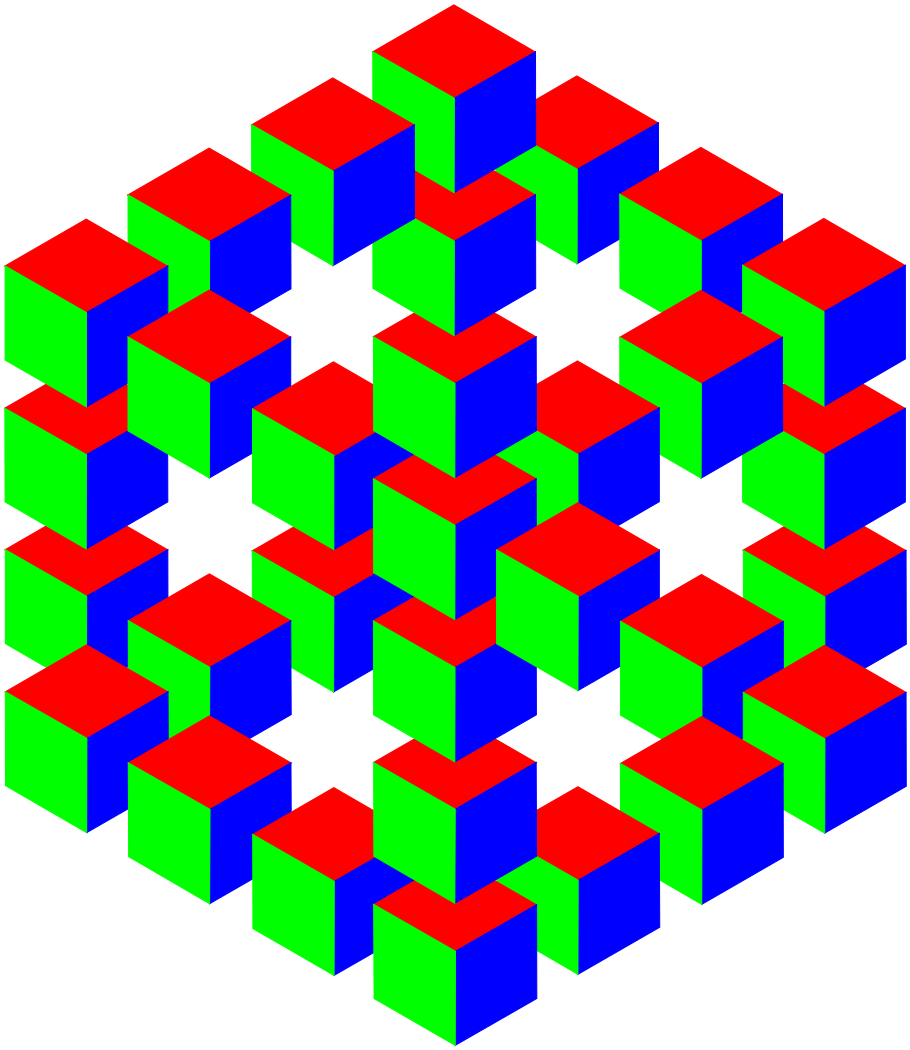
Neuroinformatik

Gerald Sommer



Lehrstuhl für Kognitive Systeme
Christian--Albrechts--Universität zu Kiel
2008

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

AKF	Autokorrelationsfunktion (engl. ACF)
Adaline	Adaptive linear element
AG	Armijo-Goldstein
AMISE	approximierter MISE
AR	Auto-Regressive
ARIMA	AR Integrated MA
ARFIMA	AR Fractional Integrated MA
ARFIMAX	ARFIMA mit eXogenen Variablen
ARMA	Zeitreihenmodell mit AR und MA Komponente
ARMAX	ARMA mit eXogenen Variablen
ART	Adaptive-Resonanz-Theorie
ASCC	Automatic Sequence Controlled Calculator
BFGS	Broyden-Fletcher-Goldfarb-Shanno
BP	Backpropagation
BSHS	Beale-Sorenson-Hestenes-Stiefel
BLUE	Best Linear Unbiased Estimator
CNN	Cellular Neural Nets
DCS	Dynamic Cell Structures
DFG	Deutsche Forschungsgemeinschaft
DFP	Davidon-Fletcher-Powell
DGP	Data Generating Process
ENIAC	Electronic Numerical Integrator and Computer
FOC	First Order Condition
FPGA	Field Programmable Gate Array
FR	Fletcher-Reeves
GED	Generalized Exponential Distribution
HON	Higher Order Networks
iid	Independent and Identically Distributed
ind	Independent
IS	Importance Sampling
KISS	Keep It Short & Simple.
KLT	Karhunen-Loeve-Transformation
kNN	Künstliches NN oder k-Nächste Nachbarn
LM	Levenberg-Marquardt
LMS	Least Mean Square
LS	Least Squares
LSE	LS Estimator
MA	Moving-Average

MAP	Maximum A Posteriori
MC	Monte-Carlo
MCMC	Monte-Carlo-Markov-Chain
MISE	Mean integreated squared error
ML	Maximum-Likelihood
MLE	ML Estimator
MLP	Multi Layer Perceptron
MSE	Mean Square Error
MMSE	Minimum MSE
NM	Nelder-Mead
NN	Neuronal Network
NR	Newton-Raphson
PACF	Partial AutoCorrelation Function
PDF	Probability Density Function
PLA	Programmable Logic Array
PR	Polak-Ribiere
QML	Quasi Maximum Likelihood
QMLE	QML Estimator
RBF	Radiale Basisfunktion
RM	Robbins-Monro
ROC	Receiver Operating Characteristics
SA	Simulated Annealing
SLN	Single-Layer-Network
SOM	Self-Organizing Map
SVM	Support Vector Maschine
TM	Turing-Maschine
VAR	Vector Autoregressive
VARMA	Vector Autoregressive Moving Average
VARMAX	VARMA with eXogenous variables
VCD	Vapnik-Chervonenskis-Dimension
VEC	*vec*-Operator
VECH	*vech*-Operator
VMA	Vector MA
YWE	Yule-Walker Estimator

Notation

In diesem Skript werden einige Notationen benutzt, die dem einen oder anderen Leser vielleicht fremd sind. Auf die folgenden besonderen Notationen wird daher an dieser Stelle explizit aufmerksam gemacht.

bedingte Funktionen

Sei $f : \mathbb{R} \to \mathbb{R}$ die Funktion $f(x, y) := x^y$. Seien $x, y \in \mathbb{R}$.
Setze $y := 3$. Dann gilt folgende Äquivalenz:

$$f_y(x) \equiv f(x; y) \equiv f(x|y)$$

bedingte Wahrscheinlichkeiten

Sei P eine Wahrscheinlichkeitsfunktion. Seien X, Y Zufallsvariable (ZV) für P.
Dann heißt $P(X|Y)$ „*Wahrscheinlichkeit von X gegeben, Y ist eingetreten*“.

