Neuronale Netze

Volker Aurich

Heinrich-Heine-Universität Düsseldorf

Inhaltsverzeichnis

1 Einführung und Grundlagen **1**

1.1 Beispiele für Abbildungen in der Realität . . . 1
1.2 Musterklassifikation und Merkmalsbildung . . . 2
1.3 Erwünschte Eigenschaften einer Realisierung . . . 4
1.4 Darstellungen von Abbildungen . . . 5
1.5 Neuronale Netze . . . 6

2 Das Perzeptron **9**

2.1 Perzeptron-Lernalgorithmus . . . 9
2.2 Der Satz von Minsky und Papert . . . 10
2.3 Bemerkungen . . . 11
2.4 Das XOR-Problem . . . 11
2.5 Rosenblatt- und Minsky-Papert-Perzeptron . . . 12

3 Adaline **13**

3.1 Das lineare L^2-Approximationsproblem . . . 13
3.2 Lösung des LMS-Problems mittels Pseudoinverse (für $|I| < \infty$) . . . 14
3.3 Die Fehlerfunktion . . . 14
3.4 Minimumsuche mit Gradientenabstieg . . . 16
3.5 Unendliche Datenfolge . . . 18
3.6 Das lineare L^2-Approximationsproblem II . . . 19
3.7 Schätzung der stochastischen Größen und der LMS-Algorithmus . . . 20
3.8 Konvergenz der Erwartungswerte der Gewichtsvektoren . . . 21
3.9 Einige stochastische Begriffe . . . 22
3.10 Adaptive lineare Filter . . . 24

4 Vorwärtsgerichtete Netze mit Backpropagation-Training **29**

4.5 L^2-Approximationsaufgaben . . . 32
4.6 Lösungsstrategien . . . 33
4.7 Einige Anwendungen . . . 38
4.8 Eingangscodierung mit Principal Component Analysis (PCA) . . . 39

5 Radial Basis Functions **41**

5.3 Varianten der Approximationsaufgaben . . . 42
5.4 Festlegung der Anzahl n_2 der versteckten Neuronen . . . 42
5.5 Lernmethoden . . . 43
5.6 Vorteile gegenüber Netzen mit BP-Training . . . 43
5.7 Normierung der Eingangswerte . . . 44
5.8 Die Grossberg-Schicht . . . 44

6 Kompetitive Netze **45**
6.1 MAXNET oder „*The winner takes all*“ . . . 45
6.2 Netze zur Vektorquantisierung . . . 45
6.3 Netze, die Treppenfunktionen realisieren . . . 46
6.4 Netze zur Clusterung von Daten . . . 46
6.5 Kohonens topologieerhaltende Abbildung . . . 50

7 Support Vector Machines **55**
7.1 Optimale lineare Trennung . . . 55
7.2 Nichtlineare Transformationen und Kerne . . . 56

8 Assoziativspeicher ohne Rückkopplung **57**
8.1 Lineare Korrelationsspeicher . . . 58
8.2 Binäre Assoziativspeicher mit spärlicher Kodierung . . . 59

9 Rückgekoppelte Assoziativspeicher **61**
9.1 Idee der Funktionsweise . . . 61
9.2 Definition der verwendeten Dynamiken . . . 62
9.3 Definitionen . . . 62
9.4 Beispiele . . . 63
9.5 Forderungen beim Einsatz als Autoassoziativspeicher . . . 63
9.6 Konstruktion von Dynamiken, die vorgegebene Prototypen als Fixpunkte haben . . . 63
9.7 Die Energiefunktion . . . 64
9.8 Asynchrone Dynamik . . . 65
9.9 Synchrone Dynamik . . . 66
9.10 Eine Abschätzung der Attraktionsradien . . . 67
9.11 Bemerkungen . . . 67
9.12 Fixpunkte als lokale Energieminima . . . 67
9.13 Binäre Hopfield-Netze . . . 68

10 Kontinuierliche dynamische Systeme **69**
10.2 Zuordnung einer Differentialgleichung . . . 70
10.3 Der Fluß einer autonomen Differentialgleichung . . . 70

11 Anwendungen von Hopfield-Netzen **73**
11.1 Das Problem des Handelsreisenden (*TSP*) . . . 73

11.2 Restauration gestörter Signale 75
11.3 Stereo Matching 78
11.4 Bemerkungen 81

12 Minimierung von Funktionen mit Hilfe stochastischer Dynamiken **83**
12.1 Neuronale Netze und Optimierungsprobleme 83
12.2 Minimierung der Fehlerfunktion (vorwärtsgerichteter Netze) 84
12.3 Stochastische Suche nach globalen Minima 85

Literaturverzeichnis **89**

Kapitel 1

Einführung und Grundlagen

Neuronale Netze (kurz: NN) sind Algorithmen, die Abbildungen realisieren und vor allem „erlernen“ können.

1.1 Beispiele für Abbildungen in der Realität

a) Bild- bzw. Mustererkennung

f : Menge aller Kammerbilder $\to \{0,1\}$, $f(B) = 1$, falls auf B ein Auto zu sehen ist, und 0 sonst.

b) Spracherkennung

f : Menge aller (isoliert) gesprochenen Wörter $\to \{0,\dots,9\} \cup \{\text{keine Ziffer}\}$.

c) Inverse Kinematik in der Robotik

Gegeben sei ein Roboterarm mit n Gelenken mit je *einem* Freiheitsgrad. Dann ist die Menge der möglichen Gelenkstellungen ein n-dimensionale Mannigfaltigkeit M. Zu jeder Gelenkstellung $x \in M$ läßt sich in relativ einfacher Weise der Punkt $g(x) \in \mathbb{R}^3$ angeben, an dem sich der Greiferendpunkt befindet.

Aufgabe: Finde zu $y \in \mathbb{R}^3$ ein x mit $y = g(x)$.

Falls so ein x existiert, wird es i.a. nicht eindeutig sein. g ist also i.a. nicht invertierbar. Man sucht vielmehr einen Schnitt von g über $g(M)$, also eine Abbildung $f : g(M) \to M$ mit $g \circ f = \mathrm{Id}$. In paraktischen Anwendungen wird man von f Zusatzeigenschaften verlangen, wie z.B. stetig, differenzierbar, $\int_{g(M)} \|D(y)\|\, dy$ möglichst klein usw.

Beachte: Selbst wenn g (lokal) umkehrbar ist, ist es schwierig, f explizit anzugeben.

In wirklichen Anwendungen ist die Situation oft noch schwieriger, weil man noch die räumliche Orientierung des Greifers und die Dynamik des Arms berücksichtigen muß.

d) Prognosen: $f(\text{Wetter heute}) = \text{Wetter morgen}$

Gibt es so eine Abbildung überhaupt? Ist der meteorologische Zustand der Atmosphäre von morgen durch den heutigen bestimmt?

„Einfacheres“ Modell: $f(\text{Wetter heute}) = 1$, falls morgen Mittag die Temperatur $\geq 20°$, und 0 sonst.

Wahrscheinlich gibt es keine solche Abbildung. Aber vielleicht gibt es eine Abbildung, die in 90% aller Fälle die richtige Vorhersage liefert. Oder eine Temperaturvorhersage, die den Fehler zur wahren Temperatur im langfristigen Mittel minimiert.

e) Stabilisierung instabiler dynamischer Prozesse: Balancieren eines Stabes oder Fahrradfahren.

f) Bewegung eines Schlägers beim Tischtennis (visuell gesteuert), visuell gesteuertes Autofahren (Gas, Bremse, Lenkkräfte).

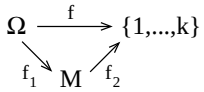
g) Karikaturzeichnungen von Menschen anfertigen.

Bemerkung. a) und b) fallen den Meschen sehr leicht, aber eine analytische Beschreibung der Abbildung f gibt es nicht. Für c) gilt ähnliches, wenn es sich um den eigenen Arm handelt, allerdings ist es schwer, so eine Abbildung zu berechnen. d) macht auch den Menschen Schwierigkeiten. Die restlichen Beispiele müssen auch vom Menschen erst erlernt werden.

1.2 Musterklassifikation und Merkmalsbildung

1.1.a) und b) sind typische Klassifikationsaufgaben. Gegeben ist eine sehr große endliche oder unendliche Menge Ω (z.B. Bilder oder gesprochene Wörter) und eine Partition $\Omega = \Omega_1 \cup \ldots \cup \Omega_k$, die jedoch nicht explizit analytisch gegeben ist, sondern nur implizit durch Beurteilung durch einen menschlichen Experten. Die Aufgabe besteht darin, die Abbildung $f : \Omega \to \{1, \ldots, k\}$, $f(\omega) = j :\Leftrightarrow \omega \in \Omega_j$, algorithmisch zu realisieren oder wenigstens zu approximieren.

Weil die Abbildung f meist höchst kompliziert und undurchsichtig ist, versucht man f als Komposition einfacher zu überblickender Abbildungen zu schreiben.



Dabei ist f_1 eine Abbildung, die i.a. hochgradig nicht injektiv ist, aber die Partition $\Omega_1 \cup \ldots \cup \Omega_k$ möglichst respektiert, das heißt, $f_1^{-1}(f_1(\Omega_j)) = \Omega_j$ erfüllen soll. $f_1 = f$ würde das natürlich tun; die Idee ist aber, daß es mit ad-hoc-Methoden (aufgrund von a-priori-Wissen) möglich ist, eine Abbildung f_1 zu konstruieren, die auf jedem Ω_j zwar nicht konstant ist, aber wenig schwankt oder so vorhersehbar schwankt, daß die Mengen $M_j := f_1(\Omega_j)$ mit relativ einfachen Mitteln getrennt werden können. f_2 ist dementsprechend zu konstruieren.

M heißt *Merkmalsraum*, $f_1(\omega)$ heißt das ω zugeordnete Merkmal. Oft ist $M \subset \mathbb{R}^n$, und $f_1(\omega)$ wird *Merkmalsvektor* genannt.

Beispiele.

a) Unterscheidung von regulären Polygonen in der Ebene: $f_1(P) :=$ (Anzahl der Ecken, Ecken, ...).

b) Unterscheidung von Tomaten, Äpfeln, Gurken anhand von Bildern, die jeweils nur ein Objekt zeigen: f_1(Bild) = (Farbe, Verhältnis der Trägheitsachsen).

c) Unterscheidung von handgeschriebenen Ziffern: $f_1(\omega) =$ (Richtung der Linien, ...).

d) Unterscheidung von gesprochenen Ziffern: $f_1(\omega) =$ Folge von Fourierspektren des Sprachsignals.

e) Bei Wetterprognosen: f_1(Wetter heute) = (Luftdruck, Temperatur, ...).

f) Bei medizinischer Bildverarbeitung: Merkmale kranker Zellen wie Farbe, Gestalt, Größe usw.

g) Bei medizinischer Signalverarbeitung: Wie kann man den für eine Krankheit typischen Verlauf eines EEG oder EKG durch Merkmale charakterisieren?

Typisch ist, daß die Merkmalsextraktion f_1 vom Menschen ad-hoc konstruiert wird. Dabei kann es durchaus passieren, daß $f_1(\Omega_i) \cap f_1(\Omega_j) \neq \emptyset$, obwohl $i \neq j$. Dann muß f_2 zwangsläufig falsch klassifizieren, das heißt, obiges Diagramm kann nicht kommutieren. Man kann dann nur noch versuchen, f_2 so zu konstruieren, daß der durch Fehlklassifikation entstehende *Schaden* (im stochastischen Mittel) minimiert wird.

Die klassische Mustererkennung beschäftigt sich mit der optimalen Konstruktion von f_2 bei gegebener Merkmalsextraktion f_1. Die Befürworter von NN-Methoden behaupten, daß sich NN für das Erlernen und Realisieren undurchsichtiger Abbildungen f eignen; in praktischen Anwendungen wird jedoch auch meist von einer gegebenen Merkmalsextraktion ausgegangen und nur die Abbildung f_2 durch ein NN erlernt. Das schließt nicht aus, daß auch f_1 durch eine NN realisiert ist; dessen Struktur ist aber meist a-priori recht genau festgelegt, so läßt sich zum Beispiel die Fouriertransformation durch ein NN realisieren. Welche Merkmale zweckmäßigerweise extrahiert werden, bleibt — zumindest bei komplizierten Situationen — dem Menschen überlassen. Ansätze

zu einer automatischen Merkmalsextraktion finden sich in der Karhunen-Loève-Transformation und Kohonens *topological map*, allerdings mit vielen wesentlichen Einschränkungen.

Oft wird als Merkmalsraum M ein metrischer Raum gewählt (z.B. $\mathbb{R}^n$ mit euklidischer Metrik), und es wird versucht, f_1 „stetig" zu machen, das heißt, Muster $\omega_1, \omega_2 \in \Omega$, die wir als ähnlich empfinden, sollen Merkmale $f_1(\omega_1)$ und $f_1(\omega_2)$ erhalten, die nicht weit voneinander entfernt sind. Man hofft, auf diese Weise die $M_j = f_1(\Omega_j)$ so zu formen, daß sie sich leicht trennen lassen; beispielsweise durch *Trennfunktionen* $h_1, \ldots, h_k$. Das sind (stetige) Funktionen $M \to \mathbb{R}$, so daß $h_j(x) > \max\{h_i(x) \mid i \neq j, i = 1, \ldots, k\}$ für alle $x \in M_j$ und j. Man braucht dann nur noch $f_2(x) := \begin{cases} j & ,\text{falls } h_j(x) > \max\{h_i(x) \mid i \neq j\} \\ \text{nicht klassifizierbar} & ,\text{sonst} \end{cases}$ zu setzen. (Die h_j sind Approximationen der χ_{M_j}, den idealen Trennfunktionen.)

In Anwendungen ist der Raum Ω der Muster oft sehr groß und f ist nur implizit gegeben (dadurch, daß ein Experte jedes einzelne vorgelegte Muster klassifiziert), so daß die Ω_j und M_j nicht explizit bekannt sind. Daher kann man nur folgendes machen: Man wählt eine endliche Folge $\omega_1, \ldots, \omega_N \in \Omega$ möglichst repräsentativ aus, läßt sie von dem Experten klassifizieren und konstruiert dann eine (überall definierte) Abbildung f_2, die die Merkmale $f_1(\omega_1), \ldots, f_1(\omega_N)$ richtig klassifiziert und gewisse Stetigkeitseigenschaften hat (also meistens nahe beieinanderliegende Merkmalsvektoren gleich klassifiziert). *Repräsentativ* soll in etwa heißen:

1. Die ω_i sind gleichmäßig über Ω verstreut; $\Omega_j \cap \{\omega_1, \ldots, \omega_N\}$ bzw. $f_1(\Omega_1) \cap f_1(\omega_1, \ldots, \omega_N)$ gibt die Gestalt von Ω_j bzw. $f_1(\Omega_j)$ gut wieder (bis auf eine vorgegebene Auflösung).
2. Ist auf Ω eine Wahrscheinlichkeitsverteilung gegeben (mit der Muster vorkommen), so sei die Folge (ω_j) eine repräsentative Stichprobe.

Führt nun auch f_1 ähnliche Muster aus derselben Klasse in benachbarte Merkmalsvektoren über, so kann man hoffen, daß $f_2 \circ f_1$ auch Muster $\omega \in \{\omega_1, \ldots, \omega_N\}$ einigermaßen richtig klassifiziert. Kann man Fehlklassifikationen einen numerischen Schaden zuordnen, so kann man versuchen, f_2 so zu wählen, daß er minimal wird. In so einem Fall kann man sogar auf einen Experten verzichten und unüberwacht klassifizieren.

Grundaufgaben der Musterklassifikation

1. Gegeben sei eine repräsentative endliche Teilmenge $\Omega_s \subset \Omega$, eine Merkmalsextraktion $f_1 : \Omega \to M$ sowie eine Klassifikationsabbildung $f_s \colon \Omega_s \to \{1, \ldots, k\}$. Sei $M_j^s := f_1(f_s^{-1}(j))$.

   Aufgabe: Finde für $M_1^s, \ldots, M_k^s$ Trennfunktionen, also Funktionen $h_1, \ldots, h_k : M \to \mathbb{R}$, die stetig oder stückweise stetig sind, so daß $h_j(x) > \max\{h_i(x) \mid i \in \{1, \ldots, k\} \backslash \{j\}\}$ für alle $x \in M_j^s$ und j. $f_2(x) := \begin{cases} j & ,\text{falls } h_j(x) > \max\{h_i(x) \mid i \neq j\} \\ \text{nicht klassifizierbar} & ,\text{sonst} \end{cases}$ ist die Klassifikationsabbildung auf dem Merkmalsraum.

2. Wie 1., nur sei zusätzlich ein Wahrscheinlichkeitsmaß p auf Ω und eine Funktion $\varrho : \Omega \times \{1, \ldots, k\} \to [0, \infty[$ gegeben, wobei man $\varrho(\omega, j)$ als Schaden, der entsteht, wenn ω mit j klassifiziert wird, interpretieren kann.

   Aufgabe: Finde Trennfunktionen wie in 1., so daß zusätzlich $\int_\Omega \varrho(\omega, f_2 \circ f_1(\omega))\, dp(\omega)$ minimal ist.

3. Gegeben seien ein Wahrscheinlichkeitsmaß p auf Ω, eine Merkmalsextraktion $f_1 : \Omega \to M$, ein $k \in \mathbb{N}$ und eine Schadensfunktion $\varrho : \Omega \times \{1, \ldots, k\} \to [0, \infty[$.

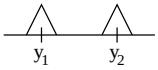
   Aufgabe: Finde $f_2 : M \to \{1, \ldots, k\}$, so daß $\int_\Omega \varrho(\omega, f_2 \circ f_1(\omega))\, dp(\omega)$ minimal ist. (Unüberwachte Klassifikation. In der Praxis muß man p meist durch Stichproben schätzen.)

Bemerkungen.

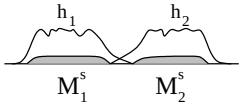
a) Statt eine Schadensfunktion zu minimieren, wird oft eine Gütefunktion maximiert.

b) Die erste Aufgabe ist ziemlich sinnlos, da trivial zu erfüllen. Weil Ω_s endlich ist, gibt es ein $\varepsilon > 0$, so daß die Kugeln $B(x, \varepsilon)$ und $B(y, \varepsilon)$ für $x, y \in f_1(\Omega_s)$ mit $x \neq y$ stets disjunkt sind. Man wähle für jedes $y \in f_1(\Omega_s)$ eine stetige Funktion $g_y : M \to [0, 1]$ mit $g_y(y) = 1$ und $g_y|_{M \backslash B(y,\varepsilon)} = 0$ und setze damit die h_j zusammen.

Beispielsweise durch $g_y(x) = 1 - \frac{1}{\varepsilon}d(x,y)$ für $x \in B(y,\varepsilon)$, und 0 sonst.



Das ist offensichtlich nicht gemeint. Oft soll eine Nebenbedingung wie in 2. erfüllt werden; meistens stellt man sich aber vor, daß die Trennfunktion h_j zwischen Merkmalen der Klasse j ähnlich große Werte annimmt und möglichst wenig schwankt (am besten die Werte konvex interpoliert, sofern das geht), während die h_i mit $i \neq j$ dort klein bleiben. Mit anderen Worten: die Merkmale zwischen Merkmalen der Klasse j sollen auch mit j klassifiziert werden.



1.3 Erwünschte Eigenschaften einer Realisierung

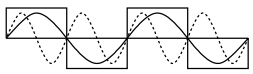
a) technische Realisierbarkeit.

b) Ausgabe des Funktionswertes innerhalb einer vorgegebenen Dauer. Das ist bei manchen Anwendungen nicht trivial; beispielsweise bei der visuellen Steuerung eines Fahrzeuges. Die Geschwindigkeitsanforderungen sind oft mit seriellen Rechnern nicht zu erfüllen. Daher muß die Abbildung in massiv paralleler Weise realisiert werden.

c) Fehlertoleranz. Auch bei Ausfall einiger Hardwarekomponenten sollte sich das Abbildungsverhalten nicht total ändern.

d) Erlernbarkeit. Bei vielen Anwendungen steht man vor dem Problem, eine Abbildung realisieren zu müssen, die man gar nicht explizit kennt. (Manchmal kann man nicht einmal einzelne Werte explizit angeben; vergleiche dazu 1.1.e)-g)) In vielen Fällen kennt man nur endlich viele Punkte aus dem Graphen. (Lernstichprobe, Trainingsfolge) Oft wird in der NN-Literatur der Eindruck erweckt, damit könne man die Abbildung erlernen. Falls es sich um irgendeine stetige Abbildung $f : \mathbb{R}^n \to \mathbb{R}^m$ handelt, kann das natürlich nicht gehen, außer f erfüllt geeignete Zusatzvoraussetzungen.

Triviales Beispiel: $f : [0,1] \to \mathbb{R}$, $S = \{(\frac{k}{4}, 0) \mid k = 0, \ldots, 4\} \subset \text{graph}(f)$.



f ist eindeutig bestimmt, wenn man fordert, daß f eine Sinusschwingung mit Frequenz $< 4\pi$ ist; denn dann ist $f \equiv 0$. (Spezialfall des Abtasttheorems von Shannon)

Allgemein kann man eigentlich nur folgendes tun:

Man wählt eine parametrisierte Familie von Abbildungen, die man realisieren kann, und versucht die Parameter so zu bestimmen, daß die vorgegebenen Punkte aus graph(f) bestmöglich approximiert werden. Wenn dies nicht eindeutig möglich ist, braucht man Zusatzbedingungen.

Die Abbildungen der Familie sollten die Eigenschaften (stetig, glatt, wenig oszillierend o.ä.) haben, die man aufgrund von a-priori-Wissen von der wirklichen Abbildung f erwartet (oder man approximiert f ganz bewußt mit Abbildungen gewisser anderer Eigenschaften, zum Beispiel um Rauschen zu unterdrücken). Soweit ist alles wie bei der klassischen Approximations- bzw. Interpolationstheorie. Nur werden dort meist große Gleichungssysteme gelöst, um die optimalen Parameter zu bestimmen und die approximierenden Familien sind oft Vektorräume von Abbildungen. Bei NN dagegen approximiert man aus nichtlinearen Familien heraus und versucht die optimalen Parameter allmählich in mehr oder weniger vielen Iterationsschritten zu bestimmen; beispielsweise jedesmal, wenn ein neuer Punkt aus graph(f) vorliegt.

1.4 Darstellungen von Abbildungen

Wir betrachten nur Abbildungen, deren Quelle und Ziel diskrete Räume (abzählbar) oder Teilmengen (Umfang, etc.) des $\mathbb{R}^n$ sind. Außerdem setzen wir voraus, daß sie stetig oder differenzierbar sind (zumindest stückweise, was immer dies im Mehrdimensionalen bedeuten mag). In Anwendungen genügt es natürlich, sie genügend genau zu approximieren. Einerseits sucht man möglichst einfache komprimierte Darstellungen, um den Implementierungsaufwand klein zu halten, andererseits aber auch redundante Darstellungen, um Fehlertoleranz zu erreichen.

1.4.1. Abbildungen, die auf einem diskreten — in der Praxis stets endlichem — Raum definiert sind, kann man einfach durch eine *Tabelle* darstellen. Das kann wegen der Tabellengröße praktisch nicht zu verwirklichen sein; beispielsweise bei der Dekodierung fehlerkorrigierender Codes mit großer Wortlänge). Ausweg: Angabe eines Algorithmus.

1.4.2. Darstellung von Abbildungen durch eine Rechenvorschrift (Algorithmus), wie zum Beispiel eine numerisch auswertbare *Formel*.

1.4.3. Darstellung einer Abbildung durch die Koeffizienten ihrer *Entwicklung nach Basisfunktionen*.

1.4.4. Viele Abbildungen lassen sich mit Hilfe einfacher Operationen aus einfachen Funktionen zusammensetzen. So zum Beispiel Boolsche Funktionen aus „$\neg$“ und „$\vee$“.

Satz (Kolmogorov 1957, Sprecher 1965)**.**
Zu jedem $N \in \mathbb{N}$, $N \geq 2$, gibt es eine monoton wachsende, surjektive Funktion $\psi : [0,1] \to [0,1]$, die Hölderstetig mit Exponent $\frac{\ln(2)}{\ln(2N+2)}$ ist und folgende Eigenschaft besitzt:

Zu jedem $\delta > 0$ existiert ein rationales $\epsilon \in]0, \delta]$ und ein $\lambda \in \mathbb{R}$, so daß es zu jeder stetigen Funktion $f : [0,1]^n \to \mathbb{R}$ mit $2 \leq n \leq N$ eine stetige Funktion $h : \mathbb{R} \to \mathbb{R}$ gibt, so daß

$$f(x) = \sum_{j=0}^{2n} h\left(\sum_{i=0}^{n} \lambda^i \psi(x_i + \epsilon j) + j\right), \text{ für jedes } x = (x_1, \ldots, x_n) \in [0,1]^n.$$

Siehe dazu: [9].

Korollar. Zu jeder stetigen Funktion $f : [0,1]^n \to \mathbb{R}$ gibt es stetige Funktionen h_j und φ_{ij} einer Veränderlichen, so daß

$$f(x) = \sum_{j=0}^{2n} h_j\left(\sum_{i=0}^{n} \varphi_{ij}(x_i)\right), \text{ für jedes } x \in [0,1]^n.$$

Dies entspricht etwa der Formulierung Kolmogorovs.

Bemerkung. Kurz gesagt: Jede stetige Funktion auf $[0,1]^n$ läßt sich *exakt* mit Hilfe von Addition und Komposition aus *endlich* vielen stetigen Funktionen einer Veränderlichen zusammenbauen. Das löst Hilberts 13. Problem.

Verzichtet man auf die exakte Darstellbarkeit und ist mit (beliebig guten) Approximationen zufrieden, so muß man nur den einfacheren Satz von Stone-Weierstraß bemühen und erhält, daß die Algebra $\mathcal{C}([0,1])^{\otimes n}$ dicht in $\mathcal{C}([0,1]^n)$ ist (bzgl. der sup-Norm), das heißt, zu jedem $\varepsilon > 0$ und jedem stetigen $f : [0,1]^n \to \mathbb{R}$ gibt es $\varphi_{ij} : [0,1] \to \mathbb{R}$, $i = 1, \ldots, n$, so daß

$$\left| f(x_1, \ldots, x_n) - \sum_{j=1}^{k} \prod_{i=1}^{n} \varphi_{ij}(x_i) \right| < \varepsilon, \text{ für jedes } (x_1, \ldots, x_n) \in [0,1]^n.$$

1.4.5. Jede Abbildung f läßt sich schreiben als $f(x) = \sum_y y \chi_{f^{-1}(y)}(x)$, wobei $\chi_{f^{-1}(y)}$ die charakteristische Funktion der Faser $f^{-1}(y)$ ist. Die Summe hat zwar unendlich viele Glieder, jedoch ist für jedes x höchstens eines ungleich 0.

Ist nun $f\colon K \to \mathbb{R}$ stetig und $K \subset \mathbb{R}^n$ kompakt, so ist $f(K)$ beschränkt (o.E. sei $f(K) \subset [0,1]$) und f gleichmäßig stetig. Deshalb existiert zu jedem $\varepsilon > 0$ ein $m \in \mathbb{N}$, so daß

$$\left| f(x) - \sum_{i=1}^{m} \tfrac{i}{m} \chi_{f^{-1}(]\frac{i-1}{m}, \frac{i}{m}])}(x) \right| < \varepsilon, \text{ für jedes } x \in K.$$

Damit diese Darstellung nützlich ist, müssen die *Niveaumengen* $\{x \mid \frac{i-1}{m} < f(x) \leq \frac{i}{m}\}$ eine einfach beschreibbare geometrische Gestalt haben, zumindest sollten sie sich durch einfach beschreibbare Mengen gut approximieren lassen; zum Beispiel durch

- *semialgebraische* Mengen; das sind Mengen, die durch polynomiale Ungleichungen definiert werden, also die Gestalt $\{x \mid P_1 < 0, \ldots, P_l < 0, Q_1(x) \leq 0, \ldots, Q_k(x) \leq 0\}$ haben, wobei P_i und Q_j Polynome sind, oder endliche Vereinigung von solchen Mengen sind.
- *Polyeder* (= endliche Vereinigung von konvexen Polyedern); das sind spezielle semialgebraische Mengen, bei denen alle definierenden Polynome affin linear sind. (McCulloch-Pitts-Neurone)

Um einen Interpolationseffekt zu erhalten, kann man statt der charakteristischen Funktionen der Niveaumengen stetige Approximationen verwenden, die man aus den definierenden Polynomen zusammenbaut.
1.4.6. Eine *implizite Darstellung* einer Abbildung $f : X \to Y$, $X \subset \mathbb{R}^n$, $Y \subset \mathbb{R}^m$, wird gegeben durch eine Abbildung $\Phi = \Delta_y E$, wobei $E : X \times Y \to \mathbb{R}$ nach y differenzierbar ist. Die übliche Interpretation ist, für jedes $x \in X$ die Funktion $E(x, \cdot)$ als Potential zu sehen, das in $y = f(x)$ sein Minimum (oder auch Maximum) hat.
1.4.7. Gegeben seien Gebiete $X \subset \mathbb{R}^n$, $Y \subset \mathbb{R}^m$ und eine („genügend schöne“) Abbildung $\Phi : X \times Y \to \mathbb{R}^m$. Für jedes $x \in X$ habe das dynamische System $u' = \Phi(x, u)$ genau einen *asymptotisch stabilen Fixpunkt* $f(x)$. Um f zu berechnen, braucht man also nur einer in den Fixpunkt laufenden Trajektorie zu folgen. Häufig ist das dynamische System ein Gradientensystem wie in 1.4.6.

Welche Funktionen f lassen sich so darstellen? Jede: $E(x, y) := (f(x) - y)^2$, $\Phi(x, y) = -2(f(x) - y)$.
1.4.8. Abbildungen zwischen diskreten Räumen werden manchmal auch mit Hilfe von *Regeln* in einem logischen Kalkül beschrieben (Expertensysteme). Mit Hilfe von *Fuzzy Logic* wird versucht, eine ähnliche Beschreibung im kontinuierlichen Fall zu erhalten. Meines Erachtens läuft dies meist (zumindest bei *Fuzzy Control*) darauf hinaus, stetige Funktionen aus einfachen Funktionen mit Hilfe einfacher Operationen zusammenzubauen.

1.5 Neuronale Netze

Tabellen und Formeln eignen sich nur in speziellen Situationen zur Realisierung von Abbildungen.

Die Entwicklung nach Basisfunktionen dagegen ist in vielen Anwendungen sehr gut einsetzbar; in der NN-Literatur findet man sie oft unter dem Stichwort *radial basis function*. Solche Netze haben eine einfache Struktur und eine übersichtliche Funktionsweise, und sie lernen sehr schnell.

Die Sätze von Kolmogorov und Sprecher sehen zwar vielversprechend aus, sind aber praktisch nicht verwendbar, weil die Funktionen h, ψ, φ_{ij} nicht explizit bekannt sind (nur ihre Existenz).

Die Approximation von Niveaumengen durch semialgebraische Mengen liefert dagegen einen in der Praxis brauchbaren Ansatz, Meist läßt man zur Vereinfachung nur affin lineare Ungleichungen, also Polyeder, zu. Die definierenden Polynome haben also die Gestalt $x \mapsto \langle w, x\rangle - \Theta$, wobei $w \in \mathbb{R}^n$ und $\Theta \in \mathbb{R}$. Um zwischen den Niveauwerten interpolieren zu können, setzt man oft noch eine *sigmoide* Funktion h dahinter, das ist eine monoton wachsende Funktion $h : \mathbb{R} \to [0, 1]$ (oder $h : \mathbb{R} \to [-1, 1]$).
Beispiele (für sigmoide Funktionen)**.**

1) $h \equiv \text{signum}$ oder $h \equiv \frac{1}{2}(\text{signum} + 1)$.

2) $h \equiv \chi_{\mathbb{R}_+}$ oder $h \equiv \chi_{[\Theta, \infty[}$.

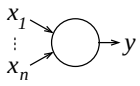
3) $h(t) = \frac{1}{\sqrt{\pi}} \int_{-\infty}^{t} \exp(-s^2)\, ds$.

4) $h(t) = \frac{1}{1+e^{-at}}$.

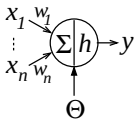
5) $h(t) = \frac{2}{\pi} \tanh(t)$.

Man hofft nun, die in Anwendungen auftretenden Funktionen f durch geeignete Verschachtelung solcher einfacher Elementarfunktionen ausreichend genau approximieren zu können.

Ein Gerät (oder Programm), das so eine Elementarfunktion realisiert, wird ein (künstliches) *Neuron* genannt. Wir symbolisieren ein solches Netron durch:



Am üblichsten sind die *McCulloch-Pitts-Neutrone*, die eine Funktion der Gestalt $x \mapsto h(w^T x - \Theta)$ mit sigmoidem h, $w \in \mathbb{R}^n$ und $\Theta \in \mathbb{R}$ realisieren. Symbolisch:



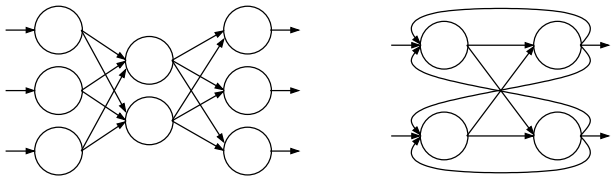
Die Eingänge heißen *Synapsen*, die w_i *Gewichte*, Θ *Schwelle* und h Aktivierungsfunktion.

Ein *neuronales Netz* (auch kurz: NN) erhält man, wenn man mehrere Neuronen miteinander verschaltet, indem man einige Ausgänge mit Eingängen verbindet. (Ein Ausgang darf mit mehreren Eingängen verbunden werden.) Die nicht beschalteten Eingänge dienen als Eingänge des gesamten Netzes; über einige Ausgänge wird der Wert der vom Netz realisierten Abbildung ausgegeben. Die Neuronen müssen nicht alle dieselbe Gestalt haben; insbesondere kann die Zahl der Eingänge variieren, und durch die Gewichte werden gerade die jeweiligen Abbildungseigenschaften der Neuronen eingestellt. Man kann das auch so auffassen: Ein NN ist ein gewichteter gerichteter Graph; die Knoten sind die Neuronen (ohne Gewichtung der Eingänge), die Kanten sind die Verbindungen zwischen den Neuronen, die Kantengewichte sind die Synapsengewichte.

Ein NN aus L Neuronen $N_1, \ldots, N_L$ heißt *vollständig*, wenn jedes Neuron N_j $L-1$ Eingänge $x_{j1}, \ldots, x_{jL-1}$ und einen Ausgang y_j hat, so daß für jedes $i = 1, \ldots, L$ der Ausgang y_i von N_i für $i \neq j$ mit x_{ji} verbunden ist mit dem Synapsengewicht w_{ij}. Offensichtlich läßt sich jedes NN aus McCulloch-Pitts-Neuronen so darstellen, indem man geeignete Synapsengewichte gleich Null setzt.

Ein NN heißt *vorwärtsgerichtet* (*feedforward net*), wenn keine zyklischen gerichteten Kantenwege existieren, ansonsten *rekurrent* (Netz mit Rückkopplung, *feedback net*).

Vorwärtsgerichtete NN werden eingesetzt, um Abbildungen in so ähnlicher Weise wie in 1.4.5 zu realisieren; rekurrente Netze realisieren eher die Darstellung von Abbildungen durch dynamische Systeme und Minimierung von Energiefunktionen.



vorwärtsgerichtetes Netz mit 3 Lagen — rekurrentes Netz

Die sigmoide Funktion h und die Schwelle Θ wird meist für alle Neurone gleich gewählt. Das NN „lernt“ die gewünschte Abbildung nur durch Veränderung der Synapsengewichte. Wie dieses Training am günstigsten erfolgt, ist ein wichtiger Forschungsgegenstand.

Im Gegensatz zu den Sätzen von Kolmogorov und Sprecher ist für solche NN überhaupt nicht klar, wieviele Neuronen und welche Verbindungen nötig sind, um eine gewünschte Abbildung zu realisieren. Zur Zeit gibt es hauptsächlich heuristische Kriterien, welche NN-Architektur für welche Anwendung günstig ist.

Unsere bisherige Darstellung gibt nur einen groben Eindruck von NN; es gibt viele unterschiedliche Varianten und Modifikationen, z.B. Neurone, die Funktionen der Gestalt $x \mapsto h(P(x))$ mit einem Polynom P realisieren ($\sum$-$\prod$-Units) oder die einen internen Zustand zwischenspeichern und ihren Ausgang gemäß einer Differentialgleichung (eventuell Funktionaldgl.) einstellen. Dazu später mehr.

Noch ein Wort zur Biologie: Die Analogie zu biologischen Neuronen und NN erscheint mir immer noch oberflächlich.

a) Biologische Neurone sind nichtlineare Oszillatoren, die von den Eingängen gesteuert werden. Die Ein- und Ausgangssignale sind Pulsfolgen (*spike trains*). Es wird allgemein angenommen, daß die momentane Frequenz als Ein- bzw. Ausgangswert zu interpretieren ist. Es gibt Neuroinformatiker, die auch künstliche

Neurone als Oszillatoren realisieren.

b) Künstliche NN umfassen höchstens einige 1000 Neurone; ein künstliches Neuron hat meist nur einige 100 Synapsen. Trotzdem sollen künstliche NN recht komplizierte Aufgaben lösen, während offensichtlich in Lebewesen schon für einfache Verarbeitungsschritte große NN eingesetzt werden.

Daten über Gehirn und Sinnesorgane

Das Gewicht des menschlichen Gehirns eines Neugeborenen beträgt etwa 300g, während es bei einem 15jährigen Menschen circa 1500g wiegt.

Bei einer Katze soll sich die Anzahl der Synapsen eines Großhirnneurons zwischen dem achten und 37. Tag von einigen wenigen auf etwa 13.000 erhöhen.

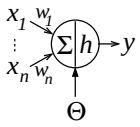
Wiederum beim Menschen schätzt man die Anzahl der Neuronen im Gehirn auf etwa 10^{10} bis 10^{11} und man nimmt eine Synapsenzahl von 10^{14} an. Man geht von etwa 10^8 Sehzellen in der Retina, 15.000 Sinneszellen im Ohr und 2.000 Geschmacksknospen aus.

Weiterführende Literatur: [4] und „Wahrnehmung und visuelles System“ erschienen im Spektrum der Wissenschaft-Verlagsgesellschaft 1986/87.

Kapitel 2

Das Perzeptron

Ein Perzeptron ist ein vorwärtsgerichtetes Netz aus Neuronen der Gestalt



mit $y = f(x) = \text{signum}(w^T x - \Theta) = \begin{cases} +1 & \text{, falls } \sum_j w_j x_j > \Theta \\ -1 & \text{, falls } \sum_j w_j x_j < \Theta \\ 0 & \text{, sonst} \end{cases}$, wobei $w = (w_1, \ldots, w_n)^T$.

Man bezeichnet sie auch als *lineare Schwellwerteinheit* (*linear thresholding unit*, LTU).

Die Menge $H := \{x \in \mathbb{R}^n \mid w^T x - \Theta = 0\}$ ist eine Hyperebene im $\mathbb{R}^n$ (falls $w \neq 0$), und zwar diejenige, welche senkrecht auf w steht und den Punkt $\frac{\Theta}{\|w\|^2} w$ enthält. Sie teilt den $\mathbb{R}^n$ in zwei Halbräume

$$H^+ := \{x \in \mathbb{R}^n \mid w^T x - \Theta > 0\} \text{ und } H^- := \{x \in \mathbb{R}^n \mid w^T x - \Theta < 0\}.$$

Eine LTU kann also eine einfache Klassifkation vornehmen, nämlich für jeden Punkt $x \in \mathbb{R}^n$ angeben, ob er in H^+, H^- oder H liegt. (H ist eigentlich eine orientierte Hyperebene.)

In Anwendungen stellt sich folgendes Problem:

Gegeben seien zwei Mengen M^+, $M^- \subset \mathbb{R}^n$. Wie findet man eine LTU die M^+ und M^- unterscheidet?
Mit anderen Worten wie findet man eine Hyperebene H, so daß $M^+ \subset H^+$ und $M^- \subset H^-$?

Triviale Beobachtung: Das geht nicht immer.

Wir nennen M^+ und M^- *linear trennbar*, wenn es eine affin lineare Funktion $f(x) = w^T x - \Theta$ gibt, so daß f auf M^+ positiv und auf M^- negativ ist; so ein f heißt dann eine *lineare Diskriminantenfunktion* für (M^+, M^-). Ist $g : M^+ \cup M^- \to \mathbb{R}$ irgendeine Funktion, so sagen wir, *g klassifiziere einen Punkt $x \in M^+ \cup M^-$ richtig*, wenn $g(x) > 0$ für $x \in M^+$ und $g(x) < 0$ für $x \in M^-$.

Eine *Trainingsfolge* für $M \subset \mathbb{R}^n$ ist eine Folge $(x_t)_{t \in \mathbb{N}}$ in M, so daß $\{t \mid x_t = x\}$ für jedes $x \in M$ unendlich ist.
Bemerkung. Oft wird eine andere Aktivierungsfunktion verwendet, nämlich $\chi_{\mathbb{R}_+}$. Man wirft also den negativen Halbraum H^- und die Hyperebene H in eine Klasse, die den Wert 0 bekommt. Der Wert 0 statt -1 ist suggestiver, wenn man Boolesche Funktionen realisieren will.

2.1 Perzeptron-Lernalgorithmus

Eingabe: Zwei endliche Mengen M^+ und M^- im $\mathbb{R}^n$ und eine Trainingsfolge $(x_t)_{t \in \mathbb{N}}$ für $M^+ \cup M^-$.

Ziel: Finde eine lineare Diskriminantenfunktion f für (M^+, M^-).

Algorithmus: Siehe nächste Seite.

Wähle $w_0 \in \mathbb{R}^n$ und $\Theta_0 \in \mathbb{R}$ beliebig;
Setze $t = 1$ und $f(x) = w_0^T x - \Theta_0$;
repeat {
  if („f klassifiziert x_t richtig“)
    then $w_t = w_{t-1}$ und $\Theta_t = \Theta_{t-1}$;
    else if ($x_t \in M^+$)
      then $w_t = w_{t-1} + x_t$ und $\Theta_t = \Theta_{t-1} - 1$;
      else $w_t = w_{t-1} - x_t$ und $\Theta_t = \Theta_{t-1} + 1$;
  $f(x) = w_t^T x - \Theta_t$;
  $t = t + 1$;
} until („f ist eine Diskriminantenfunktion für (M^+, M^-)“);

Algorithmus 2.1: Perzeptron-Lernalgorithmus

2.2 Der Satz von Minsky und Papert

Satz (1969)**.** Sind M^+ und M^- linear trennbar, so terminiert der Perzeptron-Lernalgorithmus.

Beweis. Weil M^+ und M^- endlich sind, gibt es ein $C > 0$ mit $\|x\|^2 \leq C$ für $x \in M^+ \cup M^-$. Da M^+ und M^- endlich und linear trennbar sind, gibt es $w_* \in \mathbb{R}^n$ und $\Theta_* \in \mathbb{R}$, so daß $\|w_*\|^2 + \Theta_*^2 = 1$, $w_*^T x - \Theta_* > \delta$ für jedes $x \in M^+$ und $w_*^T x - \Theta_* < -\delta$ für jedes $x \in M^-$.

Setze $u_* = (w_*, \Theta_*) \in \mathbb{R}^{n+1}$, $u_t = (w_t, \Theta_t) \in \mathbb{R}^{n+1}$, $z_t = (x_t, -1) \in \mathbb{R}^{n+1}$ und $\mu(t) = \#\{s \in \mathbb{N} \mid s \leq t, x_s$ wird innerhalb der repeat-Schleife nicht richtig klassifiziert $\}$.

Wenn x_t nicht richtig klassifiziert wird, so gelten

$$u_*^T u_t > u_*^T u_{t-1} + \delta \tag{1}$$
$$\|u_t\|^2 \leq \|u_{t-1}\|^2 + C + 1 \tag{2}$$

Beweis. Fallunterscheidung in $x_t \in M^+$ oder $x_t \in M^-$.

1. $x_t \in M^+$. Dann ist $u_t = u_{t-1} + z_t$. Es folgt

$$\begin{aligned} u_*^T u_t &= u_*^T u_{t-1} + u_*^T z_t = u_*^T u_{t-1} + w_*^T x_t - \Theta_* > u_*^T u_{t-1} + \delta \\ \|u_t\|^2 &= \|u_{t-1}\|^2 + 2\underbrace{u_{t-1}^T z_t}_{\leq 0} + \|z_t\|^2 \leq \|u_{t-1}\|^2 + \|x_t\|^2 + 1 \leq \|u_{t-1}\|^2 + C + 1 \end{aligned}$$

2. $x_t \in M^-$. Dann ist $u_t = u_{t-1} - z_t$. Die Ungleichungen folgen auf analoge Weise.

□

Aus (1) folgt induktiv

$$u_*^T u_t > u_*^T u_0 + \mu(t) \cdot \delta$$

und wegen $\|u_*\| = 1$ ergibt sich mit der Cauchy-Schwarschen-Ungleichung

$$\|u_t\| \geq u_*^T u_t > u_*^T u_0 + \mu(t) \cdot \delta. \tag{3}$$

Aus (2) folgt induktiv

$$\|u_t\|^2 \leq \|u_0\|^2 + \mu(t)(C+1). \tag{4}$$

Aus (3) und (4) ergibt sich

$$\left(u_*^T u_0 + \mu(t) \cdot \delta\right)^2 < \|u_0\|^2 + \mu(t)(C+1), \text{ für jedes } t \in \mathbb{N}.$$

Weil die linke Seite quadratisch in $\mu(t)$ ist, die rechte aber nur linear, muß μ beschränkt sein. Da μ nach Definition monoton wachsend und ganzzahlig ist, wird es schließlich konstant, das heißt, daß es ein $t_0 \in \mathbb{N}$ gibt,

so daß $\mu(t) = \mu(t_0)$ für jedes $t \geq t_0$, also alle x_t mit $t \geq t_0$ von $f(x) = w_{t_0}^T x - \Theta_{t_0}$ richtig klassifiziert werden. Weil $(x_t)_{t\in\mathbb{N}}$ eine Trainingsfolge ist, kommt jedes $x \in M^+ \cup M^-$ unter den x_t mit $t \geq t_0$ vor. Folglich ist f eine lineare Diskriminantenfunktion für (M^+, M^-) und der Lernalgorithmus terminiert spätestens nach dem t_0-ten Schleifendurchlauf. □

2.3 Bemerkungen

1) Rosenblatt baute 1958 ein Perzeptron tatsächlich auf. Er benutzte eine leicht modifizierte Lernregel: $(w_t, \Theta_t) = (w_{t-1}, \Theta_{t-1}) \pm \alpha(x_t, -1)$, wobei $\alpha > 0$ die Größe der Veränderung beeinflußt. Der Konvergenzbeweis bleibt auch dafür richtig.