Stichproben

Sei $n \in \mathbb{N}$. Sei $t \in \mathbb{N}_{\leq n}$. Sei $x \in \mathbb{R}^n$ ein Vektor von Stichproben.
Dann gilt im Zusammenhang mit Stichproben folgende äquivalente Notation:

$$x^t \equiv x_t$$

Schätzer

Sei $X(\theta)$ eine ZV mit unbekanntem Parameter $\theta \in \mathbb{R}$.
Dann wird nach gängiger Konvention ein Schätzer für θ mit $\widehat{\theta}$ notiert.

Beispiel

Sei der Erwartungswert μ von X unbekannt, also $\theta = \mu$.
Dann ist z.B. der Mittelwertoperator $\widehat{\mu} := \frac{1}{T} \sum_{t=1}^{T} x_t \equiv < x > \equiv \bar{x}$ (das arithmetische Mittel) ein Schätzer (Notation in Statistik / Physik / Mathematik) für diesen.

Operatoren / Funktionen / Abbildungen

Man verwechsele nicht Erwartungswert- und Mittelwertoperator.
Sei X eine ZV mit Ausprägungen x_t, $t \in \mathbb{N}_{\leq T}$ (Stichprobe).
Dann gilt für den Erwartungswert von X

$$E\{X\} = \lim_{T \to \infty} \frac{1}{T} \sum_{t=1}^{T} x_t,$$

wohingegen der Mittelwertoperator $\widehat{\mu}$ der Schätzer des Erwartungswerts μ ist.
Praktisch ist jedoch nur der Mittelwertoperator nutzbar, weswegen beide oft synonym verwendet werden. Gleiches gilt auch für die Varianz σ^2 und andere (unbekannten) Charakteristika einer ZV.

1 Einführung

- Übersicht der Vorlesung

  Gegenstand der Vorlesung sind Modelle künstlicher neuronaler Netze (kNN) und ihre Berechnungseigenschaften. Das NN-Paradigma hat seine Wurzeln in der statistischen Entscheidungstheorie und Mustererkennung (Pattern Recognition). Diese Verbindung herzustellen, ist das Ziel dieser Vorlesung. Darüber hinaus werden verschiedene grundlegende Typen vorwärtsgerichteter NN und ihre Anwendungen eingeführt.

- Vorstellung von Forschungsarbeiten in der Arbeitsgruppe Kognitive Systeme
  - Gesichtserkennung (Zuordnung eines Gesichtes zu einem genetischen Typus)
  - Bewegungsplanung und Navigation autonomer Roboter (Qualität und Quantität des Vorwissens beim Reinforcement-Lernen)
  - DCS-Netzwerke (Dynamische Zellstrukturen)
    Merkmale: lokale, topologieerhaltende Repräsentation von Daten und Anpassung der Netzwerktopologie bezüglich eines Fehlermaßes

    Beispiele: Poseschätzung eines Puppenkopfes oder Sakkaden-Lernen eines Stereo-Kamerakopfes
  - Clifford-Neuronen
    Sie entstehen, wenn im linearen Assoziator das Skalarprodukt durch das geometrische Produkt oder das Spinorprodukt ersetzt werden.

    Beispiele: Lernen geometrischer Transformationsgruppen oder Lernen (hyper-) sphärischer Entscheidungsgrenzen

Großhirnrinde	:	20 Mrd. Neuronen (1,5m^2 Fläche, 2mm Dicke)
Kleinhirnrinde	:	100 Mrd. Neuronen,
Zentralnervensystem (ZNS)	:	100 Mrd. Neuronen, 10^5 mm^{-3}

Tabelle 1.1: Neuronen im menschlichen Gehirn.

Großhirnrinde	$\approx 10^4$ Synapsen pro Neuron
Kleinhirnrinde	$\approx 2 \cdot 10^5$ Synapsen pro Neuron

Tabelle 1.2: Synapsen im menschlichen Gehirn.

1.1 Das biologische Neuron

a) Nervenzellen als Elemente der Informationsverarbeitung

- große Anzahl und hohe Dichte
  ca. 1 Billion Zellen im menschlichen Gehirn (siehe Tabelle 1.1)
- relativ homogene Grundstruktur bestehend aus Pyramidenzellen (verstärkend) und Sternzellen (hemmend)
- hohe Spezialisierung: ca. $10^3 - 10^4$ verschiedene Typen (in der Netzhaut/Retina mehr als 90), bislang noch wenig verstanden.

b) Strukturelle Bestandteile der Nervenzellen

- Dendriten: Aufnahme von Information
  permeable Ionenkanäle für Na^+-, K^+-, Ca^{++}-Ionen
  synaptische Spalte
- Zellkörper (Soma): Informationsverarbeitung
  (Summation dendritischer Signale)
- Axone: Informationsweiterleitung
  Übertragung eines Aktionspotentials (Spike) gesteuert durch eine Na^+-, K^+-Ionenpumpe an der Zellmembran
- Synapsen: Informationsspeicherung

  Beispiel: **Hebbsches Lernen**
  Modifikation der Synapseneffizienz durch die Rate des präsynaptischen Erregungsangebotes (Menge des Neurotransmitters Glutamat)
  dauerhafte Reduktion der Blockade postsynaptischer Dendriten-Rezeptoren durch Mg^+-Ionen aufgrund starker Erregung (Änderung der Reizschwelle)

c) Verbunde von Nervenzellen

- Bildung einer Strukturhierarchie: Säulen, Schichten, Systeme

- Rezeptive Felder sind lokale Netze
- Verbindungen untereinander relativ schwach, aber mit vielen anderen Neuronen verbunden (10^9 Synapsen pro mm^3), siehe Tabelle 1.2
- zyklische Verschaltung: nach 4-5 Neuronen wird das Ausgangsneuron erreicht
- starke Rückkopplung (rückwärts gerichtete Verschaltung)

  Beispiel: seitliche Kniehöcker (LGN/Corpus geniculatum laterale) verschalten die Netzhaut (Retina) mit der primären Sehrinde und erhalten 90% ihres Inputs durch Rückkopplung mit der primären Sehrinde

d) Eigenschaften neuronaler Informationsverarbeitung

- langsam im Vergleich zu elektronischen Schaltelementen (msec - nsec); dennoch Echtzeitfähigkeit des Gesamtsystems
- sehr simple "Prozessoren"; dennoch Realisierung komplexer nichtlinearer Prozesse
- Bildung hochkomplexer Systeme durch massive Parallelität, hohe Vernetzungsdichte und hierarchische bzw. rückgekoppelte Systeme
- Die stochastische Natur der Bausteine (hohe Toleranzen) und ihre unscharfe Reaktionen auf einen Stimulus wird kompensiert durch eine hohe Parallelität und viel Redundanz.
- verteilte Repräsentation der Information
- Die zeitliche Synchronisierung der Aktivität (Binding-Problem) in regionalen Verbunden führt zu Korrelation mit den Phänomenen in der Welt.
- Bis auf einen genetischen Anteil, der als sehr wichtig gilt, ist das System nicht programmierbar - aber lernfähig.
- enge Beziehung zwischen Struktur und Funktion. Der Ort eines Neurons bestimmt dessen Aufgabe.
- keine zentrale Kontrolle, jedoch Kontrolle durch Kommunikation
- Die semantische Lücke (Eine Semantik ergibt sich erst durch das Lernen von Strukturen.) ist auch Teil des Wesens des Nervensystems. Seine Syntax ist die Biochemie der Nervenzellen.
- Das Gehirn ist ein selbstorganisierendes, offenes, dynamisches System...

1.2 Das Berechenbarkeitsproblem

Ein kNN ist ein Netz primitiver verkoppelter Funktionen, dessen Rechenleistung über das objektiv beobachtbare Input-Output-Verhalten zu bewerten ist. Dabei werden hierarchische Verbunde von Berechnungselementen zugelassen.

1.2.1 Der klassische Computer

Zunächst die Frage: Wozu eignen sich klassische Computer, die ihre Funktionalität durch die explizite Angabe eines Algorithmus erhalten?

- arithmetische Operationen
- logische Operationen
- Textverarbeitung
- Modellierung "intelligenter" Handlungen (1956, Dartmouth College, USA: J. McCarthy, M. Minsky, A. Newell, H. Simon)
  Ziel: Modellierung intelligenter Handlungen mittels Symbolverarbeitung (Symbole als linguistische Termini, z.B. "Katze") als Abstraktion (in Form von Äquivalenzklassen) von Ausprägungen der Umwelt

1.2.2 Das biologische Neuron

Lernen ist eine inhärente Eigenschaft natürlicher Neuronen! Sie realisieren eine Signalverarbeitung durch elektrochemische Erregung. Dies entspricht einer nichtsymbolischen Repräsentation der Operanden!

Beispiel: Nicht die Äquivalenzklasse "Katze", sondern zuerst nur eine Instanz "Katze" dient als Operand (als Menge aller Signale, die von "Katze" wahrgenommen werden). Erst aus vielen ähnlichen Eindrücken wird ein Modell der Äquivalenzklasse "Katze" gebildet.

1.2.3 Künstliche Neuronale Netze

Künstliche Neuronale Netze schlagen die Brücke zwischen biologischen Neuronen und klassischen Computern.

- Anwendungen ihrer Lernparadigmen:
  - Mustererkennung (Generalisierung im Sinne von approximativer Äquivalenzklassenbildung)
  - assoziative Speicherung
  - (stochastische) Optimierung, Prädiktion und Kontrolle

  - Die Auswertung statistischer Korrelationen in Daten bildet die Grundlage für inexaktes evidentielles Schließen im Gegensatz zu scharfen logischen Regeln!

- Eigenschaften von kNN:
  - Robustheit (Unempfindlichkeit gegenüber Änderungen der Modellvoraussetzungen, hier: oft völliger Verzicht auf Modelle)
  - Adaptivität (Anpassung der Berechnungsvorschriften an Änderungen der Voraussetzungen)

- Die Realisierung (Simulation) eines kNN auf klassischen Computer kann nie dessen Berechenbarkeitsgrenzen überschreiten! Analoge kNN dagegen können die Turing-Berechenbarkeit überschreiten (H.T. Siegelmann, 1999).

  Beispiel: Eine "Katze" wird repräsentiert durch die Pixel eines digitalen Bildes (endlicher Zustandsraum zur Beschreibung der Katze).

1.2.4 KNN als alternatives Berechnungsparadigma

Seit den 30er und 40er Jahre des 20. Jahrhunderts existieren verschiedene Strömungen zur Beherrschung des Berechenbarkeitsproblems mit logischen/mathematischen und physikalischen Mitteln.

- Turing ⇒ Präzisierung des Algorithmenbegriffes durch die Turing-Maschine

- von Neumann ⇒ Architektur des heutigen Universalrechners

- Shannon ⇒ Informationstheorie

- Shannon & von Neumann ⇒ Kybernetik (verkörperlichte intelligente Maschinen)

Zu jener Zeit war es offen, ob die Entwicklung des mechanisierten/elektronischen Rechnens in die Richtung geht, welche heute die Informatik verkörpert.

Zitate: aus "Die Rechenmaschine und das Gehirn", J. von Neumann, Oldenburg, 1991 (Silliman Lectures, Yale University, 1956):

1 *Endlich ist es mein Anliegen, noch einen anderen Gesichtspunkt herauszuarbeiten. Ich vermute, dass ein gründlicheres mathematisches Studium des Nervensystems... unser Verständnis der betreffenden Gebiete der Mathematik selbst beeinflussen wird. Dadurch wird möglicherweise unsere Ansicht über die eigentliche Mathematik und Logik modifiziert.*

2 *Man sollte sich vergegenwärtigen, dass Sprache im wesentlichen ein historischer Zufall ist... Ebenso wie Sprachen, wie das Griechische oder Sanskrit, historische Tatsachen und keine absoluten Notwendigkeiten sind, kann man vernünftigerweise annehmen, dass Logik und Mathematik in ähnlicher Weise historische, zufällige Ausdrucksformen sind. ... sie können in anderen als den uns bekannten Formen existieren. In der Tat, die Beschaffenheit des Zentralnervensystems und des von ihm übertragenen Nachrichtensystems weist positiv darauf hin, dass es sich so verhält.*

1.2.5 Berechenbarkeit / Entscheidbarkeit

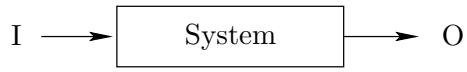


Abbildung 1.1: Ein kanonisches System.