2) Die Anzahl der benötigten Lernschritte hängt nicht nur von M^+ und M^- ab, sondern auch stark von w_0, Θ_0 und der gewählten Trainigsfolge $(x_t)_{t\in\mathbb{N}}$.

3) Für die Wahl von w_0 und Θ_0 gibt es verschiedene Heuristiken. Zum Beispiel kann man für (w_0, Θ_0) den Schwerpunkt der Punkte $(x, -1)$ mit $x \in M^+$ wählen.

4) Modifikationen zur Beschleunigung des Lernens

   a) Weil die Länge der Vektoren (w_t, Θ_t) meist mit t wächst, werden die durchgeführten Korrekturen relativ immer kleiner. Das kann man durch Normierung verhindern.

   b) Abändern der Trainingsfolge: Wird x_t falsch klassifiziert, so überprüft man nach der Korrektur der Gewichte, ob derselbe Punkt x_t jetzt richtig klassifiziert wird. Wenn nicht, werden die Gewichte wieder entsprechend korrigiert. Das wiederholt man, bis x_t richtig klassifiziert wird; erst dann geht man zu x_{t+1} über.

   c) Abändern der Gewichtskorrektur: $\alpha > 1$ sei vorgegeben.
   Wird x_t falsch klassifiziert, so sei $\delta = \frac{\alpha}{1+\|x_t\|^2} w_{t-1}^T x_t - \Theta_{t-1}$.
   Die Korrektur der Gewichte wird jetzt abgeändert zu

$$w_t = w_{t-1} - \delta x_t \qquad \text{und} \qquad \Theta_t = \Theta_{t-1} + \delta.$$

   Dann wird x_t durch $f(x) = w_t^T x - \Theta_t$ richtig klassifiziert, denn

$$f(x_t) = w_t^T x_t - \Theta_t = w_{t-1}^T x_t - \Theta_{t-1} - \delta\|x_t\|^2 - \delta = (1-\alpha)(w_{t-1}^T x_t - \Theta_t) = \begin{cases} \text{positiv für } x_t \in M^+ \\ \text{negativ für } x_t \in M^- \end{cases}.$$

   Aber Vorsicht: Ist α zu groß, so kann der Fehler für andere Punkte wieder sehr groß werden. Es ist nicht garantiert, daß der Algorithmus terminiert.
   Eine andere Möglichkeit, die Gewichtskorrektur zu modifizieren, besteht darin, α mit t zu variieren (und die Normierung wegzulassen):

$$w_t = w_{t-1} - \alpha_t(w_{t-1}^T x_t - \Theta_{t-1})x_t \qquad \text{und} \qquad \Theta_t = \Theta_{t-1} - \alpha_t(w_{t-1}^T x_t - \Theta_{t-1}).$$

   Dabei wählt man für $(\alpha_t)_{t\in\mathbb{N}}$ eine Nullfolge. (*Delta-Regel*) Damit kann man zwar die Konvergenz von $(w_t)_{t\in\mathbb{N}}$ und $(\Theta_t)_{t\in\mathbb{N}}$ erzwingen, allerdings nicht notwendiger gegen die Koeffizienten einer Diskriminantenfunktion für (M^+, M^-).

2.4 Das XOR-Problem

AND:	x_1	x_2	y
	0	0	0
	0	1	0
	1	0	0
	1	1	1

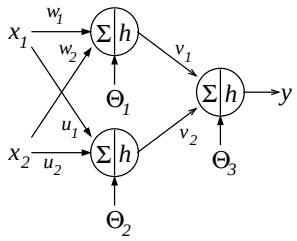
OR:	x_1	x_2	y
	0	0	0
	0	1	1
	1	0	1
	1	1	1

XOR:	x_1	x_2	y
	0	0	0
	0	1	1
	1	0	1
	1	1	0

Ersetzt man im Ausgang y den Wert 0 durch -1, so bleibt ein Klassifizierungsproblem zu lösen:

Sind die Mengen $M^+ = \{(x_1, x_2) \mid y = f(x_1, x_2) = 1\}$ und $M^- = \{(x_1, x_2) \mid y = f(x_1, x_2) = 0\}$ linear trennbar? Sowohl die AND- als auch die OR-Verknüpfung lassen sich linear trennen. Ein einlagiges Perzeptron allerdings kann die XOR-Verknüpfung nicht realisieren. Mit einem zweilagigen geht es:

Dazu stelle die XOR-Verknüpfung so dar: x_1 XOR $x_2 = (x_1$ OR $x_2)$ AND $(x_1$ NAND $x_2)$.



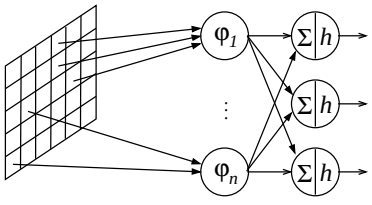
OR	NAND	XOR
$w_1 = w_2 = 1$	$u_1 = u_2 = -1$	$v_1 = v_2 = 1$
$\Theta_1 = \frac{1}{2}$	$\Theta_2 = -\frac{3}{2}$	$\Theta_3 = \frac{1}{2}$

2.5 Rosenblatt- und Minsky-Papert-Perzeptron

Eine LTU realisiert die charakteristische Funktion eines (offenen) Halbraums. Mit zwei Lagen von LTUs kann man die charakteristische Funktion jedes Konvexen Polytops (= Durchschnitt von endlich vielen Halbräumen) realisieren und mit drei Lagen die charakteristische Funktion beliebiger Vereinigungen endlich vieler Polytope (statt 0 wird der Wert -1 angenommen).

Somit müßte ein 3-lagiges Perzeptron für Klassifiaktionsaufgaben gut geeignet sein. Wenn es nicht einen gravierenden Nachteil gäbe: Es ist überhaupt nicht klar, wie ein mehrlagiges Perzeptron trainiert werden kann und wieviele LTUs es enthalten muß, um eine Klassifiaktionsaufgabe zu lösen.

Rosenblatt baute 1958 ein Perzeptron (Mark I) auf, das Buchstaben erkennen sollte. Jeder Buchstabe wurde in einem 20×20-Feld von Bildpunkten durch ein Schwarz-Weiß-Muster dargestellt. Das Perzeptron war 2-lagig; die Verbindungen zwischen den Bildpunktsensoren und den Neuronen der ersten Lage waren festverdrahtet; nur die Gewichte der 2-ten Lage wurden gelernt (512 Gewichte aus motorisierten Potentiometern).



Die Neuronen der 1. Lage realisierten jeweils eine Abbildung $\varphi : \{s, w\}^{20\times 20} \to 0, 1$, das man als Prädikat interpretieren kann, d.h. $\varphi(b) = 1$ bedeutet, daß b eine Eigenschaft hat, und $\varphi(b) = 0$, daß nicht. Gedacht war dabei an lokale geometrische Eigenschaften wie:

$$b(i,k) = s \text{ und } b(i-1,k) = s \text{ und } b(i-2,k) = s$$

oder

$$b(i,k) = s \text{ und } b(i+1,k) = s \text{ und } b(i+1,k+1) = s$$

Man hoffte durch geschickte Wahl dieser Abbildungen φ das Bild so kodieren zu können, daß in der 2. Lage lineare Schwellwerteinheiten die Klassifikation durchführen können.

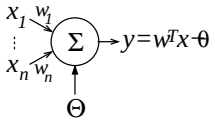
Minsky und Papert zeigten, daß dies nicht immer möglich ist, wenn man von den Abbildungen φ fordert, daß sie jeweils nur von m Bildpunkten ($m <$ Anzahl aller Bildpunkte) oder jeweils nur von Bildpunkten innerhalb eines kleines Quadrats (mit Seitenlänge $<$ Bildseitenlänge) abhängen (Perzeptron von beschränkter Ordnung oder von beschränktem Durchmesser). So kann damit zum Beispiel nicht festgestellt werden, ob die Menge der schwarzen Pixel zusammenhängt oder ob sie eine ungerade Anzahl von Pixeln hat (Paritätsproblem). Beim Paritätsproblem wächst sowohl die Anzahl der in der 1. Lage benötigten Neuronen wie auch die Anzahl der benötigten Lernschritte exponentiell mit der Anzahl der Bildpunkte.

Kapitel 3

Adaline

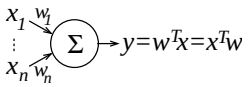
Adaline bedeutet ursprünglich *adaptive linear neuron*, heute meint man häufig *adaptive linear element*.

Ein Adaline ist ein McCulloch-Pitts-Neuron ohne sigmoider Aktivierungsfunktion (oder mit linearer), hat also die Gestalt



Die Sonderrolle der Schwelle Θ kann man eliminieren, indem man einen $(n+1)$-ten Eingang x_0 hinzunimmt, der stets den Wert -1 hat, und $w_0 = \theta$ setzt. Dann gilt $y = \overline{w}^T \overline{x}$.

Wir werden deshalb im folgenden nur Adalines mit Schwellwert 0 betrachten, die wir kurz als ALC (*adaptive linear combiner*) nennen. Sie haben die Gestalt



Wir werden nun untersuchen, wie man eine gegebene Trainingsfolge von Punkten (x, y) durch einen ALC „bestmöglich“ approximieren kann. So eine lineare Approximation wird zwar oft miserabel sein; man kann jedoch dabei schon viele Methoden und Probleme kennenlernen, die später auch für die Approximation durch nichtlineare NN typisch sind. Außerdem gibt es eine ganze Reihe wichtiger technischer Anwendungen in der Form adaptiver linearer Filter.

3.1 Das lineare L^2-Approximationsproblem

Gegeben sei eine Folge $(a_i)_{i \in I}$ von Punkten im $\mathbb{R}^n$ und eine Folge $(b_i)_{i \in I}$ reeller Zahlen; dabei sei $I = \mathbb{N}$ oder $I = \{1, \ldots, N\}$ mit $N \in \mathbb{N}$.

Gesucht ist ein $w \in \mathbb{R}^n$, so daß $\sum_{i \in I} |b_i - a_i^T w|^2 = \|(b_i - a_i^T w)_{i \in I}\|^2$ minimal wird, wobei $\| \cdot \|$ die l^2-Norm bezeichne.

Bemerkungen.

1) Statt der Euklidischen Norm kann man jede andere Norm verwenden; insbesondere die l^1- oder l^∞-Norm. Sie sind aber meist schwieriger zu behandeln.

2) Es ist nicht klar, ob es überhaupt immer eine Lösung w gibt und ob sie eindeutig ist.

3) Man bezeichnet das L^2-Approxiamtionsproblem auch häufig kurz mit lineares LMS-Problem für *linear least mean square problem*.

3.2 Lösung des LMS-Problems mittels Pseudoinverse (für $|I| < \infty$)

Sei $I = \{1, \dots, N\}$. Setze $a_i = (a_{i1}, \cdots, a_{in})^T \in \mathbb{R}^n$ und $A = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{N1} & \dots & a_{Nn} \end{pmatrix} = (a_1, \dots, a_N)^T$ und $b = (b_1, \dots, b_N)^T$. Dann gilt:

$$\sum_{i=1}^{N} |b_i - a_i^T w|^2 = \|b - Aw\|^2.$$

Gesucht wird also ein w, das das lineare Gleichungssystem $Ax = b$ „möglichst gut“ löst in dem Sinne, daß der Fehler $\|b - Ax\|$ möglichst klein wird. Häufig gibt es viele solcher w; man versucht dann oft unter diesen w eines zu finden, das minimale Norm $\|w\|$ hat.

Sei $\varphi : \mathbb{R}^n \to \mathbb{R}^N$ die lineare Abbildung $\varphi(x) = Ax$, $K = \text{Kern}(\varphi)$ und $B = \text{Bild}(\varphi)$. Weiter sei $K^\perp$ das orthogonale Komplement zu K wie $B^\perp$ das zu B (bezüglich des Euklidischen Skalarprodukts).

Es gilt: $\mathbb{R}^n = K \oplus K^\perp$ und $\mathbb{R}^N = B \oplus B^\perp$. $\Pi_B : \mathbb{R}^N \to B$ sei die orthogonale Projektion. Die Restriktion $\varphi|_{K^\perp} : K^\perp \to B$ ist bijektiv; sei $\psi : B \to K^\perp$ die Umkehrabbildung.

3.2.1 Satz. $w = \psi \circ \Pi_B(b)$ löst das lineare LMS-Problem. Jede andere Lösung hat größere Norm als w.

Beweis. Für $y \in B$ setze $h(y) = \|b - y\|^2 = \langle b - y, b - y \rangle$. Damit h in y ein Minimum hat, muß gelten $Dh(y) = -\langle b - y, y \rangle - \langle b - y, y \rangle = -2 \langle y, b - y \rangle = 0$, also $y \perp b - y$. Somit ist $y = \Pi_B(b)$ der eindeutig bestimmte Punkt aus B, der $\|b - y\|$ minimal macht. Und folglich löst $w = \psi \circ \Pi_B(b)$ das LMS-Problem. Für jede andere Lösung $u \neq w$ muß ebenfalls $\varphi(u) = \Pi_B(b)$ gelten; daher ist $w - u \in K$ und somit $w - u \perp w$ (denn $w \in K^\perp$). Nach Pythagoras gilt $\|u\|^2 = \|u - w + w\|^2 = \|u - w\|^2 + \|w\|^2 > \|w\|^2$. □

3.2.2 Bezeichnung. Die Abbildung $\psi \circ \Pi_B$ heißt die *Pseudoinverse* (*Moore-Penrose-Inverse*) von $\varphi(x) = Ax$; die sie beschreibende Matrix wird oft mit A^+ bezeichnet.

3.2.3 Probleme. Ein Problem, das sich in der Praxis stellt, ist, wie man A^+ berechnet.

Wenn A^TA (oder AA^T) invertierbar ist, gilt $A^+ = (A^TA)^{-1}A^T = A^T(AA^T)^{-1}$ und w ist die Lösung von $A^TAW = A^Tb$, die man mit üblichen Methoden zur Lösung linearer Gleichungssysteme erhält. Im allgemeinen kann man die Pseudoinverse über die Singulärwertzerlegung erhalten. Die Singulärwertzerlegung ist fast genau die Pseudoinverse bezüglich einer besonders geschickt gewählten Basis.

3.3 Die Fehlerfunktion

Sei $I = \{1, \dots, N\}$. Für jede Folge $(c_i)_{i \in I}$ in einem Vektorraum, setzen wir

$$\mathbb{E}\left((c_i)_{i \in i}\right) = \frac{1}{N} \sum_{i \in I} c_i.$$

Die Bezeichnung soll an den Erwartungswert aus der Stochastik erinnern.
Bemerkung. Wir werden oft etwas schlampig $\mathbb{E}(c_i)$ statt $\mathbb{E}((c_i)_{i \in i})$ schreiben. In manchen Anwendungen sind die c_i Realisierungen von Zufallsvariablen; sind sie beispielsweise unabhängig und identisch verteilt, so kann man $\frac{1}{|I|} \sum c_i$ als Schätzer für den wahren Erwartungswert ansehen (Ersetzung des Ensemble-Mittelwerts durch den Zeit-Mittelwert für ergodische Prozesse).

Wir betrachten jetzt das lineare L^2-Approximationsproblem und definieren für $w \in \mathbb{R}^n$ den *Einzelfehler*

$$\varepsilon_i(w) = b_i - a_i^T w$$

und den *Gesamtfehler*

$$e(w) = \mathbb{E}\left((\varepsilon_i(w))^2\right) = \tfrac{1}{N} \sum_{i \in I} (\varepsilon_i(w))^2.$$

$e(w)$ wird auch MSE (= *mean square error*) genannt. Die Funktion e nennt man *Fehlerfunktion*.

Es gilt:

$$\begin{aligned} e(w) &= \mathbb{E}\left((b_i - a_i^T w)\cdot(b_i - a_i^T w)\right) = \mathbb{E}\left(b_i^2 + w^T a_i a_i^T w - 2b_i a_i^T w\right) \\ &= \mathbb{E}\left(b_i^2\right) + w^T \mathbb{E}\left(a_i^2\right) w - 2\mathbb{E}\left(b_i a_i\right)^T w = \mathbb{E}\left(b_i^2\right) + w^T R w - 2P^T w, \end{aligned}$$

wobei

$$\begin{aligned} R &= \mathbb{E}(a_i a_i^T) = \begin{pmatrix} \mathbb{E}(a_{i1}^2) & \cdots & \mathbb{E}(a_{i1}a_{in}) \\ \vdots & & \vdots \\ \mathbb{E}(a_{in}a_{i1}) & \cdots & \mathbb{E}(a_{in}^2) \end{pmatrix} \text{ mit } a_i^T = (a_{i1}, \ldots, a_{in}) \\ \text{und } P &= \mathbb{E}(b_i a_i) = (\mathbb{E}(b_i a_{i1}), \ldots, \mathbb{E}(b_i a_{in}))^T. \end{aligned}$$

R heißt die *Eingangs-Autokorrelations-Matrix* und ist symmetrisch.

P heißt die *Kreuzkorrelation* zwischen $(a_i)_{i\in I}$ und $(b_i)_{i\in I}$.

Der Graph von e wird oft *Fehlerfläche* genannt (obwohl er natürlich nur für $n = 2$ zweidimensional ist).

3.3.1 Einige Eigenschaften der Fehlerfunktion

a) $e(w) \geq 0$ für jedes $w \in \mathbb{R}^n$.

b) e ist ein Polynom vom Grad ≤ 2. Es gilt: $\nabla e(w) = -2\mathbb{E}(\varepsilon_i(w)a_i) = 2(Rw - P)$.

c) e hat ein globales Minimum w_*, das die Gleichung $Rw_* = P$ löst. (*Normalgleichung*) Es gilt:

$$e(w_*) = \mathbb{E}(b_i^2) - P^T w_* =: e_{\min}.$$

d) Ist R invertierbar, so hat e genau ein globales Minimum, und zwar $w_* = R^{-1}P$.

e) Es gibt eine orthogonale Matrix $Q \in O(n)$, so daß $e(w) = e_{\min} + (w - w_*)^T Q^T \Lambda Q(w - w_*)$, wobei w_* ein globales Minimum von e und $\Lambda \in \mathbb{R}^{n\times n}$ eine Diagonalmatrix sind, die auf der Hauptdiagonale genau die Eigenwerte $\lambda_1, \ldots, \lambda_n$ von R stehen hat.

Man kann also durch $v = Q(w - w_*)$ zu einem neuen Koordinatensystem übergehen, in dem e die Gestalt

$$v \mapsto e_{\min} + \sum_{i=1}^{n} \lambda_i v_i^2$$

hat. Die λ_i sind die Eigenwerte von R; sie sind alle nicht negativ.

f) Über jeder durch w_* gehenden Gerade im $\mathbb{R}^n$ hat der Graph von e die Gestalt einer nach oben geöffneten Parabel mit Scheitel in $(w_*, e_{\min})$, die zu einer horizontalen Geraden ausarten kann, wenn nicht alle Eigenwerte von R positiv sind.

Beweis der Eigenschaften.

a) nach Definition von e.

b) nach Definition von e.

c) Für jedes $v \in \mathbb{R}^n$, $v \neq 0$, ist $t \mapsto e(tv)$ ein Polynom vom Grad ≤ 2. Es kann nicht Grad 1 haben, da $e \geq 0$. Damit e in w_* ein lokales Minimum hat, muß gelten $0 = \nabla e(w_*) = 2(Rw_* - P)$.

Angenommen, diese Gleichung sei nicht lösbar. Dann ist $P \neq 0$ und $P \notin \text{Bild}(R)$. Es gibt daher ein $v \in \mathbb{R}^n$, $v \neq 0$, mit $v^T R = 0$ und $v^T P \neq 0$, und es folgt, daß $\frac{d}{dt}e(tv) = v^T \nabla e(vt) = -2v^T P$ konstant, aber nicht 0 ist. $t \mapsto e(tv)$ ist also linear. Das ist ein Widerspruch zu obiger Überlegung. Folglich gibt es ein w_* mit $0 = \nabla e(w_*) = 2(Rw_* - P)$. w_* muß ein globales Minimum von e sein, weil $t \mapsto e(w_* + tv)$ für

jedes $v \in \mathbb{R}^n$ ein Polynom vom Grad ≤ 2 ist, das nicht negativ ist, also nur konstant oder quadratisch sein kann.

Wegen $Rw^* = P$ folgt

$$e(w_*) = \mathbb{E}(b_i^2) + (w_*)^T R w_* - 2P^T w_* = \mathbb{E}(b_i^2) + (w_*)^T P - 2P^T w_* = \mathbb{E}(b_i^2) - P^T w_*.$$

d) Ist R invertierbar, so gibt es genau eine Lösung von $Rw_* = P$, nämlich $w_* = R^{-1}P$.

e) Wegen $R = R^T$, $w_*^T R w = w^T R w_*$ und $P^T w_* = w_*^T P$ erhält man durch bloßes Nachrechnen

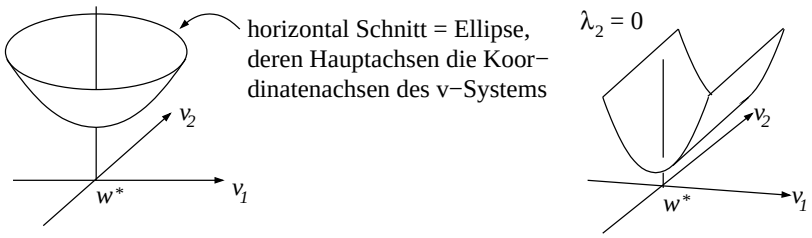
$$e_{\min} + (w - w_*)^T R (w - w_*) = e(w).$$

Da R symmetrisch ist, gibt eine orthogonale Matrix Q, so daß $R = Q^T \Lambda Q$. Weil $e \geq 0$ ist, sind alle Diagonalelemente (die Eigenwerte von R) nicht negativ.

f) folgt unmittelbar aus e).

□

Beispiele. $n = 2$



3.3.2 Dekorrelation von Fehler und Daten

Ist w_* ein Minimum von e, so gilt $\mathbb{E}(\varepsilon_i(w_*)a_i) = \frac{1}{N}\sum_{i\in I} \varepsilon(w_*)a_i = 0$, d.h. der Fehlervektor $(\varepsilon_i(w_*))_{i\in I}$ und $(a_i)_{i\in I}$ stehen „senkrecht aufeinander“ oder — stochastisch ausgedrückt — sind nicht korreliert.

Beweis. $\mathbb{E}(\varepsilon_i(w_*)a_i) = \mathbb{E}(b_i a_i) - \mathbb{E}(a_i(a_i^T w_*)) = P - Rw_* = 0$, nach Eigenschaft 3.3.1.c). □

3.4 Minimumsuche mit Gradientenabstieg

Meist ist in Anwendungen die Autokorrelationsmatrix R invertierbar. Dann hat e gemäß 3.3.1 genau ein Minimum w_*, und w_* löst das lineare Gleichungssystem $Rw_* = P$. Um w_* zu berechnen, wird man nur selten R^{-1} bestimmen, sondern übliche numerische Verfahren zur Lösung linearer Gleichungssysteme verwenden.

Wenn nun zu den bisherigen Daten $(a_i)_{i\in I}$ und $(b_i)_{i\in I}$ neue hinzukommen, also die Folgen verlängert werden zu $(\tilde{a}_i)_{i\in\tilde{I}}$ und $(\tilde{b}_i)_{i\in\tilde{I}}$, dann erhält man eine neue Fehlerfunktion $\tilde{e}(w) = \mathbb{E}(\tilde{b}_i^2) + w^T \tilde{R} w - 2\tilde{P}^T w$ mit $\tilde{R} = \mathbb{E}(\tilde{a}_i \tilde{a}_i^T)$ und $\tilde{P} = \mathbb{E}(\tilde{b}_i \tilde{a}_i)$, wobei sich der Erwartungswertoperator $\mathbb{E}$ jetzt über $\tilde{I}$ erstreckt. Um das Minimum $\tilde{w}_*$ von $\tilde{e}$ zu bestimmen, muß das lineare Gleichungssystem $\tilde{R}w = \tilde{P}$ gelöst werden. Dieses System hat wie das bisherige n Gleichungen. Wenn n groß ist und oft neue Daten hinzukommen (was in manchen Anwendungen der Fall ist), kann es zu zeitaufwendig sein, ständig diese Normalgleichung lösen zu müssen. Als Ausweg bietet sich an, von dem Minimum w_* der alten Fehlerfunktion auszugehen und das der neuen Fehlerfunktion $\tilde{e}$ zu suchen, indem man sich immer in Richtung des negativen Gradienten von $\tilde{e}$ (= Richtung des stärksten Gefälles) bewegt.

Wir wollen zeigen, daß man auf diese Weise bei jeder Fehlerfunktion obiger Gestalt ein Minimum findet, egal von welchem Startpunkt man ausgeht.

3.4.1 Satz. Jede Lösung des Differentialgleichungssystem $w' = -\nabla e(w)$ ist auf ganz $\mathbb{R}$ definiert und konvergiert für $t \to \infty$ gegen ein globales Minimum von e.

Beweis. Nach 3.3.1 hat e ein globales Minimum w_* und durch Übergang zu den Koordinaten $v = Q(w - w_*)$ erhält e die Gestalt $f(v) = e_{\min} + \sum_i \lambda_i v_i^2$, wobei $\lambda_i \geq 0$ die Eigenwerte von R sind. w_* geht dabei in 0 über, und es gilt

$$\nabla e(w) = 2(Rw - P) = 2(RQ^{-1}v + Rw_* - P) = 2RQ^{-1}v.$$

Ist $t \mapsto w(t)$ eine Lösung von $w' = -\nabla e(w)$ und ist $v = Q(w - w_*)$, so gilt

$$v' = Qw' = Q(-\nabla e(w)) = -2QRQ^{-1}v = -2\Lambda v$$

mit $\Lambda = \mathrm{Diag}(\lambda_1, \ldots, \lambda_n)$ wie in 3.3.1.e), d.h. v ist eine Lösung von $v' = -\nabla f(v) = -2\Lambda v$.

Umgekehrt ist für jede Lösung v von $v' = -2\Lambda v$ die Kurve $w = Q^{-1}v + w_*$ eine Lösung von $w' = -\nabla e(w)$. Wir brauchen also nur die Lösungen des Systems $v' = -2\Lambda v$ zu untersuchen. Dieses System ist sehr einfach, weil Λ eine Diagonalmatrix ist.

$$\begin{pmatrix} v_1' \\ \vdots \\ v_n' \end{pmatrix} = - \begin{pmatrix} 2\lambda_1 & & 0 \\ & \ddots & \\ 0 & & 2\lambda_n \end{pmatrix} \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix} = \begin{pmatrix} -2\lambda_1 v_1 \\ \vdots \\ -2\lambda_n v_n \end{pmatrix}$$

Die Differentialgleichungen sind also entkoppelt und haben die Lösungen $v_j(t) = \alpha_j \exp(-2\lambda_j t)$ mit $\alpha_j \in \mathbb{R}$.

Für jedes j mit $\lambda_j > 0$, gilt $\lim_{t\to\infty} v_j(t) = 0$; für $\lambda_j = 0$ gilt $v_j(t) = \alpha_j$ für jedes $t \in \mathbb{R}$. Jeder Punkt $(u_1, \ldots, u_n)^T \in \mathbb{R}^n$ mit $u_j = 0$ für $\lambda_j > 0$ ist aber ein Minimum von f. □

Einer Trajektorie des Gradientfeldes bis ins Minimum folgen kann man vielleicht mit Analogrechnern (auch nicht ganz, weil man es in endlicher Zeit nicht erreicht); mit Digitalrechnern dagegen kann man nur in diskreten Sprüngen vorwärtsgehen, etwa in folgender Weise.

Sei $w_0 \in \mathbb{R}^n$ beliebig. Setze $w_{k+1} = w_k - \alpha\nabla e(w_k) = (I - 2\alpha R)w_k + 2\alpha P$. Konvergiert diese Folge? Wenn ja, gegen was?

3.4.2 Satz (Diskreter Gradientenabstieg)**.**
Seien $\alpha > 0$, $\alpha\lambda_{\max} < 1$, wobei $\lambda_{\max}$ der größte Eigenwert von R ist. Sei $w_1 \in \mathbb{R}^n$. Dann konvergiert die durch $w_{k+1} = w_k - \alpha\nabla e(w_k) = (I - 2\alpha R)w_k + 2\alpha P$ definierte Folge gegen ein globales Minimum von e.

Beweis. Sei w_* ein Minimum von e. Wir gehen wieder win in 3.3.1 zu den Koordinaten $v = Q(w - w_*)$ über, in denen e die Gestalt $f(v) = e_{\min} + \sum_i \lambda_i v_i^2$ hat und die λ_i die Eigenwerte von R sind. Setze $v_k = Q(w_k - w_*)$, also $w_k = Q^{-1}v_k + w_*$. Dann gilt

$$Q^{-1}v_{k+1} + w_* = w_{k+1} = w_k - \alpha\nabla e(w_k) = Q^{-1}v_k + w_* - 2\alpha(RQ^{-1}v_k + Rw_* - Rw_*),$$

folglich ist $v_{k+1} = (I - 2\alpha QRQ^{-1})v_k = (I - 2\alpha\Lambda)v_k$. Mit Induktion ergibt sich $v_k = (I - 2\alpha\Lambda)^{k-1}v_1$ für $k \geq 1$. Weil Λ Diagonalgestalt hat, hat auch $(I - 2\alpha\Lambda)$ Diagonalgestalt, und es folgt

$$v_k = \begin{pmatrix} (1 - 2\alpha\lambda_1)^{k-1} & & 0 \\ & \ddots & \\ 0 & & (1 - 2\alpha\lambda_n)^{k-1} \end{pmatrix}.$$

Die j-te Komponente konvergiert genau dann, wenn entweder $\lambda_j = 0$ und $\alpha > 0$ beliebig oder $\lambda_j > 0$ und $|1 - 2\alpha\lambda_j| < 1$ ist. Ist $\lambda_j > 0$, so gilt stets $1 - 2\alpha\lambda_j < 1$ wegen $\alpha > 0$, und $1 - 2\alpha\lambda_j > -1$ ist äquivalent zu $\alpha\lambda_j < 1$. Gilt $\alpha\lambda_{\max} < 1$ für den größten Eigenwert $\lambda_{\max}$ von R, so gilt natürlich $\alpha\lambda_i < 1$ für alle anderen Eigenwerte λ_i auch.

Im Falle der Konvergenz gilt $v^* := \lim_{k\to\infty} v_k$ mit $v_j^* = 0$, falls $\lambda_j > 0$, und v_{1j}, sonst, und f hat in v^* offensichtlich ein globales Minimum. □

3.4.3 Bemerkung.

1) Ist $\lambda_j > 0$, so konvergiert die j-te Komponente umso schneller, je kleiner $|1 - 2\alpha\lambda_j|$ ist, je näher also $\alpha\lambda_j$ bei $\frac{1}{2}$ liegt. Sind der größte positive Eigenwert $\lambda_{\max}$ von R und der kleinste positive $\lambda_{\min}$ verschieden, so kann man $|1 - 2\alpha\lambda_j|$ nicht für alle positive Eigenwerte λ_j durch eine geeignete Wahl von α beliebig klein machen. Die erzielbare Konvergenzgeschwindigkeit hängt im wesentlichen von dem Verhältnis $\frac{\lambda_{\max}}{\lambda_{\min}}$ ab.

2) Diese Beobachtung legt nahe, für jede Komponente eine andere Schrittweite α_j zu wählen. Zurückübersetzt in die ursprünglichen Koordinaten w bedeutet dies, daß man als Rekursionsgleichung

$$w_{k+1} = w_k - A\nabla e(w_k)$$

mit einer $n \times n$-Matrix A wählt, also die Richtung des Abstiegs nicht mehr immer die Gradientenrichtung ist. Um die unterschiedlich großen Eigenwerte zu kompensieren, wählt man einfach $A = \frac{1}{2}R^{-1}$ (bzw. in den v-Koordinaten $A = \frac{1}{2}\Lambda^{-1}$); das ist nichts anderes als das aus der Numerik bekannte Newton-Verfahren, denn R ist die Hessematrix $D^2e(w)$ (die von w unabhängig ist, da e quadratisch ist). Man beachte, daß mit $A = \frac{1}{2}R^{-1}$ gilt $w_2 = w_1 - R^{-1}(Rw_1 - Rw_*) = w_*$ also das Newton-Verfahren in einem Schritt das Minimum findet. (So ist es ja gerade konstruiert.)

3) Der Graph der Funktion $k \mapsto e(w_k)$ heißt *Lernkurve*.

4) Die Abschätzung $\alpha\lambda_{\max} < 1$ ist oft schwer zu verifizieren, weil man die Eigenwerte von R nicht kennt, und weil die Eigenwerte größerer Matrizen auch nicht leicht auszurechnen sind. Eine grobe Abschätzung ist jedoch leicht möglich. Es gilt nämlich $\lambda_{\max} \leq \sum_{i=1}^{n} \lambda_i = \text{Spur}(\Lambda) = \text{Spur}(R)$; also ist $\alpha \leq \frac{1}{\text{Spur}(R)}$ hinreichend für Konvergenz.

3.5 Unendliche Datenfolge

In Anwendungen erhält man häufig zu jedem Zeitpunkt $k \in \mathbb{N}$ ein neues Datenpaar $(a_k, b_k) \in \mathbb{R}^n \times \mathbb{R}$. Idealerweise sucht man dann für jedes $k \in \mathbb{N}$ ein $w_k^* \in \mathbb{R}^n$, so dass $e_k(w) := \sum_{i=1}^{k}(b_i - a_i^T w)^2$ in w_k^* ein Minimum hat.

Manchmal berücksichtigt man auch nur ein endliches Zeitfenster der Länge $T > 0$ und sucht also für jedes $k > t$ ein Minimum der modifizierten Fehlerfunktion $e_k(w) := \sum_{i=k-T+1}^{k}(b_i - a_i^T w)^2$.

Löst man das Problem exakt mit Hilfe der Normalgleichung, muss man für jedes k die entsprechenden Matrizen R und P und die Lösung der Normalgleichung berechnen. Verwendet man Gradientenabstieg, so erspart man sich das Lösen der Normalgleichung, muss aber etliche Iterationsschritte durchführen, um dem Minimum nahe zu kommen.

Daher hat Widrow eine radikale Vereinfachung vorgeschlagen:
Man führt für jedes k nur einen einzigen Gradientenabstiegsschritt aus, wobei man von dem zum Zeitpunkt $k-1$ gefundenen Punkt w_{k-1} ausgeht. Überdies wählt man noch eine sehr grobe Näherung für den wirklichen Gradienten, und zwar ersetzt man den Gradienten $\nabla e_k(w) = -2\sum_{i=1}^{k}(b_i - a_i^T w) \cdot a_i$ durch nur den einen Summanden $-2(b_k - a_k^T w) \cdot a_k$.

3.5.1 Der LMS-Algorithmus

Gegeben seien eine Folge $(a_j)_{j\in\mathbb{N}}$ im $\mathbb{R}^n$ und eine Folge $(b_j)_{j\in\mathbb{N}}$ in $\mathbb{R}$. Seien $\alpha > 0$ und $w_0 \in \mathbb{R}^n$. Für $k \in \mathbb{N}$ setze

$$w_{k+1} \;=\; w_k + 2\alpha(b_k - w_k^T a_k) \cdot a_k$$

3.5.1 Bemerkung.

1) Die Folge der Paare $((a_j, b_j))_{j\in\mathbb{N}}$ heißt Trainingsfolge.

2) Die Folge $(w_k)_k$ muss nicht konvergieren. Falls doch, muss sie nicht gegen eine Minimumstelle der Fehlerfunktion konvergieren.

3) Trotzdem kann man in praktischen Anwendungen manchmal die Werte der Fehlerfunktion ziemlich klein bekommen. Den Parameter α darf man nicht zu groß wählen, da sonst die Folge $(w_k)_k$ oft stark oszilliert. Man kann ihre Konvergenz erzwingen, indem man α von k abhängig macht und mit wachsendem k gegen Null gehen lässt. Dadurch wird aber nicht garantiert, dass w_k in die Nähe einer Minimumstelle der Fehlerfunktion kommt. Man kann Oszillationen dämpfen, indem man die Rekursionsgleichung mit Hilfe eines gleitenden Mittels über zurückliegende Fehlerterme modifiziert:

$$w_{k+1} \;=\; w_k + 2\alpha \sum_{l=0}^{L} \beta_l (b_{k-l} - w_{k-l}^T a_{k-l}) \cdot a_{k-l}$$

wobei die Koeffizienten $\beta_l > 0$ sind und sich zu 1 summieren.

4) Der LMS-Algorithmus lässt sich auf das LMS-Problem mit endlicher Indexmenge I anwenden. Dazu wählt man als Trainingsfolge eine Folge, in der jedes Element von $\{(a_i, b_i) : i \in I\}$ unendlich oft vorkommt, z.B. indem man die Elemente zyklisch wiederholt.

5) In praktischen Anwendungen ist die Trainingsfolge oft eine Folge von Ergebnissen (a_i, b_i) unabhängiger Messungen. Man modelliert sie als unabhängige Realisierungen von zwei Zufallsvariablen (X, Y). Dies wird im nächsten Abschnitt präzisiert.

3.6 Das lineare L^2-Approximationsproblem II

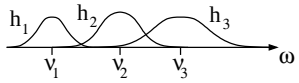
(Ω, p) sei ein Wahrscheinlichkeitsraum. $X : \Omega \to \mathbb{R}^n$ und $Y : \Omega \to \mathbb{R}$ seien Zufallsvariablen, d.h. meßbare Abbildungen.

Gesucht ist ein $w \in \mathbb{R}^n$, so daß $\mathcal{E}(|Y - w^T X|^2) = \int_\Omega (Y(\omega) - w^T X(\omega))^2 \, dp(\omega)$ minimal ist.

3.6.1 Bemerkung. Das Problem II umfaßt das Problem I in 3.1, wenn die Indexmenge I endlich ist. Sei nämlich $\Omega = \{1, \ldots, N\} = I$, $p(i) = \frac{1}{N}$, $X(i) = a_i$, $Y(i) = b_i$. Dann ist $\mathcal{E}(|Y - w^T X|^2) = \frac{1}{N} \sum_{i=1}^N |b_i - w^T a_i|^2$.

3.6.2 Beispiel. Messung einer skalaren Größe mit *fuzzy* Sensoren (*radial basis functions*)

Gemessen werden soll eine reelle Größe $\omega \in [0, M]$ (z.B. die Frequenz eines Sinustons). Dafür stehen n Sensoren zur Verfügung, deren Ausgangssignale jeweils eine glockenförmige, nicht negative Funktion h_j von ω sind. (Die Sensoren könnten z.B. Bandpaßfilter mit nachgeschlatetem Amplitudenmesser sein.)



Die Größe ω werde durch ein Zufallsexperiment bestimmt; es gebe also auf $\Omega = [0, M]$ ein Wahrscheinlichkeitsmaß p. Wir würden gerne $Y(\omega)$ wissen; uns steht aber nur die Messung $X(\omega) = (h_1(\omega), \ldots, h_n(\omega))$ der n Sensoren zur Verfügung. Deshalb möchten wir Y durch eine Linearkombination der h_j im stochastischen Mittel bestmöglich schätzen; wir suchen also ein $w \in \mathbb{R}^n$, so daß $\mathcal{E}(|Y - w^T X|^2)$ möglichst klein wird.

Um auch affin lineare Kombinationen zuzulassen, kann man $X(\omega) = (h_0(\omega), h_1(\omega), \ldots, h_n(\omega))$ mit $h_0(\omega) \equiv 1$ wählen.

3.6.3 Lösungsansätze.

1. Wenn man zusätzlich $\mathcal{E}(\|X\|^2) < \infty$ und $\mathcal{E}(Y^2) < \infty$ fordert, kann man wie in 3.2 eine Lösung mit Hilfe der Pseudoinversen von $\varphi : \mathbb{R}^n \to L^2(\Omega, p)$, $w \mapsto w^T X$ finden.

2. Für die Fehlerfunktion $e(w) = \mathcal{E}(|Y - w^T X|^2)$ gilt in völliger Analogie zu 3.3

$$e(w) = \mathcal{E}(Y^2) + w^T \mathcal{E}(XX^T) w - 2\mathcal{E}(YX)^T w = \mathcal{E}(Y^2) + w^T R w - 2P^T w,$$

wobei $R = \mathcal{E}(XX^T)$ die Autokorrelationsmatrix von X und $P = \mathcal{E}(YX)$ die Kreuzkorrelation von X und Y ist.

Alle Resultate aus 3.3 lassen sich wörtlich übertragen, ebenso die Suche nach dem Minimum durch Gradientenabstieg in 3.4.

3.7 Schätzung der stochastischen Größen und der LMS-Algorithmus

In der Praxis kennt man die Verteilungen von X und Y meist nicht und muß $R = \mathcal{E}(XX^T)$ und $P = \mathcal{E}(YX)$ und ∇e schätzen. Dazu nimmt man das Gesetz der großen Zahlen zu Hilfe, um aus unabhängigen Stichproben eine Schätzung zu erhalten. Das mathematische Modell dafür ist folgendes.

Als zugrundeliegenden Wahrscheinlichkeitsraum nimmt man $\Omega^{\mathbb{N}}$ mit dem Produktmaß $p^{\mathbb{N}} = \otimes_{i=1}^{\infty} p$ und den Zufallsvariablen $X_i : \Omega^{\mathbb{N}} \to \mathbb{R}^n$, $\omega \mapsto X(\omega_i)$ und $Y_i : \Omega^{\mathbb{N}} \to \mathbb{R}$, $\omega \mapsto Y(\omega_i)$ für $i \in \mathbb{N}$.

Interpretation: X_i und Y_i beschreiben die i-te Stichprobe.

Setze

$$R_N(\omega) = \frac{1}{N} \sum_{i=1}^{N} X_i(\omega) X_i^T(\omega) \qquad \text{und} \qquad P_N(\omega) = \frac{1}{N} \sum_{i=1}^{N} Y_i(\omega) X_i(\omega).$$

Nach dem starken Gesetz der großen Zahlen (Satz von Kolmogorov) gilt

$$\lim_{N\to\infty} R_N = R \qquad \text{und} \qquad \lim_{N\to\infty} P_N = P \qquad p^{\mathbb{N}}\text{-fast überall,}$$

falls $R = \mathcal{E}(XX^T)$ und $P = \mathcal{E}(YX)$ existieren.

Dies liefert die Rechtfertigung, R_N und P_N für große N als Schätzer für R und P zu verwenden.

3.7.1 Schätzung von R und P

Man nehme N unabhängige Stichproben $a_1, \ldots, a_N$ von X und $b_1, \ldots, b_N$ von Y und verwende

$$\hat{R} = \frac{1}{N} \sum_{i=1}^{N} a_i a_i^T \qquad \text{und} \qquad \hat{P} = \frac{1}{N} \sum_{i=1}^{N} b_i a_i$$

als Schätzwert für R bzw. P.

Man erhält somit also ein lineares L^2-Approximationsproblem I, nämlich

$$\hat{e}(w) = \mathbb{E}\left((b_i - a_i^T w)^2\right) = \mathbb{E}(b_i^2) + w^T \hat{R} w - 2\hat{P}^T w$$

zu minimieren. Dies kann wie in 3.3 durch Lösen der Normalgleichung $\hat{R}w = \hat{P}$ tun oder wie in 3.4 durch Gradientenabstieg. Die Suche nach dem Minimum von $\hat{e}$ mit Gradientenabstieg kann man auch als modifizierten Gradientenabstieg für e verstehen, bei dem der wahre Gradient $\nabla e(w)$ durch den Schätzwert $\nabla \hat{e}(w) = 2(\hat{R}w - \hat{P})$ ersetzt wird.

3.7.2 Der Fall unendlich vieler Stichproben

In manchen Anwendungen erhält man ständig neue Stichproben für X und Y, oder mit anderen Worten es kommen neue Paare (a_i, b_i) hinzu, die berücksichtigt werden sollen (z.B. wenn ständig neue experimentelle Messungen gemacht werden). Nach den bisherigen Überlegungen müßte man jedesmal bei Hinzunahme eines neuen Datums (a_{N+1}, b_{N+1}) die Größen $\hat{R}$ und $\hat{P}$ neu berechnen und wieder einen Gradientenabstieg starten. Weil man erwartet, daß sich die bisherige Fehlerfunktion durch Hinzunahme eines nicht total verändern wird, wird man das (approximative) Minimum der alten Fehlerfunktion als Startwert nehmen.

Kommen die neuen Daten sehr schnell hintereinander, so reicht eventuell die Zeitpanne nicht aus, $\hat{R}$ und $\hat{P}$ neu zu berechnen und einen Gradientenabstieg durchzuführen. Weil man aber sowieso schon in der Nähe des Minimums ist, versucht man mit einer gröberen Schätzung des Gradienten und nur einem Schritt in dieser Richtung auszukommen. Die gröbere Schätzung erhält man, indem man von der Gleichung $\nabla \hat{e}(w_k) = -2\mathbb{E}(\varepsilon_i(w_k) a_i)$ ausgeht und einfach den Operator $\mathbb{E}$ wegläßt, also $\nabla e(w_k)$ durch $-2\varepsilon_k(w_k) a_k$ schätzt.

3.7.3 Der LMS-Algorithmus von Widrow

Gegeben sei eine Folge $(a_i)_{i\in\mathbb{N}}$ im $\mathbb{R}^n$ und eine Folge $(b_i)_{i\in\mathbb{N}}$ in $\mathbb{R}$. Sei $\alpha > 0$. $w_1 \in \mathbb{R}^n$ sei beliebig. Für $t \in \mathbb{N}$ setze

$$w_{t+1} = w_t + 2\alpha(b_t - w_t^T a_t)a_t = w_t + 2\alpha\varepsilon_t(w_t)a_t.$$

Bezeichnung. Die im LMS-Algorithmus verwendete Tupelfolge $((a_i, b_i))_{i\in\mathbb{N}}$ heißt *Trainigsfolge*.
Bemerkung. Die im LMS-Algorithmus konstruierte Folge $(w_t)_{t\in\mathbb{N}}$ muß nicht konvergieren; und wenn sie gegen ein w_* konvergiert und a_i bzw. b_i unabhängige Stichproben von X und Y sind, dann muß $\mathcal{E}(|Y - w_*^T X|^2)$ nicht minimal sein. Trotzdem erzielt man in vielen Anwendungen mit dem LMS-Algorithmus gute Resultate, wenn α geeignet gewählt ist (d.h. meistens recht klein). Anfangs irrt die Folge $(w_t)_{t\in\mathbb{N}}$ zwar meist stark hin und her; befindet man sich aber einmal in der Nähe des Minimums, so werden die Oszillationen kleiner (wenn α nicht zu groß ist). Diese Situation ist aber gerade in vielen Anwendungen interessant, in denen sich die Verteilungen von X und Y langsam mit der Zeit ändern können. Nimmt man ständig neue Stichproben (a_i, b_i) hinzu und verändert man die Gewichte gemäß des LMS-Algorithmus, so hat man die Chance, daß der Gewichtsvektor stets in der Nähe des wirklichen Minimums bleibt.

Zur adäquaten Beschreibung verwendet man statt zweier Zufallsvariablen X und Y zwei Folgen $(X_t)_{t\in\mathbb{N}}$ und $(Y_t)_{t\in\mathbb{N}}$ von Zufallsvariablen, die nicht identisch verteilt sein müssen.

3.7.4 Modifikationen

1. Um das starke Oszillieren von $(w_t)_{t\in\mathbb{N}}$ zu dämpfen, wird oft in der Rekursionsgleichung der Term $\varepsilon_t(w_t)a_t$ durch einen Mittelwert über entsprechende Terme mit kleinerem Index ersetzt, also

$$w_{t+1} = w_t + 2\alpha \sum_{l=0}^{L-1} \gamma_l \varepsilon_{t-l}(w_{t-l})a_{t-l} \quad \text{mit } \gamma_l \geq 0.$$

   Arithmetisches Mittel: $\gamma_l = \frac{1}{L}$, $w_{t+1} = w_t + 2\alpha g_t$, $g_{t+1} = g_t - \frac{1}{L}\varepsilon_{t-L+2}(w_{t-L+2})a_{t-L+2} + \frac{1}{L}\varepsilon_{t+1}(w_{t+1})a_{t+1}$.
   Exponentielles Mittel: $\gamma_l = \frac{1}{2^l}$, $w_{t+1} = w_t + 2\alpha g_t$, $g_{t+1} = \frac{1}{2}(g_t + \varepsilon_{t+1}(w_{t+1})a_{t+1})$.

2. Auch auf das L^2-Approximationsproblem I wird der LMS-Algorithmus angwendet, indem man aus den gegebenen endlich vielen Daten$(a_i)_i$ und $(b_i)_i$ auf irgendeine Weise unendliche Folgen macht (z.B. indem man sie zyklisch wiederholt), die man als Trainingsfolgen verwenden kann.

3.8 Konvergenz der Erwartungswerte der Gewichtsvektoren

(Ω, p) sei ein Wahrscheinlichkeitsraum, $(X_t)_{t\in\mathbb{N}}$ sei eine Folge von Zufallsvariablen $X_t : \Omega \to \mathbb{R}^n$ und $(Y_t)_{t\in\mathbb{N}}$ eine Folge von Zufallsvariablen $Y_t : \Omega \to \mathbb{R}$. Sie seien stationär in dem Sinne, daß die Erwartungswerte $\mathcal{E}(X_t X_t^T)$ und $\mathcal{E}(Y_t X_t)$ nicht von t abhängen. Sei $w_1 \in \mathbb{R}^n$ beliebig.

Durch $w_{t+1} = w_t + 2\alpha(Y_t - w_t^T X_t)X_t$ werden für $t \in \mathbb{N}$ Zufallsvariablen auf (Ω, p) definiert.

$(*)$ Es werde vorausgesetzt, daß die Komponenten von $X_t X_t^T$ und w_t unkorreliert seien.

Ist $\alpha\lambda_{\max} < 1$, wobei $\lambda_{\max}$ der größte Eigenwert von $R = \mathcal{E}(X_t X_t^T)$ (unabhängig von t) ist, so konvergiert die Folge $(\mathcal{E}(w_t))_t$ gegen ein globales Minimum von $e(w) = \mathcal{E}(|Y_t - w^T X_t|^2)$ (auch unabhängig von t).

Beweis. Sei $P = \mathcal{E}(Y_t X_t)$ (P ist unabhängig von t). Wendet man $\mathcal{E}$ auf die Rekursionsgleichung an, so erhält man

$$\begin{aligned}
\mathcal{E}(w_{t+1}) &= \mathcal{E}(w_t) + 2\alpha\mathcal{E}(Y_t X_t) - 2\alpha\mathcal{E}(X_t X_t^T w_t) \\
&= \mathcal{E}(w_t) + 2\alpha\mathcal{E}(Y_t X_t) - 2\alpha\mathcal{E}(X_t X_t^T)\mathcal{E}(w_t) \quad \text{wegen } (*) \\
&= (I - 2\alpha R)\mathcal{E}(w_t) + 2\alpha P
\end{aligned}$$

Das ist dieselbe Rekursionsgleichung wie im diskreten Gradientenabstieg in 3.4.2. Deshalb folgt wie dort, daß $(\mathcal{E}(w_t))_t$ gegen ein globales Minimum von e konvergiert, wenn $\alpha\lambda_{\max} < 1$. □

3.8.1 Bemerkungen.

1. Etwas problematisch ist die Voraussetzung $(*)$, die in der Praxis kaum zu verifizieren ist und wahrscheinlich auch nicht immer exakt erfüllt ist. Sind die X_t unabhängig, so ist $(*)$ erfüllt.

2. Die Abweichung $w_t - \mathcal{E}(w_t)$ ist wieder eine Zufallsvariable, deren Eigenschaft man untersucht. Ebenso gibt es über den sogenannten *excess mean square error* $= e(w_t) - e(\mathcal{E}(w_t))$ Untersuchungen.

3.9 Einige stochastische Begriffe

3.9.1 Definition. Ein *diskreter stochastischer Prozeß* mit Werten im $\mathbb{R}^n$ ist eine Folge $(X_t)_{t\in\mathbb{T}}$ ($\mathbb{T} \in \{\mathbb{N}, \mathbb{Z}\}$) von Zufallsvariablen $X_t : \Omega \to \mathbb{R}^n$, die auf einem Wahrscheinlichkeitsraum (Ω, p) definiert sind.

Für jedes $\omega \in \Omega$ nennen wir die Abbildung $\mathbb{T} \to \mathbb{R}^n, t \to X_t(\omega)$ einen *Pfad* oder eine *Realisierung* des Prozesses.

Ein stochastischer Prozeß (X_t) heißt *stationär im strikten Sinn*, wenn für jedes $l \in \mathbb{N}$ und jedes $(t_1, \ldots, t_l) \in \mathbb{T}^l$ die gemeinsame Verteilung von $X_{t_1+t}, \ldots, X_{t_2+t}$ nicht von $t \in \mathbb{T}$ abhängt.

Er heißt *stationär im weiteren Sinn*, wenn $\mathcal{E}(\|X_t\|^2) < \infty$ und unabhängig von $t \in \mathbb{T}$ ist, und wenn für jedes $s \in \mathbb{N}$ die Autokorrelation $\mathcal{E}(X_t X_{t+s}^T)$ nicht von $t \in \mathbb{T}$ abhängt.

Zwei stochastische Prozesse $(X_t)_{t\in\mathbb{T}}$ mit Werten in $\mathbb{R}^n$ und $(Y_t)_{t\in\mathbb{T}}$ mit Werten in $\mathbb{R}$ heißen *gemeinsam stationär im weiteren Sinn*, wenn sie beide stationär im weiteren Sinn sind und wenn die Kreuzkorrealtion $\mathcal{E}(Y_t X_{t+s})$ nur von $s \in \mathbb{T}$ und nicht von $t \in \mathbb{T}$ abhängt.

Bei einem stochastischen Prozeß $(X_t)_{t\in\mathbb{T}}$ unterscheide man

die Ensemble-Mittelwerte $\quad \mathcal{E}(X_t) = \int_\Omega X_t(\omega)\, dp(\omega) \quad$ für jedes $t \in \mathbb{N}$

und die zeitlichen Mittelwerte $\quad \mathbb{E}\left((X_t(\omega))_{t\in\mathbb{N}}\right) = \lim_{N\to\infty} \frac{1}{N} \sum_{t=1}^{N} X_t(\omega) \quad$ für jedes $\omega \in \Omega$.

Bemerkung. Weder die Ensemble- noch die zeitlichen Mittelwerte müssen existieren.

3.9.2 Definition. Ein stochastischer Prozeß $(X_t)_{t\in\mathbb{N}}$ heißt *ergodisch*, wenn er folgende Eigenschaften hat

1. Alle Erwartungswerte $\mathcal{E}(X_t)$ existieren und haben denselben Wert η.

2. $\mathbb{E}\left((X_t(\omega))_{t\in\mathbb{N}}\right) = \eta$ für p-fast alle $\omega \in \Omega$.

3.9.3 Satz (Ergodensatz)**.**
Jeder Prozeß $(X_t)_{t\in\mathbb{N}}$ aus unabhängigen, integrierbaren, identisch verteilten Zufallsvariablen ist ergodisch.

Beweis. Siehe dazu beispielsweise Doob, Bauer. □

3.9.4 Bemerkung (zu Generatoren von Zufallsszahlen)**.**
Beim Training von NN braucht man oft Realisierungen von stochastischen Prozessen. Für Testläufe ist es oft zu mühsam, tatsächlich physikalische stochastische Prozesse zu verwenden. Man greift auf simulierte Realisierungen zurück, sogenannte *Zufallszahlenfolgen.*

Dieser Begriff ist mit Vorsicht zu genießen: Betrachten wir zum Beispiel die Folgen der Augenzahlen, die bei unabhängigem Würfeln entstehen können. Jede Folge $(r_t)_{t\in\mathbb{N}} \in \{1, \ldots, 6\}^{\mathbb{N}}$ ist gleichwahrscheinlich. Trotzdem werden wir die konstante Folge $(6)_{t\in\mathbb{N}}$ nicht als zufällig im Sinne von repräsentativ für das Würfelexperiment ansehen. Wir werden eher eine Folge als typisch werten, wenn die Häufigkeit

$$\lim_{N\to\infty} \tfrac{1}{N} \#\{j \in \mathbb{N} \mid j \le N, r_j = z\} = \tfrac{1}{6} \qquad \text{für jedes } z \in \{1, \ldots, 6\}.$$

Dies reicht aber sicher noch nicht. Denn die zyklische Wiederholung von $1, \ldots, 6$ werden wir noch nicht als ausreichend zufällig empfinden. Wir stellen und vielmehr vor, daß aufeinanderfolgende Folgenglieder nicht stark

„korreliert“ sind, z.B. im folgenden Sinn:

$$\lim_{N\to\infty}\frac{1}{N}\sum_{t=1}^{N}(r_t-3,5)(r_t-3,5)=0 \quad \text{oder} \qquad \lim_{N\to\infty}\frac{1}{N}\sum_{t=1}^{N}\prod_{l=0}^{L}(r_{t+l}-3,5)=0 \quad \text{für jedes } L\in\mathbb{N}.$$

Allgemein: Ist $(X_t)_{t\in\mathbb{N}}$ ein stochastischer Prozeß aus identisch verteilten Zufallsvariablen mit Werten in $\mathbb{R}^n$, so ist eine Folge $(r_t)_{t\in\mathbb{N}}$ im $\mathbb{R}^n$ eine geeignete Zufallsfolge, wenn die in der jeweiligen Anwendung wichtigen stochastischen Größen des Prozesses gleich den entsprechenden zeitlichen Größen der Folge sind. Das ist natürlich keine exakte Definition.

Beispiele:

$$\mathbb{E}((r_t)_{t\in\mathbb{N}})=\lim_{N\to\infty}\frac{1}{N}\sum_{t=1}^{N}r_t=\mathcal{E}(X_1)=:a$$

$$\mathbb{E}(((r_t-a)(r_{t+s}-a)^T)_{t\in\mathbb{N}})=\mathcal{E}((X_t-1)(X_{t+s}-a)^T)=\text{const},$$

falls $(X_t)_{t\in\mathbb{N}}$ stationär entsprechend für die Kovarianz mehrerer Variablen

$$\lim_{N\to\infty}\tfrac{1}{N}\#\{j\in\mathbb{N}\mid j\le N, r_j\in M\}=p(X_1^{-1}(M)) \qquad \text{für jedes meßbare } M\subset\mathbb{R}^n.$$

Häufiger Spezialfall: (Ω,p) sei ein Wahrscheinlichkeitsraum, $X:\Omega\to[0,1]$ eine Zufallsvariable, deren Verteilung $p\circ X^{-1}$ die Gleichverteilung auf $[0,1]$ ist. $(X_t)_{t\in\mathbb{N}}$ seien unabhängige Wiederholungen von X. Gesucht ist eine Folge $(r_t)_{t\in\mathbb{N}}$ in $[0,1]$, die „überall gleich dicht“ liegt und deren aufeinanderfolgende Glieder „im zeitlichen Mittel unabhängig“ sind.

Zur rekursiven Konstruktion solcher Folgen verwendet man häufig *lineare Kongruenzen*

$$r_{t+1}=a\cdot r_t+b \mod c \qquad a,b,c\in\mathbb{N}.$$

Diese Definition liefert eine Folge in $[0,c-1]$.

1. Die Folge $(r_t)_{t\in\mathbb{N}}$ ist zyklisch. Die maximale Periodenlänge ist c. Braucht man bei einer Anwendung lange Folgen, so sollte c groß sein.
2. Erzeugt man aus $(r_t)_{t\in\mathbb{N}}$ eine Folge in $\mathbb{R}^n$, indem man sie in Stücke der Länge n zerhackt, so kann man zeigen, daß diese Vektoren in der Vereinigung von höchstens $\sqrt[n]{c}$ Hyperflächen im $\mathbb{R}^n$ liegen. Sind a, b, c ungeschickt gewählt, so können es viel weniger sein. Zwischen aufeinanderfolgenden r_t bestehen also starke Abhängigkeiten.
3. Über die sonstigen statistischen Eigenschaften ist nicht sehr viel bekannt.

Transformation von Wahrscheinlichkeitsdichten

Manchmal braucht man zur Simulation von Anwendungen Zufallszahlen, die nicht gleichverteilt sind, sondern einer anderen Verteilung genügen. Häufig benutzt man die Normalverteilung (oder Gaußverteilung), die die Gaußfunktion $g(x)=\frac{1}{\sqrt{2\pi}\sigma}e^{-x^2/2\sigma^2}$ als Dichte bezüglich des Lebesguemaßes auf $\mathbb{R}$ hat. σ heißt die *Streuung*; für $\sigma=1$ spricht man von der standardisierten Normalverteilung.

Es gibt zwei Gründe dafür, daß die Normalverteilung häufig benutzt wird:

1. Man kann mit ihr relativ leicht rechnen und Beweise führen.
2. Die wirklich vorkommenden Verteilungen sind Normalverteilungen oft recht ähnlich; das wird oft mit dem zentralen Grenzwertsatz begründet.

Um nicht gleichverteilte Zufallszahlen zu erhalten, geht man meist von gleichverteilten aus und transformiert sie geeignet. Hat die gewünschte Verteilung eine Dichte, so kann man die Transformation mit der Transformationsformel für Integrale finden:

Satz. Seien $U, V \subset \mathbb{R}^n$, $\phi : U \to V$ ein Diffeomorphismus und $h : V \to \mathbb{R}$ integrierbar. Dann gilt für jede meßbare Menge $A \subset U$

$$\int_{\phi(A)} h(y)\,dy = \int_A h \circ \phi(x) \cdot |\det(D\phi(x))|\,dx.$$

Diesen Satz kann man folgendermaßen ausnutzen: Ist h die gewünschte Dichte, so suche ein ϕ mit

$$\begin{aligned} h \circ \phi(x) \cdot |\det(D\phi(x))| = 1 \text{ für alle } x \in U \quad &\Leftrightarrow \quad h(y) = |\det(D\phi(x)(\phi^{-1}(y)))|^{-1} \text{ für alle } y \in V \\ &\Leftrightarrow \quad h(y) = |\det(D\phi^{-1}(y))| \text{ für alle } y \in V. \end{aligned}$$

Simpler Spezialfall: $n = 1, U = [0,1], V = [a,b]$, $h(y) = \frac{1}{b-a}$. Setze $\phi(x) = (b-a)x+a$. Dann $D\phi(x) = b-a$ und somit $h(y) = |\det(D\phi(\phi^{-1}(y)))|$. Mit ϕ kann man Zufallszahlen, die in $[0,1]$ gleichverteilt sind, in Zufallszahlen transformieren, die in $[a,b]$ gleichverteilt sind.

Im allgemeinen ist es nicht leicht, eine geeignete Abbildung ϕ zu finden. Um standardisiert normalverteilte Zufallszahlen zu erzeugen, gibt es einen kleinen Trick, um ein geeignetes ϕ zu finden. Darauf beruht auch die C-Funktion *gasdev()*, die standardisiert normalverteilte Zufallszahlen liefert.

3.10 Adaptive lineare Filter

Bei vielen Anwendungen werden zeitlich veränderliche Größen verarbeitet. Welche mathematischen Modelle sind dafür geeignet?

Die Zeit wird üblicherweise als $\mathbb{R}$ modelliert. Will man Digitalrechner zur Signalverarbeitung einsetzen, so muß man $\mathbb{R}$ durch eine diskrete Punktfolge $(t_j)_{j\in\mathbb{Z}}$ ersetzen, die meist äquidistant gewählt werden: $t_j = j\Delta t$.

Eine reellwertige von der Zeit abhängige Größe wird man also als Abbildung $\xi : \mathbb{Z} \to \mathbb{R}$ auffassen. Weil man in der Praxis erst ab einem Zeitpunkt t_0 mit den Untersuchungen beginnt, werden wir meist nur Abbildungen $\xi : \mathbb{N} \to \mathbb{R}$ betrachten. Wir werden sie oft *Signale* nennen.

Vielfach liegt nicht die eigentlich interessierende Größe vor, sondern eine durch Meßfehler oder Störungen verfälschte. Weil diese Abweichungen nicht exakt vorhersehbar sind, modelliert man sie stochastisch, indem man den Signalwert $\xi(t)$ als Realisierung einer Zufallsvariablen auffaßt. Ein Signal wird dann also als stochastischer Prozeß $(x_t)_{t\in\mathbb{N}}$ aus Zufallsvariablen x_t auf einem Wahrscheinlichkeitsraum (Ω, p) aufgefaßt; das in einer praktischen Anwendung auftretende Signal ξ ist ein Pfad von $(x_t)_{t\in\mathbb{N}}$, d.h. die Natur wählt gemäß p zufällig ein $\omega \in \Omega$ und liefert das Signal ξ mit den Werten $\xi(t) = x_t(\omega)$.

Beispiele.

1. Rauschen bei Bandaufnahmen, Schallplatten, Telefonleitungen

   allgemein: höherfrequente Störungen, die am oberen Ende des Frequenzspektrums des Nutzsignals liegen.

2. Störgeräusche mit ähnlichen Frequenzen wie das Nutzsignal

   Pfeiftöne, Maschinenlärm bei Sprachübertragung, 50 Hz Brummen bei EKG-Signalen, Herzschlag der Mutter bei EKG von Schwangeren.

   An dem 50 Hz Brummen scheint nichts stochastisch zu sein; das stimmt nicht ganz, denn Amplitude und Phasenlage sind bei jeder neuen Messung wieder etwas anders. Das methematische Modell für das Brummen könnte folgendermaßen aussehen:

   $\Omega = [0, A] \times [0, 2\pi[$ (Amplitude und Phase); p Gleichverteilung, vernünftiger ist es wohl, zwar jede Phasenlage als gleichwahrscheinlich anzusehen, aber nicht jede Amplitude. Für $\omega = (a, \varphi) \in \Omega$ setze

   $$x_t(\omega) = a \sin(2\pi \cdot 50 \cdot t \cdot \Delta t + \varphi),$$

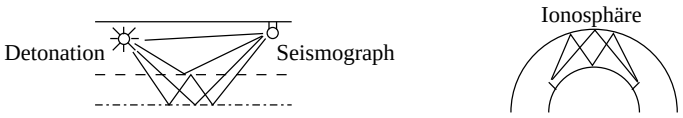
   wobei $t \in \mathbb{N}$ und Δt der Abstand zweier aufeinanderfolgender Zeitpunkte ist, in denen das kontinuierliche Signal abgetastet wird.

3.10.1 Übertragungssysteme

Signale werden meist verändert, wenn sie sich durch ein Medium ausbreiten.

Beispiele: Schall wird durch die akustischen Eigenschaften eines Raumes verändert, Sprache durch die (meist schlechten) Eigenschaften der Telefonleitungen, der Detonationsknall einer geologischen Sprengung durch die verschiedenen Erdschichten, Funkübertragung.

Solche Signaländerungen modelliert man einfach als Abbildung $S : \mathbb{R}^{\mathbb{N}} \to \mathbb{R}^{\mathbb{N}}$. Man bezeichnet S als ein *(Übertragungs-) System*. So eine Abbildung S kann natürlich sehr kompliziert sein. In vielen Situationen ist es jedoch gerechtfertigt, sie von einfacher Gestalt zu wählen. So entstehen in obigen Beispielen die Signalveränderungen zu einem wesentlichen Teil dadurch, daß das Signal an einigen Stellen teilweise reflektiert wird und somit über mehrere Wege mit unterschiedlichen Laufzeiten zum Empfänger (oder Sensor) gelangt; es summieren sich also Echos auf.



3.10.2 Lineares System

Ein *lineares, kausales System* $(n-1)$*-ter Ordnung* (oder *Filter*) ist eine Abbildung $L : \mathbb{R}^{\mathbb{N}} \to \mathbb{R}^{\mathbb{N}}$ der Gestalt

$$L\xi(t) = \sum_{l=0}^{n-1} w(l)\xi(t-l) \qquad \text{für } \xi \in \mathbb{R}^{\mathbb{N}},\ t \in \mathbb{N},$$

wobei $w(0), \ldots, w(n-1) \in \mathbb{R}$ die sogenannten Filterkoeffizienten sind und $\xi(-n+2), \ldots, \xi(0) \in \mathbb{R}$ den internen Anfangszustand zum Zeitpunkt 0 beschreiben.

Bemerkungen.

1. L ist einfach eine Summe von unterschiedlich gewichteten Echos, die um höchstens $n-1$ Zeittakte verzögert sind. Das System L braucht also einen internen Speicher für die $n-1$ jüngst vergangenen Signalwerte (man sagt, es habe Gedächtnis). Weil es nicht auf künftige Signalwerte vorgreift, ist es kausal; das ist vernünftig, wenn die Variable t die Zeit bedeutet. Beschreibt t jedoch z.B. den Ort, so spielen auch nicht kausale Systeme eine wichtige Rolle.

2. L ist eine lineare Abbildung, die translationsinvariant ist, d.h. sie kommutiert mit Translationsoperatoren.

3.10.3 Stochastische Prozesse

Hat L obige Gestalt und ist $x = (x_t)_{t\in\mathbb{N}}$ ein stochastischer Prozeß, so definiert man den stochastischen Prozeß Lx durch

$$(Lx)_t = \sum_{l=0}^{n-1} w(l)x_{t-l}.$$

3.10.4 Typische Anwendungen

a) Echounterdrückung

Infolge der veralteten Technologie von analogen Telefonen wird ein Teil des in die Sprechkapsel gesprochenen Signals beim Empfänger reflektiert und in die eigene Hörkapsel zurückgeschickt. Dieses Echo ist zeitlich verzögert. Ist die Verzögerung relativ groß (circa 200ms bei Satellitenübertragung), so stört sie den menschlichen Sprecher sehr; ist sie (bei lokalen Verbindungen) klein, so kann sie immerhin noch schnelle Modems stören. Deshalb ist es nötig, das Übertragungssystem S, das das Echo produziert, möglichst gut mit einem (linearen) System zu simulieren, um das Echo berechnen und damit eliminieren zu können.

Die Aufgabe ist also, ein lineares System L zu finden, dessen Ausgang Lx für jedes Sprachsignal x möglichst genau mit dem von S produzierten Echo y übereinstimmt. Es ist naheliegend, x und y als stochastische Prozesse zu modellieren und $\mathcal{E}(|y_t - (Lx)_t|^2)$ zu minimieren, wobei man x und y als gemeinsam stationär im weiteren Sinn annimmt, zumindest für einige Zeit.

b) Geräuschunterdrückung

Der Lärm einer Maschine soll an einem benachbarten Ort möglichst gut kompensiert werden, indem ein passendes gegenphasiges Geräusch erzeugt wird.

S sei das unbekannte System, das den Lärm zum Ohr überträgt. Das lineare System L ist so zu wählen, daß sein Ausgang möglichst genau das negative des Ausgangs von S ist.

Die Maschine produziere den Lärm x, den wir als stochastischen Prozeß auffassen. Das Mikrofon hört ein Geräusch $\tilde{x}$, das mit x stark korreliert ist. Das unkompensierte Geräusch am Ohr sei der stochastische Prozeß y (=Ausgang von S bei Eingabe von x). Man will L so konstruieren, daß y und $-Lx$ möglichst gut übereinstimmen im Sinne von $\mathcal{E}(|y_t + (Lx)_t|^2)$ minimal, wobei x und y als gemeinsam stationär im weiteren Sinn angenommen werden.

Wie findet man L? Wie kann sich L an wechselnde Lärmquellen und wechselnde Raumakustik anpassen?

c) Elimination von Störungen

$u = (u_t)_{t\in\mathbb{N}}$ und $v = (v_t)_{t\in\mathbb{N}}$ seien stochastische Prozesse, die stationär im weiteren Sinne sind.

Man stelle sich u als Sprachsignal vor; dann werden die u_t und u_{t-k} für größere Zeitunterschiede k kaum korreliert sein, also $\mathcal{E}(u_t u_{t-k}) \approx 0$ gelten.

v stelle man sich als pfeifendes Nebengeräusch vor, z.B. eine Überlagerung von Sinustönen mit zufälliger Phase und Amplitude. Dann wird v_t mit u_s wenig korreliert sein; wir setzen daher $\mathcal{E}(v_t u_s) = 0$ voraus.

Zu hören sei nun die Überlagerung $y = u + v$, d.h. $y_t = u_t + v_t$ für alle t. Wie kann man die Störung v entfernen?

Idee: v_t wird mit den Variablen y_s zu früheren Zeitpunkten stark korreliert sein, u_t dagegen nicht, wenn $s \le t - k$. Versuche daher, v_t aus $y_{t-k}, \ldots, y_{t-k-n+1}$ linear zu schätzen (n geeignet). Setze $x_t = y_{t-k}$ für $t > k + n - 1$, und 0 sonst.

Aufgabe: Finde ein lineares Filter L $(n-1)$-ter Ordnung so, daß $\mathcal{E}(|y_t - (Lx)_t|^2)$ möglichst klein ist. (Beachte, daß $\mathcal{E}(|y_t - (Lx)_t|^2)$ unabhängig von t ist.)

d) Lineare Vorhersage von Zeitreihen

Sei $(x_t)_t$ ein stochastischer Prozeß und stationär im weiteren Sinn; $y_t := x_{t+1}$. Versuche, y durch x auf lineare Weise möglichst gut zu schätzen, d.h. finde ein lineares System L, so daß $\mathcal{E}(|x_{t+1} - (Lx)_t|^2)$ möglichst klein ist.

Gelingt dies gut, so kann man jeden Pfad $(x_t(\omega))_t$ durch die Anfangswerte $x_1(\omega), \ldots, x_n(\omega)$ und die Folge der Fehler $x_{t+1}(\omega) - \sum_{l=0}^{n-1} w(l) x_{t-l}(\omega)$ beschreiben. Meist genügt es, die Fehler grob zu quantisieren. Dann erhält man eine Datenkompression (*linear predicitve coding*; einfache Variante: Deltamodulation). Für kürzere Zeitspannen sind sogar Sprachsignale genügend stationär, so daß es genügt, ab und zu neue Filterkoeffizienten zu übertragen.

3.10.5 Approximationsaufgabe

Sei $n \in \mathbb{N}$. $x = (x_t)_{t\in\mathbb{N}}$ und $y = (y_t)_{t\in\mathbb{N}}$ seien stochastische Prozesse; sie seien gemeinsam stationär im weiteren Sinn. Finde ein lineares Filter L $(n-1)$-ter Ordnung so, daß $\mathcal{E}(|y_t - (Lx)_t|^2)$ möglichst klein ist.

Umformulierung: Setze $X_t = (x_t, x_{t-1}, \ldots, x_{t-n+1})^T$, $Y_t = y_t$. Ist $w = (w(0), \ldots, w(n-1))^T$ der Vektor der Filterkoeffizienten, so gilt $\mathcal{E}((y_t - (Lx)_t)^2) = \mathcal{E}((Y_t - w^T X_t)^2)$.

Somit ist ein lineares L^2-Approximationsproblem II zu lösen. Dafür können wir die schon angeführten Lösungsmethoden verwenden, insbesondere den LMS-Algorithmus.

3.10.6 Adaptive Filter

Gegeben seien ein Pfad $\xi : \mathbb{N} \to \mathbb{R}$ von $(x_t)_{t\in\mathbb{N}}$ und ein Pfad $\eta : \mathbb{N} \to \mathbb{R}$ von $(y_t)_{t\in\mathbb{N}}$. Wähle Anfangsfiltergewichte $w_1(0), \ldots, w_1(n-1) \in \mathbb{R}$ und ein $\alpha > 0$. Setze

$$w_{t+1}(j) = w_t(j) + 2\alpha \left(\eta(t) - \sum_{l=0}^{n-1} w_t(l)\xi(t-l) \right) \cdot \xi(t-j) \qquad \text{für } t \in \mathbb{N},\ j \in \{0, \ldots, n-1\}.$$

Zu jedem Zeitpunkt t bekommt man also ein neues Filter mit den Koeffizienten $w_t(0), \ldots, w_t(n-1)$. Ist α hinreichend klein und sind die Folgen ξ und η repräsentativ für die Prozesse x und y, so konvergieren die Filtergewichte meist gegen die eines in obigem Sinne optimalen Filters. Das ist eine Erfahrungstatsache und nicht bewiesen.

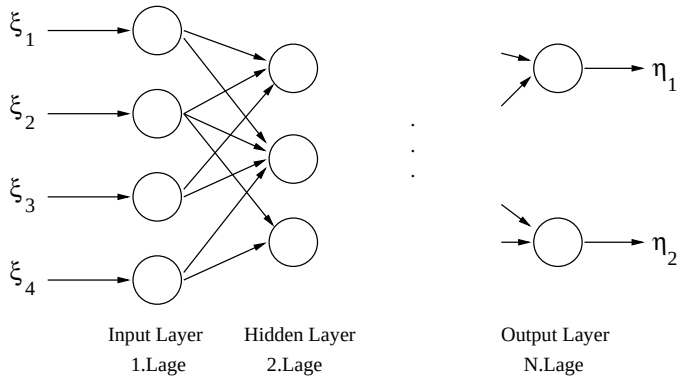
Für Anwendungen interessant ist, daß die Filterkoeffizienten nach anfänglichen Schwankungen meist relativ schnell ($t \approx 1000$) in die Nähe der optimalen Werte kommen und nur noch kleine Änderungen ausführen. Ändern sich die statistischen Eigenschaften der Folgen ξ und η zu einem Zeitpunkt sehr abrupt, so müssen die Filterkoeffizienten erst wieder konvergieren; ändern sie sich jedoch langsam, so bleiben die Koeffizienten immer nahe den optimalen Werten.

Probleme in der Praxis: Wie groß darf α sein? Wie groß muß man die Filterordnung n wählen?

Kapitel 4

Vorwärtsgerichtete Netze mit Backpropagation-Training

Wir betrachten mehrlagige Netze aus McCulloch-Pitts-Neuronen, die folgendermaßen aufgebaut sind.



Die Neuronenmenge des gesamten Netzes ist in N Lagen aufgeteilt, so daß gilt:

- Jedes Neuron der 1. Lage hat genau einen Eingang; diese Eingänge sind mit keinem Ausgang eines Neurons verbunden. Sie dienen als Eingänge des gesamten Netzes.
- Die Ausgänge der Neuronen der N. Lage sind mit keinem Eingang eines Neurons verbunden. Sie dienen als Augänge des gesamten Netzes.
- Der Ausgang jedes Neurons der j. Lage ist mit einem Eingang jedes Neurons der $j+1$. Lage verbunden.

Sonst gibt es keine Verbindungen.

Wir setzen im folgenden voraus, daß die Netztopologie fest gewählt ist, und daß ebenso die Aktivierungsfunktionen der Neuronen fest gewählt sind (sie müssen nicht gleich sein, werden aber vielfach identisch gewählt). Das einzige, was variieren darf, sind die Gewichte der Synapsen; wir fassen sie zu einem Vektor w zusammen.

$\xi = (\xi_1, \ldots, \xi_n)^T$ seien die Werte an den Netzeingängen, $\eta = (\eta_1, \ldots, \eta_m)^T$ die an den Netzausgängen. Für jeden Gewichtsvektor w realisiert das Netz eine Abbildung $\eta = F_w(\xi)$. In Anwendungen wird man w so wählen, daß F_w möglichst gut mit einer gewünschten Abbildung übereinstimmt.

Welche Abbildungen F_w erhält man auf diese Weise?

Wenn die Aktivierungsfunktionen nicht linear sind und das Netz mehrlagig ist, ist schwer zu überblicken, welche Funktionen durch das Netz dargestellt werden können. Mir sind keine Aussagen bekannt.

Vereinbarung. Von nun an setzen wir voraus, daß die Neuronen der 1. Lage die identische Abbildung realisieren, d.h. sie dienen nur zur Verteilung der Eingabewerte.

Fixiert man lediglich die Anzahl der Lagen und eine für alle Neuronen (außer denen der 1. Lage) gemeinsame Aktivierungsfunktion, aber nicht die Anzahl der Neuronen in den versteckten Lagen, so gibt es Beschreibungen der approximierbaren Funktionen.

4.1 Satz. Zu jedem $\varepsilon > 0$ und zu jeder Abbildung $f \in L^2([0,1]^n, \mathbb{R}^m)$ gibt es ein 3-lagiges Netz, das eine Funktion F mit $\|F - f\|_2 < \varepsilon$ realisiert.

4.2 Definition. Eine Funktion $h : \mathbb{R} \to [0,1]$ (oder $h : \mathbb{R} \to [-1,1]$) heißt *sigmoid*, wenn sie monoton wächst und $\lim_{t\to\infty} h(t) = 1$ und $\lim_{t\to-\infty} h(t) = 0$ (bzw. $\lim_{t\to-\infty} h(t) = -1$).

4.3 Satz. Seien $f : [0,1]^n \to \mathbb{R}^m$ stetig und h sigmoid. Zu jedem $\varepsilon > 0$ gibt es ein 3-lagiges Netz, das eine Funktion F mit $\|F - f\|_\infty < \varepsilon$ realisiert. Das Netz kann so gewählt werden, daß die Neuronen der versteckten Lage h als Aktivierungsfunktion haben und die Neuronen der Ausgangslage linear sind.

An dieser Stelle wird nur eine Beweisskizze angegeben.

4.3.1 Lemma. $g \geq 0$ sei eine stetige sigmoide Funktion, $h \geq 0$ eine beliebige sigmoide Funktion und $\varepsilon > 0$. Dann gibt es ein $L \in \mathbb{N}$, $\alpha_1, \ldots, \alpha_L \in \mathbb{R}$, affin lineare Abbildungen $A_1, \ldots, A_L$ von $\mathbb{R}$ nach $\mathbb{R}$, so daß für

$$\varphi(t) := \sum_{l=1}^{L} \alpha_l h(A_l(t)) \quad \text{gilt, daß} \quad \sup_{t\in\mathbb{R}} |g(t) - \varphi(t)| < \varepsilon.$$

Beweis. Ohne Einschränkung sei $\varepsilon < 1$. Wähle $L \in \mathbb{N}$ so, daß $\frac{1}{L+1} < \frac{\varepsilon}{4}$. Setze $\alpha_l := \frac{1}{L+1}$ für $l \in \{1, \ldots, L\}$. Weil h sigmoid ist, gibt es ein $M > 0$, so daß $h(-M) < \frac{\varepsilon}{2(L+1)}$ und $h(M) > 1 - \frac{\varepsilon}{2(L+1)}$. Weil g stetig ist, existieren $r_l = \sup\{t \mid g(t) = \frac{l}{L+1}\}$ für $l \in \{1, \ldots, L\}$ und $r_{L+1} = \sup\{t \mid g(t) = 1 - \frac{1}{2(L+1)}\}$.

Für jedes $l \in \{1, \ldots, L\}$ sei A_l diejenige affin lineare Transformation von $\mathbb{R}$, die $[r_l, r_{l+1}]$ auf $[-M, M]$ abbildet. Für $l \in \{1, \ldots, L\}$ und $t \in]r_l, r_{l+1}]$ gilt

$$\begin{aligned}
h(A_j(t)) &< \frac{\varepsilon}{2(L+1)} && \text{für } j > l \\
h(A_j(t)) &> 1 - \frac{\varepsilon}{2(L+1)} && \text{für } j < l \\
\frac{l}{L+1} &< g(t) < \frac{l+1}{L+1} \\
\frac{\varepsilon}{2(L+1)} &\leq h(A_l(t)) \leq 1 - \frac{\varepsilon}{2(L+1)}
\end{aligned}$$

und damit

$$\begin{aligned}
\varphi(t) &= \sum_{j=1}^{l-1} \alpha_j h(A_j(t)) + \alpha_l h(A_l(t)) + \sum_{j=l+1}^{L} \alpha_j h(A_j(t)) \geq \frac{1}{L+1}\left((l-1)\left(1 - \frac{\varepsilon}{2(L+1)}\right) + \frac{\varepsilon}{2(L+1)}\right) \\
&\geq \frac{1}{L+1}\left(l - 1 - \frac{\varepsilon}{2}\right) \geq \frac{l}{L+1} - \frac{1}{L+1} - \frac{\varepsilon}{2(L+1)} \geq \frac{l}{L+1} - \frac{\varepsilon}{2}
\end{aligned}$$

sowie

$$\varphi(t) \leq \frac{l-1}{L+1} \cdot 1 + \frac{1}{L+1} + \frac{L-l-1}{L+1} \cdot \frac{\varepsilon}{2(L+1)} \leq \frac{l}{L+1} + \frac{\varepsilon}{2},$$

also $|g(t) - \varphi(t)| < \varepsilon$, für $t \in]r_l, r_{l+1}]$. Außerdem gelten:

$$\begin{aligned}
|g(t) - \varphi(t)| &\leq |g(t)| + |\varphi(t)| \leq \frac{1}{L+1} + \frac{L}{L+1} \cdot \frac{\varepsilon}{2(L+1)} < \frac{\varepsilon}{4} + \frac{\varepsilon}{4} \cdot \frac{\varepsilon}{2} < \varepsilon, \text{ für } t < r_1, \\
g(t) - \varphi(t) &< 1 - \frac{L}{L+1}\left(1 - \frac{\varepsilon}{2(L+1)}\right) = \frac{1}{L+1} + \frac{L}{L+1} \cdot \frac{\varepsilon}{2(L+1)} < \frac{\varepsilon}{4} + \frac{\varepsilon}{2} < \varepsilon, \text{ für } t > r_{L+1} \text{ und} \\
\varphi(t) - g(t) &\leq 1 - \left(1 - \frac{1}{2(L+1)}\right) = \frac{1}{2(L+1)} < \frac{\varepsilon}{8} < \varepsilon, \text{ für } t > r_{L+1}.
\end{aligned}$$

□

4.3.2 Lemma. Zu jeder sigmoiden Funktion $h : \mathbb{R} \to [0,1]$, jedem $\varepsilon > 0$ und jedem $M > 0$ gibt es ein $L \in \mathbb{N}$ und $\alpha_1, \ldots, \alpha_n \in \mathbb{R}$ und affin lineare Transformationen $A_1, \ldots, A_L$ von $\mathbb{R}$, so daß für

$$\gamma(t) = \sum_{l=1}^{L} \alpha_l h(A_l(t)) \quad \text{gilt, daß} \quad \sup\{|\gamma(t) - \cos(t)| \mid t \in [-M, M]\} < \varepsilon.$$

Beweis. Setze

$$g(t) = \begin{cases} 0, & \text{für } t < -\frac{\pi}{2} \\ \frac{1}{2} + \frac{1}{2}\cos(t + \frac{3}{2}\pi), & \text{für } t \in [-\frac{\pi}{2}, \frac{\pi}{2}] \\ 1, & \text{für } t > \frac{\pi}{2} \end{cases} .$$

g ist sigmoid und stetig.

Auf $[-M, M]$ kann man cos durch eine endliche affine Linearkombination von k affind linear transformierten Versionen von g darstellen. Nach dem letzten Lemma kann man g auf $\mathbb{R}$ durch φ mit der Gestalt $\sum_l \alpha_l h \circ A_l$ bis auf $\frac{\varepsilon}{k}$ approximieren. Das gilt entsprechend für die affin linear verzerrten Versionen. Mit der Dreiecksungleichung folgt die Behauptung. □

Beweis von 4.3 (für $m = 1$). Nach Stone-Weierstraß gibt es eine Funktion P der Gestalt

$$\sum_{j=1}^{s} \lambda_j \prod_{k=1}^{n_j} \cos \circ A_{j,k},$$

wobei $s, n_j \in \mathbb{N}$, $\lambda_j \in \mathbb{R}$, $A_{j,k} : \mathbb{R}^n \to \mathbb{R}$ affin linear, so daß $|f(x) - P(x)| < \frac{\varepsilon}{2}$ für $x \in [0,1]^n$.

Durch mehrmaliges Anwenden des Additionstheorems

$$\cos(\alpha)\cos(\beta) = \tfrac{1}{2}\left(\cos(\alpha + \beta) + \cos(\alpha - \beta)\right)$$

kann man jedes trigonometrische Polynom obiger Art umformen in die Gestalt

$$\sum_{j=1}^{r} \beta_j \cos \circ B_j,$$

wobei $\beta_j \in \mathbb{R}$ und $B_j : \mathbb{R}^n \to \mathbb{R}$ affin linear sind. Es gibt ein $M > 0$, so daß $B_j([0,1]^n) \subset [-M, M]$ für jedes $j \in \{1, \ldots, r\}$. Setze $\varepsilon' = \frac{\varepsilon}{2}\left(\sum_{j=1}^{r} |\beta_j|\right)^{-1}$. Nach vorhergehendem Lemma gibt es ein γ der Gestalt $\gamma = \sum_{l=1}^{L} \alpha_l h \circ A_l$ mit $|\gamma(t) - \cos(t)| < \varepsilon'$ für $t \in [-M, M]$. Damit folgt

$$\left| \sum_{j=1}^{r} \beta_j \cos\left(B_j(x)\right) - \sum_{j=1}^{r} \beta_j \gamma\left(B_j(x)\right) \right| \leq \sum_{j=1}^{r} |\beta_j| \cdot |\cos\left(B_j(x)\right) - \gamma\left(B_j(x)\right)| < \sum_{j=1}^{r} |\beta_j| \varepsilon' = \tfrac{\varepsilon}{2},$$

für $x \in [0,1]^n$. Insgesamt folgt

$$\left| f(x) - \sum_{j=1}^{r} \sum_{l=1}^{L} \beta_j \alpha_l h\left(A_l \circ B_j(x)\right) \right| < \varepsilon, \text{ für } x \in [0,1]^n.$$

□

4.4 Bemerkung. Der Satz 4.3 besagt, daß man mit 3-lagigen Netzen jede stetige Funktion auf einem Kompaktum im $\mathbb{R}^n$ beliebig genau approximieren kann. Er sagt aber nicht aus, daß es ein 3-lagiges Netz gibt, mit dem man allein durch Verändern der Gewichte jede stetige Funktion beliebig gut approximieren kann.

Probleme in der Praxis:

1. Wieviele Neuronen soll man in der versteckten Lage wählen? Die Zahl der Eingangs- und Ausgangsneuronen ist festgelegt.

2. Welche Aktivierungsfunktion(en) soll man wählen? Der Satz 4.3 sagt zwar aus, daß man die Ausgangsneuronen linear wählen darf und für die versteckten Neuronen eine beliebige sigmoide Aktivierungsfunktion verwenden kann; jedoch könnte die benötigte Anzahl von versteckten Neuronen wesentlich von dieser Wahl abhängen.

3. Wie soll man die optimalen Gewichte finden?

Übliche Vorgehensweise in der Praxis: Wähle irgendeine sigmoide Aktivierungsfunktion für die versteckten Neuronen (eventuell auch für die Ausgangsneuronen) und irgendeine Anzahl von versteckten Neuronen. Dieses Netz stellt dann eine Familie $(F_w)_w$ von Abbildungen $\eta = F_w(\xi)$ dar, die durch den Vektor aller Gewichte parametrisiert ist. Nun versucht man durch eine Lernregel (die eigentlich immer ein modifizierter Gradientenabstieg ist) den Gewichtsvektor w solange zu ändern, bis die gewünschte Abbildung durch F_w gut genug angenähert wird. Wenn dies nicht gelingt, verändert man die Anzahl der versteckten Lagen oder die ihrer Neuronen oder die Aktivierungsfunktion oder die Anfangsgewichte und beginnt von vorn. Es gibt kaum begründete Richtlinien für die Wahl dieser Parameter, höchstens Erfahrungswerte. Welche Abbildung $(F_w)_w$ durch ein gwähltes Netz dargestellt werden, ist kaum zu übersehen. F_w hängt i.a. hochgradig nichtlinear von w ab.

Wir werden im folgenden davon ausgehen, daß die Netztopologie fest gewählt ist und nur noch die Synapsengewichte gewählt werden dürfen. Dafür formulieren wir Approximationsprobleme in völliger Analogie zu denen für ein Adaline.

Statt $\eta = F_w(\xi)$ werden wir meist $\eta = F(w, \xi)$ schreiben.

4.5 L^2-Approximationsaufgaben

Gegeben sei ein vorwärtsgerichtetes Netz mit den Eingängen $\xi = (\xi_1, \ldots, \xi_n)$ und den Ausgängen $(\eta_1, \ldots, \eta_m)$. Die Synapsengewichte seien zu einem Vektor w zusammengefaßt; sie werden als variabel angesehen. Für jedes w wird durch das Netz eine Abbildung $\eta = F(w, \xi)$ realisiert.

4.5.1 Aufgabe I

Gegeben sei die Folgen $(a_i)_{i\in I}$ im $\mathbb{R}^n$ und $(b_i)_{i\in I}$ im $\mathbb{R}^m$, wobei $I = \mathbb{N}$ oder $I = \{1, \ldots, N\}$ mit $N \in \mathbb{N}$.

Gesucht ist ein w, so daß $e(w) = \sum_{i\in I} \|b_i - F(w, a_i)\|^2$ minimal ist.

4.5.2 Aufgabe II

(Ω, p) sei ein Warscheinlichkeitsraum, und $X : \Omega \to \mathbb{R}^n$ und $Y : \Omega \to \mathbb{R}^m$ seien Zufallsvariablen.