Frage: Läßt sich für jedes beliebige Ein-/Ausgabeverhalten ein System (siehe Abbildung 1.1) entwerfen?
Antwort: Nein, es gibt entscheidbare/berechenbare Probleme und solche, die es nicht sind (D. Hilbert (1900), Paris: 23 math. Probleme).
Eine Funktion ist berechenbar, wenn es für sie einen Algorithmus (d.h. ein Verfahren, welches zu beliebigen Argumenten in endlich vielen Schritten einen Funktionswert liefert) gibt. Die lösbaren Probleme sind von unterschiedlicher Komplexität, wobei Komplexität ein Begriff ist, der für die einzusetzenden Ressourcen steht (z.B. Zeit) und nicht für die Repräsentation des Problems.
Die meisten Probleme der Mustererkennung und des Lernens einer Aufgabe sind NP-vollständig (NP - nichtdeterministisch polynomial).

Einschub: NP - vollständig (Rojas, 1996, Seite 271)

> *Die gutartigsten Probleme sind von linearer oder sublinearer Komplexität. Dies ist ein Spezialfall polynomialer Zeitkomplexität.*
>
> **Komplexitätsklasse P:** *Ein Problem kann in polynomialer Zeitkomplexität gelöst werden. Auf dem Modell einer Turing-Maschine lässt sich das Problem als Eingabe der Länge n kodieren. Wenn irgend eine Instanz x eines Problems X von der Größe n in polynomialer Zeit $p(n)$ lösbar ist, dann gilt $X \in P$.*
>
> **Komplexitätsklasse NP:** *In polynomialer Zeit kann ein vorgeschlagener konkreter Lösungsweg überprüft werden, ob er eine Lösung des Problems X darstellt. Es kann in polynomialer Zeit aber nicht die allgemeine Lösung des Problems X berechnet werden. Hierfür kann*

das Modell einer Turing-Maschine mit Orakel dienen. Das Orakel schlägt den konkreten Lösungsweg zufällig vor. Deshalb nennt man diese Komplexitätsklasse nichtdeterministisch-polynomial.

Beispiel: Problem des Handelsreisenden (TSP)

Interessant ist die Beziehung zwischen diesen beiden Klassen. Offensichtlich gilt $P \subseteq NP$. Hieraus folgt die Implikation: Wenn das Problem X zu P gehört, dann ist auch in nichtdeterministischer Weise in polynomialer Zeit überprüfbar, ob es eine Lösung darstellt.

Der Beweis, dass $P \neq NP$, ist noch nicht erbracht und wahrscheinlich auch nicht möglich.

Neben P gibt es noch die Klasse NPv als Teilmenge von NP. NPv heißt NP-vollständig und steht für die Klasse der schwierigsten Probleme aus NP.

Komplexitätsklasse NPv: *Ein Problem $X \in NP$ gehört zu NPv, wenn es ein anderes Problem $X' \in NP$ gibt, das in polynomialer Zeit als Instanz x des Problems X identifiziert werden kann.*

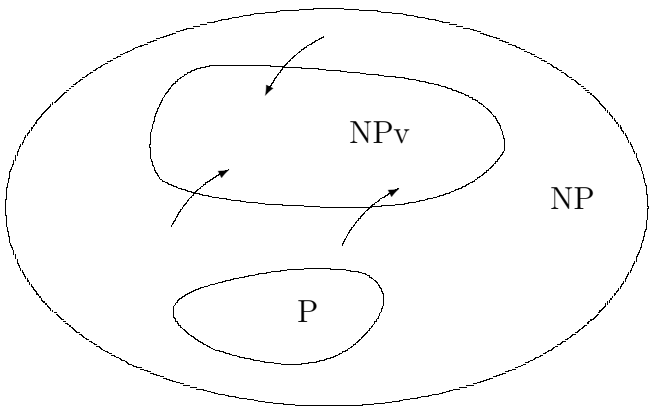


Abbildung 1.2: Die wichtigsten Komplexitätsklassen.

Beispiel: Gegeben sei das Bild einer Katze. Gesucht sei die Entscheidung, dass es sich um eine Katze handelt.

- Ein klassisches Computerprogramm (Menge von Regeln zur expliziten Modellierung der Äquivalenzklasse "Katze“) kann die Aufgabe nicht mit endlichem Aufwand lösen.

- Künstliche Neuronale Netzwerke (realisiert auf einem klassischen Computer) repräsentieren das erlernte Wissens implizit (ohne dies statisch in einem Programmcode zu repräsentieren) und können die Aufgabe mit endlichem Aufwand aber mit einer gewisser Unsicherheit lösen.

- Der Mensch löst die Aufgabe ebenfalls mit einer gewissen Ungenauigkeit durch Lernen anhand von Beobachtungen.

Das allgemeine Lernproblem ist also NP-vollständig, kann aber mit kNN approximiert werden.

Das Berechenbarkeitsprobleme hat viele Gesichter:

1.2.5.1 Mathematik / Logik (logisches Modell)

Eine zahlentheoretische Funktion f heiße **berechenbar**, wenn es einen Algorithmus gibt, der es gestattet, zu jedem Argumentwert n den Wert $f(n)$ anzugeben.

- D. Hilbert (1926): Vermutung, dass die Klasse der **primitiv rekursiven Funktionen** (primitive Rekursion: Regeln zu Komposition, Projektion und Iteration gewisser Basisfunktionen) alle berechenbaren Funktionen enthält.

- A. Church, St. Kleene (1936): **Allgemein rekursive Funktionen** (allgemeine Rekursion: enthält auch nichtdeterministische Rekursion) sind im Formalismus des λ-Kalküls ausdrückbar.

- **Church-These:** Alle berechenbaren Funktionen sind allgemein-rekursive Funktionen.

- K. Gödel (1930/32): Die Vollständigkeit eines Systems von Axiomen läßt sich nicht innerhalb des Axiomensystems beweisen (Unvollständigkeitssatz).
  $\hookrightarrow$ Berechenbarkeit ist in mathematisch invarianter Weise definierbar.
  $\hookrightarrow$ Die Entscheidbarkeit eines Problems ist probleminhärent und nicht darstellungsabhängig.