Gesucht ist ein w, so daß $e(w) = \mathcal{E}(\|Y - F(w, X)\|^2)$ minimal ist.

4.5.3 Aufgabe III

$(x_t)_{t\in\mathbb{N}}$ und $(y_t)_{t\in\mathbb{N}}$ seien zwei stochastische Prozesse, so daß die gemeinsame Verteilung von x_t und y_t (also die Verteilung von (x_t, y_t)) für alle $t \in \mathbb{N}$ dieselbe sei.

Gesucht ist ein w, so daß $e(w) = \mathcal{E}(\|y_t - F(w, x_t)\|^2)$ minimal ist. Dieser Erwartungswert hängt dabei nicht von t ab.

4.6 Lösungsstrategien

In allen drei Aufgaben ist eine Fehlerfunktion e zu minimieren, die leider nicht mehr quadratisch ist, wenn die Aktivierungsfunktionen der versteckten Neuronen nicht linear sind. Es ist nicht klar, ob e ein globales Minimum ist besitzt; falls ja, gibt es meist mehrere Stellen, wo es angenommen wird, weil e unter etlichen Permutationen der Synapsengewichte invariant ist.

Außer globalen Minima kann e auch noch lokale Minima haben, die nicht global sind. Deshalb genügt es nicht, die Gleichung $\nabla e(w) = 0$ zu lösen wie im Falle von Adalines. Überdies ist diese Gleichung nicht leicht zu lösen, denn

1. sie ist hochgradig nichtlinear,
2. sie hat viele Variablen, nämlich alle Gewichte des Netzes,
3. sie ist in den Aufgaben II und III meist garnicht exakt bekannt, weil die Verteilung von (X, Y) bzw. (x_t, y_t) nicht exakt bekannt ist.

Wie kann man also ein globales Minimum von e finden?

Eine Möglichkeit ist die Benutzung stochastischer Optimierungsmethoden; sie sind jedoch meist langwierig. Wir werden in späteren Abschnitten darauf zurückkommen.

Ungeachtet aller oben genannten Schwierigkeiten ist es weit verbreitet, wie bei Adalines mit Gradientenabstiegsmethoden zu arbeiten. In Aufgabe I kennt man wenigstens die Fehlerfunktion e und hat somit die Chance mit Gradientenabstieg in ein dem Anfangsgewichtsvektor benachbartes lokales Minimum zu laufen. In Aufgabe II und III dagegen muß man überdies noch die Verteilungen der Zufallsvariablen und damit e und ∇e schätzen. Es gelten dieselben Überlegungen wie bei Adalines. In Aufgabe II zieht man von X und Y viele unabhängige Stichprobenund schätzt daraus e. Man kann Aufgabe II auch als Aufgabe III formulieren, indem für $(x_t)_{t\in\mathbb{N}}$ und $(y_t)_{t\in\mathbb{N}}$ unabhängige Wiederholungen von X und Y nimmt.

In der Praxis ergibt sich für die Aufgabe III folgende Situation: Man wählt einen Startwert w_1. Zu jedem Zeitpunkt t erhält man Realisierungen a_t von x_t und b_t von y_t, d.h. man erhält die Anfangsstücke $(a_s)_{s\le t}$ und $(b_s)_{s\le t}$ zweier Pfade. Daraus berechnet man einen Schätzwert $\hat{\gamma}(t)$ von $\nabla e(w(t))$ und setzt $w(t+1) = w(t) - \alpha\hat{\gamma}(t)$, wobei $\alpha > 0$ eventuell auch noch von t abhängen darf.

Wie wählt man den Schätzwert $\hat{\gamma}(t)$ von $\nabla e(w(t))$. Es gilt

$$\nabla e(w) = \nabla(\mathcal{E}(\|y_t - F(w, x_t)\|^2)) = \mathcal{E}(\nabla \langle y_t - F(w, x_t), y_t - F(w, x_t)\rangle) = -2\mathcal{E}(D_1F(w, x_t)^T(y_t - F(w, x_t))),$$

wobei D_1F die partielle Ableitung nach w bezeichne.

Man könnte nun $\mathcal{E}$ durch den zeitlichen Mittelwert $-\frac{2}{t}\sum_{s=1}^{t} D_1F(w, a_s)^T(b_s - F(w, a_s))$ ersetzen. Die Berechnung dieses Mittelwerts kann jedoch recht aufwendig sein, weil D_1F eine wesentlich komplizierte Gestalt als im Falle eines Adalines hat (wo $D_1F = a_s^T$ war). Daher ist es verführerisch, wie schon bei Adalines jegliche Mittelwertbildung einfach wegzulassen.

4.6.1 Der LMS-Algorithmus für Netze oder die verallgemeinerte Deltaregel

Seien $\alpha > 0$ und $w(1)$ ein Vektor von Anfangsgewichten, $(a_t)_{t\in\mathbb{N}}$ eine Folge von Eingangsvektoren, $(b_t)_{t\in\mathbb{N}}$ eine Folge von Ausgangsvektoren. Für jeden Gewichtsvektor w sei $F(w, \cdot)$ die vom Netz realisierte Abbildung.

Für $t \in \mathbb{N}$ setze

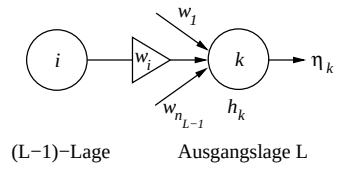
$$w(t+1) = w(t) + 2\alpha D_1F(w(t), a_t)^T(b_t - F(w(t), a_t)),$$

die sogenannte verallgemeinerte Deltaregel (*generalized delta rule, GDR*).
Bemerkung. Man hofft (wie schon bei Adalines), daß die Folge $(w_t)_{t\in\mathbb{N}}$ gegen eine globale Minimumstelle von e konvergiert.

Um die Folge $(w_t)_{t\in\mathbb{N}}$ in der Praxis berechnen zu können, müssen wir D_1F explizit ausrechnen. Dabei wird verständlich werden, warum das Lernen mit der GDR auch als *Backpropagation* bezeichnet wird.

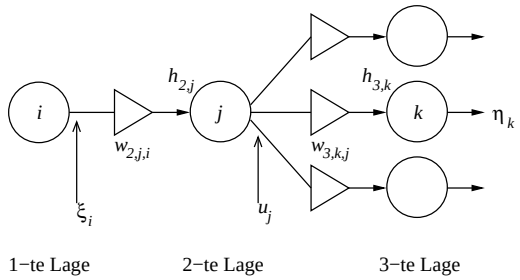
4.6.2 Berechnung von D_1F

Wir betrachten zunächst zwei Spezialfälle, nämlich die partiellen Ableitungen nach einem Gewicht der Ausgangslage bzw. einem Gewicht der versteckten Lage eines 3-lagigen Netzes. Die vom Netz dargestellte Funktion $\eta = F_w(\xi)$ schreiben wir jetzt in der Form $\eta = F(w, \xi)$. F_k sei die k-te Komponente, also $\eta_k = F_k(w, \xi)$.



$$\frac{\partial}{\partial w_i} F_k(w, \xi) = h_k' \left(\sum_{j=1}^{n_L - 1} w_j u_j \right) u_i,$$

wobei u_j der Ausgangswert des j-ten Neurons der vorletzten Lage ist.

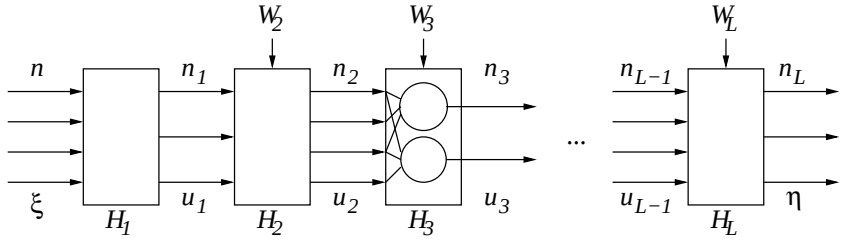


$$\begin{aligned} \frac{\partial F_k}{\partial w_{2,j,i}}(w, \xi) &= \frac{\partial \eta_k}{\partial u_j} \left(\sum_{r=1}^{n_2} w_{3,k,r} u_r \right) \frac{\partial u_j}{\partial w_{2,j,i}} \\ &= h_{3,k}' \left(\sum_{r=1}^{n_2} w_{3,k,r} u_r \right) w_{3,k,j} \cdot h_{2,j}' \left(\sum_{r=1}^{n_1} w_{2,j,r} \xi_r \right) \xi_i \end{aligned}$$

Berechnung von D_1F im allgemeinen Fall

Das Netz habe L Lagen; die l-te Lage habe n_l Neuronen, die nummeriert seien. Jedes dieser Neuronen hat n_{l-1} Eingänge; die Verbindung vom i-ten Neuron der $(l-1)$-ten Lage zum j-ten Neuron der l-ten Lage habe das Gewicht $w_{l,j,i}$. Setze $w_{l,j} = (w_{l,j,1}, \ldots, w_{l,j,n_{l-1}})$ für $l \in \{2, \ldots, L\}$.

Die Werte der n_l Ausgänge der l-ten Lage sind eine Funktion der n_{l-1} Ausgangswerte der $(l-1)$-ten Lage $u_{l-1} = (u_{l-1,1}, \ldots, u_{l-1,n_{l-1}})^T \in \mathbb{R}^{n_{l-1}}$ und des (Spalten-)Gewichtsvektors $w_l = (w_{l,1}, \ldots, w_{l,n_l})^T \in \mathbb{R}^{n_{l-1} \cdot n_l}$; sie werde mit $H_l : \mathbb{R}^{n_{l-1} \cdot n_l} \times \mathbb{R}^{n_{l-1}} \to \mathbb{R}^{n_l}$ bezeichnet.



$h_{l,j}$ sei die Aktivierungsfunktion des j-ten Neurons der l-ten Lage; sie sei differenzierbar. Dann hat H_l die Gestalt

$$H_l(w_l, u_{l-1}) = \begin{pmatrix} h_{l,1}(\langle w_{l,1}, u_{l-1} \rangle) \\ \vdots \\ h_{l,n_l}(\langle w_{l,n_l}, u_{l-1} \rangle) \end{pmatrix}.$$

Die durch das gesamte Netz realisierte Abbildung $\xi \mapsto F_w(\xi)$ schreiben wir jetzt als $F(w, \xi)$; sie hat die Gestalt

$$F(w, \xi) = H_L(w_L, H_{L-1}(w_{L-1}, \ldots, H_2(w_2, \xi)) \ldots),$$

wobei $w = (w_1^T, \ldots, w_L^T) \in \prod_{l=2}^{L} \mathbb{R}^{n_l \cdot n_{l-1}}$ und $\xi \in \mathbb{R}^n = \mathbb{R}^m$; denn $H_1 = \mathrm{Id}_{\mathbb{R}^n}$. Sei

$$u_j = H_j(w_j, H_{j-1}(w_{j-1}, \ldots, H_2(w_2, \xi)) \ldots) \quad \text{für jedes } j \in \{2, \ldots, L\}.$$

Dann gilt nach der Kettenregel für jedes l

$$\frac{\partial F}{\partial w_l}(w,\xi) = D_2H_L(w_L,u_{L-1})\cdot D_2H_{L-1}(w_{L-1},u_{L-2})\cdot\ldots\cdot D_2H_{l+1}(w_{l+1},u_l)\cdot D_1H_l(w_l,u_{l-1}). \tag{1}$$

$$D_1H_l(w_l,u_{l-1}) = \begin{pmatrix} h'_{l,1}(\langle w_{l,1},u_{l-1}\rangle)u_{l-1}^T & 0 & \cdots & 0 \\ 0 & h'_{l,2}(\langle w_{l,2},u_{l-1}\rangle)u_{l-1}^T & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & h'_{l,n_l}(\langle w_{l,n_l},u_{l-1}\rangle)u_{l-1}^T \end{pmatrix}.$$

$$D_2H_l(w_l,u_{l-1}) = \begin{pmatrix} h'_{l,1}(\langle w_{l,1},u_{l-1}\rangle)w_{l,1} \\ \vdots \\ h'_{l,n_l}(\langle w_{l,n_l},u_{l-1}\rangle)w_{l,n_l} \end{pmatrix}.$$

Mit

$$A_l = \begin{pmatrix} h'_{l,1}(\langle w_{l,1},u_{l-1}\rangle) & & 0 \\ & \ddots & \\ 0 & & h'_{l,n_l}(\langle w_{l,n_l},u_{l-1}\rangle) \end{pmatrix} \quad \text{und} \quad W_l = (w_{l,1}^T,\ldots,w_{l,n_l}^T)$$

ist dann $D_1H_l^T = u_{l-1}\otimes A_l$ und $D_2H_l^T = (w_{l,1},\ldots,w_{l,n_l})\cdot A_l = W_l\cdot A_l$.[1]

Mit $D_1F = \left(\frac{\partial F}{\partial w_2},\ldots,\frac{\partial F}{\partial w_L}\right)$ ergibt sich

$$\nabla_w F(w,\xi) = D_1F(w,\xi)^T = \begin{pmatrix} \frac{\partial F}{\partial w_2}^T \\ \vdots \\ \frac{\partial F}{\partial w_L}^T \end{pmatrix} = \begin{pmatrix} u_1\otimes A_2\cdot W_3A_3\cdot W_4A_4\cdot\ldots\cdot W_LA_L \\ u_2\otimes A_3\cdot W_4A_4\cdot\ldots\cdot W_LA_L \\ \vdots \\ u_{L-1}\otimes A_L \end{pmatrix}, \tag{2}$$

wobei wieder $u_l = (u_{l-1},\ldots,u_{l,n_l})^T$ und $u_{l,j}$ der Ausgangswert des j-ten Neurons in der l-ten Lage, wenn ξ der Netzwerkinput ist.

4.6.3 Die verallgemeinerte Deltaregel als Fehler-Backpropagation

Die Bezeichnungen seien wie oben gewählt. Das Netz habe also L Lagen, die l-te Lage habe n_l Neuronen, und $w_{l,j,i}$ sei das Gewicht der Verbindung des i-ten Neurons der $(l-1)$-ten Lage mit dem j-ten der l-ten Lage. Setze $w_{l,j} = (w_{l,j,1},\ldots,w_{l,j,n_{l-1}}) \in \mathbb{R}^{n_{l-1}}$ und $w_l = (w_{l,1},\ldots,w_{l,n_l})^T = (w_{l,1,1},\ldots,w_{l,1,n_{l-1}},w_{l,2,1},\ldots,w_{l,n_l,n_{l-1}})^T$. Es ist $w_l \in \mathbb{R}^{n_l\cdot n_{l-1}}$. Der Zeilenvektor $w = (w_2^T,\ldots,w_L^T)$ besteht aus den aneinandergehängten Zeilen w_l^T.

Für jedes t sei $\epsilon(t) = b(t) - F(w(t),a(t)) \in \mathbb{R}^m$ der *Fehlervektor* zum Zeitpunkt t. Dann läßt sich die verallgemeinerte Deltaregel in 4.6.1 mit Hilfe von Gleichung (2) in 4.6.2 folgendermaßen formulieren:

Für jedes $t\in\mathbb{N}$ und $l\in\{2,\ldots,L-1\}$ sei

$$\begin{aligned} w_l(t+1) &= w_l(t) + 2\alpha\, u_{l-1}(t)\otimes(A_l(t)\cdot W_{l+1}(t)A_{l+1}\cdot\ldots\cdot W_L(t)A_L(t)\cdot\epsilon(t)) \quad \text{und} \\ w_L(t+1) &= w_L(t) + 2\alpha\, u_{L-1}(t)\otimes(A_L(t)\cdot\epsilon(t)) \end{aligned}$$

Dabei ist $u_l(t)$ wieder der Vektor der Ausgangswerte der Neuronen der l-ten Lage, wenn $a(t)$ der Input des Netzes und $w(t)$ der Gewichtsvektor ist. Diese Gleichungen kann man noch etwas übersichtlicher umformen.

[1] Das Kronecker-Produkt „$\otimes$“ ist dabei für $A\in\mathbb{R}^{n\times m}$ und $B\in\mathbb{R}^{k\times l}$ definiert durch

$$B\otimes A = \begin{pmatrix} a_{11}B & a_{12}B & \cdots & a_{1m}B \\ \vdots & & & \vdots \\ a_{n1}B & a_{n2}B & \cdots & a_{nm}B \end{pmatrix} \in \mathbb{R}^{nk\times ml}$$

Backpropagation-Lernregel

Für jedes $t \in \mathbb{N}$ seien $\epsilon(t) = b(t) - F(w(t), a(t))$ und $\delta_L(t) = A_L(t)\,\epsilon(t)$ und für $l \in \{2, \ldots, L-1\}$

$$\delta_l(t) = A_l(t) W_{l+1}(t) \cdot \delta_{l+1}(t) \tag{BP1}$$

$$\text{und} \qquad w_l(t+1) = w_l(t) + 2\alpha\, u_{l-1}(t) \otimes \delta_l(t) \tag{BP2}$$

Interpretation: Der Fehler $\epsilon(t)$ am Ausgang wird von Lage zu Lage gemäß $\delta_l = A_l W_{l+1} \cdot \delta_{l+1}$ weitergegeben; daher der Name *Backpropagation*.

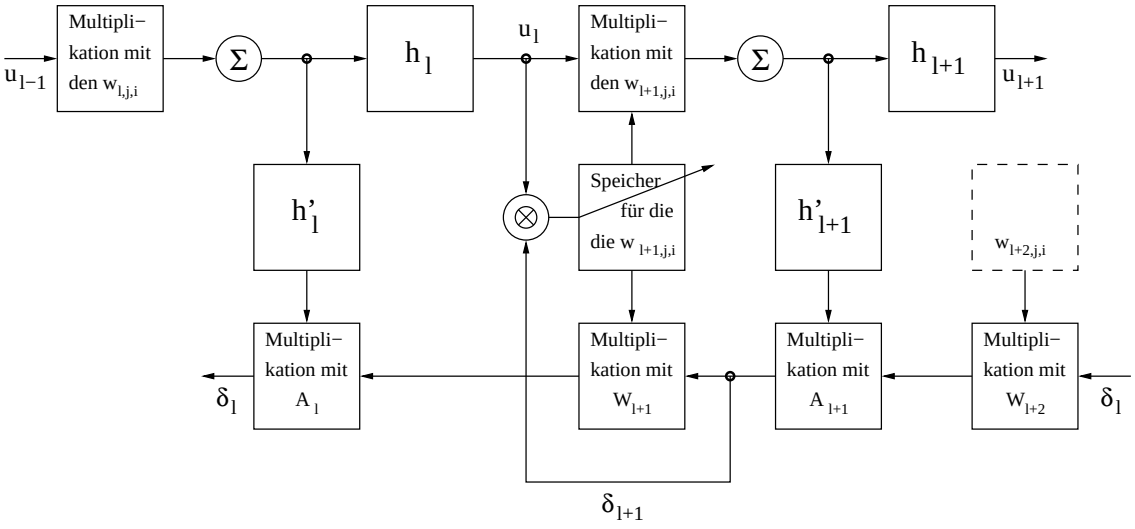
Explizite Summendarstellung

$$\begin{aligned}
\delta_{L,k}(t) &= h'_{L,k}(\langle w_{L,k}(t), u_{L-1}(t)\rangle)\,\epsilon_k(t), \\
\epsilon_k(t) &= b_k(t) - F_k(w(t), a(t)), \\
\text{(BP1)} \qquad \delta_{l,j}(t) &= h'_{l,j}(\langle w_{l,j}(t), u_{l-1}(t)\rangle) \sum_{k=1}^{n_l} w_{l+1,k,j}(t)\,\delta_{l+1,k}(t), \text{ für } l \in \{2, \ldots, L-1\}, \\
\text{(BP2)} \qquad w_{l,j,i}(t+1) &= w_{l,j,i}(t) + 2\alpha\,\delta_{l,j}(t)\,u_{l-1,i}(t)
\end{aligned}$$

Spezialfall: Backpropagation für 3-lagige Netze

$$\begin{aligned}
\delta_3(t) &= A_3(t)\,\epsilon(t) = \begin{pmatrix} h'_{3,1}(\langle w_{3,1}(t), u_2(T)\rangle)\,\epsilon_1(t) \\ \vdots \\ h'_{3,n_3}(\langle w_{3,n_3}(t), u_2(t)\rangle)\,\epsilon_{n_3}(t) \end{pmatrix}, \\
\epsilon(t) &= b(t) - F(w(t), a(t)) = b(t) - u_3(t), \\
w_{3,j,i}(t+1) &= w_{3,j,i}(t) + 2\alpha\,\delta_{3,j}(t)\,u_{2,i}(t), \\
\delta_2(t) &= A_2(t) W_3(t) \cdot \delta_3(t), \text{ also } \delta_{2,j}(t) = h'_{2,j}(\langle w_{2,j}, u_1(t)\rangle) \sum_{k=1}^{n_3} w_{3,k,j}(t)\,\delta_{3,k}(t), \\
w_{2,j,i}(t+1) &= w_{2,j,i}(T) + 2\alpha\,\delta_{2,j}(t)\,u_{1,i}(t).
\end{aligned}$$

Dabei ist $u_1(t) = a(t)$ der Netzinput, $u_3(t) = F(w(t), a(t))$ der Netzoutput und $u_2(t)$ der Vektor der Ausgangswerte der versteckten Neuronen. Übliche Abkürzung: $\text{net}_{l,j} = \langle w_{l,j}, u_{l-1}\rangle$.



4.6.4 Zusammenfassung der Backpropagation-Prozedur

Für jeden Zeitpunkt t tue folgendes

1. Gib den Vektor $a(t)$ in die Netzeingänge und berechne die Ausgabevektoren $u_2(t), \ldots, u_L(t)$ aller Lagen (in der Reihenfolge aufsteigender Indizes)
$$u_1 = a(t) \quad \text{und} \quad u_{l+1} = \left(h_{l+1,1}\left(\langle w_{l+1,1}, u_l\rangle\right), \ldots, h_{l+1,n_{l+1}}\left(\langle w_{l+1,n_{l+1}}, u_l\rangle\right)\right)^T.$$
2. Berechne den Fehlervektor $\epsilon = b(t) - u_L$ und berechne mit der Backpropagationregel (BP1) die Fehlervektoren $\delta_L, \ldots, \delta_2$ für die einzelnen Lagen (in dieser Reihenfolge)
$$\delta_L = A_L \epsilon \quad \text{und} \quad \delta_l = A_l W_l \, \delta_{l+1}.$$
3. Berechne die neuen Gewichte für den Zeitpunkt $t+1$ gemäß der verallgemeinerten Deltaregel (BP2) (in beliebiger Reihenfolge)
$$w_l(t+1) = w_l(t) + 2\alpha \, u_{l-1} \otimes \delta_l.$$

4.6.5 Bemerkungen zum praktischen Einsatz

Die Probleme des LMS-Algorithmus für Adalines treten beim Training von allgemeinen Netzen mit der Backpropagationregel verstärkt auf. Denn es kommen (mindestens) zwei neue Problemquellen hinzu:

1) Die Fehlerfunktion e hat viele lokale Minima, die nicht alle global sein müssen.

2) Die Fehlerfunktion e hat oft in großen Bereichen einen sehr flachen Verlauf. Das ist eine Folge der sigmoiden Aktivierungsfunktionen, deren Wert sich bei großen Argumenten kaum ändert, auch wenn das Argument sich stark ändert. Bekommt ein Neuron an seinen Eingängen große Werte angeboten, so bewirken größere Gewichtsänderungen fast keine Änderungen des Ausgangswertes (außer wenn sich große Komponenten gerade kompensieren).

1) bewirkt, daß mit Gradientenabstieg (und BP ist eine Art stochastischer Gradientenabstieg) nicht immer ein globales Minimum gefunden wird. Oft wird gesagt, die Erfahrung zeige, daß mit BP nur selten ein nicht globales Minimum gefunden werde. Meines Erachtens kann man das nur behaupten, wenn man globale Minima kennt, was nur selten der Fall ist. Vielleicht ist eher folgendes gemeint: Wenn das BP-Training überhaupt konvergiert, so häufig gegen eine Stelle, in der die Fehlerfunktion nur wenig größer als ihr globales Minimum ist.

2) bewirkt, daß das BP-Training oft nur sehr langsam den Fehler verkleinert. Backpropagation-Training ist sehr zeitaufwendig. Oft sind einige 10.000 Schritte notwendig. Bei vielen Anwendungen hat man gar nicht so viele Trainingsdaten. Dann muß man die wenigen mehrmals verwenden. Verwendet man sie zu oft, so kann sich sogenanntes *Overtraining* einstellen, d.h. das Netz stellt sich zu sehr auf die speziellen statistischen Eigenschaften der Trainingsfolge ein, die aber nicht wirklich repräsentativ für die in der Anwendung vorkommenden Daten ist. Trainiert man also mit dieser einen Trainigsfolge zu lange, so wird der Fehler beim praktischen Einsatz des Netzes wieder größer. Man spricht davon, daß die „Verallgemeinerungsfähigkeit“ des Netzes wieder schlechter werde.

Es stellt sich die Frage: Wann soll man mit dem Training aufhören?
Eine allgemeingültige Regel gibt es (wohl) nicht dazu. Hier sind lediglich Heuristiken gefragt. Eine wichtige Rolle spielt der Lernparameter *alpha*, der die Schrittweite bestimmt. Wählt man α groß, so konvergieren Gewichte nicht, sondern oszillieren mehr oder weniger stark. Wählt man α zu klein, so stagniert das Training, sobald man in eines der in 2) beschriebenen flachen Plateaus von e hineingerät. Um aus diesem Dilemma herauszukommen, wurde das bisher beschriebene BP-Training (sogenanntes *Online-Training*) auf viele Weisen modifiziert, von denen wir nur einige skizzieren.

Adaption des Lernparameters α: Man wählt α zeitlich variabel. Zum Beispiel kann man α vergrößern, wenn der Fehler noch relativ groß ist und einige Zeitschritte lang keine wesentliche Verkleinerung des Fehlers eingetreten ist. Dadurch kann man sich schneller über flache Plateaus von e bewegen.

Momentum Training: Die Gewichtsänderung $2\alpha \, u_{l-1}(t) \otimes \delta_l(t)$ in der verallgemeinerten Deltaregel wird ersetzt durch einen Mittelwert über die entsprechenden Terme zu einigen zurückliegenden Zeitpunkten.

Batch Training: Die Gewichtsänderungen $2\alpha\, u_{l-1}(t) \otimes \delta_l(t)$ werden über einen längeren Zeitraum gemittelt. Erst dann werden die Gewichte mit diesem Mittelwert geändert; also nicht zu jdem Zeitschritt.

Wahl der Startwerte für w: Zufällig oder mit Vorwissen?

Veränderung der Anzahl der Neuronen in den versteckten Lagen: Es gibt mehrere Vorschläge zu zwei verschiedenen Vorgehensweisen:

- Man beginnt mit vielen versteckten Neuronen. Neuronen, deren Ausgangswert sich während des Trainings kaum ändert, werden weggelassen. Ebenso wird eines von zwei Neuronen weggelassen, wenn ihre Ausgangswerte stets fast gleich sind.
- Man fügt neue versteckte Neuronen hinzu, wenn das Training in einem lokalen Minimum oder einem flachen Plateau stagniert.

Die Erfahrung zeigt, daß viele versteckte Neuronen die sogenannte „Verallgemeinerungsfähigkeit“ des Netzes vermindern, d.h. es stellt sich zu sehr auf die statistischen Eigenschaften der speziellen Trainingsfolge ein.

Veränderung der Anzahl versteckter Lagen: Es kann vorkommen, daß ein 3-lagiges Netz sehr viele versteckte Neuronen haben muß, um den Fehler recht klein machen zu können, während ein Netz mit mehr Lagen viel weniger versteckte Neuronen braucht. Gibt es Kriterien, wieviele versteckte Lagen man wählen soll? Vergleiche hierzu auch [1].

4.7 Einige Anwendungen

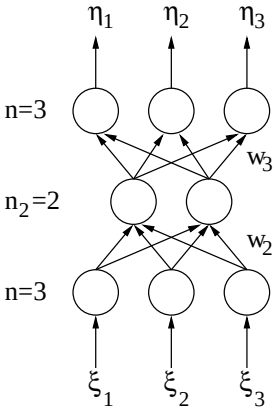
Überall, wo analytisch undurchsichtige Abbildungen auftauchen, wird versucht, diese Abbildungen durch NN zu realisieren, die man mit Backpropagation trainiert. Ein Paradebeispiel war NETtalk, ein Netz, das aus ASCII-Text Phoneme macht, die einem Spracherzeugungschip übergeben werden.

Wir führen noch zwei Anwendungen an, an die man vielleicht nicht gleich denkt.

4.7.1 Datenkompression

Beispiel: Ein Bild mit 512×512 Pixeln werde in 8×8 Blöcke zerlegt. Wenn jedes Pixel einen Grauwert aus $\{0, \ldots, 255\}$ annehmen kann, so ist $\{0, \ldots, 255\}^{8\times 8}$ die Menge aller möglichen Blöcke. Nun werden in natürlichen Bildern nicht alle Blöcke gleich wahrscheinlich vorkommen. Daher kann man auf die Idee kommen, nur die häufigen (oder typischen) Blöcke zu verwenden und die anderen damit zu approximieren. Dadurch könnte man eine Datenkompression (mit Verlust) erreichen.

Ansatz: Sei $B \subset \mathbb{R}^n$ (im Beispiel $n = 8\times 8$, $B = \{0, \ldots, 255\}^{8\times 8}$). Wähle ein 3-lagiges vorwärtsgerichtetes Netz mit n Eingängen, n Ausgängen und n_2 versteckten Neuronen. Es sei $n_2 < n$ und p sei ein Wahrscheinlichkeitsmaß auf B (im Beispiel die Häufigkeitsverteilung der Blöcke). $F(w, \cdot)$ sei die durch das Netz realisierte Abbildung.



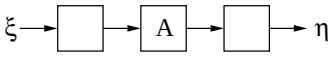
Versuche, das Netz so zu trainieren, daß $F(w, \cdot)$ möglichst gut die Identität auf B approximiert, also $\int_B \|x - F(w,x)\|_2^2\, dp(x)$ minimal.
Sei der Gewichtsvektor $w = (w_1, w_2)^T$ in diesem Sinn optimal. Jedes $x \in B$ kodiert man dann durch den Vektor u_2 der Ausgangswerte der versteckten Neuronen, wenn das Netz mit x gespeist wird.
Die Kodierung wird also durch das Teilnetz aus den beiden ersten Lagen realisiert, also n Eingänge und n_2 Ausgänge. Die Dekodierung wird ebenfalls durch ein 2-lagiges Netz realisiert, dessen erste Lage wie üblich die Eingangswerte nur verteilt und dessen zweite Lage gleich der dritten des obigen Netzes ist; es hat also n_2 Eingänge und n Ausgänge.

Um daraus eine praktisch einsetzbare Kodierung zu machen, wird man die Eingangs- und Ausgangswerte quantisieren (was man in Digitalrechnern sowieso muß). Sofern man die Ausgangswerte der versteckten Neuronen (also die Komponenten der Codewörter) nicht mit höherer Auflösung quantisiert als die Eingangswerte des Netzes, erhält man eine Kompressionswirkung, weil $n_2 < n$.

4.7.2 Lösung von Aufgaben aus der Matrizenrechnung

Invertierung von Matrizen

Sei $A \in \mathbb{R}^{n\times n}$ eine nicht singuläre Matrix. Wähle ein 3-lagiges Netz, das in jeder Lage n Neuronen hat. Alle Neuronen seien linear, d.h. ihre Aktivierungsfunktion sei die Identität. Die Gewichte der zweiten Lage seien so gewählt, daß gerade die Multiplikation mit der Matrix A durchgeführt wird; die Gewichtsvektoren $w_{2,1}^T, \ldots, w_{2,n}^T$ seien also gerade die Zeilen von A.



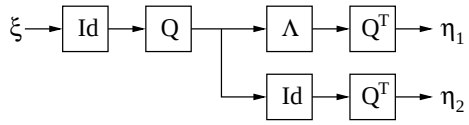
Das Netz wird nun nur durch Verändern der Gewichte der dritten Lage so trainiert, daß es möglichst gut die identische Abbildung realsisiert. Die Gewichte der dritten Lage sind dann näherungsweise die Koeffizienten von A^{-1}; $w_{3,j}^T$ entspricht der j-ten Zeile.

Die Güte der Konvergenz hängt natürlich von der Wahl der Trainingsfolge ab.

Hauptachsentransformation

Gegeben sei eine symmetrische Matrix $A \in \mathbb{R}^{n\times n}$. Gesucht ist eine orthogonale Matrix $Q \in O(n, \mathbb{R})$ und eine Diagonalmatrix $\Lambda \in \mathbb{R}^{n\times n}$, so daß $A = Q\Lambda Q^T$.

Wir verwenden wieder ein vorwärtsgerichtetes Netz aus linearen Neuronen. Zur Abkürzung schreiben wir in jede Lage die Matrix, deren Koeffizienten durch die Gewichte dieser Lage approximiert werden sollen.



Das Netz besteht aus 4 Lagen; die 3. und die 4. Lage zerfallen in zwei Teile; entsprechend wird

$$\eta = (\eta_1, \eta_2) = (F_1(w, \xi), F_2(w, \xi)) \quad \text{und} \quad w = (Q, \lambda_1, \ldots, \lambda_n)$$

gesetzt.

Der Gewichtsvektor w soll so gewählt werden, daß $F_1(w, \xi) = A\xi$ und $F_2(w, \xi) = \xi$ für alle ξ. Dies versucht man zu erreichen, indem man die Fehlerfunktion $\|A\xi - F_1(w, \xi)\|_2^2 + \|\xi - F_2(w, \xi)\|_2^2$ mit Hilfe des LMS-Algorithmus angewendet auf eine Trainingsfolge $(\xi(t))_t$ zu minimieren versucht. Die Berechnung des Gradienten erfolgt wie in 4.6; allerdings ist zu berücksichtigen, daß in den mit Q und Q^T bezeichneten Lagen dieselben Gewichte verwendet werden.

In ähnlicher Weise lassen sich andere Aufgaben lösen wie die LU-Zerlegung, QR-Faktorisierung und Singulärwertzerlegung von Matrizen.

4.8 Eingangscodierung mit Principal Component Analysis (PCA)

Damit Neuronale Netze in Anwendungen vorteilhaft eingesetzt werden können, ist es oft wesentlich, dass die Eingabeparameter geeignet codiert sind. Insbesondere bei Klassifikationsaufgaben ist dies wichtig, für die Darstellung der Merkmalsvektoren ein Koordinatensystem so zu wählen, dass die einzelnen Komponenten der Vektoren Teilmerkmalen entsprechen, die weitgehend unabhängig voneinander sind. Man wählt dann für jede Komponente einen Netzeingang und kann sicher sein, nicht unnötig viele oder zu wenig Eingänge gewählt zu haben.

Häufig sind die in einer Anwendung möglichen Eingangsparameter Vektoren in einem $\mathbb{R}^n$, die nicht beliebig im Raum verteilt sind, sondern sich zumindest grob ellipsoidförmig verteilt sind. Betrachtet man die Trainingsvektoren als Massepunkte im $\mathbb{R}^n$, so kann man für diese Masseverteilung das Trägheitsellipsoid mit dem Schwerpunkt als Mittelpunkt berechnen. Die Hauptachsen des Ellipsoid stehen senkrecht aufeinander. Bezüglich des Koordinatensystems aus diesen Hauptachsen mit Ursprung im Schwerpunkt sind unterschiedliche Komponenten der Traainingsvektoren nicht mehr korreliert.

Das Trägheitsellipsoid hat in den einzelnen Hauptachsenrichtungen meist unterschiedliche Dicken (außer im Falle einer Sphäre). In Richtungen mit großer Dicke variieren die Trainingsvektoren stärker als in Richtungen mit kleiner Dicke. Deshalb lässt man bei den Eingabevektoren des Netzes die Komponenten weg, die Richtungen mit kleiner Dicke entsprechen. Das Neuronale Netz hat dann entsprechend weniger Eingänge und wird weniger komplex.

Mathematisch ausgedrückt berechnet man für die Trainingsvektoren in den ursprünglichen Koordinaten die Autokorrelationsmatrix R wie schon in Kapitel 3 bei der Fehlerfunktion für ein Adaline. Sie ist symmetrisch. Deshalb gibt es eine orthogonale Matrix Q, so dass $\Lambda = QRQ^T$ Diagonalgestalt hat. Der durch Q definierte Koordinatenwechsel heißt Hauptachsentransformation von R. Die Elemente λ_j auf der Diagonale von Λ sind die Eigenwerte von R; die neuen Koordinatenachsen sind die dazu gehörenden Eigenrichtungen. Das Trägheitsellipsoid ist, ausgedrückt in dem neuen Koordinatensystem, die Menge der Punkte $x \in \mathbb{R}^n$, die die Gleichung $1 = x^T \Lambda x = \sum_{j=1}^{n} \lambda_j x_j^2$ erfüllen.

Kapitel 5

Radial Basis Functions

Bei der Untersuchung von L^2-Approximationsaufgaben ist es naheliegend, Hilbertraummethoden einzusetzen, insbesondere die Entwicklung nach Orthonormalbasen (ONB). Dies wurde in zahllosen Anwendungen auch getan; man denke nur an die Fouriertechniken in der Signalverarbeitung.

Die Entwicklungen nach ONB haben den riesigen Vorteil, daß man die Koeffizienten leicht berechnen kann: Ist H ein separabler Hilbertraum und $(\varphi_j)_{j\in\mathbb{N}}$ eine ONB in H, so gilt für jedes $f \in H$: $f = \sum_{j=1}^{\infty} \langle f, \varphi_j \rangle \varphi_j$.

Meist handelt es sich in Anwendungen um Funktionenräume, und eine gute L^2-Approximation ist nicht unbedingt eine gute Approximation in einer anderen Norm, beispielsweise der Supremumsnorm (man denke z.B. an das Gibbsche Phänomen). Will man eine Funktion approximativ durch Stichproben aus ihrem Graph lernen, so wünscht man meist, daß die Funktion dort, wo die Stichproben dichter liegen, auch besser approximiert wird (z.B. in der Supremumsnorm). Das ist mit vielen ONB nicht gut zu erreichen, weil die Basisfunktionen recht große Träger haben. Was man braucht, sind Basen aus Funktionen mit kompaktem Träger, am besten sogar so, daß beliebig kleine Träger vorkommen. Ein Ansatz dafür sind die sogenannten *Wavelets*.

Ein naiver Ansatz besteht darin, „glockenähnliche Hütchenfunktionen“ zu wählen, deren Zentrum und Weite variabel sind. Mit Linearkombinationen solcher Funktionen kann man stetige Funktionen sogar recht gut in der Supremumsnorm approximieren; nur ist es nicht so leicht, die Entwicklungskoeffizienten zu berechnen, weil die Funktionen nur dann orthogonal sind (in $L^2(\mathbb{R})$), wenn ihre Träger disjunkt sind.

5.1 Bezeichnung. Unter einer *radialen Funktion* φ wollen wir eine Funktion $\varphi : \mathbb{R}^n \to \mathbb{R}$ verstehen, die die Gestalt $\varphi(x) = \phi(\|x - z\|_2)$ hat, wobei ϕ eine Funktion $[0, \infty[\to \mathbb{R}$ ist und $z \in \mathbb{R}^n$ das *Zentrum* von φ heißt.

Beispiele.

a) $\phi(r) = \exp(-r^2/2\sigma^2)$.

b) $\phi(r) = \sqrt{r^2 + \sigma^2}^{-1}$.

c) $\phi(r) = r$.

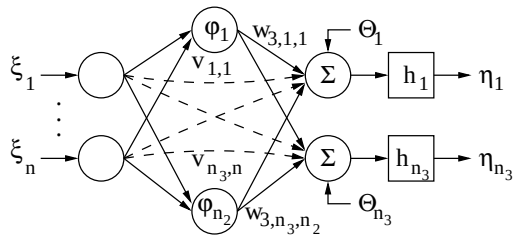
d) $\phi(r) = r^3$.

e) $\phi(r) = \sqrt{r^2 + \sigma^2}$.

Gegebenfalls werden die Funktionen „abgeschnitten“, so daß sie kompakten Träger haben.

5.2 Bezeichnung. Ein *RBF-Netz* sei ein 3-lagiges vorwärtsgerichtetes Netz mit folgenden Eigenschaften:

- die Eingangsneuronen realisieren die identische Abbildung; sie dienen nur zum Verteilen der Eingangswerte an die Neuronen der versteckten Lage.
- jedes Neuron der versteckten Lage realisiert eine radiale Funktion.



$\eta_k = h_k\left(\sum_{i=1}^{n_2} w_{3,k,i}\varphi_i(\xi) + \Theta_k\right)$.

$\Theta_1, \ldots, \Theta_n \in \mathbb{R}$ (*bias*), h_k Aktivierungsfunktion.

Manchmal läßt man auch noch direkte Verbindungen von der ersten Lage zur dritten zu; ihre Gewichte seien mit $v_{1,1}, \ldots, v_{1,n}, v_{2,1}, \ldots, v_{n_3,n}$ bezeichnet. Dann hat die vom Netz realisierte Abbildung die Gestalt

$$\eta_k = h_k\left(\sum_{i=1}^{n_2} w_{3,k,i}\,\varphi_i(\xi) + \sum_{j=1}^{n} v_{k,j}\,\xi_j + \Theta_k\right).$$

Jede der L^2-Approximationsaufgaben I-III in 4.5 läßt sich entsprechend für RBF-Netze formulieren. Es ist jedoch üblich und sinnvoll, nicht nur die Gewichte zu variieren, um eine gute Approximation zu erreichen.

5.3 Varianten der Approximationsaufgaben

5.3.1 Aufgabe I

Die Anzahl n_2 der versteckten Neuronen und ihre radialen Funktionen $\varphi_1, \ldots, \varphi_{n_2}$ seien fest gewählt. Minimiere den Fehler allein durch die Variation der Gewichte $w_{3,k,i}$ und eventuell $v_{k,i}$ und Θ_k.

5.3.2 Aufgabe II

n_2 sei fest gewählt. Minimiere den Fehler durch Variation der Gewichte w, v und Θ und der radialen Funktionen $\varphi_1, \ldots, \varphi_{n_2}$ innerhalb einer vorgegebenen Familie.

5.3.3 Aufgabe III

Minimiere den Fehler durch Variation von n_2, w, v, Θ und $\varphi_1, \ldots, \varphi_{n_2}$.
Bemerkungen. Aufgabe I ist einfach das Training eines Adalines, nämlich des Ausgangsneurons. Die Wahl von n_2 und der $\varphi_1, \ldots, \varphi_{n_2}$ erfolgt aufgrund von Vorwissen.

Bei Aufgabe II hat die vorgegebene Familie radialer Funktionen oft die Gestalt $\phi(\|x - z\|)$ mit $z \in \mathbb{R}^n$, d.h. außer den Gewichten $w_1, \ldots, w_{n_2}$ dürfen auch die Zentren $z_1, \ldots, z_{n_2}$ variiert werden. Besitzt ϕ eine Gestalt wie zum Beispiel $\exp(-r^2/2\sigma^2)$, so läßt man manchmal auch noch eine Veränderung der Werte σ zu.

Während man bei BP-trainierten Netzen mit sigmoiden Aktivierungsfunktionen oft keinen Anhaltspunkt hat, wieviele versteckten Neuronen man wählen soll, wird bei RBF-Netzen tatsächlich häufig diese Anzahl n_2 mitgelernt. Das geht natürlich nicht mit irgendeiner Art von Gradientenabstieg.

5.4 Festlegung der Anzahl n_2 der versteckten Neuronen

Die Grundlegende Idee ist es, dort wo die Trainingseingangswerte dichter liegen, mehr Zentren der radialen Funktionen hinzulegen.

5.4.1 Erster Fall

Ist die Anzahl der Trainingsdaten klein, so wählt man einfach jeden Trainingseingangswert als Zentrum der radialen Funktion eines versteckten Neurons. (In diesem Fall ist ein BP-Training meist nicht sehr sinnvoll.)

5.4.2 Zweiter Fall

Ist die Anzahl der Trainingsdaten groß, so führt man eine Clusterung der Trainingseingangswerte durch — d.h. man teilt sie in Haufen dicht liegender Punkte. Je nachdem wie stark die Trainingseingangswerte variieren, wählt man für jedes Cluster ein oder mehrere Zentren für radiale Funktionen (z.B. den Schwerpunkt des Clusters). Gegebenenfalls wählt man die Breite σ der Funktionen so, daß ihre Träger sich nicht allzu sehr überschneiden, aber das Cluster überdecken.

Hat man n_2 festgelegt, so kann man die optimalen Werte der Zentren, Breiten und Gewichte durch irgendeine Art von Gradientenabstieg z.B. mit dem LMS-Algorithmus zu bestimmen versuchen.

5.5 Lernmethoden

Zunächst wählt man die Anzahl der versteckten Neuronen und ihre radialen Funktionen wie im letzten Abschnitt beschrieben. Anschließend versucht man durch Verändern der Gewichte $w_{3,j,i}$, der Schwellen Θ_j und eventuell auch der $v_{j,i}$ sowie der Zentren $z_1, \ldots, z_{n_2}$ und Weiten $\sigma_1, \ldots, \sigma_{n_2}$ die Fehlerfunktion zu minimieren. Dazu verwendet man irgendeine Art von Gradientenabstieg, meist irgendeine Modifikation des LMS-Algorithmus. Ist $\eta = F(w, v, \Theta, z, \sigma, \xi)$ die von Netz realisierte Abbildung und hat die Fehlerfunktion die Gestalt

$$e(w, v, \Theta, z, \sigma) = \mathcal{E}(\|Y - F(w, v, \Theta, z, \sigma, X)\|_2^2),$$

so muß man also $\nabla e = \left(\frac{\partial e}{\partial w}, \frac{\partial e}{\partial v}, \frac{\partial e}{\partial \Theta}, \frac{\partial e}{\partial z}, \frac{\partial e}{\partial \sigma}\right)^T$ berechnen. Mit der Kettenregel folgt

$$\frac{\partial e}{\partial w} = -2\mathcal{E}\left((Y - F(\ldots, X))^T \frac{\partial F}{\partial w}\right), \ldots, \frac{\partial e}{\partial \sigma} = -2\mathcal{E}\left((Y - F(\ldots, X))^T \frac{\partial F}{\partial \sigma}\right).$$

Nun ersetzt man wieder den wirklichen Gradienten durch einen Schätzwert; im einfachsten Fall läßt man den Erwartungswertoperator $\mathcal{E}$ weg. Dann erhält man als Lernregel für eine Trainingsfolge $(a(t), b(t))$

$$w_{3,j,i}(t+1) = w_{3,j,i}(t) + 2\alpha \left(\frac{\partial F}{\partial w_{3,j,i}}(\ldots, a(t))\right)^T (b(t) - F(\ldots, a(t)))$$

entsprechend für die anderen Parameter. Hat das Netz nur einen Ausgang, so ist F skalar und das Transponieren ist überflüssig. Der Lernparameter α wird oft für die verschiedenen Gruppen von Parametern w, v, Θ, z, σ unterschiedlich gewählt.

Beachte, daß hier die Komponenten von DF nicht durch eine Backpropagationmethode berechnet werden können. Die partiellen Ableitungen nach den unterschiedlichen Parametern können sehr verschieden aussehen.

Ansonsten gelten aber die Überlegungen, die wir zum BP-Lernen im vorigen Abschnitt angestellt haben, auch hier. Insbesondere werden auch die dort genannten Varianten des Momentum Training und Batch Training eingesetzt; angeblich zeigt das Batch Training in der Praxis gegenüber dem Online Training Vorteile. Die Wahl der Startwerte, zumindest für die Parameter z und σ, ist nicht ganz unkritisch.

In dem NN-Simulator „SNNS“ gibt es dafür mehrere unterschiedliche Initialisierungsroutinen, die teilweise recht aufwendig sind. Es lohnt sich im Manual den Abschnitt 8.10 zu lesen.

5.6 Vorteile gegenüber Netzen mit BP-Training

RBF-Netze lassen sich meist sehr viel schnelle als BP-Netze trainieren, vor allem dann wenn nur die Gewichte der Ausgangslage variabel sind. Sie lassen sich auch dann trainieren, wenn so wenige Trainingsdaten vorliegen, daß das BP-Training von Netzen versagt.Der Geschwindigkeitsvorteil kann einige Zehnerpotenzen betragen.

5.7 Normierung der Eingangswerte

Verwendet man Gaußfunktionen als radiale Funktionen der versteckten Neuronen, so wird das Lernen der Zentren (z.B. durch Clusterung) erleichtert, wenn die Eingangsvektoren des Netzes stets die euklidische Länge 1 haben und ebenso die Zentren. Denn es gilt $\|\xi - z\|^2 = \langle \xi - z, \xi - z\rangle = \|\xi\|^2 + \|z\|^2 - 2\langle \xi, z\rangle = 2 - 2\langle \xi, z\rangle$, folglich $\exp\left(-\frac{\|\xi - z\|^2}{2\sigma^2}\right) = \exp\left(\frac{\langle \xi, z\rangle - 1}{\sigma^2}\right) = g(\langle \xi, z\rangle)$, wobei $g(t) = \exp\left(\frac{t-1}{\sigma^2}\right)$.

Das bedeutet, daß sich die versteckten Neuronen als McCulloch-Pitts-Neuronen mit der Aktivierungsfunktionen g auffassen lassen. Das Netz hat dann die Gestalt wie in Abschnitt 4 und läßt sich auch mit BP trainieren. Es realisiert die Abbildung mit den Komponenten $\eta_k = h_k\left(\sum_{j=1}^{n_2} w_{3,k,j}\, g(\langle \xi, z_j\rangle)\right)$ die Zentren z_j spielen also die Rolle der Gewichte der versteckten Neuronen.

Weil sowohl die Zentren z_j als auch die Eingangsvektoren in der Einheitssphäre S^{n-1} liegen, kann man $\langle \xi, z_j\rangle$ als Cosinus des Winkels zwischen z_j und ξ deuten; $g(\langle \xi, z_j\rangle)$ ist somit nur vom sphärischen Abstand zwischen ξ und z_j abhängig und zwar monoton fallend.

Bemerkung. Es gibt auch Varianten von RBF-Netzen, die den Vektor der Ausgangswerte der versteckten Lage normieren. Seine Komponenten sind dann gerade diejenigen einer Partition der Eins.

5.8 Die Grossberg-Schicht

Während die Normierung eines Vektors mit Digitalrechnern kein Problem ist, ist unklar, wie sie in Analogrechnern oder biologischen Systemen durchgeführt werden kann. Dort ist sie aber gerade besonders nötig, weil die Systeme meist nur eine eingeschränkte Dynamik haben und bei zu großen Werten Sättigungserscheinungen auftreten, die die Funktionsweise beeinträchtigen (*noise saturation dilemma*). Grossberg hat sich 1982 eine Neuronenschicht ausgedacht, die approximativ einen Eingabevektor normieren kann, allerdings in der L^1-Norm. Seine Lage von Neuronen realisiert nicht einfach eine Abbildung, sondern ein gewöhliches Differentialgleichungssystem (mit der Zeit als Variable); die Lösung wird an den Ausgängen ausgegeben und strebt für $t \to \infty$ gegen eine approximativ normierten Vektor.

Die Lage habe n Neuronen; $\xi_1, \ldots, \xi_n$ seien die Eingangswerte; es sei $\xi_i \geq 0$ für jedes i, $\|\xi\|_1 = \sum_{i=1}^n \xi_i$, $\alpha, \beta > 0$. $u_1(t), \ldots, u_n(t)$ seien die Ausgangswerte der Neuronen zum Zeitpunkt t. Sie mögen folgendem Dgl-System genügen:

$$\dot{u}_i = -\alpha u_i + (\beta - u_i)\xi_i - u_i \sum_{j \neq i} \xi_j \quad \text{für } i = 1, \ldots, n.$$

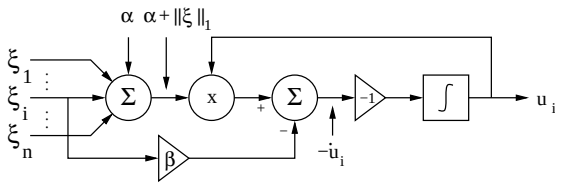
Beachte, daß die Gleichungen nicht gekoppelt sind.

Durch Umformung der rechten Seite erhält man

$$\dot{u}_i = -(\alpha + \|\xi\|_1)u_i + \beta\xi_i.$$

Ist α oder $\alpha + \|\xi\|_1$ groß, so klingt der homogene Lösungsanteil sehr schnell ab, und der Ausgangswert des i-ten Neurons liegt nahe bei $\frac{\beta\xi_i}{\alpha + \|\xi\|_1}$. Dieser Wert ist zwar nicht unabhängig von $\|\xi\|_1$ ändert sich aber für große $\|\xi\|_1$ kaum noch und bleibt vor allem beschränkt.

Realisierung in analoger Hardware



Kapitel 6

Kompetitive Netze

6.1 MAXNET oder „*The winner takes all*“

Ein *MAXNET* oder *WTA*-Netz ist ein Netz mit ebenso viel Ausgängen $\eta_1, \ldots, \eta_n$ wie Eingängen $\xi_1, \ldots, \xi_n$, so daß für jedes j gilt:

$$\eta_j = \begin{cases} 1 & , \xi_j = \max\{\xi_1, \ldots, \xi_n\} \\ 0 & , \text{sonst} \end{cases} .$$

Modifikationen: Statt 1 kann natürlich ein anderer Wert genommen werden, der $\neq 0$, aber nicht für alle j dergleiche sein muß. Man kann auch verlangen, daß ξ_j das einzige maximale Element ist.

Die Realisierung auf Digitalrechnern ist natürlich trivial und wird nicht in Gestalt eines Netzes erfolgen. Eine Realisierung in Gestalt eines Netzes erfordert dagegen Verbindungen zwischen den Neuronen derselben Schicht und ist nicht so trivial. Eine Realisierung mit analoger Hardware läßt sich ähnlich wie bei einer Grossberg-Schicht erreichen, allerdings durch ein System gekoppelter Differentialgleichungen. Statt mit kontinuierlicher Zeit kann man auch mit diskreter Zeit arbeiten und die folgende Rekursionsgleichung verwenden.

Es sei für $j = 1, \ldots, n$ $\quad u_j(0) = \xi_j$,
und für $t \in \mathbb{N}_0$ $\quad u_j(t+1) = \varphi\left(u_j(t) - \varepsilon \sum_{i \neq j} u_i(t)\right)$,

wobei $\varepsilon > 0$ und $\varphi(u) = u \cdot \chi_{[0,\infty[}(u)$, also $\varphi(u) = 0$ für $u \leq 0$ und $\varphi(u) = u$ für $u > 0$. Oft wird φ leicht modifiziert, z.B.

$$\varphi(u) = \begin{cases} 0 & , \text{für } u \leq 0 \\ \alpha u & , \text{für } u > 0 \text{ mit } \alpha > 0 \end{cases} \quad \text{oder} \quad \varphi(u) = \begin{cases} 0 & , \text{für } u \leq 0 \\ \alpha u & , \text{für } 0 < u < \alpha^{-1} \\ 1 & , \text{für } u \geq \alpha^{-1} \end{cases} ;$$

diese Funktion heißt *threshold logic activationfunction*.

Interpretation: Gegeben seien n Neuronen, die einen Ausgangswert u_j haben, der zum Zeitpunkt 0 mit ξ_j übereinstimmt und in den folgenden Zeitpunkten sich gemäß obiger Rekursionsgleichung entwickelt. Die Ausgangswerte u_i mit $i \neq j$ wirken sich hemmend aus, falls sie nicht 0 sind, während sich u_j fördernd auswirkt.

6.2 Netze zur Vektorquantisierung

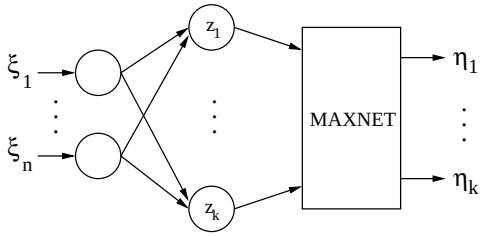
d sei eine Metrik im $\mathbb{R}^n$, $z_j \in \mathbb{R}^n$. $U_j = \{x \in \mathbb{R}^n \mid d(x, z_j) < d(x, z_i) \text{ für } i \neq j\}$. Wähle V_j so, daß $U_j \subset V_j \subset \overline{U_j}$ und $\cup_j V_j = \mathbb{R}^n$. Die Familie $(V_j)_j$ nennt man eine *Voronoi-Parkettierung* des $\mathbb{R}^n$; die V_j *Voronoi-Zellen*.

Ist d die euklidische Metrik, so werden die V_j durch Hyperebenenstücke begrenzt. Sind die z_j Punkte eines regelmäßigen Gitters, so sind die Voronoi-Zellen Quader.

Anwendung: Kodierung mit Kompression.
Man kann irgendwelche Muster durch Merkmalsextraktion im $\mathbb{R}^n$ beschreiben (z.B. 8×8-Blöcke aus Grauwertbildern durch Vektoren aus $\mathbb{R}^{64}$). Um die Datenmenge zu reduzieren, möchte man nur mit endlich vielen typischen Merkmalsvektoren $z_1, \ldots, z_k$ weiterarbeiten. Dafür will man jeden beliebigen Merkmalsvektor $\xi \in \mathbb{R}^n$ das (oder ein) ähnlichstes z_j zuordnen. Wir setzen voraus, daß sich zwei Vektoren umso ähnlicher sind, je kleiner ihr Abstand bezüglich der Metrik d ist. Man will also für jedes $\xi \in \mathbb{R}^n$ entscheiden, in welcher Voronoi-Zelle V_j es liegt.

Die Abbildung $\mathbb{R}^n \to \{1, \ldots, k\}$, $\xi \mapsto j$ mit $\xi \in V_j$ nennt man die zur Voronoi-Zerlegung gehörende *Vektorquantisierung*. Sie läßt sich durch ein Netz folgender Gestalt realisieren:

Man nimmt eine Eingangslage aus n Neuronen, eine zweite Lage aus k Neuronen, die die Funktionen $e^{-d(\xi,z_j)}$ (oder $e^{-d(\xi,z_j)^2}$) realisieren, und schaltet ein MAXNET dahinter.

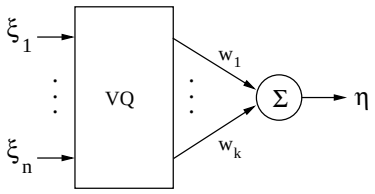


$$\eta_j = \begin{cases} 1 & \text{, wenn } \xi \in V_j \\ 0 & \text{, sonst} \end{cases} = \chi_{V_j}(\xi).$$

Der Ausgangswert der Vektorquantisierung ist also kodiert, und zwar entspricht j dem j-ten kanonischen Einheitsvektor.

6.3 Netze, die Treppenfunktionen realisieren

Seien $z_1, \ldots, z_k \in \mathbb{R}^n$, $V_1, \ldots, V_k$ eine zugehörige Voronoi-Zerlegung (bezüglich einer Metrik). VQ sei ein Netz, das die entsprechende Vektorquantisierung durchführe. Für jeden Eingangsvektor ξ habe nur ein Ausgang j mit $\xi \in V_j$ den Wert 1, die anderen alle den Wert 0. Hinter VQ sei ein lineares Neuron mit den Gewichten $w_1, \ldots, w_k$ geschaltet.



Dann realisiert das Netz die Treppenfunktion

$$\eta = \sum_{j=1}^{k} w_j \, \chi_{V_j}(\xi).$$

Anwendung: Approximation von Funktionen
Sei $K \subset \mathbb{R}^n$ (meist kompakt) und $f : K \to \mathbb{R}$ eine Funktion. Damit obige Treppenfunktion als Approximation von f verwenden zu können, wird man für w_j einen Wert wählen, der für f auf V_j repräsentativ ist (wenn es so einen gibt) z.B. den Mittelwert $\int_{V_j \cap K} f(x)\, dx$.

Problem: Selbst wenn f stetig ist, braucht man für eine ausreichend gute Approximation sehr viele Voronoi-Zellen, also sehr viele Neuronen. Also ist eine geschickte Wahl der Voronoi-Zerlegung wesentlich. Aber wie?

6.4 Netze zur Clusterung von Daten

Bisher ist uns schon mehrfach das Problem begegnet, eine Punktmenge $M \subset \mathbb{R}^n$ so in Teilmengen $M_1, \ldots, M_k$ aufzuteilen, daß die Punkte in jedem M_i „enger miteinander benachbart“ als mit den Punkten aus den M_j mit $j \neq i$; diese Aussage muß natürlich noch präzisiert werden. Die übliche Interpretation ist die, daß die Punkte in M Merkmalsvektoren von Mustern sind, die sich umso ähnlicher sind, je näher ihre Merkmalsvektoren zusammen liegen (bezüglich der Metrik d auf $\mathbb{R}^n$). Man will nun M in Haufen von einander ähnlichen Vektoren einteilen, so daß aber die Vektoren aus verschiedenen Haufen nicht sehr ähnlich sind.

Damit das Problem nicht trivial wird, darf die Anzahl der Haufen nicht beliebig gewählt werden dürfen. Wir werden voraussetzen, daß die Anzahl k fest vorgegeben ist und i.a. $k \ll \#M$.

Wie sollen die Haufen beschrieben werden? In unserem Kontext läge es nahe, sie durch lineare oder polynomiale Ungleichungen, also als semi-algebraische Mengen zu beschreiben. Wir wollen jedoch eine in gewissem Sinn noch einfachere Beschreibung verwenden, indem wir für jeden Haufen M_i ein Zentrum $z_i \in \mathbb{R}^n$ wählen und verlangen, daß $M_i = M \cap V_i$, wobei $V_1, \ldots, V_k$ eine Voronoi-Zerlegung bezüglich $z_1, \ldots, z_k$ und der Metrik d ist.

Es gibt nun mehrere Möglichkeiten, die oben vage formulierten Wünsche an eine Aufteilung in Haufen zu präzisieren. Dabei gibt es wieder zwei grundsätzliche Sichtweisen:

a) Gegeben sei eine Folge von Punkten $(x_j)_{j=1,\ldots,N}$ im $\mathbb{R}^n$.

b) Gegeben sei eine Zufallsvariable X mit Werten im $\mathbb{R}^n$ und Verteilung p.

Wir setzen voraus, daß die Verteilung p von X eine Dichte (bezüglich des Lebesguemaßes auf $\mathbb{R}^n$) hat, damit die Ränder von Voronoi-Zellen Nullmengen sind. d sei eine Metrik, die die euklidische Topologie (aber nicht notwendig die übliche uniforme Struktur) erzeugt.

Sind $z_1, \ldots, z_k \in \mathbb{R}^n$ und $U_i = \{x \in \mathbb{R}^n \mid d(x, z_i) < d(x, z_j) \text{ für jedes } j \neq i\}$, so gilt für jede Voronoi-Zerlegung $V_1, \ldots, V_k$ zu $z_1, \ldots, z_k$, daß $U_i \subset V_i \subset \overline{U_i}$, also $p(U_i) = p(V_i) = p(\overline{U_i})$.

Für $Z \subset \mathbb{R}^n$ und $x \in \mathbb{R}^n$ sei $d(x, Z) = \inf\{d(x, z) \mid z \in Z\}$.

Cluster-Aufgaben

Sei $k \in \mathbb{N}$ gegeben.

6.4.1 Aufgabe 1

a) Finde $Z \subset \mathbb{R}^n$ mit $\#Z = k$, so daß $\sum_j d(x_j, Z)^2$ minimal ist.

b) Finde $Z \subset \mathbb{R}^n$ mit $\#Z = k$, so daß $\int_{\mathbb{R}^n} d(x_j, Z)^2 \, dp(x)$ minimal ist.

6.4.2 Aufgabe 2

a) Finde eine Voronoi-Zerlegung $V_1, \ldots, V_k$ mit Zentren $z_1, \ldots, z_k \in \mathbb{R}^n$, so daß in jedem V_i gleich viele Glieder der Folge $(x_j)_j$ liegen, also $\#\{j \in \{1, \ldots, N\} \mid x_j \in V_i\} = \frac{N}{k}$ für jedes $i = 1, \ldots, k$.

b) Finde eine Voronoi-Zerlegung $V_1, \ldots, V_k$ mit Zentren $z_1, \ldots, z_k \in \mathbb{R}^n$, so daß für jedes $i = 1, \ldots, k$

$$p(V_i) = \tfrac{1}{k} \text{ gilt.}$$

Weil die zweite Aufgabe oft nicht exakt lösbar ist, kann man eine „schwächere“ Aufgabe stellen:

6.4.3 Aufgabe 3

Finde eine Voronoi-Zerlegung $V_1, \ldots, V_k$, so daß

a) $\displaystyle\sum_{i=1}^{k} \left(\#\{j \mid x_j \in V_i\} - \tfrac{N}{k}\right)^2$ minimal ist.

b) $\displaystyle\sum_{i=1}^{k} \left(p(V_i) - \tfrac{1}{k}\right)^2$ minimal ist.

Bemerkungen.

1. Wie schon oft erwähnt, kennt man in praktischen Anwendungen das Wahrscheinlichkeitsmaß p nicht und muß $p(V_i)$ aus unabhängigen Stichproben von X schätzen; man landet dann bei der entsprechenden Aufgabe a).
2. Aufgabe 1 ist eine klassische Cluster-Aufgabe, zu deren Lösungen es Algorithmen gibt.

Lösungsansatz zu 6.4.1: Stochastischer Gradientenabstieg mit dem LMS-Schätzwert

Wir setzen jetzt voraus, daß d die euklidische Metrik sei.

Die Fehlerfunktionen

$$e_a(z_1,\ldots,z_k)=\sum_{j=1}^{N} d(x_j,\{z_1,\ldots,z_k\})^2 \quad \text{bzw.} \quad e_b(z_1,\ldots,z_k)=\int_{\mathbb{R}^n} d(x,\{z_1,\ldots,z_k\})^2\,dp(x)$$

sind nicht überall differenzierbar. Aber ein Term $d(x,\{z_1,\ldots,z_k\})^2$ ist überall dort differenzierbar, wo es nur ein i mit $d(x,z_i)=\min\{d(x,z_1),\ldots,d(x,z_k)\}$ gibt, denn dort gilt ja $d(x,\{z_1,\ldots,z_k\})^2=\|x-z_i\|^2$. Liegt also kein x_j auf dem Rand einer Voronoi-Zelle $V_1,\ldots,V_k$ zu $z_1,\ldots,z_k$, so gilt

$$\frac{\partial}{\partial z_l}e_a(z_1,\ldots,z_k)=\sum_{i=1}^{k}\sum_{j:x_j\in V_i}\frac{\partial}{\partial z_l}\|x_j-z_i\|^2=\sum_{j:x_j\in V_l}-2\cdot(x_j-z_l)^T=-2\cdot\sum_{j=1}^{N}(x_j-z_l)^T\chi_{V_l}(x_j).$$

Um bezüglich einer aus (x_j) gebildeten Trainingsfolge $(x(t))_{t\in\mathbb{N}}$ einen stochastischen Gradientenabstieg zu machen, nimmt man als LMS-Schätzwert zum Zeitpunkt t

$$\widehat{\nabla e_a}(t)=-2\begin{pmatrix}(x(t)-z_1(t))\,\chi_{V_1}(x(t))\\ \vdots\\ (x(t)-z_k(t))\,\chi_{V_k}(x(t))\end{pmatrix}.$$

Man läßt also wieder die Summe weg.

6.4.4 Kohonens Lernregel

$(x(t))_{t\in\mathbb{N}}$ sei eine Folge im $\mathbb{R}^n$. $V_1(1),\ldots,V_k(1)$ sei eine Voronoi-Zerlegung mit den Zentren $z_1(1),\ldots,z_k(1)$, $\alpha\in]0,1]$. Für $t\in\mathbb{N}$ und $i\in\{1,\ldots,k\}$ sei

$$z_i(t+1)=z_i(t)+\alpha(x(t)-z_i(t))\cdot\chi_{V_i(t)}(x(t))$$

und $V_1(t+1),\ldots,V_k(t+1)$ eine Voronoi-Zerlegung mit den Zentren $z_1(t+1),\ldots,z_k(t+1)$.

Interpretation: $\chi_{V_i(t)}$ ist die charakteristische Funktion von $V_i(t)$. Deshalb ist $\chi_{V_i(t)}(x(t))$ genau dann gleich 1, wenn $x(t)\in V_i(t)$, sonst 0. $x(t)\in V_i(t)$ bedeutet, daß kein $z_j(t)$ mit $j\neq i$ näher bei $x(t)$ ist als $z_i(t)$. Bis auf Ausnahmefälle ist also $z_i(t)$ *das* zu $x(t)$ nächstgelegene Zentrum. Durch die Rekursionsgleichung wird nun $z_i(t)$ ein kleines Stück zu $x(t)$ hin bewegt. Beachte, daß die Rekursionsgleichung auch in folgender Form geschrieben werden kann:

$$z_i(t+1)=\begin{cases}(1-\alpha)\,z_i(t)+\alpha\,x(t) & \text{, falls } x(t)\in V_i(t)\\ z_i(t) & \text{, sonst}\end{cases}.$$

Falls $x(t)\in V_i(t)$, liegt $z_i(t+1)$ also auf der Strecke von $z_i(t)$ nach $x(t)$ und teilt sie im Verhältnis $(1-\alpha):\alpha$.

Bemerkungen.

1. Zu Zentren $z_1,\ldots,z_k$ gibt es mehrere Voronoi-Zerlegungen, deren Zellen sich aber nur durch Teilmengen ihrer Ränder unterscheiden. In Kohonens Lernregel legt man eine Auswahl willkürlich fest, z.B.

$$V_j=\{x\in\mathbb{R}^n\mid d(x,z_i)>d(x,z_j)\text{ für }i<j\text{ und }d(x,z_i)\geq d(x,z_j)\text{ für }i>j.\}$$

2. Die charakteristischen Funktionen $\chi_{V_i(t)}$ lassen sich, wie in 6.2 ausgeführt, durch ein Netz zur Vektorquantisierung realisieren. Deshalb ist Kohonens Lernregel eigentlich eine Regel zum Training von Netzen zur Vektorquantisierung.

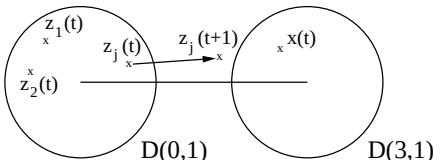
3. Die Wunschvorstellung ist, daß die $z_i(t)$ gegen Zentren z_i konvergieren, die die Cluster-Aufgaben lösen, wenn $(x(t))_t$ eine Folge ist, die im Fall a) aus geeigneten Wiederholungen der gegebenen Punktfolge $x_1, \ldots, x_N$ und im Fall b) aus unabhängigen Realisierungen der Zufallsvariablen X besteht.

   Leider ist sie nicht richtig, vergleiche aber die Bemerkung in [1], S.67, dass Tsyphin gezeigt hat, dass Kohonens Lernregel die erste Cluster-Aufgabe löst, wenn α in geeigneter Weise von t abhängt, sowie die Konvergenzuntersuchungen von Ritter und Schulten.

   Selbst wenn die $z_i(t)$ gegen eine Lösung von 6.4.1 konvergieren, müssen sie noch nicht gegen eine Lösung der zweiten Aufgabe konvergieren. Die Verteilung der z_i kann weit von einer Gleichverteilung im Sinn von 6.4.2 entfernt sein wie folgendes Beispiel zeigt.

6.4.5 Probleme

Sei $n = 2$. Die Startwerte $z_1(0), \ldots, z_k(0)$ seien in der Einheitskreisscheibe $D(0,1)$ im $\mathbb{R}^2$. Die Trainingsdaten $(x(t))_t$ mögen alle in $D(0,1) \cup D(3,1)$ liegen.



Sei t der erste Zeitpunkt, für den $x(t) \in D(3,1)$ und sei $z_j(t)$ das zu $x(t)$ nächstgelegene Zentrum. Dann wird $z_j(t)$ ein Stück in Richtung $x(t_0)$ bewegt; nehmen wir an, es wird so weit bewegt (und das kann leicht passieren), daß

$$d(z_j(t+1), D(3,1)) < d(D(0,1), D(3,1)).$$

Dann wird $z_j(s)$ für alle Zeiten $s \geq t$ näher an den Trainingsdaten $x(s)$, die in $D(3,1)$ liegen, sein als alle anderen $z_i(s)$ mit $i \neq j$. Somit werden die Trainingsdaten in $D(3,1)$ nur durch den einen Vektor z_j repräsentiert, während die in $D(0,1)$ durch alle restlichen repräsentiert werden. Das wird i.a. bedeuten, daß in der Voronoi-Zelle mit Zentrum z_j viel mehr Datenpunkte liegen als in den Voronoi-Zellen der anderen Zentren.

Es gibt mehrere Vorschläge, dieses sogenannte *stuck-vector*-Problem zu lösen. Wir skizzieren den von Desieno.

6.4.6 Desienos Modifikation von Kohonens Lernregel

$(x(t))_{t\in\mathbb{N}}$ sei eine Folge im $\mathbb{R}^n$, $\alpha, \beta \in]0,1]$ und $\gamma > 0$. $V_1(1), \ldots, V_k(1)$ sei eine Voronoi-Zerlegung des $\mathbb{R}^n$ mit den Zentren $z_1(1), \ldots, z_k(1)$. Setze $\mu_1(1) = \mu_2(1) = \ldots = \mu_k(1) = 0$.

Für $t \in \mathbb{N}$ und $i \in \{1, \ldots, k\}$ sei

$$\begin{aligned}
\mu_i(t+1) &= \mu_i(t) + \beta(\chi_{V_i(t)}(x(t)) - \mu_i(t)) \\
d_i(t) &= d(z_i(t), x(t)) - \gamma(\tfrac{1}{k} - \mu_i(t)) \\
z_i(t+1) &= \begin{cases} (1-\alpha)\, z_i(t) + \alpha\, x(t) & \text{, falls } d_i(t) < d_j(t) \text{ für alle } j \neq i \\ z_i(t) & \text{, sonst} \end{cases}
\end{aligned}$$

β ist typischerweise eine recht kleine Zahl. $\mu_i(t)$ ist ein Maß dafür, wie oft in letzter Zeit das i-te Zentrum den Wettbewerb gewonnen hat und in Richtung des Trainingsdatums geschoben wurde. μ_i wächst, wenn dies häufig der Fall ist, und nimmt wieder ab, wenn dies selten passiert.

Die Abweichung dieser Zahl μ_i von der gewünschten Häufigkeit $\frac{1}{k}$ wird benützt, um den Abstand $d(z_i(t), x(t))$ künstlich zu erhöhen oder zu vermindern und damit dem i-ten Zentrum den Gewinn des Wettbewerbs (wer ist der nächstgelegene?) zu erschweren oder zu erleichtern.

In vielen Anwendungen kann man auf diese Weise erreichen, daß sich die z_i sehr gleichmäßig im Sinn von 6.4.2 oder 6.4.3 verteilen.

6.5 Kohonens topologieerhaltende Abbildung

Neurophysiologische Experimente deuten darauf hin, daß die Sinneszellen im Auge (oder Ohr oder Finger usw.) so mit Neuronen im Gehirn verbunden sind, daß bei Reizung einer Sinneszelle die Position der dadurch erregten Gehirnzelle stetig mit der Position der gereizten Sinneszelle variiert und daß häufiger gereizte Bereiche von Sinneszellen oder Bereiche von dichter liegenden Sinneszellen entsprechend größeren Bereichen im Gehirn entsprechen.

Kohonen u.a. haben Modelle entwickelt, wie sich solche Zuordnungen selbst ohne Lehrer herausbilden können (*self-organizing maps*). Solange die Modelle kontinuierlich sind, macht es keine Schwierigkeiten die angestrebten Eigenschaften der Abbildung durch Begriffe wie stetig, topologisch, differenzierbar zu beschreiben. Im diskreten Fall verlieren diese Begriffe ihren Sinn oder beschreiben nicht mehr die (unpräzise) Wunschvorstellung. Wir machen hier den Versuch, eine mathematisch sinnvolle, adäquate Beschreibung zu geben.

Wir betrachten das Gitter $\mathbb{Z}^n$ als Graphen, indem wir alle Punkte $(i_1,\ldots,i_n),(j_1,\ldots,j_n) \in \mathbb{Z}^n$ durch eine Kante verbinden, für die $|i_l - j_l| \leq 1$ für alle $l = 1,\ldots,n$ gilt.

Sei $[\![a,b]\!] := \{l \in \mathbb{Z} \mid a \leq l \leq b\}$. Als Quader in $\mathbb{Z}^n$ bezeichnen wir eine Menge der Gestalt $[\![i_1,j_1]\!] \times \ldots \times [\![i_h,j_n]\!]$; sind alle $i_l = 0$ und alle $j_l > 0$, so nennen wir sie einen *Standardquader* in $\mathbb{Z}^n$.

6.5.1 Aufgabenstellung für den Fall gleicher Dimensionen

a) Kontinuierlich

$G \subset \mathbb{R}^n$ sei ein beschränktes Gebiet, p ein Wahrscheinlichkeitsmaß auf G mit Dichte f bezüglich des Lebesgue-Maßes λ. Finde einen Diffeomorphismus $\phi : G \to G$, so daß $p(\phi(B)) = \frac{1}{\lambda(G)}\lambda(B)$ für jede meßbare Menge $B \subset G$ (d.h. $f \circ \phi |\det(D\phi)| = \frac{1}{\lambda(G)}$ wegen des Transformationssatzes) und $\phi|_{\partial G} = \mathrm{Id}_{\partial G}$.

Anschauliche Interpretation für $n = 2$: Kann man eine in einen Rahmen eingespannte Gummimembran so verzerren, daß ihre Dichte konstant wird?

Man kann zeigen, daß es unter geeigneten Zusatzvoraussetzungen so einen Diffeomorphismus ϕ gilt.

b) Diskret

$G \subset \mathbb{R}^n$ sei ein Quader, p ein Wahrscheinlichkeitsmaß auf G, Q ein Standardquader in $\mathbb{Z}^n$. Finde eine Abbildung $\phi : Q \to G$ mit folgenden Eigenschaften:

Sind $(V_q)_{q\in Q}$ die Voronoi-Zellen einer Voronoi-Zerlegung von G bezüglich der Zentren $(\phi(q))_{q\in Q}$ ($\phi(q)$ sei das Zentrum von V_q), so gelte:

1) $p(V_q) = \frac{1}{|Q|}$ für alle $q \in Q$.

2) Die Zuordnung $q \mapsto V_q$ ist ein Graphisomorphismus von Q auf den Nachbarschaftsgraphen $\mathcal{V}$ der Voronoi-Zerlegung. (Die Knoten von $\mathcal{V}$ sind genau die Voronoi-Zellen V_q, $q \in Q$, und zwischen V_q und V_r existiert genau dann eine Kante, wenn $\overline{V}_q \cap \overline{V}_r \neq \emptyset$.)

Bemerkung. Diese Aufgabenstellung ist eine Erweiterung der Aufgabe 6.4.2. Und zwar trägt die Indexmenge der Zentren eine Nachbarschaftsstruktur, und es wird gefordert, daß sie gleich der der Voronoi-Zerlegung ist. Das bedeutet für die Vektorquantisierungsabbildung

$$G \to Q,\ x \mapsto q \quad \text{mit} \quad x \in V_q.$$

eine Art von Stetigkeit in dem Sinn, daß q nur zu einem Nachbarwert in $\mathbb{Z}^n$ springt, wenn x in eine benachbarte Voronoi-Zelle (=Intervalle) in aufsteigender Reihenfolge durchzunumerieren.

Graphische Darstellung für $n = 2$: als Netz, d.h. zeichne die Verbindungsstrecke von $\phi(q)$ nach $\phi(r)$, wenn q und r sich genau einer Komponente unterscheiden, also 4-Nachbarn in $\mathbb{Z}^2$ sind.

6.5.2 Aufgabenstellung für den Fall ungleicher Dimensionen

Häufig füllen die Datenpunkte, die geclustert werden sollen, nicht ein ganzes Gebiet G in $\mathbb{R}^n$ aus, sondern konzentrieren sich auf eine Umgebung einer niederdimensionalen Untermannigfaltigkeit, weil zwischen ihren Komponenten Abhängigkeiten bestehen. Haben sie z.B. alle (nahezu) die Länge 1, so liegen sie auf (bzw. in einer kleinen Umgebung von) der Einheitssphäre. In so einem Fall will man natürlich auch die Zentren der Voronoi-Zellen einer Vektorquantisierung als Knoten eines entsprechend niederdimensionalen Gitters wählen. Wir versuchen, dies im diskreten Fall etwas präziser zu fassen.

p sei ein Wahrscheinlichkeitsmaß auf $\mathbb{R}^n$, Q ein Standardquader in $\mathbb{Z}^m$, $m < n$. Finde eine Abbildung $\phi : Q \to \mathbb{R}^n$ und ein $\varepsilon > 0$ mit folgenden Eigenschaften:

Sind die $(V_q)_{q\in Q}$ die Voronoi-Zellen einer Voronoi-Zerlegung von $\mathbb{R}^n$ bezüglich der Zentren $(\phi(q))_{q\in Q}$ und ist $V_q^\varepsilon = V_q \cap B(\phi(q), \varepsilon)$, so gelte:

1. $p(V_q^\varepsilon) = \frac{1}{|Q|}$ für alle $q \in Q$.
2. Die Zuordnung $q \mapsto V_q$ ist ein Graphisomorphismus von Q auf den Nachbarschaftsgraphen $\mathcal{V}$ von $(V_q^\varepsilon)_{q\in Q}$.

Bemerkung.

1. Obige Aufgabenstellung ist nicht ganz befriedigend, weil das Abschneiden der V_q durch $B(\phi(q), \varepsilon)$ eigentlich nur in einer Richtung erfolgen sollte, die zu dem m-dimensionalen Gitter mit den Knoten $\phi(q)$ orthogonal ist. Dies ist jedoch schwierig zu präzisieren im diskreten Fall. Nicht dagegen im kontinuierlichen Fall.
2. Die Aufgabenstellung läßt sich natürlich modifizieren, z.B. auf folgende Weise:

   Sei $f : \cup_{q\in Q} V_q^\varepsilon \to Q$ die Abbildung $f(x) = q$ mit $x \in V_q^\varepsilon$.

   Dann werde zusätzlich gefordert, daß $\int d(x, \phi(f(x)))^2\, dp(x)$ minimal ist.
3. Eine analoge Aufgabenstellung läßt sich im Kontinuierlichen leichter präzise formulieren:

   Gesucht ist eine parametrisierte Umkehrfunktion $\phi : B \to \mathbb{R}^n$ der Dimension $m < n$, B Gebiet im $\mathbb{R}^m$, und die Parametrisierung $\tilde{\phi} : B \times B(0, \varepsilon) \to U_\varepsilon$ einer ε-Tubenumgebung von $\phi(B)$, so daß $p \circ \tilde{\phi}^{-1}$ die konstante Verteilung ist. Stattdessen oder zusätzlich kann man fordern, daß $\int_{u_\varepsilon} d(x, \phi(B))^2\, dp(x)$ minimal ist. (Zu jedem $x \in U_\varepsilon$ gibt es genau einen Punkt $y \in \phi(G)$ mit $d(x, \phi(G)) = d(x, y)$, wenn d die euklidische Metrik ist.)

Spezialfall: $B = \mathbb{R}$, ϕ linear, also $\phi(B)$ eindimensionaler Untervektorraum. Dann bedeutet die Minimalität von $\int d(x, \phi(B))^2\, dp(x)$ gerade, daß $\phi(B)$ die *Hauptträgheitsachse* der Massenverteilung p ist.

Es gibt etliche Verfahren, die obige Aufgaben ansatzweise lösen sollen; teilweise beruhen sie auf biologischen Analogien. Das wegen seiner Einfachheit am häufigsten verwendete stammt von Kohonen, der seine Lernregel 6.4.4 dadurch modifizierte, daß nicht nur der Gewinner, sondern auch seine Nachbarn bei jedem Lernschritt verändert werden.

6.5.3 Kohonens Lernregel für selbstorganisierende Abbildungen

$(x(t))_{t\in\mathbb{N}}$ sei eine Folge im $\mathbb{R}^n$, $m \le n$, $Q \in \mathbb{Z}^m$ ein Standardquader, $\alpha \in]0, 1]$. $h : [0, \infty[\to [0, \infty[$ sei eine monoton fallende Funktion. (*Interaktionsfunktion*) $\phi_1 : Q \to \mathbb{R}^n$ sei eine Abbildung; $z_q(1) = \phi_1(q)$ für $q \in Q$. $(V_q(1))_{q\in Q}$ sei eine Voronoi-Zerlegung mit den Zentren $(z_q(1))_{q\in Q}$.

Für $t \in \mathbb{N}$ gibt es genau ein $r \in Q$ mit $x(t) \in V_r(t)$. Setze für jedes $q \in Q$

$$z_q(t+1) = z_q(t) + \alpha\, h(\|q - r\|)(x(t) - z_q(t)) \quad \text{und} \quad \phi_{t+1}(q) = z_q(t+1),$$

und wähle eine Voronoi-Zerlegung $(V_q(t+1))_{q\in Q}$ mit den Zentren $(z_q(t+1))_{q\in Q}$.

6.5.4 Bemerkungen

1. Die Folge $(x(t))_{t\in\mathbb{N}}$ muß man sich als unabhängige Stichproben vorstellen, die gemäß der in 6.5.1 oder 6.5.2 gegebenen Verteilung p gezogen sein. Oft findet man eine andere Sprechweise: Die Gitterpunkte von Q heißen Neuronen und das Zentrum z_q heißt das Gewicht oder der Gewichtsvektor von q.

2. Erstaunlicherweise verändern sich die Zentren $z_q(t)$ tatsächlich vielfach so, daß ϕ_t Graphisomorphismen im Sinn von 6.5.1 oder 6.5.2 sind. Für h wählt man meist glockenähnliche Funktionen, z.B. $h(u) = \exp(-\frac{u^2}{2\sigma^2})$. Bei Experimenten stellt man meist folgendes Verhalten fest:

   1. Phase: Zunächst entwirrt sich das Netz, bis ϕ_t ein Graphisomorphismus ist. Damit das wirklich eintritt, darf die Breite der Glocke h (also bei der Gaußfunktion die Varianz σ) nicht zu klein sein. Der Ordnungseffekt beruht wesentlich darauf, daß genügend viele Nachbarn des jeweiligen gewinnenden Neurons ihren Gewichtsvektor ebenfalls verändern.

   2. Phase: Ist ϕ_t einmal ein Graphisomorphismus, so tritt oft keine große Veränderung von ϕ_t mehr ein. Man hat den Eindruck, daß die Zentren $z_q(t)$ nur noch um einen Mittelwert oszillieren, und zwar infolge der statistischen Fluktuationen der Stichproben $(x(t))_t$. Deshalb wählt man in dieser Phase α und h nicht mehr konstant, sondern läßt sowohl α wie auch die Breite von h gegen 0 gehen. Ist die Breite klein oder Null, so lernen die Nachbarn des Gewinners nicht mehr mit; ist dagegen α sehr klein, so findet fast keine Veränderung mehr statt. Es ist deshalb wichtig, daß α nicht zu schnell gegen 0 geht.

   Genauere Untersuchungen zum Konvergenzverhalten in der 2. Phase findet man in [6], [7] und [8].

   Dort wird ein Modell mit kontinuierlicher Zeit entworfen. Als notwendige und hinreichende Bedingung für eine fast sichere Konvergenz von ϕ_t gegen den stochastischen gleichgewichteten Mittelwert erweist sich:

$$\lim_{t\to\infty} \alpha(t) = 0 \quad \text{und} \quad \lim_{t\to\infty} \int_0^t \alpha(s)\, ds = \infty.$$

   Diese Bedingungen sind erfüllt, wenn $\alpha(t) = \text{const} \cdot t^{-\varrho}$ mit $\varrho \in]0, 1]$.

   Zur 1. Phase stellte Kohonen in einigen Arbeiten Überlegungen an, allerdings mehr heuristischer Art und nur zum eindimensionalen Fall. Einen Beweis im eindimensionalen Fall haben Cottrell und Fort gegeben.

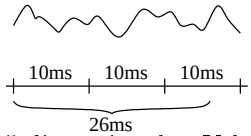
   In zweidimensionalen Beispielen sieht man meist leicht, ob ϕ_t bereits ein Graphenisomorphismus ist und somit die 1. Phase als beendet angesehen werden kann. In höherdimensionalen Situationen kann das viel schwerer festzustellen sein.

3. Ein Problem in Anwendungen ist die geeignete Festlegung der Dimension m des Gitters. Ist die Dimension ungeeignet, so legt sich das Gitternetz in Falten. In höheren Dimensionen ist dies nicht immer leicht zu bemerken. Es gibt Arbeiten, die sich damit beschäftigen, wie man m festlegen kann und wie man merken kann, daß m nicht geeignet ist.

4. Obwohl Kohonen behauptete, daß sich mit seiner Lernregel die Zentren so einstellen, daß in jeder Voronoi-Zelle annähernd gleich viele Stichproben liegen, zeigten Ritter und Schulten, daß dies nicht immer stimmt, zumindest dann nicht, wenn das Wahrscheinlichkeitsmaß deutlich von der Gleichverteilung abweicht, außer für $m = n = 2$.

6.5.5 Oft zitierte Anwendungen

- **Phonem-Karte**



Sprachsignal abtasten und digitalisieren
Alle 10ms wird ein 26ms langes Stück Sprache Fouriertransformiert. Der gesamte Frequenzbereich wird in 15 Intervalle $B_1, \ldots, B_{15}$ unterteilt; für jeden Bereich wird die Intensität des Spektrums berechnet. Diese Folge von 15-dimensionalen Vektoren dient als Trainingsfolge für eine zweidimensionale Kohonen-Abbildung.

Nach dem Training wird durch Musterbeispiele der 21 finnischen Phoneme festgestellt, welches Neuron welchem Phonem entspricht.

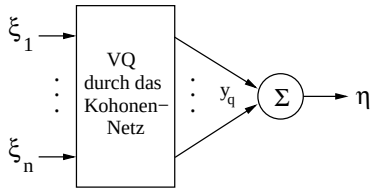
Anschließend kann man die so konstruierte Vektorquantisierung verwenden, um jedem Sprachsignal einen Weg in den Neuronen, also eine Folge von Phonemen zuzuordnen.

- **Approximation von Funktionen durch Treppenfunktionen**

  Mit Hilfe von Kohonen-Netzen kann man, wie schon in 6.3 ausgeführt, Funktionen durch Netze, die Treppenfunktionen realisieren, approximieren.

  $K \subset \mathbb{R}^n$ sei kompakt, $f : K \to \mathbb{R}^l$. $(x(t))_{t\in\mathbb{N}}$ sei eine Folge in K, $y(t) = f(x(t))$. Q sei ein Standardquader in $\mathbb{Z}^2$.

  Zunächst konstruiert man mit Kohonens Lernregel 6.5.2 ein Kohonen-Netz $\phi : Q \to \mathbb{R}^n$ bezüglich der Folge $(x(t))_t$. Sei $(V_q)_{q\in Q}$ eine Voronoi-Zerlegung mit den Zentren $(\phi(q))_{q\in Q}$. y_q sei ein Mittelwert von f in der Menge $V_q \cap K$; z.B. kann während der Konstruktion von ϕ zusätzlich $y_q(t+1) = y_q(t)+\gamma(y(t)-y_q(t))$ brechnet werden, wobei $\gamma \in]0,1[$.

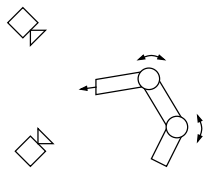
  $\eta = \sum_{q\in Q} y_q \chi_{v_q}(\xi)$ ist die vom Netz realisierte Abbildung.

  Damit f dadurch gut approximiert wird, müssen natürlich einige Voraussetzungen erfüllt sein, z.B. sollte $(x(t))_t$ in K gleichmäßig verteilt sein (oder entsprechend einer in der jeweiligen Anwendung wichtigen Verteilung

  Meist braucht man sehr viele Gitterpunkte, also große Standardquader Q, um eine gute Approximation zu erreichen. Manchmal kommt man mit weniger Gitterpunkten aus, wenn man die konstanten Werte y_q auf V_q auf geeignete Weise interpoliert. Dabei ist es vorteilhaft, daß das Kohonen-Netz die Topologie erhält.

  Hecht-Nielsen nennt solche Netze *Counterpropagation-Netze*. Sie benötigen oft mehr Neuronen als BP-Netze. Ihr Vorteil ist jedoch, daß sie meist sehr viel schneller zu trainieren sind und daß der Approximationsfehler abschätzbar ist im Gegensatz zu BP-Netzen, bei denen man für nicht trainierte Eingabewerte unliebsame Überraschungen erleben kann.

- **(Visuomotorische) Steuerung eines Roboterarms**

  Gegeben: ein Roboterarm mit drei Freiheitsgraden; zwei Kameras, die die Position des Greifers sehen. B_1 und B_2 seien die Bildebenen der Kameras, $G \subset \mathbb{R}^3$ sei die Menge der möglichen Greiferendpunkte. Jeder Punkt $x \in G$ wird durch die Kameras auf ein Punktepaar $\beta(x) \in B_1 \times B_2$ abgebildet. Die Kameras seien so positioniert, daß diese Abbildung β injektiv und eigentlich ist. Dann ist $\beta(G)$ eine dreidimensionale Untermannigfaltigkeit von $B_1 \times B_2$.