1.2.5.2 Informatik (logisch-konstruktives Modell)

- A. Turing (1936): "On Computable Numbers with an Application to the Entscheidungsproblem", Formulierung der Turing-Maschine als mechanisches Berechnungsmodell:
  1. unendlich langes beschreibbares und lesbares Band
  2. rechts-/links-verschiebbarer Lese-/Schreibkopf
  3. Zustandsübergangs- und Ausgabetabelle
  4. Kontrolle durch endlichen Automaten

- **Turing-These**: Jede berechenbare Funktion kann durch eine TM realisiert werden.
- Turing (1937): Äquivalenz von Turing-These und Church-These ("Programmierung auf Papier" wurde möglich)
- Verbindung der Berechenbarkeit mit dem Begriff des **Algorithmus**:
  - Allgemeinheit: Lösung einer Problemklasse
  - Endlichkeit bzgl. Aufwand und Terminierung
  - Determiniertheit: eindeutige Abhängigkeit der Ausgabe von der Eingabe

1.2.5.3 Informatik (praktisches Modell)

Eine Übersicht der Computergeschichte findet sich z.B. im Internet unter `http://de.wikipedia.org/wiki/Computergeschichte`.

- K. Zuse (Rechenautomat Z1, 1938): nur Kompositionen, keine Iteration, erfaßt nicht einmal die Menge primitiv rekursiver Funktionen
- IBM/Harvard-Universität (ASCC bzw. Harvard MARK I, 1944): Iterationen aber keine While-Schleifen, erfaßt primitiv rekursive Funktionen
  Anwendungen: sphärische Bessel- und Hankel-Funktionen (u.a. Flugbahn und Zündzeitpunkt der Nagasaki-Atombombe)
- Harvard-Universität (ENIAC, 1946): keine Trennung zwischen Speicher und Prozessor
  Anwendungen: Geschossbahnen, Berechnungen für die Wasserstoffbombe
- Universität Manchester (Manchester MARK I, 1948): erster speicherprogrammierbarer Rechner, erfaßt allgemein rekursive Funktionen

1.2.5.4 Informatik (Zellularautomaten)

- J. v. Neumann: gleichzeitige, hochgradig parallele Verarbeitung der Daten; Problem: Koordination und Kommunikation zwischen den Zellen (alle berechenbaren Funktionen sind auch mit Zellularautomaten berechenbar)
- J.H. Conway (1970): Game-Of-Life (Simulation eines zellulären Automaten)
- Bildverarbeitung (1970 - 1980er): Legendy, Duff
- systolische Prozessoren (für gleichartige Daten)
- Cellular Neural Nets (CNN) (1988)
- PLA $\Rightarrow$ FPGA (programmierbarer Logik-Baustein)

digitale Simulation:	endlicher Zustandsraum ↪ Turing-Äquivalenz (det. abz. unendl.)
analoge Simulation:	reelle Funktionen / unendlicher Zustandsraum ↪ Überschreitung der Turing-Berechenbarkeit

Tabelle 1.3: Simulationsgrenzen in Berechnungsmodellen.

1.2.5.5 Informatik (Neuronale Netze)

Jedes Berechnungsmodell erfordert eine Menge primitiver Funktionen und Kompositionsregeln für deren Kombination zu komplexen Funktionen. Im konventionellen Computer sind dies die Rechenstrukturen und die Termalgebra (von-Neumann-Architektur). NN sind jedoch:

- anders als von-Neumann-Rechner: keine Trennung von Verarbeitung und Speicherung
- anders als Turing-Maschine: nicht sequentiell
- anders als Zellularautomat: ex-hierarchische Schichtstrukturen
- so, dass alle Berechnungselemente verbunden sein können
- so, dass keine Programme gespeichert, sondern Netztopologie und -parameter antrainiert werden

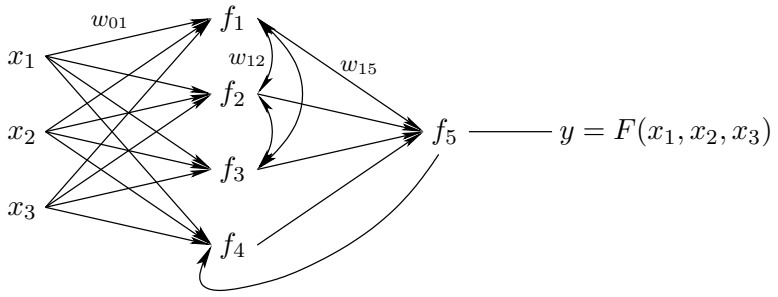


Abbildung 1.3: Ein neuronales Netz als Graph.

Ein NN ist als **gerichteter Graph** interpretierbar (siehe Abbildung 1.3) mit

- Knoten (Neuronen), welche die primitiven Funktionen tragen und
- Kanten (Verbindungen), welche die die Verarbeitungsregeln tragen

und wird charakterisiert durch die

- Struktur der Knoten,
- Verbindungsmuster (Topologie des Netzes) und

- Lernregeln, die Netzparameter (Verbindungen und Gewichte) modifizieren.

Das Berechenbarkeitsproblem begegnet uns in der Neuroinformatik (es ist jedoch nicht Gegenstand der Vorlesung) als Nachweis der **universellen Approximationsfähigkeit** einer gegebenen kNN-Architektur. Für bestimmte Architekturen (MLP, RBF) ist dieser Nachweis mit Mitteln der Funktionalanalysis (Satz von Stone und Weierstraß) bereits geführt worden.
In der **Praxis** ist das eben behandelte Berechenbarkeitsproblem aber von nachgeordneter Bedeutung, weil es keinerlei Aussagen über die nötigen Ressourcen macht. Vielmehr ist für die Lösbarkeit eines Problems in der Praxis eine Reihe von Randbedingungen ausschlaggebend. Dabei können die anfallenden Aufwendungen Anlaß zur Formulierung eines (suboptimalen) Ersatzproblems geben.

1.3 Historischer Abriß des NN-Paradigmas

- McCulloch, Pitts (1943): A Calculus of Ideas Immanent to Nervous Activity
  logisches Neuronenmodell (MP-Zelle, siehe Abschnitt 2.1)
  Netzwerke von (**gedächtnislosen**) MP-Zellen definieren die Klasse der regulären Ausdrücke.

- Hebb (1949): Modell des Lernens neuronaler Systeme (Hebb-Regel)
  Ein Trainingsmuster erzeugt durch **unüberwachtes Lernen** ein Aktivitätsmuster von Neuronen.

- Hodgkin, Huxley (1949-56): Untersuchung und Modellierung der axonalen Reizleitung der Neuronen
  Modellierung der auftretenden Depolarisationswellen mittels Telegraphengleichen (DGL) und Postulat, dass die Dynamik der Nervenimpulse durch elektrische Schaltungen realisierbar ist (ART-Netze, Grossberg).