  Zu jeder Gelenkstellung $\vartheta \in \mathbb{R}^3$ des Arms sei $f(\vartheta) \in B_1 \times B_2$ das Paar der Bildpunkte des Greifers. Die Geometrie des Arms und der Kamera seien nicht bekannt, so daß für f keine analytische Beschreibung vorliegt.

  Gesucht: eine Realisierung der inversen Abbildung $f^{-1} : \beta(G) \to \mathbb{R}^3$. Das Erlernen soll ohne Überwachung ablaufen.

  Ansatz: Konstruiere ein dreidimensionales Kohonen-Netz $\phi : Q \to B_1 \times B_2$, das die dreidimensionale Umkehrfunktion $\beta(G)$ „diskret approximiert". Konstruiere gleichzeitig zwei Abbildungen $\Theta : Q \to \mathbb{R}^3$ und $A : Q \to \mathbb{R}^{3\times 4}$, so daß für jedes $q \in Q$ gilt:

  > Für alle $u \in \beta(G)$, die in der Voronoi-Zelle mit Zentrum $\phi(q)$ liegen, ist $\Theta_q + A_q(u - \phi(q))$ eine gute Approximation für $f^{-1}(u)$.

  Das heißt, f^{-1} wird durch eine Abbildung approximiert, die auf jeder Voronoi-Zelle affin linear ist.

  Training: ϕ wird durch die Kohonen-Lernregel erlernt, indem einfach eine repräsentative Menge von Greiferpositionen vorgegeben wird. Gleichzeitig werden Θ und A durch eine Mischung aus LMS-Regel und Kohonenregel erlernt. Das Training erfolgt unüberwacht ohne Vorgabe von Sollwerten für Θ_q und A_q. Man versucht lediglich, die Abweichung der tatsächlichen Greiferposition von der gewünschten in den Bildern der Kameras mit einem Gradientenabstiegsverfahren zu minimieren.

Kapitel 7

Support Vector Machines

7.1 Optimale lineare Trennung

Wir kommen zurück auf die Trennung zweier linear separierbarer Mengen $M^+ \subset \mathbb{R}^n$ und $M^- \subset \mathbb{R}^n$ durch eine Hyperebene.

Der Perzeptron-Lernalgorithmus von Minsky-Papert in Kapitel 2 ist nicht sehr effizient und liefert eine Trennebene, die nicht unbedingt optimal ist in dem Sinne, dass sie möglichst großen Abstand zu M^+ und M^- hat. Eine solche optimale Hyperebene kann man aber leicht finden:

C^+ bzw. C^- seien die konvexen Hüllen von M^+ bzw. M^-. Bestimme Punkte $p^+ \in C^+$ und $p^- \in C^-$, so dass

$$d(p^+, C^-) = \min\{d(p, C^-) : p \in C^+\} \text{ und } d(p^-, C^+) = \min\{d(p, C^+) : p \in C^-\}.$$

Die Hyperebene, die auf der Strecke von p^+ nach p^- senkrecht steht und durch deren Mittelpunkt geht, ist optimal.

Analytisch lässt sich das folgendermaßen beschreiben. Jede Hyperebene lässt sich durch eine Gleichung $u^T x + a = 0$ mit $u \in \mathbb{R}^n, u \neq 0$ und $a \in \mathbb{R}$ beschreiben. Gilt $\|u\|_2 = 1$, so ist $u^T x + a$ der Abstand (mit Vorzeichen) von x zur Hyperebene.

Ist $\delta = d(C^+, C^-)$, so gilt für die optimale Hyperebene $u^T x + a = 0$ mit $\|u\|_2 = 1$, dass

$$\min\{u^T x + a : x \in M^+\} \geq \frac{\delta}{2} \qquad \text{und} \qquad \max\{u^T x + a : x \in M^-\} \leq -\frac{\delta}{2}$$

Setze $u = \frac{\delta}{2} w$ und $a = \frac{\delta}{2} b$. Dann gilt $w^T x + b \geq 1$ für $x \in M^+$ und $w^T x + b \leq -1$ für $x \in M^-$. Außerdem ist $\delta = \frac{2}{\|w\|_2}$.

Somit ergibt sich folgendes Optimierungsproblem: Finde w und b so, dass $\frac{1}{2} w^T w$ unter der Nebenbedingung $w^T x + b \geq 1$ für $x \in M^+$ und $w^T x + b \leq -1$ für $x \in M^-$ minimal ist.

Meist wählt man für die Punkte in $M = M^+ \cup M^-$ eine Aufzählung $(x_i)_{i=1,\ldots,m}$ und setzt $y_i := +1$ für $x_i \in M^+$ und $y_i = -1$ für $x \in M^-$. Dann lautet das Optimierungsproblem:
Finde w und b so, dass $\frac{1}{2} w^T w$ unter den Nebenbedingungen $y_i(w^T x_i + b) \geq 1$ für $i = 1, \ldots, m$ minimal ist.

Mit Hilfe von Lagrangemultiplikatoren kann man das Problem umformulieren zu folgendem

Optimierungsproblem: Finde α_i , so dass

$$\frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j x_i^T x_j \; - \sum_{i=1}^{m} \alpha_i \;\text{ unter den Bedingungen }\; \sum_{i=1}^{m} \alpha_i y_i = 0 \;\text{ und } 0 \leq \alpha_i \text{ minimal ist.}$$

Dann steht der Vektor $w = \sum_{i=1}^{m} \alpha_i y_i x_i$ senkrecht auf der optimalen Hyperebene .

Verallgemeinerung für linear nicht trennbare Mengen

Wenn M^+ und M^- nicht linear trennbar sind, ihre konvexen Hüllen sich jedoch nur wenig überlappen, kann man versuchen, eine Hyperebene zu finden, die die beiden Mengen bis auf wenige Ausnahmepunkte trennt. Dazu geht man von sogenannten reduzierten konvexen Hüllen der beiden Mengen aus. Eine reduzierte konvexe Hülle einer endlichen Menge $X = \{x_1, \ldots, x_k\}$ besteht aus allen Punkten der Gestalt $\sum_{i=1}^{k} \lambda_i x_i$ mit $\sum_{i=1}^{k} \lambda_i = 1$ und $0 \leq \lambda_i \leq \rho$, wobei $\rho \in]0, 1]$ eine vorgegebene Schwelle ist. Für $\rho = 1$ bekommt man die übliche konvexe Hülle; verkleinert man ρ, so wird der Einfluss der Punkte x_i in den konvexen Linearkombinationen lokaler. Besteht X z.B. aus drei Punkten, so werden durch Verkleinern von ρ mehr und mehr die Ecken der aufgespannten Dreiecksfläche abgeschnitten.

Man kann nun versuchen, eine optimale Trennebene für reduzierte konvexe Hüllen von M^+ und M^- zu finden. Sie wird M^+ und M^- natürlich nur mit einem gewissen Fehler trennen. Dies führt auf ein Optimierungsproblem der obigen Art, wobei lediglich die Nebenbedingungen $0 \leq \alpha_i$ durch $0 \leq \alpha_i \leq \rho$ ersetzt werden.

7.2 Nichtlineare Transformationen und Kerne

Um auch nicht linear separierbare Klassen trennen zu können, versucht man, mit einer Abbildung $\Phi\colon \mathbb{R}^n \to \mathbb{R}^N$ die Musterklassen M^+ und M^- so in einen hochdimensionalen Raum abzubilden, dass ihre Bilder linear separabel sind und mit obigem Ansatz eine optimale Trennebene bestimmt werden kann.

Zu minimiere ist dann die Funktion

$$\frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \Phi(x_i)^T \Phi(x_j) \; - \; \sum_{i=1}^{m} \alpha_i$$

unter der Bedingung $\quad \sum_{i=1}^{m} \alpha_i y_i = 0, \; \alpha_i \geq 0.$

Der Parameter w der optimalen Trennebene ergibt sich dann zu $w = \sum_{i=1}^{m} \alpha_i \Phi(x_i)$ und der optimale Parameter b durch Einsetzen der Stützwerte. Die Klassifikation der x_i erfolgt dann durch die Funktion

$$f(x_j) = \text{signum}\big(\sum_{i=1}^{m} \alpha_i y_i \Phi(x_i)^T \Phi(x_i) + b\big)$$

Man beachte, dass lediglich Terme $\Phi(x_i)^T \Phi(x_i)$ vorkommen. $\Phi(x)$ kommt nicht allein vor. Deshalb braucht man nicht erst Φ zu wählen, sondern kann gleich zu sogemannten Kernfunktionen $K(u.v)$ übergehen. In obigem Beispiel ist $K(u.v) = \Phi(u)^T \Phi(v)$.

Beispiele für Kernfunktionen

- $K(u, v) = (u^T v + 1)^p$
- Der Gaußkern $K(u.v) = \exp(-\frac{\|u-v\|^2}{2\sigma^2}$ führt zu RBF-Netzen.
- $K(u, v) = h(u^T v - \theta)$ mit sigmoider Funktion h.

Die Kunst bei der Anwendung von SVMs liegt in der Wahl der Kernfunktion und einer geeigneten Codierung der rohen Eingangsdaten. Vorteilhaft ist, dass es robuste Lösungsalgorithmen für die Optimierung gibt, dass lokale Minima kein Problem darstellen und die Resultate reproduzierbar sind.

Eine kurze Übersicht über SVM-Methoden findet man in
K.P. Benett, C. Campbell: *Support Vector machines: Hype or Hallelujah.* SIGKDD Explorations, 2000, Vol. 2, 1-13.

Kapitel 8

Assoziativspeicher ohne Rückkopplung

Wer ist der Autor des Buches „Der kleine Prinz“? Wahrscheinlich wissen Sie es sofort: Saint-Exupéry.
Wie finden Sie die Antwort so schnell?

Wer ist der Übersetzer der deutschen Ausgabe? Wahrscheinlich wissen Sie sofort, daß Sie es nicht wissen. Wie können Sie ihren Gedächtnisspeicher so schnell durchsuchen? Bedenken Sie, daß Nervenzellen verhältnismäßig langsam sind.

Ähnliche Aufgaben sind z.B. bei Datenbanken zu lösen: Man möchte zu einem (eventuell unvollständigen) Schlüsselwort (beispielsweise Name) die in der Datenbank gespeicherten Angaben ausgegeben bekommen oder eine entsprechende Mitteilung, daß unter diesem Schlüsselwort nichts gespeichert ist.

Es muß also eine partielle Abbildung $f : A \supset D \to B$ zwischen meist sehr großen endlichen Mengen realisiert werden und zwar so, daß zu jedem $a \in A$ sehr schnell festgestellt werden kann, ob f auf a definiert ist, und welchen Wert es gegebenenfalls annimmt. Ein Konzept, das dies leistet, heißt *Assoziativspeicher*.
Welche Datenstrukturen bietet die übliche Informatik dafür an?

Tabellen und verkettete Listen sind meist ineffizient, weil man sie linear durchsuchen muß, um a zu finden oder festzustellen, daß f auf a nicht definiert ist. Um solche Suchen zu beschleunigen benötigt man auf A eine Ordnung, mit deren Hilfe man den Graphen von f in Gestalt eines Suchbaumes darstellen kann. Trotzdem ist die Suchgeschwindigkeit manchmal zu langsam, und man hat überlegt, ob es nicht noch andere Realisierungsmöglichkeiten gibt. Dazu kommt, daß in vielen Anwendungen zwei Arten von *Fehlertoleranz* wünschenswert sind:

1. Wenn $a' \in A$ nicht im Definitionsbereich von f ist, aber sehr „ähnlich“ zu einem a im Definitionsbereich ist, so soll automatisch der Wert $f(a)$ ausgegeben werden. Beispiel: $A =$ Buchstabenstring, a ein Name, a' derselbe Name mit Tippfehler.

2. Auch wenn Teile des Assoziativspeichers ausfallen, soll die realisierte Abbildung sich nicht allzu sehr verändern.

Unter Berücksichtigung von 1. definieren wir:

Ein *Assoziativspeicher* für eine partielle Abbildung $f : A \supset D \to B$ ist ein Konzept, das eine Abbildung $\psi : A \to B$ realisiert, so daß gilt: Es gibt eine Partition $(A_x)_{x \in D}$ von A, so daß $x \in A_x$ und $\psi|_{A_x} = f(x)$.

Anschaulich soll A_x alle $a \in A$ enthalten, die zu x so ähnlich sind, daß ihnen derselbe Wert zugeordnet wird. A_x heißt oft *Einzugsbereich* von x (*basin of attraction*). Die Elemente von D werden oft die gespeicherten *Prototypen* genannt.

Beschreibt man die Ähnlichkeit mit einer Metrik d, so kann man für $(A_x)_x$ eine Voronoi-Zerlegung wählen. Ein Assoziativspeicher ist dann einfach die Realisierung einer Treppenfunktion durch ein kompetitives Netz wie in 6.3. Allerdings sind die typischen Anwendungssituationen meist etwas anders als beispielsweise bei der Vektorquantisierung:

1. Die Anzahl der gespeicherten Prototypen ist sehr groß; beispielsweise die Wörter eines Lexikons, Bilder, Fingerabdrücke u.v.m.

2. Die Menge A der möglichen Eingabemuster ist Teil eines Vektorraums hoher Dimension; z.B. $\{0,\ldots,26\}^{50}$ oder $[\![0,255]\!]^{512\times 512}$.

3. Die Metrik d ist oft nicht die euklidische Metrik, ja nicht einmal explizit gegeben. Meist hat man nur eine vage Vorstellung, was ähnlich bedeuten soll und wie die Einzugsbereiche A_x aussehen sollen. Manchmal läßt sich diese Vorstellung durch eine Metrik (z.B. die Hamming-Metrik) präzisieren, oft wird jedoch ψ auf andere Weisen konstruiert.

Wir werden im folgenden $A \subset \mathbb{R}^n$ und $B \subset \mathbb{R}^m$ voraussetzen. Gilt $A \subset \{0,1\}^n$, so spricht man von *binären Mustern*; gilt $A \subset \{-1,1\}^n$, so von *bipolaren Mustern*. Ein großer Teil der Assoziativspeicher wurde nur für binäre oder bipolare Muster entwickelt; andere Muster müssen erst geeignet kodiert werden. Da man sich in der Literatur nicht auf eine Art geeignet hat, muß man sich daran gewöhnen, öfters die simple, aber doch lästige Übersetzung

$$\{0,1\} \to \{-1,1\} \text{ bzw. } \{-1,1\} \to \{0,1\} \quad \text{durch} \quad x \mapsto 2x-1 \text{ bzw. } y \mapsto \tfrac{1}{2}(y+1)$$

vorzunehmen.

Wir werden meist die übliche Notation verwenden und die Punkte in D abzählen und als Folge $x_1,\ldots,x_N$ schreiben und dementsprechend auch den Graphen von f als Folge (x_j,y_j), $j=1,\ldots,N$ mit $y_j = f(x_j)$.

8.1 Lineare Korrelationsspeicher

Seien $(x_1,y_1),\ldots,(x_N,y_N) \in \mathbb{R}^n \times \mathbb{R}^m$. Gesucht ist eine Matrix $W \in \mathbb{R}^n \times \mathbb{R}^m$, so daß $y_j = Wx_j$ für jedes j. So eine Matrix gibt es natürlich nicht immer. Setze $X = (x_1,\ldots,x_N)$, $Y = (y_1,\ldots,y_N)$.

1. Fall $x_1,\ldots,x_N$ sind orthonormal (bzgl. des euklidischen Skalarprodukts).

Dann setzt man einfach $W = y_1x_1^T + \ldots + y_Nx_N^T$, und es gilt $Wx_j = \sum_{i=1}^{N} y_i x_i^T x_j = y_j$ für jedes j.

Man kann $((x_j,y_j))_j$ auch als Trainingsfolge betrachten und W iterativ konstruieren:
$W(0) := 0$, $W(t+1) := W(t) + y_{t+1}x_{t+1}^T$ für $t \in \{0,\ldots,N-1\}$, $W := W(N)$. (*Hebbsche Lernregel*)
Mit $X = (x_1,\ldots,x_N)$ und $Y = (y_1,\ldots,y_N)$ gilt $W = YX^T$.

2. Fall $x_1,\ldots,x_N$ linear unabhängig, aber nicht notwendig orthogonal.

Dann läßt sich $x_1,\ldots,x_N$ zu einer Basis $x_1,\ldots,x_n$ des $\mathbb{R}^n$ ergänzen und man setzt

$$\psi\left(\sum_{i=1}^{n} \alpha_i x_i\right) := \sum_{i=1}^{N} \alpha_i y_i.$$

ψ ist eine lineare Abbildung, ihre Matrix bezüglich der Standardbasen im $\mathbb{R}^n$ bzw. $\mathbb{R}^m$ ist die gesuchte Matrix W. Es gilt $W = YX^+$, wobei X^+ die Pseudoinverse von X ist.

3. Fall $x_1,\ldots,x_N$ linear abhängig.

Dann kann man nicht mehr für beliebige $y_1,\ldots,y_N$ so eine Matrix finden. Man kann nur noch eine bestmögliche L^2-Approximation wie im 3. Kapitel finden.

Weil im $\mathbb{R}^n$ mehr als n Vektoren linear abhängig sind, scheinen lineare Abbildungen und ihre Matrizen als Assoziativspeicher nicht geeignet. Man kann jedoch durch eine andere Blickweise doch noch einen fruchbaren Ansatz finden. Und zwar definiert man einfach die Matrix W wie im 1. Fall, auch wenn die x_j nicht orthogonal oder sogar linear abhängig sind, also

$$W = \sum_{i=1}^{N} \alpha_i y_i x_i^T \quad \text{mit} \quad \alpha_i = \tfrac{1}{\|x_i\|_2^2} \quad \text{(es seien alle } x_i \neq 0\text{)}.$$

Dann gilt für jedes $j \in \{1, \ldots, N\}$

$$Wx_j = \alpha_j y_j \cdot x_j^T x_j + \sum_{i \neq j} \alpha_i y_i \cdot x_i^T x_j = y_j + \underbrace{\sum_{i \neq j} \alpha_i y_i \cdot x_i^T x_j}_{\text{Störterm } (\textit{cross talk})} .$$

Anschauliche Interpretation: $x_i^T x_j = x_{i,1}x_{j,1} + \ldots + x_{i,n}x_{j,n}$ läßt sich als Kreuzkorrelation der beiden Muster x_i und x_j auffassen. Sind die x_i unkorreliert, so wird die Zuordnung $x_i \to y_j$ durch W exakt gegeben; sind die x_i korreliert, so wird die exakte Zuordnung durch die Kreuzkorrelationsterme gestört.

Idee: Unterdrücke die Störterme durch eine geeignete Schwelle.
Dazu läßt man für die y_i nur noch bipolare oder binäre Muster zu; wir wollen hier $y_i \in \{0,1\}^n$ wählen. Setze wieder

$$W = \sum_{i=1}^{N} \frac{y_i x_i^T}{\|x_i\|^2}, \qquad w_{k,l} = \sum_{i=1}^{N} \frac{y_{ik} x_{il}^T}{\|x_i\|^2}, \qquad \text{also } W = (w_{kl}).$$

Es seien $\Theta \in]0,1[$ und $h = \chi_{[0,\infty[}$ die Heavyside-Funktion. Definiere $\psi : \mathbb{R}^n \to \{0,1\}^m$, $\eta = \psi(\xi)$ durch $\eta_k = h\left(\sum_{l=1}^{n} w_{kl}\xi_l - \Theta\right)$, also $\eta_k = \begin{cases} 1 & \text{, falls die } k\text{-te Komponente von } W\xi \text{ größer oder gleich } \Theta \\ 0 & \text{, sonst} \end{cases}$.

Wählt man z.B. $\Theta = \frac{1}{2}$ und sind die Übersprecherterme genügend klein, so ist $\psi(x_j) = y_j$ für jedes j und nahe bei x_j liegende Punkte x bekommen denselben Wert zugewiesen.

Ob die Übersprechterme klein sind, hängt nicht nur von den Kreuzkorrelationstermen ab, sondern auch davon, wieviele Einsen in jeder Komponente der anderen y_i vorkommen. Es kann also eine wesentliche Rolle spielen, wie die Ausgangsdaten binär kodiert werden. Noch deutlicher wird das bei dem folgenden Korrelationsspeicher.

8.2 Binäre Assoziativspeicher mit spärlicher Kodierung

$(x_j, x_j) \in \{0,1\}^n \times \{0,1\}^m$, $j = 1, \ldots, N$, seien binäre Muster, $x_j = (x_{j,1}, \ldots, x_{j,n})^T$, $y_j = (y_{j,1}, \ldots, y_{j,m})^T$.
Setze:
$w_{k,l} := \max\{y_{1,k}x_{1,l}, \ldots, y_{N,k}x_{N,l}\}$ für $k \in \{1, \ldots, m\}$, $l \in \{1, \ldots, n\}$, $W := (w_{k,l})$. (*Palmsche Lernregel*)
Dies wird etwas übersichtlicher, wenn wir $A \vee B$ für zwei Matrizen $A, B \in \mathbb{R}^{m \times n}$ als die Matrix definieren die durch komponentenweise Maximumbildung entsteht. Dann gilt nämlich:
$W(0) := 0 \in \mathbb{R}^{m \times n}$, $W(t+1) := W(t) \vee \Delta W(t+1)$ für $t \in \{0, \ldots, N-1\}$, $W = W(N)$, wenn wir setzen $\Delta W(t+1) = y_{t+1} x_{t+1}^T$. $\Delta W(t+1)$ ist eine binäre $m \times n$-Matrix, deren

$$l\text{-te Spalte} = \left\{ \begin{array}{ll} y_{t+1} & \text{, wenn } x_{t+1,l} = 1 \\ 0 & \text{, sonst} \end{array} \right\} \text{ und deren } k\text{-te Zeile} = \left\{ \begin{array}{ll} x_{t+1}^T & \text{, wenn } y_{t+1,k} = 1 \\ 0 & \text{, sonst} \end{array} \right\}.$$

W entsteht also, indem man alle $\Delta W(t+1)$ durch komponentenweise Maximumbildung verknüpft. Da im binären Fall die Maximumbildung lediglich eine $\vee$-Verknüpfung ist, läßt sich W also sehr schnell und einfach berechnen.

Beispiel:

$$y_1 = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} \underbrace{\begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}}_{\Delta W(1) = W(1)} = x_1^T \qquad y_2 = \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \underbrace{\begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix}}_{\Delta W(2)} = x_2^T \qquad W = W(2) = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

Bemerkung. In der Literatur wird meist die transponierte Matrix dargestellt.

Verwendung von W als Assoziativspeicher

Wie sieht die k-te Komponente ξ_k von Wx_t aus? Es gilt

$$\xi_k = \sum_{l=1}^{n} w_{k,l}x_{t,l} = \sum_{l=1}^{n} \max\{y_{1,k}x_{1,l}, \ldots, y_{N,k}x_{N,l}\} = \sum_{l=1}^{n} \max\{y_{1,k}x_{1,l}x_{t,l}, \ldots, y_{N,k}x_{N,l}x_{t,l}\}$$

$$\begin{cases} \geq & \sum_{l=1}^{n} x_{t,l}^2, \text{ wenn } y_{t,k} = 1 \\ \leq & \#\{l : x_{t,l} = 1 \text{ und mindestens ein } x_{j,l} = 1 \text{ mit } j \neq t\}, \text{ wenn } y_{t,k} = 0 \end{cases}$$

Fazit: Um von ξ_k durch eine Schwellwertentscheidung auf den Wert von $y_{t,k}$ schließen zu können, sollte es ein n_1 geben, so daß

1) in jedem x_j mindestens n_1 Komponenten gleich 1 sind,

2) in jedem x_j eine Komponente existiert, die den Wert 1 hat, während die gleiche Komponente der anderen x_i, $i \neq j$, stets 0 ist.

Dann kann man $\Theta := n_1$ setzen und $\eta_k := \begin{cases} 1 & \text{, wenn } \xi_k \geq n_1 \\ 0 & \text{, sonst} \end{cases}$.

Um eine Fehlertoleranz gegen Störungen der Eingangsmuster zu erhalten, kann man 2) ersetzen durch

2') in jedem x_j gibt es mindestens r Komponenten mit dem Wert 1, während diegleichen Komponenten der anderen x_i alle 0 sind.

Wie kann man 1) und 2) (oder 2')) erfüllen?
Während 1) eine Mindestzahl von Einsen vorschreibt, verlangt 2), daß die x_i nicht zu viele gemeinsame Einsen haben. Bei einer vorgegebenen Anzahl N von Prototypen x_i können diese Forderungen höchstens dann erfüllt werden, wenn die Vektoren x_i sehr lange sind (also die Anzahl n der Komponenten sehr groß ist) und viele Nullen enthalten. Solche binären Muster nennt man *spärlich kodiert*.

Solche binären Assoziativspeicher mit spärlicher Kodierung wurden von Willshaw (1969) u.a. und insbesondere von Palm (1980) untersucht. Im Fall $n = m$ (x_i und y_i gleiche Dimension) besagen seine Resultate grob folgendes. Wählt man die x_i und y_i zufällig aus den binären Vektoren der Länge n mit $\log_2(n)$ Einsen, so kann man für große n im Mittel etwa $N = n^2 \frac{\ln(2)}{\log_2(n)^2}$ Prototypen (x_i, y_i) speichern, bevor der Ausgabevektor des Speichers bei Eingabe von x_i in mehr als einer Komponente von y_i abweicht. Für $n \to \infty$ kann so ein Speicher im Mittel etwa 0.69 Bit pro Matrixelement speichern (im Gegensatz zu 1 Bit pro Matrixelement für traditionelle Speicher).

8.3 Bemerkung. Wenn die Eingangsmuster eines binären Assoziativspeichers unterschiedlich viele Einsen haben, ist es sinnvoll, die Entscheidungsschwelle Θ vom Eingangsmuster x abhängig zu machen, z.B.

$$\Theta = \text{Anzahl der Einsen in } x \quad \text{oder} \quad \Theta = \text{Anzahl der Einsen in } x + b$$

zu wählen, wobei b ein für alle Eingangsmuster fester Offset ist.

Kapitel 9

Rückgekoppelte Assoziativspeicher

Wir betrachten im folgenden *autoassoziative* — d.h. für die partielle Abbildung $f : D \to B$ gilt $f = \mathrm{Id}_D$ — Speicher für bipolare Muster. Der Speicher bestehe aus n Speicherzellen, die jeweils 1 Bit, nämlich den Wert $+1$ oder -1, speichern können. Diese Speicherzellen werden auch Neuronen genannt. Sei $z_i(t) \in Z = \{\pm 1\}$ der Zustand des i-ten Neurons zum Zeitpunkt $t \in \mathbb{N}$. Dann wird der Zustand des Speichers zum Zeitpunkt t vollständig beschrieben durch den Vektor $z(t) = (z_1(t), \ldots, z_n(t))^T \in Z^n$.

Damit der Speicher als Assoziativspeicher funktionieren kann, sei eine (zeitlich diskrete) *Dynamik* auf Z^n gegeben, das ist eine Folge $(\phi_t)_{t\in\mathbb{N}}$ von Abbildungen $\phi_t : Z^n \to Z^n$.

9.1 Idee der Funktionsweise

Das Eingabemuster $(\xi_1, \ldots, \xi_n) \in Z^n$ wird als Anfangszustand $z(0)$ gewählt. Von nun an verändern sich die Neuronenzustände gemäß der Rekursionsgleichung

$$z(t+1) = \phi_t(z(t)) \qquad \text{für } t \in \mathbb{N}_0.$$

Die Dynamik $(\phi_t)_t$ soll so gewählt sein, daß die Folge $(z(t))_t$ konvergiert, was für den Zustandsraum Z^n einfach bedeutet, daß sie ab einem $t_0 \in \mathbb{N}$ konstant wird, also $\phi_t(z(t_0)) = z(t_0)$ gilt und somit $z(t_0)$ ein Fixpunkt der Dynamik ist. $z(t_0)$ ist das zu ξ assoziierte Muster, daß ausgegeben wird.

Literaturhinweise

Derartige neuronale Netze wurden von vielen Forschern untersucht, insbesondere von Amari in den 70ern und von Hopfield in den 80ern. Es gibt starke Zusammenhänge zur statistischen Physik (Spingläser). Recht ausführliche Darstellungen findet man in [3] und [5].

Natürlich werden nicht irgendwelche Dynamiken verwendet, sondern meist eine recht spezielle Art, die wir gleich vorstellen wollen. Selbst dann ist es nicht leicht, eine Übersicht über das Verhalten des Netzes zu erhalten. Dies gelingt erst, indem man Energiefunktionen oder Lyapunov-Funktionen verwendet.

Weil Hopfield Anfang der 80er neuen Schwung in die Untersuchung dieser rückgekoppelten Netze gebracht hat, werden sie meist *Hopfield-Netze* genannt; auch wenn die Dynamik nicht immer genau dieselbe wie in Hopfields Arbeiten ist.

9.2 Definition der verwendeten Dynamiken

Üblicherweise sind die Abbildungen ϕ_t der Dynamik auf einfache Weise aus Funktionen $\varphi_1, \ldots, \varphi_n : Z^n \to Z$ aufgebaut. Im folgenden seien $W = (w_{ij} \in \mathbb{R}^{n \times n}$, $\Theta = (\Theta_1, \ldots, \Theta_n)^T \in \mathbb{R}^n$ und für $z = (z_1, \ldots, z_n)^T \in Z^n$

$$\varphi_i(z) = \begin{cases} +1 & \text{, falls } \sum_{j=1}^n w_{ij} z_j - \Theta_i > 0 \\ -1 & \text{, falls } \sum_{j=1}^n w_{ij} z_j - \Theta_i < 0 \\ z_i & \text{, sonst} \end{cases} .$$

9.2.1 Synchrone Dynamik

Jedes ϕ_t hat die Gestalt $(\varphi_1, \ldots, \varphi_n)$, d.h. zu jedem Zeitpunkt t werden alle Komponenten des Zustandsvektors neu berechnet nach der Regel $z_(t+1) = \varphi_i(z(t))$. Wir werden kurz $\phi_t(z) = \text{signum}(Wz - \Theta)$ schreiben.

9.2.2 Asynchrone Dynamik

a) Deterministisch: Sei σ eine Permutation von $\{1, \ldots, n\}$. $\pi_j : Z^n \to Z$ sei die Projektion auf die j-te Komponente. Für jedes $t \in \mathbb{N}$ sei die j-te Komponente von

$$\phi_t = \begin{cases} \varphi_j & \text{, wenn } j = \sigma(t \mod n) \\ \pi_j & \text{, sonst} \end{cases} .$$

Zu jedem Zeitpunkt wird nur eine Komponente des Zustandsvektors gemäß $z_(t+1) = \varphi_i(z(t))$ verändert. Der Index wird zyklisch durchgezählt.

b) Zufällig: Wähle zu jedem Zeitpunkt t zufällig ein $j \in \{1, \ldots, n\}$ aus und setze

$$z_i(t+1) = \begin{cases} z_i(t) & \text{, falls } i \neq j \\ \varphi_i(z(t)) & \text{, sonst} \end{cases} .$$

9.2.3 Block-sequentielle Dynamik

Durchlaufe zyklisch eine Partition $I_1, \ldots, I_l$ von $\{1, \ldots, n\}$ und arbeite auf jedem I_k synchron.

9.2.4 Vereinbarung. Im folgenden sei $(\phi_t)_t$ stets eine der obigen Dynamiken.

9.3 Definitionen

$\xi \in Z$ heißt *Fixpunkt* von (ϕ_t), wenn $\xi = \phi_t(\xi)$ für alle $t \in \mathbb{N}_0$ gilt. Dies ist äquivalent dazu, daß $\varphi_i(\xi) = \xi_i$ für jedes $i \in \{1, \ldots, n\}$, und somit unabhängig davon, ob die Dynamik synchron oder asynchron ist. Für $x, y \in Z$ sei $d(x,y) = \#\{i \in \{1, \ldots, n\} \mid x_i \neq y_i\} = \frac{1}{2} \sum_{i=1}^n |x_i - y_i|$ der *Hamming-Abstand*. $B(x,r) = \{y \in Z \mid d(x,r) \leq r\}$ sei die Kugel mit Radius r und Mittelpunkt x.

Für jedes $z(1) \in Z$ heißt die durch $z(t+1) = \phi_t(z(t))$ definierte Folge $(z(t))_{t \in \mathbb{N}_0}$ der in $z(0)$ beginnende *Pfad*; $z(0)$ heißt *Anfangspunkt* des Pfades. Ein Pfad $(z(t))_t$ heißt *zyklisch*, wenn es $t_0, T \in \mathbb{N}$ gibt mit $z(t+T) = z(t)$ für jedes $t \geq t_0$; ansonsten heißt er *zyklenfrei*.

Für jeden Fixpunkt $\xi \in Z$ sei der *Attraktionsbereich* $A(\xi)$ die Menge der Anfangspunkte aller Pfade, die durch ξ laufen (und dann konstant gleich ξ sind).

$A_1(\xi) = \{x \in Z \mid \varphi_0(x) = \xi\}$ heißt der *direkte Attraktionsbereich* von ξ.

$B_1(\xi) =$ größte Kugel $B(\xi, r)$ in $A_1(\xi)$ heißt das *Bassin direkter Attraktion* von ξ. Ihr Radius heißt *direkter Attraktionsradius*.

$B_k(\xi) =$ größte Kugel $B(\xi, r)$, so daß $\phi_{k-1} \circ \ldots \circ \phi_0(x) = \xi$ für jedes $x \in B(\xi, r)$; sie heißt *Attraktionsbassin k-ter Ordnung*.

9.4 Beispiele

a) $W = \begin{pmatrix} 2 & 1 \\ 1 & \frac{1}{2} \end{pmatrix}$, $\Theta = 0$, synchrone Dynamik, $n = 2$. Es gelten

$$\phi_0\left((1,1)^T\right) = (1,1)^T,\ \phi_0\left((-1,-1)^T\right) = (-1,-1)^T,\ \phi_0\left((1,-1)^T\right) = (1,1)^T,\ \phi_0\left((-1,1)^T\right) = (-1,-1)^T,$$

also sind $\xi = (1,1)^T$ und $\eta = (-1,-1)^T$ Fixpunkte und $B_1(\xi) = \{\xi\} \neq \{\xi, (1,-1)^T\} = A_1(\xi)$.

b) $W = \begin{pmatrix} 1 & -2 \\ -2 & 1 \end{pmatrix}$, $\Theta = 0$, $n = 2$, $\xi = (1,-1)^T$ und $\eta = (-1,1)^T$ sind Fixpunkte.

Wählt man eine asynchrone Dynamik, in der die erste vor der zweiten Komponente geändert wird, so gilt $\phi_0\left((-1,-1)^T\right) = (1,-1)^T = \xi$ und $\phi_0\left((1,1)^T\right) = (1,-1)^T = \eta$. Ändert dagegen die Dynamik die zweite vor der ersten Komponente, so gilt $\phi_0\left((-1,-1)^T\right) = (-1,1)^T = \eta$ und $\phi_0\left((1,1)^T\right) = (1,-1)^T = \xi$.
Die Attraktionsbassins hängen also bei asynchroner Dynamik von der Reihenfolge ab.

9.5 Forderungen beim Einsatz als Autoassoziativspeicher

Die Eingabemuster müssen geeignet in Z^n kodiert werden, und die Dynamik soll so gewählt werden, daß gilt:

9.5.1 Die Prototypen sind Fixpunkte.

9.5.2 Es gibt keine anderen Fixpunkte (sogenannte *parasitären* Fixpunkte).

9.5.3 Jeder Pfad ist zyklenfrei und trifft einen Fixpunkt.

9.5.4 Die Attraktionsbereiche sollten den Vorstellungen über die Ähnlichkeit von Mustern entsprechen.

Um diese Forderungen zu erreichen, können folgende Größen geeignet gewählt werden: n = Dimension der Muster, die Matrix W und der Schwellenvektor Θ.

Wir werden im folgenden zu jeder Forderung — soweit möglich — einige Ergebnisse darstellen. Wir beschränken uns dabei auf deterministische Dynamiken.

9.6 Konstruktion von Dynamiken, die vorgegebene Prototypen als Fixpunkte haben

Wir erinnern:

$$\xi \in Z^n \text{ Fixpunkt} \Leftrightarrow \phi_i(\xi) = \xi_i \text{ für jedes } i \in \{1,\ldots,n\} \Leftrightarrow \xi = \text{signum}(W\xi - \Theta),$$

wobei signum die komponentenweise Vorzeichenbildung sei, allerdings mit der Konvention, daß bei jeder Komponente der Wert 0 durch ξ_j ersetzt wird.

Wir betrachten zunächst den Fall $\Theta = 0$. Dann git: ξ Fixpunkt $\Leftrightarrow \xi = \text{signum}(W\xi - \Theta)$. Seien $x_1, \ldots, x_N \in Z^n$ linear unabhängige Prototypen. $X = (x_1, \ldots, x_N)$ hat dann den Rang $N \leq n$.
Gesucht ist ein $W \in \mathbb{R}^{n \times n}$ mit $X = \text{signum}(WX)$.

Eine triviale Lösung ist $W = \text{Id}$. Aber dann ist jeder Punkt in Z^n ein Fixpunkt mit Attraktionsradius 0. Das ist meist nicht erwünscht.

9.6.1 Die Hebbsche Regel

Setze $W = \frac{1}{N}\sum_{k=1}^{N} x_k x_k^T = \frac{1}{N} XX^T$. Für alle Koeffizienten $w_{i,j}$ gilt $|w_{i,j}| \leq 1$.

Sind die Prototypen $x_1, \ldots, x_N$ orthogonal, so gilt $Wx_j = \frac{1}{N}\|x_j\|^2 x_j$, also ändert die Multiplikation mit W die Vorzeichen der Komponenten nicht, folglich ist jedes x_j ein Fixpunkt, allerdings auch jedes $-x_j$.

Sind die $x_1, \ldots, x_N$ nicht orthogonal, so besteht eine Chance, daß sie Fixpunkte sind, wenn ihre Kreuzkorrelationen, d.h. Skalarprodukte $x_i^T x_j$ klein sind. In jedem Fall ist die mit der Hebbschen Regel konstruierte Matrix W symmetrisch und positiv semidefinit. Für die Diagonalelemente $w_{i,i}$ gilt $w_{i,i} = \frac{1}{N}\sum_{k=1}^{N} x_{k,i}^2 = 1$. Manchmal werden die Diagonalelemente von W gleich 0 gesetzt. (Hebbregel mit Nulldiagonale)

9.6.2 Die Pseudoinversen-Regel oder Projektions-Regel

Sei X^+ die Pseudoinverse von X, also $X^+ = (X^T X)^{-1} X^T$. Dann ist $X^+ X = \mathrm{Id}_n$. Setze $W = XX^+$. Dann gelten

1) $WX = X$.

2) $W^2 = XX^+XX^+ = XX^+ = W$.

3) $W^T = W$, wegen $W^T = (XX^+)^T = (X^+)^T X^T = X(X^T X)^{-1} X^T = XX^+ = W$, weil $X^T X$ symmetrisch.

Folglich ist W die orthogonale Projektion auf den von $x_1, \ldots, x_p$ aufgespannten Teilraum. Für die Diagonalelemente $w_{i,i}$ gilt wegen $W^2 = W$

$$w_{i,i} = \sum_{j=1}^{n} w_{i,j} w_{j,i} \overset{3)}{=} \sum_{j=1}^{n} w_{i,j}^2,$$

also $w_{i,i}(1 - w_{i,i}) = \sum_{j \neq i} w_{i,j}^2$ und folglich $0 \leq w_{i,i} \leq 1$ und $|w_{i,j}| \leq 1$ für alle i, j.

W ist positiv semidefinit, weil $z^T W z \overset{2)}{=} z^T W W z \overset{3)}{=} z^T W^T W z = \|Wz\|^2 \geq 0$.

Wegen 1) ist jeder Prototyp x_i ein Fixpunkt. Allerdings ergibt sich natürlich, daß für jeden Vektor $z \in Z^n$, der eine Linearkombination der $x_1, \ldots, x_p$ ist, $Wz = z$ gilt und z somit ein Fixpunkt ist. Manchmal werden die Diagonalelemente von W durch 0 ersetzt (z.B. in Hopfields Arbeiten); man bildet also $\tilde{W} = W - \mathrm{Diag}(w_{1,1}, \ldots, w_{n,n})$. Dies ist ebenfalls eine symmetrische Matrix. Es gilt $\tilde{W}X = X - \mathrm{Diag}(w_{1,1}, \ldots, w_{n,n})X$, und wegen $0 \leq w_{i,i} \leq 1$ ändert die Multiplikation mit $\tilde{W}$ das Vorzeichen der Komponenten der x_i nicht. Daher sind die x_i auch bei Verwendung von $\tilde{W}$ Fixpunkte.

9.6.3 Bemerkung

Um zu verhindern, daß mit $z \in Z^n$ auch stets $-z$ ein Fixpunkt ist, wählt man den Schwellwertvektor Θ von 0 verschieden.

9.7 Die Energiefunktion

Gegeben sei ein Hopfieldnetz mit Zustandsraum Z^n, Matrix $W \in \mathbb{R}^{n \times n}$ und Schwellwertvektor $\Theta \in \mathbb{R}^n$; $W = (w_{i,j})$, $\Theta = (\Theta_i)$. Dann wird ihm die folgende sogenannte *Energiefunktion* zugeordnet. Für $z \in Z^n$ sei

$$H(z) = -\frac{1}{2} z^T W z + z^T \Theta.$$

Ist W symmetrisch, so gilt für $z = (z_1, \ldots, z_n)^T \in Z^n$

$$H(z) = -\sum_{i>j} w_{i,j} z_i z_j + \sum_{i=1}^{n} z_i \Theta_i - \frac{1}{2} \sum_{i=1}^{n} w_{i,i},$$

weil stets $z_i z_i = 1$.

Der Name Energiefunktion wurde wegen der formalen Ähnlichkeit zur physikalischen Energie gewählt.

9.8 Asynchrone Dynamik

9.8.1 Satz. Ist W symmetrisch und sind die Diagonalelemente nichtnegativ, so ist jeder Pfad der asynchronen Dynamik zyklenfrei.

Beweis. Ohne Einschränkung werde zum Zeitpunkt t die erste Komponente des Zustandsvektors $z(t)$ gemäß φ_1 wie in 9.2.2 geändert. Wir verwenden folgende Schreibweise: $x = z(t)$, $y = z(t+1)$ $x = (x_1, \tilde{x}^T)^T$, $y = (y_1, \tilde{y}^T)^T$, $\Theta = (\Theta_1, \tilde{\Theta}^T)^T$, $W = \begin{pmatrix} w_{11} & \tilde{w}^T \\ \tilde{w} & \tilde{W} \end{pmatrix}$, also $\tilde{x} = \tilde{y}$. Dann gilt:

$$\begin{aligned}
H(z(t)) &= -\frac{1}{2}\left(w_{11}x_1^2 + x_1\tilde{w}^T\tilde{x} + \tilde{x}^T\tilde{w}x_1 + \tilde{x}^T\tilde{W}\tilde{x}\right) + \Theta_1 x_1 + \tilde{\Theta}^T\tilde{x} \\
&= -\frac{1}{2}\left(w_{11}x_1^2 + 2x_1\tilde{w}^T\tilde{x} + \tilde{x}^T\tilde{W}\tilde{x}\right) + \Theta_1 x_1 + \tilde{\Theta}^T\tilde{x}. \\
H(z(t+1)) &= -\frac{1}{2}\left(w_{11}y_1^2 + 2y_1\tilde{w}^T\tilde{y} + \tilde{y}^T\tilde{W}\tilde{y}\right) + \Theta_1 y_1 + \tilde{\Theta}^T\tilde{y}. \\
H(z(t+1)) - H(z(t)) &= -\frac{1}{2}\left(w_{11}y_1^2 - 2x_1y_1w_{11} + w_{11}x_1^2\right) - x_1y_1w_{11} + w_{11}x_1^2 - (y_1 - x_1)\left(\tilde{w}^T\tilde{x} - \Theta_1\right) \\
&= -\frac{1}{2}w_{11}\left(y_1 - x_1\right)^2 - (y_1 - x_1)\left(w_{11}x_1 + \tilde{w}^T\tilde{x} - \Theta_1\right) \\
&= -\frac{1}{2}w_{11}\left(z_1(t+1) - z_1(t)\right)^2 - (z_1(t+1) - z_1(t))\left(\textstyle\sum_{j=1}^{n} w_{1,j}z_j(t) - \Theta_1\right).
\end{aligned}$$

Sei $\Delta H = H(z(t+1)) - H(z(t))$ und $A_k(z_j(t)) = \sum_{j=1}^{n} w_{k,j} z_j(t) - \Theta_k$.

Es können folgende Fälle eintreten:

$A = 0$: Dann ist nach 9.2.2 $z_1(t+1) = z_1(t)$, also $z(t+1) = z(t)$ und $\Delta H = 0$.

$A \neq 0$: $\left\{ \begin{array}{l} \text{Ist } A > 0, \text{ so ist } z_1(t+1) = +1, \text{ also } z_1(t) = -1 \\ \text{Ist } A < 0, \text{ so ist } z_1(t+1) = -1, \text{ also } z_1(t) = +1 \end{array} \right\}$ wenn $z(t+1) \neq z(t)$, sonst $\Delta H = 0$.

In beiden Fällen ist $(z_1(t+1) - z_1(t)) \cdot A_1(z_j(t)) > 0$ und $\Delta H < 0$, weil $w_{11} \geq 0$.

Damit ist gezeigt, daß jede Änderung des Zustandsvektors eine echte Abnahme von H bewirkt. Daher müssen die Pfade zyklenfrei sein. □

9.8.2 Korollar. Ist W symmetrisch und sind die Diagonalelemente nichnegativ, so erreicht jeder Pfad der asynchronen Dynamik nach endlich vielen Schritten einen Fixpunkt.

Beweis. Weil Z^n endlich ist, ist H nach unten beschränkt und ebenso

$$\mu = \min\{|A_i(z_j)| \mid i \in \{1, \ldots, n\} \text{ und } z \in Z^n \text{ mit } A_i(z_j) \neq 0\} > 0.$$

Aus obigem Beweis folgt, daß $\Delta H = H(z(t+1)) - H(z(t)) \leq -\mu$, wenn $z(t+1) \neq z(t)$. Also muß der Pfad nach endlich vielen Schritten stationär werden. □

9.8.3 Korollar. Wird die Gewichtsmatrix eines Hopfield-Netzes gemäß der Hebbregel oder der Pseudoinversenregel (mit oder ohne Nulldiagonale) gebildet, so ist jeder Pfad der asynchronen Dynamik zyklenfrei und erreicht nach endlich vielen Schritten einen Fixpunkt.

Da die Energie beschränkt ist und jede Änderung kleiner als eine negative Konstante ist, erreicht jeder Pfad nach endlich vielen Schritten einen Fixpunkt. Die Anzahl der Schritte kann man nach oben abschätzen.

9.8.4 Satz. Ist W symmetrisch und sind die Diagonalelemente positiv, so erreicht jeder Pfad der asynchronen Dynamik nach höchstens $\frac{1}{d}(M + \sum_{i=1}^{n} |\Theta_i|)$ Schritten einen Fixpunkt. Dabei ist $d = \min\{w_{i,i} \mid i = 1, \ldots, n\}$ und M die Summe der $\lfloor \frac{n^2}{4} \rfloor$ größten Koeffizientenbeträge $|w_{i,j}|$ mit $i \neq j$. Gilt zusätzlich $|w_{i,j}| \in \{0, 1\}$ für alle (i, j), so reichen $\frac{5}{2}n^2$ Schritte, um einen Fixpunkt zu erreichen.

9.9 Synchrone Dynamik

9.9.1 Satz. Ist W symmetrisch und auf $\{-1, 0, 1\}^n$ positiv semidefinit, so ist jeder Pfad der synchronen Dynamik zyklenfrei.

Beweis. Für zwei aufeinanderfolgende Zustände $z(t)$ und $z(t+1)$ gilt

$$H(z(t+1)) - H(z(t)) = -\frac{1}{2}z(t+1)^T W z(t+1) + z(t+1)^T\Theta + \frac{1}{2}z(t)^T W z(t) - z(t)^T\Theta$$

$$\begin{aligned} = \; & -\frac{1}{2}z(t+1)^T W z(t+1) + \frac{1}{2}z(t+1)^T W z(t) + \frac{1}{2}z(t)^T W z(t+1) - \frac{1}{2}z(t)^T W z(t) + z(t+1)^T\Theta \\ & - \frac{1}{2}z(t+1)^T W z(t) - \frac{1}{2}z(t)^T W z(t+1) + \quad z(t)^T W z(t) - z(t)^T\Theta \\ = \; & -\frac{1}{2}\underbrace{(z(t+1) - z(t))^T W (z(t+1) - z(t))}_{(1)} - \underbrace{(z(t+1) - z(t))^T (Wz(t) - \Theta)}_{(2)}, \text{ da } W = W^T. \end{aligned}$$

Ist $z(t+1) \neq z(t)$, so haben nach 9.2.1 die Komponenten von $z(t+1) - z(t)$ dasselbe Vorzeichen wie die entsprechenden Komponenten von $Wz(t) - \Theta$; folglich ist der Term (2) positiv. Aufgrund der Zustandsänderungen bei der synchronen Dynamik gilt $z(t+1) - z(t) \in \{-2, 0, 2\}^n$, und daher ist der Term (1) nichtnegativ. Somit ist $H(z(t+1)) - H(z(t)) < 0$ und infolgedessen ist jeder Pfad zyklenfrei. □

9.9.2 Korollar. Ist W symmetrisch und auf $\{-1, 0, 1\}^n$ positiv semidefinit, so erreicht jeder Pfad der synchronen Dynamik nach endlich vielen Schritten einen Fixpunkt.

Beweis. Weil Z^n endlich ist, ist H beschränkt und ΔH gleich 0 oder kleiner als eine negative Konstante. □

9.9.3 Korollar. Wird die Gewichtsmatrix eines Hopfield-Netzes gemäß der Hebbregel oder der Pseudoinversenregel gebildet, so ist jeder Pfad der synchronen Dynamik zyklenfrei und erreicht nach endlich vielen Zeitschritten einen Fixpunkt.

9.9.4 Satz. Ist W symmetrisch und positiv definit, so erreicht jeder Pfad der synchronen Dynamik nach höchstens $\frac{1}{\lambda_{\min}}(M + \sum_{i=1}^{n} |\Theta_i|)$ Schritten einen Fixpunkt. Dabei ist $\lambda_{\min}$ der kleinste Eigenwert von W und M die Summe der $\lfloor \frac{n^2}{4} \rfloor$ größten Koeffizientenbeträge $|w_{i,j}|$ mit $i \neq j$.

Beweis. Sind $z(t)$ und $z(t+1)$ zwei aufeinanderfolgende Zustände eines Pfades, so gilt wie im Beweis von 9.9.1 $\Delta H = H(z(t+1)) - H(z(t)) = -\frac{1}{2}(z(t+1) - z(t))^T W (z(t+1) - z(t)) - (z(t+1) - z(t))^T (Wz(t) - \Theta)$.

Für symmetrische Matrizen W gilt $\lambda_{\min} = \min\left\{\frac{x^T W x}{\|x\|^2} \mid x \neq 0\right\}$. Weil W positiv definit ist, gilt $\lambda_{\min} > 0$. Für $x \in \{-2, 0, 2\}^n$, $x \neq 0$, gilt $x^T W x \geq \|x\|^2 \lambda_{\min} \geq 4\lambda_{\min}$. Ist $z(t) \neq z(t+1)$, so ist $z(t+1) - z(t) \in \{-2, 0, 2\}^n$, und es folgt $|\Delta H| \geq \frac{1}{2} \cdot 4\lambda_{\min} = 2\lambda_{\min}$. Man schätzt dann die Differenz Δ von $\max H$ und $\min H$ durch $\Delta \leq 2(M + \sum_{i=1}^{n} |\Theta_i|)$ ab. Damit ergibt sich, daß jeder Pfad nach höchstens $\frac{\Delta}{\lambda_{\min}}$ Schritten in einer absoluten Minimumstelle von H landet. □

9.10 Eine Abschätzung der Attraktionsradien

Der Radius $r_k(\xi)$ des Attraktionsbassins k-ter Ordnung $B_k(\xi)$ eines Fixpunktes $\xi \in Z^n$ heißt der *Attraktionsradius k-ter Ordnung.*

Wir setzen in diesem Abschnitt voraus, daß $\Theta = 0$ und $|w_{i,j}| \leq 1$ für alle i, j. Wir betrachten nur die synchrone Dynamik. $z \mapsto \phi(z) = \text{signum}(Wz)$. Für $z \in Z^n$ sei $h_i(z) = \sum_{j=1}^{n} w_{i,j} z_j$ das sogenannte *synaptische Potential* der i-ten Komponente.

9.10.1 Satz. Ist ξ ein Fixpunkt, so gilt $r_1(\xi) \geq \frac{1}{2} \min\{|h_i(\xi)| \mid i = 1, \ldots, n\}$.

Beweis. Da ξ ein Fixpunkt ist, gilt $\text{signum}(\xi_i) = \text{signum}(h_i(\xi))$ für jedes i. Sei $z \in Z^n$ mit

$$d(z, \xi) \leq \frac{1}{2} \min\{|h_i(\xi)| \mid i = 1, \ldots, n\}.$$

Wegen $|w_{i,j}| \leq 1$ ist dann $|h_j(z) - h_j(\xi)| \leq 2d(z, \xi) \leq \min\{|h_i(\xi)| \mid i = 1, \ldots, n\}$. Folglich haben $h_j(z)$ und $h_j(\xi)$ das gleiche Vorzeichen, und es gilt $\phi(z) = \xi$, also $z \in B_1(\xi)$.

Infolgedessen gilt $r_1(\xi) \geq \frac{1}{2} \min\{|h_i(\xi)| \mid i = 1, \ldots, n\}$. □

9.10.2 Satz. Sei ξ ein Fixpunkt. Die Koordinaten seien so numeriert, daß $|h_1(\xi)| \leq |h_2(\xi)| \leq \ldots \leq |h_n(\xi)|$ gilt. Für $1 \leq k \leq m$ sei R_k definiert durch

$$R_1 = \left\lfloor \frac{1}{2} |h_1(\xi)| \right\rfloor, \quad R_k = \left\lfloor \frac{1}{2} |h_{R_{k-1}+1}(\xi)| \right\rfloor$$

Dann gilt: $r_k(\xi) \geq R_k$.

Beweis. $r_1(\xi) \geq R_1$ wurde im vorhergehenden Satz gezeigt. Wir führen eine Induktion nach k durch.

Sei $z \in B(\xi, R_k)$. Wegen $|w_{i,j}| \leq 1$ und $|h_j(z) - h_j(\xi)| \leq 2R_k \leq |h_{R_{k-1}+1}(\xi)|$ gilt $\text{signum}(h_i(z)) = \text{signum}(h_i(\xi))$ für $i \geq R_{k-1} + 1$. Somit unterscheidet sich $y = \text{signum}(Wz)$ von ξ in höchstens R_{k-1} Komponenten, also $y \in B(\xi, R_{k-1})$. Nach Induktionsvoraussetzung ist $r_{k-1}(\xi) \geq R_{k-1}$, also $y \in B_{k-1}(\xi)$, und somit $z \in B_k(\xi)$ und $r_k(\xi) \geq R_k$. □

9.11 Bemerkungen

1. Für die block-sequentielle Dynamik kann man ähnliche Sätze zeigen wie für die (a)synchrone.
2. Es ist schwierig, allgemeine Aussagen über die Attraktionsbereiche oder die Kapazität (d.h. wieviele Prototypen als Fixpunkte gespeichert werden können) oder parasitäre Fixpunkte zu machen. Mit stochastischen Techniken ist es möglich, solche Aussagen in asymptotischen Sinn zu beweisen.

9.12 Fixpunkte als lokale Energieminima

Es wurde gezeigt, daß unter geeigneten Bedingungen an W die Energie $H(z) = -\frac{1}{2} z^T W z + z^T \Theta$ längs der Pfade streng monoton abnimmt, bis der Pfad einen Fixpunkt erreicht.

Ist jeder Fixpunkt x auch ein lokales Minimum in dem Sinne, daß $H(z) \geq H(x)$ für alle z mit $d(z, x) = 1$, wobei d der Hammingabstand ist? Nein; nicht notwendigerweise.

Sowohl für die synchrone wie die asynchrone Dynamik gilt:
Unterscheiden sich x und z in der i-ten Komponente (und nur in dieser wegen $d(z, x) = 1$), so ergibt sich wie in 9.8.1 bzw. 9.9.1

$$\Delta H = H(z) - H(x) = -\frac{1}{2} w_{ii} (z_i - x_i)^2 - (z_i - x_i) A_i(x_j).$$

Weil x ein Fixpunkt ist, muß $z_i \neq \text{signum}(A_i(x_j))$ sein; weil $z_i \neq x_i$, also $z_i = -x_i$, ist somit der zweite Term $-(z_i - x_i)(A_i(z_j))$ stets nicht negativ.

Der erste Term ist nach den Voraussetzungen in 9.8.1 (alle $w_{ii} \geq 0$) bzw. 9.9.1 ($W \geq 0$) stets nicht positiv und kann bewirken, daß $\Delta H < 0$ ist.

9.12.1 Korollar. Ist W symmetrisch und sind alle Diagonalelemente 0, so ist jeder Fixpunkt ein lokales Minimum der Energiefunktion $H(z) = -\frac{1}{2} z^T W z + z^T \Theta$.

Bemerkung. Dies legt die Idee nahe, Hopfield-Netze zur Minimierung von Funktionen zu verwenden. Und in der Tat ist es bei manchen Optimierungsproblemen möglich, das zu minimierende Kostenfunktional als Energiefunktion eines Hopfield-Netzes aufzufassen.

9.13 Binäre Hopfield-Netze

Manchmal ist es in Anwendungen naheliegender, binäre statt bipolarer Neuronen zu verwenden. Man wählt also $\tilde{Z}^n = \{0,1\}^n$ als Zustandsraum und definiert die Dynamik (synchron oder asynchron) wie in 9.2, allerdings mit einem modifizierten φ_i, nämlich

$$\varphi_i(z) = \begin{cases} 1 & \text{, falls } A_i(z_j) > 0 \\ 0 & \text{, falls } A_i(z_j) < 0 \\ z_i & \text{, sonst} \end{cases} .$$

Man prüft leicht nach, daß dann obige Resultate, insbesondere 9.8.1 und 9.9.1, entsprechend gelten, und zwar mit derselben Energiefunktion $H(z) = \frac{1}{2} z^T W z + z^T \Theta$.

Kapitel 10

Kontinuierliche dynamische Systeme

Bei der Untersuchung rückgekoppelter neuronaler Netze wurden vielfach Neuronen mit kontinuierlichen Zustandswerten und kontinuierlicher Zeit betrachtet. Die Ausgangswerte der Neuronen (und ihre inneren Aktivitätswerte) sind die Komponenten der Lösung eines gewöhnlichen Differentialgleichungssystem; im folgenden auch kurz (gew.) Dgl.system genannt. Das kontinuierliche Analogon zu dem in vorigen Abschnitt eingeführten Hopfield-Netz wird durch ein Differentialgleichungssystem folgender Gestalt gegeben:

$$\frac{du}{dt} = -u + W\varphi(u) - \Theta.$$

Dabei ist $u(t)$ $\mathbb{R}^n$-wertig, $W \in \mathbb{R}^{n\times n}$, $\Theta \in \mathbb{R}^n$, $\varphi = (\varphi_1(x_1), \ldots, \varphi_n(x_n))$ mit monoton wachsenden $\mathcal{C}^1$-Funktionen φ_i. u beschreibt die innere Aktivität der Neuronen, $\varphi(u)$ ihre Ausgangswerte. Die Funktionsweise so eines Netzes ist analog der eines diskreten rückgekoppelten Netzes. Der Eingabevektor u_0 wird als Anfangswert der Differentialgleichung genommen, zu dem es eine eindeutige Lösung u mit $u(0) = u_0$ gibt. Die Differentialgleichung sollte die Eigenschaft haben, daß $u(t)$ für $t \to \infty$ konvergiert. Dann ist $\lim_{t\to\infty} u(t)$ der Ausgabevektor des Netzes.

Um solche Netze einsetzen zu können, muß man also die (kontinuierliche) Dynamik eines Differentialgleichungssystems gut kennen.

Bemerkungen.

1. Solche gew. Dgl.systeme lassen sich oft in Form mechanischer oder elektronischer Systeme realisieren.

2. Diskretisiert man obige Dgl., indem man immer um einen Zeitpunkt $\Delta t = 1$ weitergeht, so erhält man $u(t+1) = u(t) + \Delta u(t) = u(t) - u(t) + W\varphi(u(t)) - \Theta = W\varphi(u(t)) - \Theta$. Mit $z(t) = \varphi(u(t))$ wird daraus $z(t+1) = \varphi(Wz(t) - \Theta)$, also eine zur synchronen Dynamik 9.2.1 ähnliche Dynamik.

Bei unseren (spärlichen) Untersuchungen diskreter Dynamiken war die Einführung einer sogenannten Energiefunktion H sehr nützlich, die längs jedes Pfades monoton abnimmt. Zur Untersuchung kontinuierlicher Dynamiken wurden solche Funktionen von Lyapunov 1892 in seiner Dissertation eingeführt. Wir wollen die Grundidee skizzieren.

10.1 Definition. Ein *globaler Fluß* oder *globales dynamisches System* auf einer offenen Menge $G \subset \mathbb{R}^n$ ist eine stetig diff'bare Abbildung $\psi : \mathbb{R} \times G \to G$ derart, daß für die Abbildungen $\psi_t : G \to G$, $x \mapsto \psi(t, x)$ gilt:

a) $\psi_0 : G \to G$ ist die Identität.

b) $\psi_{t+s} = \psi_t \circ \psi_s$ für alle $s, t \in \mathbb{R}$.

Bemerkungen. Die Abbildungen ψ_t sind stetig diff'bar. Wegen $\psi_t \circ \psi_{-t} = \psi_{-t} \circ \psi_t = \mathrm{Id}$ sind sie sogar Diffeomorphismen. Die Abbildung $t \mapsto \psi(t, x)$ heißt der *Pfad* durch x.

Beispiel. $\psi_t = e^{tA}$, wobei $A \in \mathcal{L}(\mathbb{R}^n, \mathbb{R}^n)$ und $G = \mathbb{R}^n$.

10.2 Zuordnung einer Differentialgleichung

$\psi : \mathbb{R} \times G \to G$ sei ein globaler Fluß. Definiere $f : G \to \mathbb{R}^n$ durch $f(x) = \frac{d}{dt}\psi(t,x)|_{t=0}$. f ist ein *Vektorfeld* auf G. Für jedes $x \in G$ ist $u_x(t) = \psi(t,x)$ eine Lösung des *autonomen Dgl.systems* $u' = f(u)$.

Bemerkung. Oft bezeichnet man auch dieses Vektorfeld oder diese Dgl. als dynamisches System. Das ist insofern gerechtfertigt, als man ψ daraus wieder berechnen kann. Das geht sogar in gewissem Sinn für eine beliebige autonome Dgl.

Beispiel. $\psi_t = e^{tA}$, $f(x) = Ax$, lineare Dgl.

10.3 Der Fluß einer autonomen Differentialgleichung

$G \subset \mathbb{R}^n$ sei offen, $f : G \to \mathbb{R}^n$ sei lokal Lipschitzstetig. Dann gibt es zu jedem Anfangswert x eine eindeutig bestimmte maximale Lösung u der Dgl. $u' = f(u)$ mit $u(0) = x$, und diese Lösungen hängen stetig diff'bar von dem Anfanswert ab. Etwas präziser bedeutet dies folgendes:

$$\Omega = \{(t,x) \in \mathbb{R} \times G \mid \exists \text{ Lösung } u \text{ von } u' = f(u) \text{ die auf } \left\{ \begin{array}{l} [0,t], \text{ falls } t > 0 \\ [t,0], \text{ falls } t < 0 \end{array} \right\} \text{ definiert ist und } u(0) = x \text{ gilt}\}$$

Ω ist offen. Und es gibt eine stetig diff'bare Abbildung $\psi : \Omega \to G$ derart, daß für jedes $x \in G$ die Abbildung $u(t) = \psi(t,x)$ die maximale Lösung von $u' = f(u)$ mit $u(0) = x$ ist.

ψ heißt der *Fluß* der Dgl. Es gilt $\psi(0,x) = x$ für jedes $x \in G$. Allerdings ist nicht immer $\Omega = \mathbb{R} \times G$, d.h. die Lösungen der Dgl. existieren nicht für alle $t \in \mathbb{R}$.

Ist jedoch jede Lösung auf ganz $\mathbb{R}$ definiert, so ist der Fluß der Dgl. ein globales dynamisches System im Sinn von 10.1, denn $\psi_{t+s} = \psi_t \circ \psi_s$ gilt dann wegen der eindeutigen Lösbarkeit.

Sprechweise: Unter einem *dynamischen System* auf $G \subset \mathbb{R}^n$ versteht man sowohl ein lokal Lipschitzstetiges Vektorfeld $f : G \to \mathbb{R}^n$ als auch die entsprechende Dgl. $u' = f(u)$ oder den zugehörigen Fluß $\psi : \Omega \to G$.

Wir betrachten von nun an nur dynamische Systeme $u' = f(u)$ mit stetig diff'barem f.

10.4 Definition. $u' = f(u)$ sei ein dynamisches System auf G.

a) $x \in G$ heißt *stationärer Punkt* oder *Gleichgewichtspunkt* oder *Fixpunkt* oder *singulärer Punkt* des dynamischen Systems, wenn $f(x) = 0$.

b) Ein stationärer Punkt x heißt *Senke*, wenn alle Eigenwerte von $Df(x)$ negative Realteile haben.

10.5 Satz. $u' = f(u)$ sei ein dynamisches System, $f \in \mathcal{C}^1(G)$. Ist x eine Senke des Systems, so gibt es eine Umgebung $\mathcal{U} \subset G$ von x, so daß gilt:
Jede Lösung u mit $u(0) \in \mathcal{U}$ ist für alle $t \geq 0$ definiert, bleibt für $t \geq 0$ in $\mathcal{U}$, und $\lim_{t\to\infty} u(t) = x$.

Bemerkung. Anschaulich Interpretation des letzten Satzes: Alle Lösungskurven, die in der Nähe einer Senke beginnen, laufen in die Senke hinein.

10.6 Definition. $u' = f(u)$ sei ein dynamisches System, $f \in \mathcal{C}^1(G)$. Ein Fixpunkt x des Systems heißt *stabil*, wenn jede Umgebung V von x in G eine Umgebung $\mathcal{U}$ von x enthält, so daß jede Lösung u, deren Anfangswert $u(0)$ in $\mathcal{U}$ liegt, für alle $t > 0$ definiert ist und in $\mathcal{U}$ bleibt. Gilt zusätzlich $\lim_{t\to\infty} u(t) = x$ für alle Lösungen u mit $u(0) \in \mathcal{U}$, so heißt x *asymptotisch stabil*.

Will man ein dynamisches System als Assoziativspeicher einsetzen, so ist es wichtig, die Fixpunkte, insbesondere die asymptotisch stabilen, zu bestimmen. Für eine spezielle Klasse von dynamischen Systemen gibt es wohlbekannte Kriterien.

10.7 Definition. Ein dynamisches System $u' = f(u)$ auf G heißt ein *Gradientensystem*, wenn es eine zweimal stetig diff'bare Funktion $H : G \to \mathbb{R}$ gibt mit $f = -\nabla H$, also das dynamische System die Gestalt $u' = -\nabla H(u)$ hat. H heißt *Potential-* oder *Energiefunktion* des Systems.

10.8 Satz. Ist x ein isoliertes Minimum der stetig diff'baren Funktion $H : G \to \mathbb{R}$, so ist x ein asymptotisch stabiler Fixpunkt des dynamischen Systems $u' = -\nabla H(u)$.

Die Aussage ist wohlbekannt und bildet die Grundlage für jede Art von Gradientenabstiegsverfahren zur Suche von Minima. Sie folgt aus dem etwas allgemeineren Satz 10.9.

Wir sind jedoch an der umgekehrten Anwendung interessiert: Wir wollen asymptotisch stabile Fixpunkte finden, indem wir Minima von H suchen. Lyapunov überlegte sich 1892, daß auch für dynamische Systeme, die nicht Gradientensysteme sind, manchmal ähnliches möglich ist.

Bezeichnung. $u' = f(u)$ sei ein dynamisches System auf G. Ist $\mathcal{U} \subset G$ offen und $H : \mathcal{U} \to \mathbb{R}$ diff'bar, so sei $\dot{H} : \mathcal{U} \to \mathbb{R}$ definiert durch $\dot{H}(x) = DH(x) \cdot f(x)$. Ist u die Lösung von $u' = f(u)$ mit $u(0) = x$, so gilt $\dot{H}(x) = \frac{d}{dt} H(u(t))|_{t=0}$, d.h. $\dot{H}(x)$ ist die Ableitung von H längs der Lösungskurve u im Punkt 0.

10.9 Satz. $u' = f(u)$ sei ein dynamisches System auf G. $x \in G$ sei ein Fixpunkt, $\mathcal{U} \subset G$ sei eine Umgebung von x, $H : \mathcal{U} \to \mathbb{R}$ sei stetig, $H : \mathcal{U}\backslash\{x\} \to \mathbb{R}$ sei diff'bar. Es gelte $H(x) = 0$ und $H(y) > 0$ für $y \neq x$.

a) Ist $\dot{H}(y) \leq 0$ für $y \in \mathcal{U}\backslash\{x\}$, so ist x stabil.

b) Ist $\dot{H}(y) < 0$ für $y \in \mathcal{U}\backslash\{x\}$, so ist x symptotisch stabil.

Im Fall a) heißt H eine *Lyapunov-Funktion* für das System und im Fall b) eine *strikte* Lyapunov-Funktion.

Beweis. Sei $V \subset G$ eine Umgebung von x. $\mathcal{U} \cap V$ enthält eine kompakte Umgebung K von x, zum Beispiel eine Kugel $\overline{B(x, \delta)}$. Dann ist $\alpha = \min\{H(y) \mid y \in \partial(K)\} > 0$ und $\mathcal{U}_1 = \{y \in K^\circ \mid H(y) < \alpha\}$ ist eine offene Umgebung von x. Wegen $\dot{H} \leq 0$ ist $H(u(t))$ für jede Lösungskurve $u(t)$ monoton abnehmend. Folglich bleibt jede Lösungskurve $u(t)$ mit $u(0) \in \mathcal{U}_1$ für $t > 0$ in dem Kompaktum $\{y \in \mathcal{U}_1 \mid H(y) \leq H(u(0))\} \Subset \mathcal{U}_1$ und muß daher für alle $t > 0$ definiert sein. Damit ist gezeigt, daß x stabil ist.

Ist $\dot{H}(y) < 0$ für $y \in \mathcal{U}\backslash\{x\}$, so nimmt H längs jeder Lösungskurve streng monoton wachsend ab. Angenommen, $u(t)$ ist eine Lösungskurve mit $u(0) \in \mathcal{U}_1$, die für $t \to \infty$ nicht gegen x konvergiert. Dann gibt es eine Folge $(t_j)_{j \in \mathbb{N}}$ und ein $z_0 \in \mathcal{U}_1$ mit $z_0 \neq x$ und $\lim_{j \to \infty} u(t_j) = z_0$. Es gilt:

$$(*) \quad H(u(t)) > H(z_0) \text{ für alle } t > 0 \text{ und } \lim_{j \to \infty} H(u(t_j)) = H(z_0).$$

Sei ψ der zu $u' = f(u)$ gehörende Fluß. Für $s > 0$ ist $H(\psi(s, z_0)) < H(z_0)$, weil H auf der Lösungskurve $t \mapsto \psi(t, z_0)$ durch z_0 streng monoton abnimmt. Sei $s_0 > 0$. Da ψ stetig ist, gibt es eine Umgebung W von z_0, so daß $H(\psi(s_0, y)) < H(z_0)$ für alle $y \in W$. Weil H auf den Lösungskurven $t \mapsto \psi(t, y)$ mit $y \in W$ streng monoton abnimmt, gilt sogar $H(\psi(s, y)) < H(z_0)$ für alle $s \geq s_0$ und $y \in W$. Weil $u(t_j)$ gegen z_0 konvergiert, ist $u(t_j) \in W$ für genügend große j und somit

$$H(z_0) > H(\psi(s, u(t_j))) = H(\psi_s(u(t_j))) = H(\psi_s(\psi_{t_j}(u(0)))) = H(\psi_{s+t_j}(u(0))) = H(u(s + t_j))$$

im Widerspruch zu $(*)$. □

10.10 Bemerkungen.

1. Ist $u' = -\nabla H(u)$ ein Gradientensystem und x ein isoliertes Minimum von H, so ist $H - H(x)$ eine strikte Lyapunov-Funktion für das System und somit x ein asymptotischer Fixpunkt. Jeder Punkt von $G\backslash\{\nabla H = 0\}$ ist im Einzugsbereich eines Fixpunktes.

2. Leider gibt es kein allgemeines Rezept, wie man für ein allgemeines dynamisches System bei einem Fixpunkt eine Lyapunov-Funktion finden kann und somit eine (asymptotische) Stabilität feststellen kann. Beschreibt das dynamische System ein physikalisches, mechanisches oder elektrisches System, so ist oft dessen Energie eine Lyapunov-Funktion.

   Getreu dem Motto: Physikalische Systeme suchen Energieminima auf, weil sie (asymptotisch) stabil sind.

3. In den Beweisen des vorigen Abschnitts, daß die Pfade der diskreten Dynamiken unter geeigneten Voraussetzungen zyklenfrei sind, benutzten wir ein diskretes Analogon der Lyapunov-Funktionen, nämlich eine Funktion H, die längs der Pfade monoton fällt.

10.11 Beispiel.

$$\left.\begin{aligned} u_1' &= -u_1 + u_1 u_2 = u_1(u_2 - 1) \\ u_2' &= -u_2 + u_1 u_2 = u_2(u_1 - 1) \end{aligned}\right\} = f(u_1, u_2)$$

Der Punkt $(0,0)^T$ ist ein Fixpunkt. Das System ist kein Gradientensystem, denn wäre $f = -\nabla H$, so müßte $\frac{\partial f_1}{\partial x_2} = -\frac{\partial^2 H}{\partial x_1 \partial x_2} = -\frac{\partial^2 H}{\partial x_2 \partial x_1} = \frac{\partial f_2}{\partial x_1}$ gelten; es ist aber $\frac{\partial f_1}{\partial x_2}(u) = u_1 \neq u_2 = \frac{\partial f_2}{\partial x_1}(u)$.

$H(u_1, u_2) = u_1^2 + u_2^2$ ist eine strikte Lyapunov-Funktion in $B(0,1)$:

$$\dot{H}(u) = \frac{\partial H}{\partial x_1}(u) \cdot f_1(u) + \frac{\partial H}{\partial x_2}(u) \cdot f_2(u) = 2u_1 \cdot u_1(u_2 - 1) + 2u_2 \cdot u_2(u_1 - 1) = 2\left(u_1^2(u_2 - 1) + u_2^2(u_1 - 1)\right),$$

also $\dot{H}(u) < 0$ für $(u_1, u_2) \in B(0,1)$.

Somit ist 0 ein asymptotisch stabiler Fixpunkt, und $B(0,1)$ liegt in seinem Einzugsbereich.

Kapitel 11

Anwendungen von Hopfield-Netzen

Wie am Ende des 9. Kapitels erwähnt wurde, kann man versuchen, Optimierungsprobleme mit Hilfe von Hopfieldnetzes zu lösen. Dazu muß das zu minimierende Kostenfunktional als Energiefunktion eines Hopfield-Netzes dargestellt werden; dann folgt man einem Pfad der Dynamik bis in einen Fixpunkt. Dabei gibt es folgende prinzipiellen Probleme:

1. Bei vielen Optimierungsproblemen wird ein Kostenfunktional unter Nebenbedingungen minimiert. Solche Nebenbedingungen müssen irgendwie mit in die Energiefunktion eingebaut werden. Dabei kann es passieren, daß die Energiefunktion viele Minima bekommt, die keine beste Lösungen des Optimierungsproblems darstellen (suboptimale Lösungen).

2. Oft gelingt es nicht, die Energiefunktion so zu definieren, daß die Diagonalelemente der Gewichtsmatrix W verschwinden. Damit ist nicht garantiert, daß die Pfade der Dynamik ein lokales Energieminimum oder überhaupt einen Fixpunkt erreichen.

3. Selbst wenn die Pfade ein lokales Energieminimum erreichen, ist nicht gewährleistet, daß man bei irgendeinem Anfangszustand eine optimale Lösung erreicht.

11.1 Das Problem des Handelsreisenden (*TSP*)

Gegeben seien r Städte $S_1, \ldots, S_r$. $d_{i,j}$ sei der Abstand zwischen S_i und S_j.

Gesucht ist eine Permutation $\sigma \in \mathcal{T}_r$, so daß $L(\sigma) = \sum_{i=1}^{r} d_{\sigma(i),\sigma(i+1)}$ möglichst klein ist; dabei werde die Addition der Indizes modulo r ausgeführt, also $r+1 \equiv 1$ und $\sigma(r+1) = \sigma(1)$.

Die Rundreise σ werde als Matrix $(x_{i,j}) \in \{0,1\}^{r\times r}$ kodiert in folgender Weise:

$$x_{i,j} = \delta_{i,\sigma(j)} = \begin{cases} 1 & \text{, wenn } \sigma(j) = i \\ 0 & \text{, sonst} \end{cases} .$$

Es gilt:

$$\begin{aligned} H_1(x) &= \frac{1}{2}\sum_{i=1}^{r}\sum_{j=1}^{r}\sum_{k=1}^{r} d_{i,j}(x_{i,k}x_{j,k+1} + x_{i,k+1}x_{j,k}) \\ &= \frac{1}{2}\sum_{i=1}^{r}\sum_{j=1}^{r}\sum_{k=1}^{r} d_{i,j}(\delta_{i,\sigma(k)}\delta_{j,\sigma(k+1)} + \delta_{i,\sigma(k+1)}\delta_{j,\sigma(k)}) \\ &= \frac{1}{2}\sum_{k=1}^{r}(d_{\sigma(k),\sigma(k+1)} + d_{\sigma(k+1),\sigma(k)}) = \sum_{k=1}^{r} d_{\sigma(k),\sigma(k+1)} \\ &= L(\sigma). \end{aligned}$$

Für jedes $(i,j) \in \{1,\ldots,r\}\times\{1,\ldots,r\}$ wählen wir ein binäres Neuron; wir benutzen also den binären Zustandsraum $Z^n = \{0,1\}^n$ mit $n = r^2$. Jede Matrix $x \in \{0,1\}^{r\times r}$ interpretieren wir als Zustand $z \in Z^n$, indem wir und die Spalten von x untereinandergeschrieben denken. Wir werden jedoch die Zustände weiter als Matrizen schreiben und $(i,j) \in \{1,\ldots,r\}^2$ als Indizes der Komponenten und Neuronen verwenden.

Wählt man als Synapsengewicht zwischen Neuron (i,m) und Neuron (j,l) $\tilde{w}_{(i,m),(j,l)} = -d_{i,j}(\delta_{m+1,l} + \delta_{m,l+1})$, so ist die daraus gebildete Matrix $\tilde{W}$ symmetrisch und hat verschwindende Diagonalelemente, und H_1 ist gerade die zugehörige Energiefunktion. Weil die Diagonalelemente verschwinden, ist die Dynamik zyklenfrei.

Leider ist damit das TSP nicht vollständig spezifiziert; denn es ist noch nicht berücksichtigt, daß jede Stadt auf der Rundreise genau einmal besucht werden soll. Dies bedeutet, daß die der Rundreise entsprechende Matrix x in jeder Zeile und in jeder Spalte genau eine Eins hat. Wir versuchen, daß diese Nebenbedingungen dadurch zu erzwingen, daß wir geeignete Terme zu H_1 addieren, die ebenfalls minimiert werden müssen. Definiere

$$H_2(x) = \frac{1}{2}\sum_{j=1}^{r}\left(\sum_{i=1}^{r} x_{i,j} - 1\right)^2 \qquad \text{und} \qquad H_3(x) = \frac{1}{2}\sum_{i=1}^{r}\left(\sum_{j=1}^{r} x_{i,j} - 1\right)^2.$$

Es gilt $H_2 \geq 0$ und $H_3 \geq 0$, sowie

$$\begin{aligned} H_2(x) = 0 \quad &\Leftrightarrow \quad \text{in jeder Spalte von } x \text{ steht genau eine 1} \\ H_3(x) = 0 \quad &\Leftrightarrow \quad \text{in jeder Zeile von } x \text{ steht genau eine 1.} \end{aligned}$$

Des weiteren

$$\begin{aligned} H_2(x) &= \frac{1}{2}\sum_{j=1}^{r}\left(\left(\sum_{i=1}^{r} x_{i,j}\right)^2 - 2\sum_{i=1}^{r} x_{i,j} + 1\right) \\ &= \frac{1}{2}\sum_{j=1}^{r}\left(\sum_{i=1}^{r}\sum_{k\neq i} x_{i,j}x_{k,j} + \sum_{i=1}^{r} x_{i,j}^2 - 2\sum_{i=1}^{r} x_{i,j} + 1\right) \\ &= \frac{1}{2}\sum_{j=1}^{r}\left(\sum_{i=1}^{r}\sum_{k\neq i} x_{i,j}x_{k,j} - \sum_{i=1}^{r} x_{i,j} + 1\right), \quad \text{weil } x_{i,j}^2 = x_{i,j}. \\ H_3(x) &= \frac{1}{2}\sum_{i=1}^{r}\left(\sum_{j=1}^{r}\sum_{k\neq j} x_{i,j}x_{i,k} - \sum_{j=1}^{r} x_{i,j} + 1\right). \end{aligned}$$

Die Konstanten 1 ziehen wir ab, weil sie für die Minimierung belanglos sind, und setzen für $\lambda > 0$

$$H(x) = H_1(x) + \lambda\left(H_2(x) + H_3(x) - r\right) = \frac{1}{2}\sum_{i=1}^{r}\sum_{j=1}^{r}\sum_{k=1}^{r}\sum_{l=1}^{r} w_{(i,k),(j,l)}x_{i,k}x_{j,l} - \sum_{i=1}^{r}\sum_{j=1}^{r} x_{i,j}\Theta_{(i,j)},$$

wobei alle $\Theta_{(i,j)} = 1$ und $w_{(i,k),(j,l)}$ die Koeffizienten einer Matrix $W = \left(w_{(i,k),(j,l)}\right)_{(i,k),(j,l)\in\{1,\ldots,r\}^2}$ sind, die symmetrisch ist und deren Diagonalelemente verschwinden. $\Theta = \left(\Theta_{(i,j)}\right) = (1,\ldots,1)^T \in \mathbb{R}^{r^2}$.

Durch W und Θ wird eine asynchrone Dynamik beschrieben, deren Pfade lokale Minima von H erreichen. Es ist allerdings nicht garantiert, daß jedes lokale Minimum x tatsächlich eine optimale Rundreise, also eone Lösung des TSP ist. Es kann nämlich folgendes passieren:
x stellt keine erlaubte Reise dar, weil eine Stadt fehlt oder weil zu einem Zeitpunkt mehr als eine Stadt besucht wird. Oder x ist eine erlaubte Rundreise, aber nicht die kürzeste.

Mit dem Parameter λ kann man beeinflussen, ob eher erlaubte Rundreisen, die vielleicht zu lang sind, gefunden werden oder ob primär die Länge der Reise minimiert wird. Um bessere Resultate zu erreichen, gibt es raffinierte Ansätze; siehe dazu die in Brause und Cichochi-Unbehanen genannte Literatur.

11.2 Restauration gestörter Signale

11.2.1 Typische Probleme

a) Ein Ton- oder ein Bildsignal ist von Rauschen überlagert. Rekonstruiere das unverrauschte Originalsignal.

b) Ein Foto ist verwischt, weil die Kamera sich bewegt oder das Bild nicht scharf gestellt war. Versuche diese Störung zu eliminieren.

Beide Probleme sind natürlich höchstens dann lösbar, wenn man genügend Vorwissen über die Art der Störung hat. Wir wollen hier nur einen sehr einfachen Fall betrachten.

Sei $f : [0,T] \to \mathbb{R}$ ein Tonsignal, $g = f + \nu$ sei das durch das Rauschen ν gestörte Signal. Wir nehmen nun an, daß das Rauschen deutlich schneller hin und her schwankt als f. Deshalb suchen wir als Schätzung für f eine $\mathcal{C}^2$-Funktion $\tilde{f}$, so daß $\|g - \tilde{f}\|^2 + \lambda\|\tilde{f}''\|^2$ möglichst klein ist, wobei $\|\cdot\|$ die L^2-Norm ist. $\lambda \geq 0$ ist ein Parameter, mit dem eingestellt werden kann, ob es wichtiger erscheint, daß $\tilde{f}$ möglichst wenig von g abweicht oder daß die Krümmung von $\tilde{f}$ nicht zu groß ist.

Um dieses Minimierungsproblem mit einem diskreten Hopfield-Netz lösen zu können, muß man sowohl den Definitionsbereich $[0,T]$ wie den Wertebereich der Funktionen diskretisieren.

Sei $T = m \cdot \Delta t$ mit $m \in \mathbb{N}$ und $\Delta t > 0$. Außerdem nehmen wir an, daß g und $\tilde{f}$ beschränkt sind und ohne Einschränkung nur Werte im Intervall $[0,Q]$ mit $Q \in \mathbb{N}$ (z.B. $Q = 255$) annehmen. Dann setzen wir $g_k = \lfloor g(k\Delta t) \rfloor$ für $1 \leq k \leq m$. Ersetzen wir die 2. Ableitung durch die 2. Differenzen, so ergibt sich das folgende diskrete Minimierungsproblem:

$$\text{Finde } u_k \in [\![0,Q]\!],\ 1 \leq k \leq m, \text{ so daß } \sum_{k=1}^{m}(g_k - u_k)^2 + \lambda \sum_{k=2}^{m-1}(u_{k-1} - 2u_k + u_{k+1})^2 \text{ möglichst klein ist.}$$

Das zu minimierende Funktional ist also ein quadratisches Polynom. Es ist jedoch noch nicht als Energiefunktion eines Hopfield-Netzes interpretierbar; denn die Variablen u_k sind weder binär noch bipolar, sondern können ganzzahlige Werte in $[0,Q]$ annehmen. Wir benötigen also noch eine geeignete binäre Darstellung der ganzen Zahlen in $[0,Q]$.

11.2.2 Binäre Darstellungen ganzer Zahlen

a) Dualdarstellung: Verwende die Folge der Koeffizienten der Dualentwicklung: $a = \sum_{j=0}^{l} \alpha_j 2^j \mapsto \alpha_l \ldots \alpha_0$. Diese Darstellung ist nicht sehr fehlertolerant; ein einzelner Bitfehler kann große Änderungen der Zahl bewirken.

b) Darstellung durch kanonische Einheitsvektoren: Sei $e_j \in \mathbb{R}^{Q+1}$, $e_j = (\delta_{j,i})_i = (0,\ldots,0,1,0,\ldots,0)^T$, $j \in \{0,\ldots,Q\}$. Verwende die Zuordnung $[\![0,Q]\!] \to \{e_0,\ldots,e_Q\}$, $a \mapsto e_a$.

c) Summendarstellung: Sei $\varphi : \{0,1\}^Q \to [\![0,Q]\!]$, $(\alpha_1,\ldots,\alpha_Q)^T \mapsto \sum_{j=1}^{Q} \alpha_j$. Als Summendarstellung einer Zahl $a \in [\![0,Q]\!]$ wird jede 0-1-Folge $\alpha \in \{0,1\}^Q$ mit $\varphi(\alpha) = a$ bezeichnet; es handelt sich also um eine nicht eindeutige Darstellung. Dies mag ungewohnt erscheinen; die Nichteindeutigkeit kann aber bei der Verwendung in Hopfield-Netzen Vorteile haben, weil mehr Punkte im Zustandsraum globale Energieminima sind. Diese Darstellung ist recht fehlertolerant.

d) Gruppenweise Summendarstellung: Stelle a beispielsweise als Dezimalzahl dar und verwende für jede Dezimale die Summendarstellung.

Siehe dazu auch [10].

Wir kommen jetzt zur Minimierung von $\sum_{k=1}^{m}(g_k - u_k)^2 + \lambda\sum_{k=2}^{m-1}(u_{k-1} - 2u_k + u_{k+1})^2$ zurück. Wir kodieren die u_k mit der Summendarstellung, indem wir für jedes k binäre Variablen $z_{k,i}$, $1 \leq i \leq Q$, einführen und $u_k = \sum_{i=1}^{Q} z_{k,i}$ setzen. Jedes $z_{k,i}$ fassen wir als Zustand eines binären Neurons auf.

Obige Funktion erhält damit die Gestalt

$$\begin{aligned}
H(z) &= \sum_{k=1}^{m}\left(g_k - \sum_{i=1}^{Q} z_{k,i}\right)^2 + \lambda \sum_{k=2}^{m-1}\left(\sum_{i=1}^{Q}(z_{k-1,i} - 2z_{k,i} + z_{k+1,i}\right)^2 \\
&= \sum_{k=1}^{m}\left(g_k^2 - 2\sum_{i=1}^{Q} g_k z_{k,i} + \sum_{i=1}^{Q}\sum_{j=1}^{Q} z_{k,i}z_{k,j}\right) + \\
&\qquad \lambda \sum_{k=2}^{m-1}\left(\sum_{i=1}^{Q}\sum_{j=1}^{Q}(z_{k-1,i} - 2z_{k,i} + z_{k+1,i})\cdot(z_{k-1,i} - 2z_{k,i} + z_{k+1,i})\right) \\
&= \sum_{k=1}^{m}\left(g_k^2 - 2\sum_{i=1}^{Q} g_k z_{k,i} + \sum_{i=1}^{Q}\sum_{j=1}^{Q} z_{k,i}z_{k,j}\right) + \\
&\qquad \lambda \sum_{k=2}^{m-1}\sum_{i=1}^{Q}\sum_{j=1}^{Q}(z_{k-1,i}z_{k-1,j} + 4z_{k,i}z_{k,j} + z_{k+1,i}z_{k+1,j} - 2z_{k-1,i}z_{k,j} + \\
&\qquad\qquad z_{k-1,i}z_{k+1,j} - 2z_{k,i}z_{k-1,j} - 2z_{k,i}z_{k+1,j} + z_{k+1,i}z_{k-1,j} - 2z_{k+1,i}z_{k,j}) \\
&= -\frac{1}{2}z^T W z + z^T\Theta + \sum_{k=1}^{m} g_k^2,
\end{aligned}$$

wobei $z = (z_{k,i})_{(k,i)\in I}$ und $\Theta = (\Theta_{k,i})_{(k,i)\in I}$ mit $I = \{1,\ldots,m\}\times\{1,\ldots,Q\}$, sowie $\Theta_{k,i} = -2g_k$, $W \in \mathbb{R}^{I\times I}$, $W = (w_{(k,i),(l,j)})$ mit

$$\begin{aligned}
w_{(k,i),(k,i)} &= -2(1+6\lambda), && 2 \le k \le m-1 \\
w_{(1,i),(1,i)} &= -2(1+\lambda) && \\
w_{(m,i),(m,i)} &= -2(1+\lambda) && \\
w_{(k,i),(l,j)} &= 0, && \text{falls } |k-l| > 2 \\
w_{(k,i),(l,j)} &= w_{(l,j),(k,i)}, && \text{falls } |k-l| \le 2
\end{aligned}$$

Die anderen Koeffizienten $w_{(k,i),(l,j)}$ kann man ausrechnen, indem man die obige Darstellung von E nach den Summanden $z_{k,i}z_{l,j}$ ordnet. Bemerkenswert ist, daß die Koeffizienten von W nicht von g und somit nicht vom Rauschen ν abhängen, sondern a priori ausgerechnet werden können. Θ dagegen hängt von g ab. Die Wahl von λ richtet sich nach dem erwarteten Signal-Rausch-Verhältnis SNR $= \frac{\|f\|^2}{\|\nu\|^2}$. Je kleiner SNR ist, desto größer wird man λ wählen, um zu erzwingen, daß das rekonstruierte Signal nicht zu sehr den Schwankungen von ν folgt. Weil die Diagonalelemente von W negativ sind, ist nicht garantiert, daß die Pfade der (synchronen oder asynchronen) Dynamik zyklenfrei sind oder in lokale Energieminima laufen. Die Energie kann längs eines Pfades sogar zunehmen. Deshalb verwendet man eine modifizierte Dynamik.

11.2.3 Modifizierte Dynamik

Sei $z(t)$ der Zustand zum Zeitpunkt t.

NEXT: Wähle (zufällig oder gemäß einer det. Regel) ein Neuron (k,i) aus.
Berechne $h_{k,i} = \sum_{(l,j)} w_{(k,i),(l,j)} z_{l,j}(t) + \Theta_{k,i}$ und
setze $u_{l,j} = z_{l,j}(t)$ für $(l,j) \neq (k,i)$,

$$u_{k,i} = \begin{cases} 1 & \text{, falls } h_{k,i} > 0 \\ 0 & \text{, falls } h_{k,i} < 0 \\ z_{k,i}(t) & \text{, sonst} \end{cases}, \; u = (u_{l,j}).$$

Berechne $\Delta H = H(u) - H(z(t))$.