- Rosenblatt (1961): Vorschlag des Perzeptrons als Neuronenmodell (siehe Kapitel 2.2)
  Kombination zum einschichtigen Perzeptron-Netz und Angabe der Lernregel (**überwachtes Lernen**)
  Die Verbindungsstärke der Knoten ändert sich in Abhängigkeit von der Abweichung des aktuellen Aktivitätsmusters der Neuronen von einem geforderten Aktivitätsmuster. Die Verbindungsstärke (Gewichte) der Knoten repräsentiert das Gedächtnis.

- Minsky, Papert (1969): "Perceptrons"
  Sorgfältige Analyse von einfachen einschichtigen Netzen aus Perzeptronen (SLN - single layer perceptron net).
  Bereits bei relativ einfachen Problemen führt dieses Netz zu einer kombinatorischen Explosion des Trainingsaufwandes, weswegen das Perzeptron-Modell einen ernsthaften Rückschlag erlitt.

- Rumelhart, McClelland (1986): MLP - Multilayer Perceptron (siehe Kapitel 4)
  Angabe eines überwachten Trainingsalgorithmus für MLP: **Backpropagation** - Lernalgorithmus.
  Ein MLP stellt gegenüber einem SLN eine Hierarchie von Neuronen dar.
  Lange Zeit wußte niemand, wie der Beitrag eines (verdeckten) Neurons dem festgestellten Fehler des Outputs zugeordnet und korrigiert werden kann.

- Hopfield (1982): Hopfield-Netz
  Vorschlag eines voll verbundenen rekursiven Netzes mit symmetrischen Verbindungen.
  Verwendung als autoassoziatives Netz

Ableitung des Modelles aus der statistischen Mechanik (Ferromagnetismus, Spingläser)

- Kohonen (1982): SOM
  Vorschlag der topologie-erhaltenden Abbildung des Zusammenhanges gegebener Daten in einem Netz von Neuronen durch **Selbstorganisation**.

- Girosi, Poggio (1990): RBF
  Vorschlag radialer Basisfunktionen als Bausteine eines neuronalen Netzes

- Bernhard Schölkopf (1997): Support Vector Learning (Dissertation)
  Einführung von SVM

- Alexander Johannes Smola (1998): Learning With Kernels (Dissertation)

1.4 Charakterisierung neuronaler Netze

In diesem Abschnitt werden einführend einige Begrifflichkeiten und Grundkonzepte von kNN erläutert.

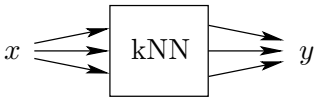


Abbildung 1.4: Ein kNN als Blackboxfunktion.

1.4.1 Netzwerktopologie

Eine Betrachtung der Netzwerktopologie eines kNN kann auf unterschiedlichen Ebenen erfolgen, z.B.:

a) System (siehe Abbildung 1.4)
Ein kNN kann als Funktion $f : \mathbb{R}^n \to \mathbb{R}^m$ interpretiert werden.
Input-Vektor $x := (x_1, \ldots, x_n)^T$ bildet Eingabeschicht
Output-Vektor $y := (y_1, \ldots, y_m)^T$ bildet Ausgabeschicht

b) Schichten
Eingabeschicht x
Ausgabeschicht y
verborgene/verdeckte Schichten h (hidden layers)

c) Menge von Verarbeitungselementen (Knoten/Neuronen, units)
Meist werden in einem kNN gleichartige Knoten verwendet. In jüngerer Zeit wurden auch kNN mit Knoten unterschiedlicher Funktionalität untersucht.

d) Netzwerkstruktur
Die Netzwerkstruktur wird durch eine Menge von Kanten repräsentiert. Die Kanten verbinden die Knoten zwischen den Schichten und innerhalb der Schichten. Sie können gewichtet oder ungewichtet auftreten. Die Gewichte w_{ij} zwischen den Schichten i und j wirken als Gedächtnis des NN. Sie speichern den aktuellen Zustand des Systems.

$$\left.\begin{array}{ll} \text{Knoten:} & \text{Verarbeitungselement} \\ \text{Kanten:} & \text{Verbindungsstruktur} \end{array}\right\} \text{gerichteter, gewichteter Graph}$$

Systeme mit Gedächnis generieren in Abhängigkeit von aktuellen Systemzustand und der momentanen Eingabe einen neuen Systemzustand sowie eine Ausgabe.

1.4.2 Allgemeine Begrifflichkeiten

Ein **NN** ist formal ein Quadrupel (K, V, I, O) mit
K Knotenmenge
V Kantenmenge $V \subset K \times K$
I Eingabeknoten $I \subset K$
O Ausgabeknoten $O \subset K$

Verdeckte Knoten: $H = K \setminus (I \cup O)$

Sei i der Index eines Knotens in der aktuellen Schicht, und sei j der Index eines Knotens in der vorherigen Schicht.
Dann heißt $w_{ij} > 0$ **exzitatorische Aktivierung eines Knotens** und $w_{ij} < 0$ **inhibitorische Aktivierung eines Knotens**.

(vollständiges) Fan-in:
Jeder Knoten der aktuellen Schicht bekommt Input von (jedem) Knoten der vorherigen Schicht.

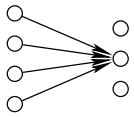


Abbildung 1.5: Vollständiges Fan-in für einen einzelnen Knoten.

(vollständiges) Fan-out:
Jeder Knoten der aktuellen Schicht ist mit (jedem) Knoten der darauffolgenden Schicht verbunden.

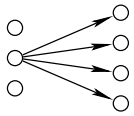


Abbildung 1.6: Vollständiges Fan-out für einen einzelnen Knoten.