Setze $z(t+1) = \begin{cases} u & \text{, falls } \Delta H \le 0 \\ z(t) & \text{, falls } \Delta H > 0 \end{cases}$.

GOTO NEXT

11.2.4 Verallgemeinerungen

a) Wir haben oben nur eindimensionale Signale betrachtet; in völlig analoger Weise kann man natürlich auch mehrdimensionale Signale behandeln. Man muß lediglich die Folge der Abtastzeitpunkte $t_k = k\Delta t$ durch die Punkte eines mehrdimensionalen Gitters ersetzen. Bilder einer Kamera bekommt man meist vom Videodigitizer schon in so einer Form geliefert, d.h. als Familie $(g_p)_{p\in P}$ der Helligkeitswerte g_p der Pixel p; die Menge P aller Pixel des Bildes wird meist mit $\{0,\ldots,N_x\}\times\{0,\ldots,N_y\}$ identifiziert; oft gilt $N_x = N_y = 511$.

In obigem Kostenfunktional werden dann die Summen über $k \in \{1,\ldots,m\}$ durch Summen über $p \in P$ ersetzt. Außerdem wird man in dem zweiten Summanden die diskrete Approximation der 2. Ableitung durch eine diskrete Approximation eines Differentialoperators 2. Ordnung, z.B. den Laplace-Operator, ersetzen. Man wird ihn meist linear wählen und in der Form $\sum_q \alpha_q u_{p+q}$ schreiben. Dann hat das zu minimierende Kostenfunktional die Gestalt

$$\sum_{p\in P}(g_p - u_p)^2 + \lambda \sum_{p\in P}\left(\sum_{q\in P}\alpha_q u_{p+q}\right)^2,$$

wobei die Addition der Indizes modulo $P = \{0,\ldots,N_x\}\times\{0,\ldots,N_y\}$ erfolgt (wodurch aber unerwünschte Randeffekte entstehen können). Man kann nun wie oben weiterverfahren.

b) Wie schon in 11.2.1.b) erwähnt ist in manchen Anwendungen das Signal nicht nur durch einen additiven Rauschterm gestört, sondern auch noch auf andere Weise verfälscht, z.B. ein Bild verschwommen, weil die Kamera nicht scharf gestellt war oder sich bewegt hat.

Wenn man die Art der Verfälschung kennt, kann man manchmal das Originalbild annähernd rekonstruieren. Nehmen wir beispielsweise an, $(f_p)_{p\in P}$ sei das Bild der unbewegten Kamera; bewegt sich die Kamera während der Aufnahme in horizontaler Richtung, so daß das Bild um r Pixel horizontal verschoben wird, so liefert die Kamera den Mittelwert, also $g_p = \frac{1}{r+1}\sum_{i=0}^{r} f_{p+(i,0)}$, $p \in P$, von Randeffekten abgesehen.

Allgemein läßt man Verfälschungen der Gestalt $g_p = \sum_{q\in P} h(q) f_{p+q}$ zu. Mit Hilfe der Fouriertransformation kann man sehen, daß nicht jede solche Verfälschung rückgängig gemacht werden kann.

Läßt man auch noch einen additiven Rauschterm ν zu, so hat das von der Kamera gelieferte Bild die Helligkeitswerte $g_p = \sum_{q\in P} h(q) f_{p+q} + \nu(p)$.

Daraus möchte man f möglichst gut rekonstruieren, indem man das Kostenfunktional

$$\sum_{p\in P}\left(g_p - \sum_{q\in P} h(q) u_{p+q}\right)^2 + \lambda \sum_{p\in P}\left(\sum_{q\in P}\alpha_q u_{p+q}\right)^2$$

minimiert; dabei sind die α_q wieder die Koeffizienten einer diskreten Approximation eines geeigneten Differentialoperators 2. Ordnung.

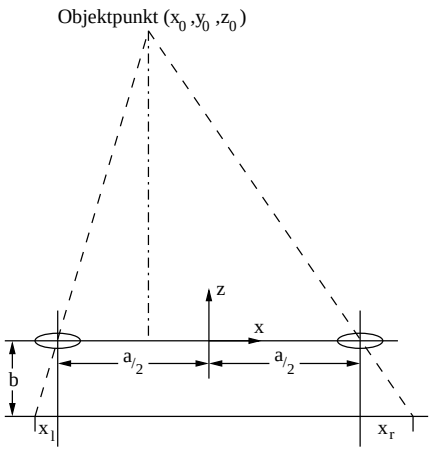
In diesem Funktional ersetzt man nun wieder die u_p durch eine Summe $\sum_{i=0}^{Q} z_{p,i}$ binärer Variablen und schreibt die entstehende Funktion in der Gestalt der Energiefunktion $-\frac{1}{2}z^T W z + z^T\Theta$ eines Hopfield-Netzes. Man kann sich überlegen, daß W nur von den h_q und α_q, aber nicht von g_q abhängt und somit a priori berechnet werden kann.

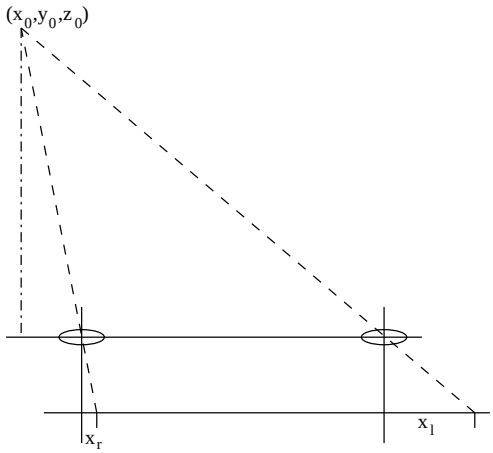
Solche Restaurationsmethoden sind nützlich, um Translations- und Rotationsbewegungen in Bildern, die von Raumfahrzeugen aufgenommen werden, zu eliminieren.

Literaturhinweis: [12]

11.3 Stereo Matching

Prinzip des Stereo-Sehens (mit parallelen optischen Achsen)





$$\frac{-x_l}{b} = \frac{\frac{1}{2}a + x_0}{z_0} \quad \text{und} \quad \frac{x_r}{b} = \frac{-x_0 + \frac{1}{2}a}{z_0}$$

$$\Rightarrow \quad \frac{x_r - x_l}{b} = \frac{a}{z_0} \quad \text{also} \quad z_0 = \frac{ab}{|x_r - x_l|}$$

$$\frac{x_r}{b} = \frac{-x_0 + \frac{1}{2}a}{z_0} \quad \text{und} \quad \frac{x_l}{b} = \frac{-x_0 - \frac{1}{2}a}{z_0}$$

$$\Rightarrow \quad \frac{x_r - x_l}{b} = \frac{a}{z_0} \quad \text{also} \quad z_0 = \frac{ab}{|x_r - x_l|}$$

Voraussetzung: Die Kameras seien so montiert, daß ihre optischen Achsen parallel sind.

In der Bildebene jeder Kamera sei ein Koordinatensystem eingeführt, dessen Ursprung der Schnittpunkt der optischen Achse der Kamera mit der Bildebene ist. Die Bildebenen der beiden Kameras und die x-Achsen der beiden Koordinatensysteme sollen übereinstimmen.
Bemerkung. Beim Menschen sind die optischen Achsen der Augen *nicht* parallel, sondern schneiden sich im Fixationspunkt.

Dann wird jeder Objektpunkt auf Punkte (x_l, y_l) und (x_r, y_r) abgebildet mit $y_l = y_r$, und aus der sogenannten *Disparität* $|x_r - x_l|$ kann die Entfernung $z_0 = \frac{ab}{|x_r - x_l|}$ des Objektpunkts berechnet werden, wenn der Abstand b der Bildebene zum Linsenmittelpunkt und der Abstand a der optischen Achsen bekannt ist.

Stereo-Matching-Problem: Finde Punkte p_l und p_r im linken bzw. rechten Bild, die Bilder desselben Objektpunkts sind. p_l und p_r heißen dann *korrespondierende* Punkte. Aus der Disparität von p_l und p_r kann man die Entfernung des Objektpunkts berechnen.
Bemerkung. Sind die Kameras – wie oben skizziert – montiert, so müssen die korrespondierende Punkte p_l und p_r dieselben Ordinaten haben. Das Stereo-Matching-Problem ist dann eindimensional.

Schwierigkeiten:

1. Ein Objektpunkt kann von einer Kamera aus zu sehen sein und von der anderen nicht.
2. Die Bildpunkte einer homogenen Objekt-Oberfläche sind nicht unterscheidbar.

Wegen 1. ist das Stereo-Matching-Problem nicht in dem Sinn lösbar, daß jeder Punkt in einem Bild stets (überhaupt oder genau) einen korrespondierenden Punkt im anderen hat. Wegen 2. wird man das Matching-Problem auf solche Bildpunkte beschränken, in denen die Bilder markante Eigenschaften haben wie starke Helligkeits- oder Farbänderungen. Wir werden im folgenden nur Grauwertbilder betrachten. Dann könnte man das Matching-Problem dadurch zu lösen versuchen, daß man jede Zeile des linken Bildes so auf die entsprechende Zeile des rechten Bildes transformiert, daß dabei dur Ableitungen der beiden Helligkeitsfunktionen (in Zeilenrichtung) möglichst gut zusammenpassen, d.h. sind f_l und f_r die beiden Helligkeitsfunktionen und so ist für jedes y eine Funktion $\delta(\cdot, y) : y$-te Zeile im linken Bild $\to \mathbb{R}$ gesucht, so daß $\int |\frac{\partial f_l}{\partial x}(x, y) - \frac{\partial f_r}{\partial x}(x + \delta(x, y), y)|^2 \, dx$ möglichst klein ist. δ wird dann als Disparität betrachtet.

Nun haben natürliche Bilder oft flächenhafte Strukturen (da reale Objekte endliche Ausdehnung haben), in denen sich die Disparität nut stetig und relativ langsam ändert (da sich die Entfernung der Punkte einer Objekt-Oberfläche stetig ändert außer an Silhouetten-Rändern). Daher kann man annehmen, daß die Funktion δ an den meisten Stellen stetig ist.

Um diese Überlegungen praktisch anwenden zu können, muß man alle Begriffe diskretisieren.

Jedes Bild habe $N_x \times N_y$ Pixel, die mit $P = [\![0, N_x - 1]\!] \times [\![0, N_y - 1]\!]$ indiziert seien. Jedes Pixel habe einen Grauwert aus $[\![0, Q-1]\!]$. In jedem Pixel (x, y) des linken (und des rechten) Bildes sei ein Schätzwert $f_l'(x, y+1)$ für $\frac{\partial f_l}{\partial x}(x, y)$ (bzw. $f_r'(x, y)$ für $\frac{\partial f_r}{\partial x}(x, y)$) gegeben. (Die Differenzen $f_l(x, y+1) - f_l(x, y)$ sind dafür nicht sehr geeignet, weil sie zu rauschempfindlich sind. Wir wollen hier jedoch nicht darauf eingehen, wie bessere Schätzer aussehen.)

Der Term $\int |\frac{\partial f_l}{\partial x}(x, y) - \frac{\partial f_r}{\partial x}(x + \delta(x, y), y)|^2 \, dx$ wird nun ersetzt durch $\sum_{x=0}^{N_x-1} |f_l'(x, y) - f_r'(x + \delta(x, y), y)|^2$, und die Matching-Aufgabe besteht darin, durch geeignete Wahl von δ

$$\sum_{(x,y)\in P} |f_l'(x, y) - f_r'(x + \delta(x, y), y)|^2 \quad \text{zu minimieren.} \tag{1}$$

Ohne weitere Bedingungen kann es passieren, daß das Minimum für ziemlich „verrückte“ δ angenommen wird. Wir wollen daher zusätzlich annehmen, daß die Disparität δ nur Werte in einem ganzzahligen Intervall $[\![0, D]\!]$ annimmt und nicht allzu sehr schwankt in dem Sinn, daß

$$\sum_{p\in P} \sum_{q\in(p+\mathcal{U})\cap P} \varphi(\delta(p), \delta(q)) \quad \text{nicht zu groß wird;} \tag{2}$$

dabei ist $\mathcal{U}$ eine geeignet gewählte Umgebung von $(0, 0)$ in $\mathbb{Z}^2$ und $\varphi(\delta(p), \delta(q)) = \begin{cases} 0 & \text{, wenn } \delta(p) = \delta(q) \\ 1 & \text{, sonst} \end{cases}$.

Minimierungsaufgabe für das Stereo-Matching

Sei $\lambda > 0$. Bestimme $\delta : P \to [\![0, D]\!]$ so, daß

$$\sum_{(x,y)\in P} \left(f_l'(x, y) - f_r'(x + \delta(x, y), y)\right)^2 + \lambda \sum_{p\in P} \sum_{q\in(p+\mathcal{U})\cap P} \varphi(\delta(p), \delta(q)) \tag{3}$$

minimal wird.

Mit dem Parameter λ gibt man an, um wieviel wichtiger die Minimierung des zweiten Terms gegenüber dem des ersten ist.

Um dieses Kostenfunktional als Energiefunktional eines Hopfield-Netzes interpretieren zu können, muß man die Werte von δ binärisieren. Wir gehen dabei ähnlich wie in 11.2.2.b) vor und wählen für jedes Pixel $p \in P$ $D+1$ binäre Neuronen, die wir mit $(p, 0)$ bis (p, D) indizieren. Ihre Zustände seien mit $z_{p,i}$, $i \in [\![0, D]\!]$ bezeichnet. Es gelte

$$\delta(p) = j \Leftrightarrow z_{p,a} = 1 \text{ und } z_{p,i} = 0 \text{ für } i \neq j.$$

Damit haben wir $N_x \cdot N_y \cdot (D+1)$ Neuronen, deren Zustandsraum $Z^{P\times[\![0,D]\!]}$, $Z = \{0, 1\}$, ist. Die möglichen Disparitätsfunktionen δ entsprechen genau den Zuständen $z \in Z^{P\times[\![0,D]\!]}$ mit der Eigenschaft, daß es für jedes $p \in P$ nur ein $j \in [\![0, D]\!]$ gibt mit $z_{p,j} = 1$; z ist also einfach die charakteristische Funktion des Graphen von δ in $P \in [\![0, D]\!]$. Die Funktion (3) kann man daher umformen zu

$$\sum_{(x,y)\in P} \sum_{j=0}^{D} \left(f_l'(x, y) - f_r'(x + j, y)\right)^2 z_{p,j} + \frac{\lambda}{2} \sum_{p\in P} \sum_{q\in(p+\mathcal{U})\cap P} \sum_{j=0}^{D} (z_{p,j} - z_{q,j})^2.$$

Um diese Funktion leichter als Energiefunktion eines Hopfield-Netzes darstellen zu können, nehmen wir eine kleine Modifikation vor. Wir nehmen an, die Nullumgebung $\mathcal{U}$ sei ein Quadrat $\mathcal{U} = [\![-a, a]\!]^2$ und lassen die

Summen statt über P nur noch über $P_a = [\![a, N_x - 1 - a]\!] \times [\![a, N_y - 1 - a]\!]$ laufen.

$$\sum_{(x,y)\in P_a} \sum_{j=0}^{D} \left(f_l'(x,y) - f_r'(x+j,y)\right)^2 z_{p,j} + \frac{\lambda}{2} \sum_{p\in P_a} \sum_{q\in\mathcal{U}} \sum_{j=0}^{D} (z_{p,j} - z_{p+q,j})^2$$

$$\begin{aligned} \sum_{p\in P_a} \sum_{q\in\mathcal{U}} (z_{p,j} - z_{p+q,j})^2 &= \sum_{p\in P_a} \sum_{q\in\mathcal{U}} (z_{p,j}^2 - 2 z_{p,j} z_{p+q,j} z_{p+q,j}^2) \\ &= \sum_{p\in P_a} A z_{p,j}^2 - 2 \sum_{p\in P_a} \sum_{q\in\mathcal{U}} z_{p,j} z_{p+q,j} + A \sum_{p\in P_a} z_{p,j}^2 + \text{ Terme } z_{p,j} \text{ mit } p \in P \backslash P_a, \end{aligned}$$

wobei $A = (2a+1)$.

Wir lassen die letzten Terme $z_{p,j}$ mit $p \in P\backslash P_a$ einfach weg und definieren

$$H(z) = \sum_{(x,y)\in P_a} \sum_{j=0}^{D} \left(f_l'(x,y) - f_r'(x+j,y)\right)^2 z_{p,j} + \lambda \sum_{p\in P_a} \sum_{j=0}^{D} \left(A z_{p,j}^2 - \sum_{q\in\mathcal{U}} z_{q,j} z_{p+q,j} \right).$$

Es gilt $H(z) = -\frac{1}{2} z^T W z + z^T \Theta$, wenn man

$$w_{(p,j),(r,i)} = -2\lambda \left(A\delta_{(p,j),(r,i)} - \sum_{q\in\mathcal{U}} \delta_{(p+q,j),(r,u)} \right) \text{ und } \Theta_{p,j)} = \left(f_l'(p) - f_r'(p+(j,0))\right)^2.$$

setzt.

$W = (w_{(p,j),(r,i)})$ ist eine symmetrische Matrix, weil $\mathcal{U}$ eine symmetrische Nullumgebung ist. Allerdings verschwinden die Diagonalelemente nicht, denn $w_{(p,j),(r,i)} = -2\lambda(A-1)$. Daher besteht dasselbe Problem wie schon in 11.2, daß nämlich die Energie längs der Pfade der asynchronen Dynamik stellenweise auch zunehmen kann. Deswegen wird man eine modifizierte Dynamik wie in 11.2.3 vorziehen. Allerdings sollte man noch weitere Veränderungen vornehmen, um zu bewirken, daß nur zulässige Zustände ausgesucht werden. Dabei soll ein Zustand z *zulässig* heißen, wenn er eine Disparität darstellt, wenn also für jedes p genau eines der $z_{p,j}$, $j \in [\![0, D]\!]$, den Wert 1 hat.

Modifizierte Dynamik

Sei $z(t)$ ein zulässiger Zustand zum Zeitpunkt t.

NEXT: Wähle (zufällig oder gemäß einer det. Regel) ein $p \in P$ (oder P_a aus.
Synchrone Neuberechnung der $z_{p,j}$:
Berechne $h_{p,j} = \sum_{(r,i)} w_{(p,j),(r,i)} z_{r,i}(t) + \Theta_{r,i}$ für jedes $j \in [\![0, D]\!]$.
Bestimme ein $j_0 \in [\![0, D]\!]$ so, daß $h_{p,j_0} = \max\{h_{p,j} \mid j \in [\![0, D]\!]\}$.
Setze $u_{p,j_0} = 1$
$u_{p,j} = 0$ für $j \neq j_0$ $\qquad u = (u_{q,j})$
$u_{q,j} = z_{q,j}(t)$ für alle j und $q \neq p$
Berechne $\Delta H = H(u) - H(z)$ und setze $z(t+1) = u$, falls $\Delta H \leq 0$, und $z(t+1) = z(t)$, sonst.
GOTO NEXT

Alternative Darstellung von H

Weil $z_{p,j}^2 = z_{p,j}$ gilt, kann man auch definieren

$$w_{(p,j),(r,i)} = 2\lambda \sum_{q\in\mathcal{U}} \delta_{(p+q,j),r(r,i)} \text{ und } \Theta_{(p,j)} = \left(f_l'(p) - f_r'(p+(j,0))\right)^2 + \lambda A$$

Dann ist $W = (w_{(p,j),(r,i)})$ symmetrisch und hat positive Diagonalelemente. Damit ist die asynchrone Dynamik zyklenfrei.

11.4 Bemerkungen

1. Wendet man Hopfield-Netze in obiger Weise an, so ist der Eingabevektor der Gewichtsvektor und der Ausgabevektor der Zustand, gegen den das Netz konvergiert. Da die Pfade oft schon nach relativ wenigen Schritten konstant werden, scheint man eine einfache Methode zur Lösung von Minimierungsproblemen zu haben.

   Das gilt jedoch nicht uneingeschränkt. Denn aus dem konkreten Optimierungsproblem müssen erst die Gewichte des Netzes berechnet werden. Das ist prinzipiell nicht schwierig, kann aber eine hohe (auch exponentielle) Rechnezeit erfordern, so daß insgesamt der Einsatz eines Netzes nicht unbedingt Vorteile gegenüber anderen Lösungsverfahren bringt.

2. In obigen Beispielen wurden globale Nebenbedingungen der Form $C(z) = 0$ oder $|C(z)| =$ minimal als sogenannte schwache Nebenbegingungen formuliert, d.h. statt die eigentliche Kostenfunktion H zu minimieren, wird die Funktion $H + \lambda|C|$ mit $\lambda > 0$ minimiert. Es kann günstig sein, λ von der Zeit abhängen zu lassen und langsam wachsen zu lassen.

Kapitel 12

Minimierung von Funktionen mit Hilfe stochastischer Dynamiken

12.1 Neuronale Netze und Optimierungsprobleme

Alle neuronalen Netze, die wir betrachtet haben, realisieren eine Abbildung $F(w, x)$, die von den Netzparametern w, den Gewichten, und dem Eingabevektor x abhängt.

Dabei gibt es unterschiedliche Ansätze:

12.1.1 Rückgekoppelte Assoziativspeicher

Gegeben ist eine Energiefunktion $H(w, z)$, die von den Gewichten w und den Zuständen z der Neuronen abhängt. Für jede Eingabe x sei $F(w, x)$ eine x nächstgelegene lokale Minimumstelle der partiellen Funktion $H(w, \cdot)$.

12.1.2 Rückgekoppelte Netze zur Lösung von Optimierungsproblemen

Gegeben ist eine Energiefunktion $H(w, z)$, die von den Gewichten w und den Zuständen z der Neuronen abhängt. In der Gestalt von H ist die Art des Optimierungsproblems kodiert; aus den jeweiligen Parametern des Optimierungsproblems berechnen sich die Gewichte w des Netzes, d.h. die Gewichte w dienen hier als Eingabewerte.

Die Ausgabe $F(w)$ des Netzes zu gegebenem w ist eine der globalen Minimumstellen der partiellen Funktion $H(w, \cdot)$. Die Gewichte werden nicht erlernt.

12.1.3 Vorwärtsgerichtete Netze

Die vom Netz realisierte Abbildung $F(w, x)$ ist bei gegebenen Gewichten w relativ einfach ohne Lösung eines Optimierungsproblems zu berechnen. Allerdings ist beim Lernen der Gewichte ein aufwendiges Minimierungsproblem zu lösen, das folgende Gestalt hat.

$h(y, F(w, x))$ sei der Fehler, der entsteht, wenn bei Eingabe von x die Ausgabe y sein soll, aber $F(w, x)$ ist. X und Y seien zwei Zufallsvariablen, deren Verteilungen angeben, wie häufig in einer bestimmten Anwendung die verschiedenen Ein- und Ausgabewerte vorkommen.

Gesucht ist dann ein w, das den mittleren Fehler $H(w) = \mathcal{E}(h(Y, F(w, X)))$ global minimiert.

Bemerkung. Dieses Minimierungsproblem ist schwieriger zu lösen als die in 12.1.1 und 12.1.2, weil die zu minimierende Funktion H meist gar nicht exakt bekannt ist, weil die Verteilungen von X und Y nicht genau bekannt sind. Man kann H nur aus vielen Stichproben von X und Y schätzen.

12.2 Minimierung der Fehlerfunktion (vorwärtsgerichteter Netze)

Wir verwenden die Bezeichnungen aus 12.1.3. X und Y seien auf einem Wahrscheinlichkeitsraum (Ω, p) definiert. $(X_t)_{t\in\mathbb{N}}$ und $(Y_t)_{t\in\mathbb{N}}$ seien unabhängige Wiederholungen.

Für $n \in \mathbb{N}$ sei $H_n(w) = \frac{1}{n}\sum_{t=1}^{n} h(Y_t, F(w, X_t))$ für jedes w.

$H_n(w)$ ist eine auf Ω definierte Zufallsvariable, und wegen des starken Gesetzes der großen Zahlen konvergiert $H_n(w)$ für jedes w gegen $H(w)$ p-fast überall auf Ω.

Zur Vereinfachung nehmen wir an, für jedes $\omega \in \Omega$ habe die Funktion

$$w \mapsto H_n(w)(\omega) = \frac{1}{n}\sum_{t=1}^{n} h(Y_t(\omega), F(w, X_t(\omega)))$$

genau ein Minimum $w_n^*(\omega)$.

Dann ist $w_n^*(\omega)$ eine Zufallsvariable auf Ω. Allgemein wählt man als w_n^* eine Zufallsvariable, so daß $w_n^*(\omega)$ für p-fast alle ω ein globales Minimum von $w \mapsto H_n(w)(\omega)$ ist.

Unter einigen zusätzlichen Voraussetzungen gilt nun, daß für p-fast jedes $\omega \in \Omega$ die Folge $(w_n^*(\omega))_n$ sich nur gegen globale Minima von H häuft.

Damit ist die Minimierung von H in stochastischem Sinn zurückgeführt auf die Minimierung von Stichproben $H_n(\cdot)(\omega)$ von $H_n(\cdot)$. Somit verbleibt wie in 12.1.2 die Suche nach den globalen Minima einer Funktion.

12.2.1 Interpretation der verallgemeinerten Deltaregel

Weil die Suche nach globalen Minima einer Funktion f vieler Variablen w auf einem großen Raum sehr schwierig sein kann, begnügt man sich manchmal damit, nur ein lokales Minimum zu suchen (oder gar nur einen stationären Punkt, d.h. eine Nullstelle der Ableitung). Dafür kann man Gradientenabstiegsmethoden verwenden, also im diskreten Fall z.B. die rekursiv definierte Folge

$$w(t+1) = w(t) - \alpha(t)\nabla f(w(t)) \text{ für } t \in \mathbb{N},\ w(1) \text{ beliebig},$$

wobei $(\alpha(t))_t$ eine Nullfolge aus positiven Zahlen, die aber nicht zu schnell gegen 0 konvergieren darf.

Hat f die Gestalt $f(w) = H_n(w)(\omega) = \frac{1}{n}\sum_{j=1}^{n} h(Y_j(\omega), F(w, X_j(\omega)))$ und ist $h(x,y) = \|x-y\|^2$, so gilt mit $a_j = X_j(\omega)$ und $b_j = Y_j(\omega)$

$$\nabla f(w) = \frac{1}{n}\sum_{j=1}^{n} 2D_1F(w, a_j)^T(b_j - F(w, a_j)).$$

Für jedes $n \in \mathbb{N}$ kann man so mit der Gradientenabstiegsmethode eine Folge $(w_n(t))_t$ durch

$$w_n(t+1) = w_n(t) + 2\alpha_n(t)\frac{1}{n}\sum_{j=1}^{n} D_1F(w(t), a_j)^T(b_j - F(w(t), a_j)),\ w_n(1) \text{ beliebig},$$

konstruieren, die für geeignetes $(\alpha_n(t))_t$ gegen ein lokales Minimum $w_n^*(\omega)$ von $H_n(\cdot)(\omega)$ konvergiert.

Man kann nun auf die kühne Idee kommen, daß eventuell eine Art von Diagonalfolge $w_t(t)$ — also $n = t$ — gegen ein lokales Minimum von $H(w) = \mathcal{E}(\|Y - F(w,X)\|^2)$ konvergieren könnte. Beispielsweise liefert die LMS-Regel so eine Art von Diagonalfolge

$$w(t+1) = w(t) + 2\alpha(t)D_1F(w(t), a_t)^T(b_t - F(w(t), a_t)),\ w(1) \text{ beliebig}.$$

Natürlich kann man nicht für jede Realisierung $(a_t)_t$, $(b_t)_t$ von (X_t), (Y_t) Konvergenz erwarten. Man kann aber beweisen, daß die Folge $(w(t))_t$ für p-fast jede Realisierung $(a_t)_t$, $(b_t)_t$ gegen einen stationären Punkt w^* von $\mathcal{E}(\|Y - F(w,X)\|^2)$ konvergiert, wenn außer einigen technischen Bedingungen noch $\sum_{t\in\mathbb{N}} \alpha(t) = \infty$ und $\sum_{t\in\mathbb{N}} \alpha^2(t) < \infty$ gilt.

12.3 Stochastische Suche nach globalen Minima

12.3.1 Gradientenabstieg mit additivem Rauschen

H sei eine differenzierbare Funktion auf einer offenen Teilmenge $\mathcal{U}$ des $\mathbb{R}^n$. $x(0) \in \mathcal{U}$.

Diskret: $$x(t+1) = x(t) - \alpha(t)\nabla H(x(t)) + \sigma(t)\nu(t),$$

wobei $\alpha(t)$, $\sigma(t)$ und $(\nu(t))_t$ ein stochastischer Prozeß (meist mit unabhängigen, identisch verteilten Variablen, d.h. weißes Rauschen).

Kontinuierlich: $$x' = -\nabla H(x) + \sigma(t)\frac{dB}{dt},$$

wobei $\sigma(t)$ und B die $\mathbb{R}^n$-wertige *Brownsche Bewegung* sind, dabei ist $\frac{dB}{dt}$ nicht im klassischen Sinn definierbar. Äquivalent dazu ist die integrale Form

$$x(t) = x(0) - \int_0^t \nabla H(x(s))\,ds + \int_0^t \sigma(s)\,dB(s),$$

Auch hier ist $\int_0^t \sigma(S)\,dB(s)$ kein übliches Lebesgue-Stieltjes-Integral, sondern ein Wiener-Integral (oder Ito-Integral), denn die Pfade der Brownschen Bewegung haben keine beschränkte Variation.

Unter geeigneten Voraussetzungen existiert eine Lösung $x(t)$ obiger Differential- oder Integralgleichung und konvergiert mit Wahrscheinlichkeit 1 gegen ein globales Minimum vun H. Wesentlich ist die Voraussetzung, daß $\sigma(t)$ für $t \to \infty$ gegen 0 geht, allerdings nicht zu schnell; es genügt, daß $(\sigma(t))^2 \geq \frac{a}{ln(t)}$.

12.3.2 Der Metropolis-Algorithmus

Z sei eine endliche Menge, die oft Zustandsraum genannt wird. $H : Z \to \mathbb{R}$ sei die Funktion, von der ein globales Minimum gesucht wird. H sei nicht konstant.

Metropolis-Algorithmus

Sei $T > 0$. Wähle ein $z(0) \in Z$. Definiere $(z(t))_t$ rekursiv auf folgende Weise:

1. Wähle ein $y \in Z$ zufällig aus – bzgl. einer Wahrscheinlichkeitsverteilung Q auf Z, die von $z(t)$ abhängen darf.
2. Berechne $\Delta H = H(y) - H(z(t))$.
   (a) Falls $\Delta H \leq 0$, setze $z(t+1) = y$.
   (b) Falls $\Delta H > 0$, führe ein Zufallsexperiment durch, das zwei Ergebnisse a und $\overline{a}$ mit den Wahrscheinlichkeiten $p(a) = \exp\left(-\frac{\Delta H}{T}\right)$ und $p(\overline{a}) = 1 - \exp\left(-\frac{\Delta H}{T}\right)$ hat. Ist das Resultat des Experiments gleich a, so setze $z(t+1) = y$ („y wird akzeptiert“); ist es $\overline{a}$, so setze $z(t+1) = z(t)$.

Der Metropolis-Algorithmus definiert also eine stochastische Dynamik auf Z. Im Gegensatz zu den in Abschnitt 9 betrachteten Dynamiken kann es passieren, daß H längs eines Pfades stellenweise auch zunimmt, allerdings ist eine Zunahme umso seltener, je größer sie ist. Der Vorteil dieser Dynamik ist, daß die Pfade nicht notwendigerweise durch lokale Minima von H angezogen werden, sondern mit einer gewissen Wahrscheinlichkeit bergauf laufen können und lokalen Minima entkommen. Andererseits werden sie aber auch nicht bei globalen Minima verbleiben, sondern auch ihnen entkommen. Die Hoffnung ist, daß sie länger in der Nähe globaler Minima verweilen.

Zu diesem Thema gibt es in der Literatur ausführliche Untersuchungen. Wir werden nur ein grundlegendes Resultat etwas präziser formulieren.

Die Auswahl von y in Schritt 1 erfolgt bezüglich einer Verteilung Q, die vom momentanen Zustand $z(t)$ abhängen darf; d.h. man gibt sich für jedes $x \in Z$ eine Wahrscheinlichkeitsverteilung $Q(x, \cdot)$ auf Z vor, also

nicht negative Zahlen $Q(x,y)$, $y \in Z$, mit $\sum_{y\in Z} Q(x,y) = 1$. So eine Familie Q heißt dann *Markov-Kern* oder Übergangswahrscheinlichkeit.

Ist ν eine Wahrscheinlichkeitsverteilung auf Z (also $\nu : Z \to [0,1]$ mit $\sum_{x\in Z} \nu(x) = 1$), so ist durch

$$\nu Q(y) = \sum_{x\in Z} \nu(x) Q(x,y), y \in Z,$$

wieder eine Wahrscheinlichkeitsverteilung νQ auf Z definiert.

Ist P ein weiter Markov-Kern, so ist durch $QP(x,y) = \sum_{z\in Z} Q(x,z)P(z,y)$ wieder ein Markov-Kern QP auf Z gegeben.

Schritt 2 im Metropolis-Algorithmus läßt sich mit Hilfe von Markov-Kernen formulieren. Für $x, y \in Z$ sei

$$\pi_T(x,y) = \begin{cases} Q(x,y)\exp\left(-\frac{1}{T}(H(y)-H(x))\right)^+ & \text{, für } x \neq y \\ 1 - \sum_{x\neq z\in Z} \pi_T(x,z) & \text{, für } x = y \end{cases} \qquad u^+ = \max\{u, 0\}.$$

π_T ist ein Markov-Kern auf Z, und Schritt 2 kann folgendermaßen beschrieben werden:

2.′ Wähle $z(t+1) \in Z$ zufällig gemäß der Verteilung $\pi_T(z(t), \cdot)$ aus.

Setze

$$\Pi_T(x) = \frac{\exp\left(-\frac{1}{T}H(x)\right)}{\sum_{z\in Z}\exp\left(-\frac{1}{T}H(z)\right)} \quad \text{für } x \in Z.$$

Π_T ist eine Wahrscheinlichkeitsverteilung auf Z, das sogenannte *Gibbsmaß* (oder *Gibbsfeld*) für H zur Temperatur T.

Satz. Z sei eine endliche Menge, $H : Z \to \mathbb{R}$ nicht konstant, $T > 0$, Π_T das Gibbsmaß für H zur Temperatur T. Q sei ein Markov-Kern auf Z, so daß $Q(x,y) = Q(y,x) > 0$ für alle $x, y \in Z$. π_T sei der aus Q und H wie oben gebildete Markov-Kern.

Dann gilt für jedes $x \in Z$ und jede Wahrscheinlichkeitsverteilung ν auf Z

$$\nu\pi_T^n(x) \to \Pi_T(x) \quad \text{für } n \to \infty.$$

Dabei steht π_T^n für die n-fache Komposition.

Bemerkung. Eine Folge von Wahrscheinlichkeitsverteilungen der Gestalt $\nu\pi_T^n$ heißt eine *homogene Markov-Kette*; homogen deshalb, weil stets derselbe Markov-Kern verwendet wird. Die spezielle Folge $(\nu\pi_T^n)_n$ heißt ein *Metropolis-Sampler*.

Interpretation: Das Gibbsmaß Π_T ist ein Wahrscheinlichkeitsmaß, das Zuständen umso höhere Wahrscheinlichkeiten zuordnet, je kleiner der Wert von H in ihnen ist. Wählt man also ein $z \in Z$ zufällig gemäß Π_T aus, so wird z mit hoher Wahrscheinlichkeit in der Nähe eines globalen Minimums von H sein. Im allgemeinen kann man aber Π_T nicht explizit berechnen, denn man müßte dafür alle Werte $H(z)$, $x \in Z$, berechnen, und dann würde man sowieso schon alle globalen Minima von H kennen. Einen Ausweg bietet obiger Satz.

Statt eine Stichprobe gemäß Π_T zu ziehen, kann man eine Stichprobe gemäß $\nu\pi_T^n$ ziehen. Weil $\nu\pi_T^n$ punktweise gegen Π_T konvergiert, wird man dadurch für große n keinen allzu großen Fehler machen und immer noch mit hoher Wahrscheinlichkeit ein x in der Nähe eines globalen Minimums ziehen. Der Vorteil besteht darin, daß die Auswahl eines x gemäß $\nu\pi_T^n$ leicht konstruktiv durchgeführt werden kann:

Als Anfangsverteilung ν kann man beispielsweise die Gleichverteilung auf Z nehmen oder sogar die Verteilung, die einem beliebigen $z(0) \in Z$ die Wahrscheinlichkeit 1 gibt und allen anderen 0. Das heißt, man darf einfach irgendein $z(0) \in Z$ nehmen.

Eine Stichprobe $z(n+1)$ gemäß $\nu\pi_T^{n+1}$ erhält man rekursiv aus einer Stichprobe $z(n)$ von $\nu\pi_T^n$, indem man den Schritt 2 des Metropolis-Algorithmus durchführt.

Der Metropolis-Algorithmus liefert also eine Folge von Stichproben $z(n)$ von $\nu\pi_T^n$. diese $z(n)$ liegen mit hoher Wahrscheinlichkeit Π_T in der Nähe von globalen Minima von H, konvergieren aber nicht dagegen, sondern unterliegen Schwankungen, die auch für $n \to \infty$ nicht verschwinden.

12.3.3 Simulated Annealing

Wir verwenden die Bezeichnungen des vorherigen Abschnitts.

Satz. Π_T sei das Gibbsmaß für H zur Temperatur T. Und M sei die Menge der globalen Minimumstellen von H. Dann gilt

$$\lim_{T\to\infty} \Pi_T(x) = \begin{cases} \frac{1}{|M|} & \text{, für } x \in M \\ 0 & \text{, sonst} \end{cases},$$

d.h. $(\Pi_T)_T$ konvergiert punktweise gegen die Gleichverteilung auf M.

Für kleine Temperaturen T werden also Stichproben gemäß Π_T mit hoher Wahrscheinlichkeit sehr nahe bei globalen Minima von H liegen. Um Stichproben gemäß Π_T approximativ zu erhalten, kann man den Metropolis-Algorithmus anwenden.

Es gibt auch Ansätze, den Metropolis-Algorithmus mit einer Senkung von T zu kombinieren. Dabei sind prinzipiell zwei Möglichkeiten denkbar:

Homogenes Simulated Annealing: Bei festem T wird der Metropolis-Algorithmus durchgeführt, bis die Häufigkeitsverteilung der durchlaufenen Zustände $z(t)$ sich nicht mehr merklich ändert. Dann wird die Temperatur etwas abgesenkt, und wieder der Metropolis-Algorithmus durchgeführt, bis sich keine Änderung mehr zeigt (thermisches Gleichgewicht) usw.

Inhomogenes Simulated Annealing: Nach jedem Schritt des Metropolis-Algorithmus wird die Temperatur ein wenig gesenkt.

Es gibt ausgedehnte Untersuchungen darüber, unter welchen Bedingungen diese Verfahren konvergieren. Interessant ist vor allem, wie schnell die Temperatur gesenkt werden darf. Wir zitieren hier nur ein erstes fundamentales Resultat.

Satz. Z sei eine endliche Menge, $H : Z \to \mathbb{R}$ nicht konstant. W sei ein Markov-Kern auf Z, so daß $Q(x,y) = Q(y,x) > 0$ für alle $x, y \in Z$. Für $T > 0$ und $x, y \in Z$ sei

$$\pi_T(x,y) = \begin{cases} Q(x,y)\exp\left(-\frac{1}{T}(H(y)-H(x))^+\right) & \text{, für } x \neq y \\ 1 - \sum_{x\neq z\in Z} \pi_T(x,z) & \text{, für } x = z \end{cases}.$$

Sei $\Delta = \max\{H(y)-H(x) \mid x, y \in Z\}$ und $(T_n)_n$ eine monoton fallende Nullfolge, so daß $T_n \geq \frac{\Delta}{\ln(n)}$ für große n. Dann gilt für jede Wahrscheinlichkeitsverteilung ν auf Z und jedes $x \in Z$

$$\lim_{n\to\infty} \nu\pi_{T_1} \circ \ldots \circ \pi_{T_n}(x) = \begin{cases} \frac{1}{|M|} & \text{, für } x \in M \\ 0 & \text{, sonst} \end{cases},$$

wobei M die Menge der globalen Minimumstellen von H ist.

12.3.4 Der Gibbs-Sampler

Der Zustandsraum habe die Gestalt $Z = A^S$, wobei S eine endliche Menge von Neuronen und A eine endliche Menge von Zuständen ist, die jedes Neuron annehmen kann. $H : Z \to \mathbb{R}$ sei eine nicht konstante, Π_T ihr Gibbsmaß zur Temperatur T. Für $s \in S$, $x, y \in Z$ sei

$$\Pi_{T,s}(x,y) = \begin{cases} \frac{\exp\left(-\frac{1}{T}H(y)\right)}{\sum_{z\in Z_{x,s}} \exp\left(-\frac{1}{T}H(z)\right)} & \text{, falls } x_t = y_t \text{ für alle } t \neq s \\ 0 & \text{, sonst} \end{cases},$$

wobei $Z_{x,s} = \{z \in Z \mid z_t = x_t \text{ für } t \in S\backslash\{s\}\}$.

$\Pi_{T,s}$ ist ein Markov-Kern auf Z. S habe r Elemente; wähle eine Aufzählung $s_1, \ldots, s_r$. Setze

$$P_T = \Pi_{T,s_1} \circ \ldots \circ \Pi_{T,s_r}.$$

Π_T ist ebenfalls ein Markov-Kern auf Z — „*sweep*“ genannt. Für jede Wahrscheinlichkeitsverteilung ν auf Z heißt die Folge $(\nu P_T^n)_{n\in\mathbb{N}}$ ein *Gibbs-Sampler* für Π_T; denn es gilt

Satz. Für jede Wahrscheinlichkeitsverteilung ν auf Z und jedes $x \in Z$ gilt

$$\nu P_T^n(x) \to \Pi_T(x) \quad \text{für } n \to \infty \text{ (sogar glm.).}$$

Bemerkung. Statt bei einem „sweep“ die Elemente von S in einer vorgegebenen Reihenfolge zu durchlaufen, darf man sie auch in geeigneter Weise zufällig auswählen.

Satz (Simulated Annealing für Gibbs-Sampler)**.**
$(T_n)_n$ sei eine monoton fallende Nullfolge, so daß $T_n \geq \frac{\Delta}{\ln(t)}$ für große n, wobei $\Delta = \max\{|H(x) - H(y) \mid x, y \in Z \text{ mit Hammingabstand } 1\}$. Dann gilt

$$\lim_{n\to\infty} \nu P_{T_1} \circ \ldots \circ P_{T_n}(x) = \begin{cases} \frac{1}{|M|} & \text{, für } x \in M \\ 0 & \text{, sonst} \end{cases}$$

gleichmäßig für alle Wahrscheinlichkeitsverteilungen ν auf Z, wobei M die Menge der globalen Minimumstellen von H ist.

12.3.5 Boltzmann-Maschinen

Wir betrachten den Gibbs-Sampler für den Fall $A = \{\pm 1\}$, also $Z = \{\pm 1\}^S$. Dann gilt für $s \in S$, $x, y \in Z$ mit $x_t = y_t$ für $t \neq s$

$$\Pi_{T,s}(x,y) = \frac{\exp\left(-\frac{1}{T}H(y)\right)}{\exp\left(-\frac{1}{T}H(y)\right) + \exp\left(-\frac{1}{T}H(x)\right)} = \frac{1}{1 + \exp\left(-\frac{1}{T}(H(y) - H(x))\right)} = \frac{1}{1 + \exp\left(-\frac{1}{T}\Delta H(x,y)\right)},$$

wobei $\Delta H(x,y) = H(y) - H(x)$.

Wählt man statt einer deterministischen Aufzählung der Neuronen eine zufällige Auswahl (z.B. bezüglich der Gleichverteilung), so läßt sich die Dynamik des Gibbs-Samplers folgendermaßen beschreiben:

Sei $x = z(t)$ der Zustand zum Zeitpunkt t.

Wähle zufällig ein $s \in S$ aus. y bestehe aus x durch Veränderung des Vorzeichens der s-ten Komponente x_s.

Wähle mit Wahrscheinlichkeit $\left(1 + \exp\left(-\frac{1}{T}\Delta H(x,y)\right)\right)^{-1}$ $z(t+1) = y$ und mit Wahrscheinlichkeit $1 - (\ldots)^{-1}$ $z(t+1) = x$.

Ein Netz, das mit dieser Dynamik arbeitet, heißt *Boltzmann-Maschine*.

Der Übergang von $z(t)$ zu $z(t+1)$ kann wieder druch einen Markov-Kern P_T beschrieben werden, und der Satz über das Simulated Annealing für Gibbs-Sampler gilt entsprechend.

Ein empfehlenswertes Lehrbuch zu dem Thema „Stochastische Optimierung“ ist: [11]

Literaturverzeichnis

[1] Hecht-Nielsen, *Theory of the backpropagation neural network*, 1990

[2] K. Hornick, M. Stinchcombe und H. White, *Multilayer feedforward networks are universal approximators*, Neuronal Networks 2, 1989, 359-366

[3] Y. Kamp, M. Hasler, *Recursive neuronal networks for associative memory*, Wiley 1990

[4] A. Maelicke, *Vom Reiz der Sinne*, VCH 1990

[5] B. Müller, J. Reinhardt, *Neural networks*, Springer 1990

[6] Ritter, Martinek, Schulten, *Neuronale Netze*, Addison Wesley, 1991

[7] Ritter, Schulten, *On the stationary state of Kohonen's self-organizing sensory mapping*, Biology Cybernatics 54, 1986, 99-106

[8] Ritter, Schulten, *Convergence properties of Kohonen's topology conserving maps: fluctuations, stability and dimension selection*, Biology Cybernatics 60, 1986, 59-71

[9] D.A. Sprecher, *On the structure of continous functions of several variables*, TAMS 115, 1965, 340-355

[10] M. Takeda, J. W. Goodman, *Neural Networks for Computation: Number Representations and Programming Complexity*, Appl. Optics 25, 1986, 3033-3046

[11] G. Winkler, *Image Analysis, Random Fields and Dynamic Monte Carlo Methods. An Introduction to Mathematical Aspects*, Springer, 1995 (2002)

[12] Y. T. Zhou, R. Chellappa, *Artificial Neuronal Networks for Computer Vision*, Springer, 1992