Konnektivitätsmatrix bzw. Adjazenzmatrix des NN-Graphen:

$$W := [w_{ij}] \in \mathbb{R}^{n \times n}, \quad n \in \mathbb{N}$$

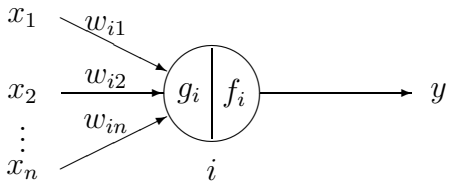
Repräsentationsmodus von Information:

- holistisch, holographisch: verteilt (über mehrere Knoten)
- "Großmutterzelle": lokalisiert (auf einem Knoten)

Beispiel: Repräsentation der Ziffern $0, \ldots, 9$ durch ein binäres Muster (verteilt: 4-Bit-Code im Stellenwertsystem; lokalisiert: 10-Bit-Code einer Dualzahl)

1.4.3 Knotendynamik

Die Knotendynamik beschreibt, wie ein einzelner Knoten seine Aktivierung bzw. seinen Zustand und den Output berechnet. Dies erfolgt in Abhängigkeit von seiner aktuellen Aktivierung und dem Input von anderen Knoten, siehe Abbildung 1.7.



$$y := f_i(u_i) \; mit \; u_i := g_i(x_1, \ldots, x_n \,|\, w_{i1}, \ldots, w_{in})$$

Abbildung 1.7: Generisches Neuron.

g: **Propagierungs- oder Inputfunktion** Berechnung der aktuellen Eingabe an internen Knoten anhand der Ausgabe vorgeschalteter Knoten und der Gewichte der Verbindungen.

f: **Aktivierungs- oder Übertragungsfunktion** Bestimmung des neuen Aktivierungszustandes des internen Knotens in Abhängigkeit vom propagierten Wert und eventuell vom vorherigen Aktivierungszustand.

1.4.3.1 Beispiele für Propagierungsfunktionen

Sei $n \in \mathbb{N}$, sei die Gewichtsmatrix $W = (w_{ij})_{i,j \leq n}$ des NN gegeben, und sei $g_i : \mathbb{R}^n \to \mathbb{R}$ die Propagierungsfunktion des Knotens/Neurons mit Index i.

- $u_i = g_i(x) := \sum_j w_{ij} x_j$ (linearer Assoziator)
- $u_i = g_i(x) := \prod_j w_{ij} x_j$ (nichtlinearer Assoziator)
- $u_i = g_i(x) := \max_j \{w_{ij} x_j\}$ (Maximum über alle gewichteten Eingaben)
- $u_i = g_i(x) := \sum_j s_j$ mit $s_j := \begin{cases} +1 & : \text{ falls } (w_{ij} x_j > 0) \\ -1 & : \text{ falls } (w_{ij} x_j < 0) \end{cases}$

  Majorität: Anzahl der positiv gewichteten Eingaben minus Anzahl der negativ gewichteten Eingaben
- $u_i = g_i(x) := \sum_j w_{ij} \prod_k x_{jk}$ $(\sum - \prod - \text{Netz})$

1.4.3.2 Beispiele für Aktivierungsfunktionen

Die **Sigmoidfunktion** $f_\sigma(u) := \frac{1}{1+e^{-\beta u}}$ wird in ihrem Funktionsverlauf durch kleines β flacher und durch großes β steiler. Sie ist praktikabel, da sie

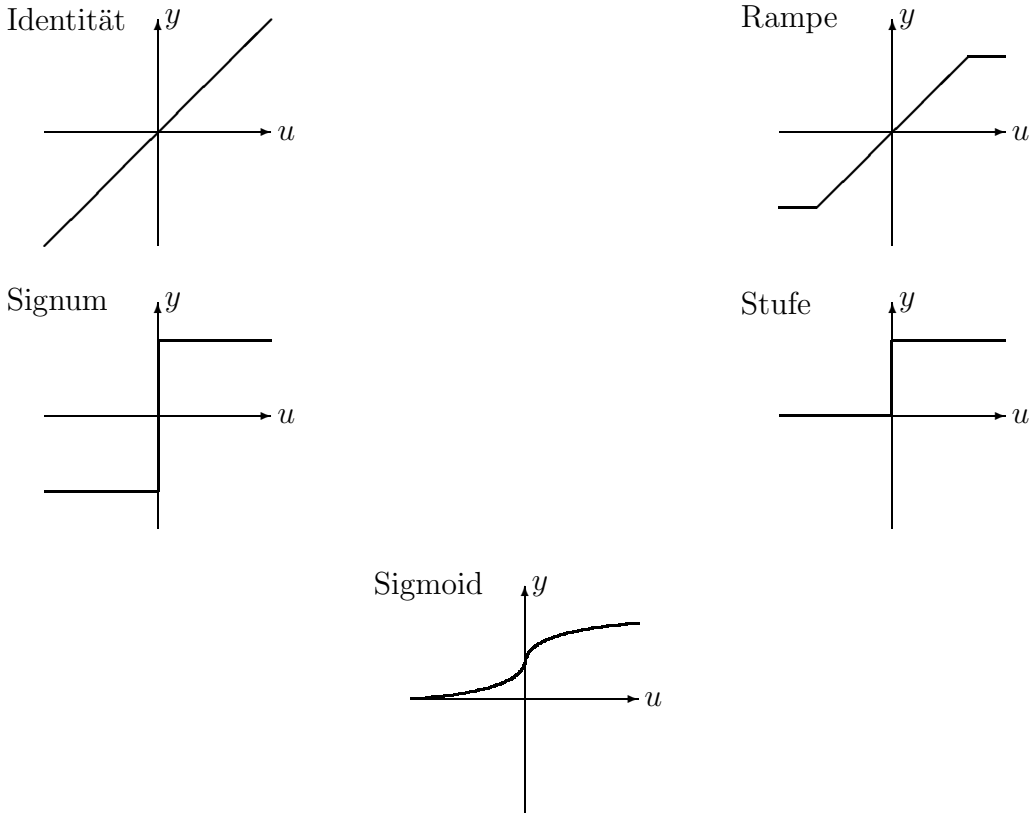


Abbildung 1.8: Aktivierungsfunktionen.

- differenzierbar ist (Lernalgorithmus nach Gradientenabstieg auf Fehlerfunktion wird ermöglicht),

- einen kontinuierlichen Output erzeugt, und

- der Output als Wahrscheinlichkeit interpretiert werden kann.

Neben der Sigmoidfunktion finden aber auch andere differenzierbare Funktionen (z.B. *tanh*) Verwendung. Abbildung 1.8 zeigt die Funktionsverläufe von einigen gebräuchlichen Aktivierungsfunktionen.

1.4.4 Erweiterung des Neuronenmodells

1. Einführung eines Schwellenwertes $\theta \in \mathbb{R}$
   Oft muß der propagierte Wert eines Neurons erst eine Schwelle überschreiten, damit ein spezieller Output erzeugt wird.
   Gängigerweise wird der Schwellenwert dem Gewichtsvektor der Propagierungsfunktion als zusätzlicher Eintrag hinzugefügt und mit 0 indiziert! Ebenso erhält der Inputvektor x einen 1-Eintrag an vorderster Stelle (siehe Abbildung

1.9):

$$x := (\underbrace{1}_{=:x_0}, x_1, \ldots, x_n)^T, \quad w_i := (\underbrace{-\theta}_{=:w_{i0}}, w_{i1}, \ldots, w_{in})^T$$

$$u_i := w_i^T x = w_{i0} + \sum_{k=1}^{n} w_{ik} x_k = \sum_{k=0}^{n} w_{ik} x_k$$

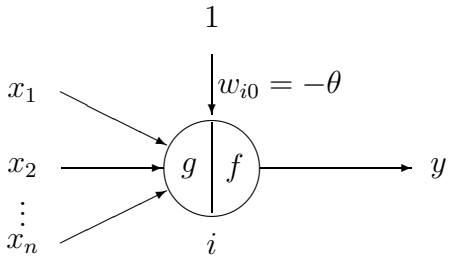


Abbildung 1.9: Neuron mit Schwellenwert.

2. Wechselwirkungen höherer Ordnung (HON)
   Multiplikative Verknüpfung von Paaren, Tripeln etc. von Eingaben:

$$u_i := w_{i0} + \sum_j w_{ij} x_j + \sum_{k,l} w_{ikl} x_k x_l + \ldots$$

3. Modifikation der Ausgabe
   - allgemein: keine Modifikation, also Identität: $output_i := y_i$
   - **winner-take-all**: Wettbewerb unter den Knoten

$$output_k := \begin{cases} WTA\{y_k\} & \text{für} \quad y_k = \max\{y_i \mid i \in \mathbb{N}_{\leq n}\} \\ 0 & \text{sonst} \end{cases}$$

   Der am stärksten aktivierte Knoten erzeugt den Output.

1.4.5 Netzwerkdynamik

Die Netzwerkdynamik wird bestimmt durch Topologie, Gewichte, Knotendynamik und Aktualisierungsmodus, wobei letzterer bestimmt, wann ein einzelnes Neuron seinen Aktivierungszustand aktualisiert. Hierfür ist die Kontrollstrategie des Informationsflusses von Bedeutung.

- **Vorwärts gerichtetes Netz**
  Es existiert ein ausschließlich vorwärtsgerichteter Informationsfluß (feed-forward), siehe Abbildung 1.10. Das Netz propagiert die Information vom

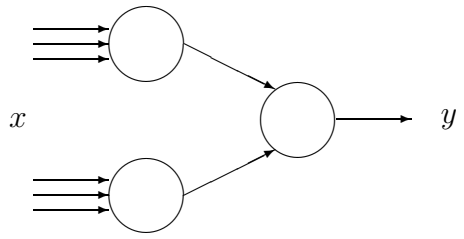
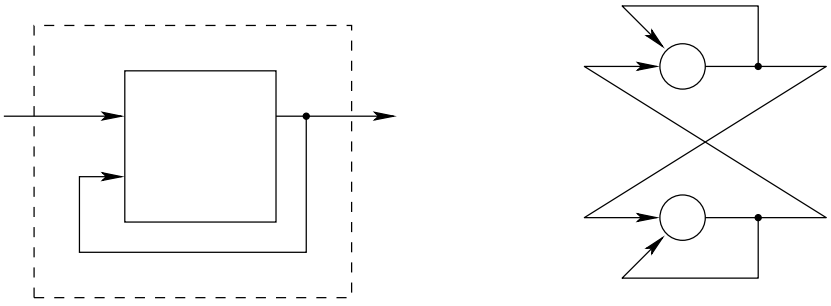


Abbildung 1.10: Vorwärts gerichtetes Netz.

Input zum Output in vorhersagbarer Weise.
Insbesondere wenn verdeckte Schichten vorhanden sind, sollte die Knotendynamik nicht zufällig sein, weil ansonsten die Propagierung der Knoten zu unübersichtlich wird (KISS-Prinzip).

- **synchrones Netz:** Die Ausgabe aller Neuronen wird gleichzeitig taktgesteuert berechnet.
- **paralleles Netz**: Der Zeitpunkt der Berechnung spielt für die Ermittlung des Outputs keine Rolle.
- **heteroassoziatives Netz**: Das Netz lernt in überwachtem Training, den Input mit dem Output zu assozieren.

- **Rückgekoppeltes (rekursives, rekurrentes) Netz**
  Es existiert ein Informationsfluß mit Rückkopplung(en), siehe Abbildungen 1.11 und 1.12.



(oft ist die Rückkopplung unbekannt)

(vollständige Rückkopplung)

Abbildung 1.11: Rückkopplung in Netzen.

Der Berechnungsprozeß ist nicht mehr allein durch die Vernetzung festgelegt, denn die zeitliche Abfolge der einzelnen Berechnungen hat Auswirkungen. Ferner sind nun Fixpunkte von Interesse.
Rekurrente Netze werden auch **asynchron** oder **sequentiell** genannt.

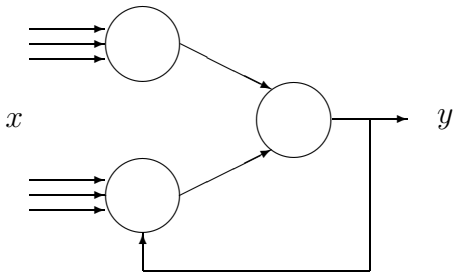


Abbildung 1.12: Ein kleines rückgekoppeltes Netz.

Die Dynamik eines zeitdiskreten Netzes ist mittels Differenzengleichung beschreibbar.
Die Dynamik eines zeitkontinuierlichen Netzes ist mittels Differentialgleichung beschreibbar.

1.4.6 Neuronales Lernen

Neuronales Lernen wird allgemein durch Modifikation

- der Gewichte und/oder
- der Struktur/Topologie

vollzogen. Eine Lernregel bestimmt dabei, wie und wann die Gewichte und die Topologie des Netzes zu verändern sind, um es an ein gegebenes Problem geeignet anzupassen.

Lernkonzepte:

- *Überwachtes Lernen* (**supervised learning**): Erfordert einen Lehrer, der eine Menge von Stichprobenelementen x^k zusammen mit den dafür korrekten Sollantworten y^k $(k = 1, 2, \ldots)$ vorgibt. Das Lernen besteht in der Minimierung der Differenz zwischen geforderten und tatsächlichen Antworten des kNN.
  Beispiel: Backprop-Algorithmus

- *Verstärkendes Lernen* (**reinforcement learning**): Eine fundamentale Aufgabe für jedes Lebewesen ist es, Strategien zu erlernen, die sein Verhalten in seinem Lebensraum optimieren. Das Lernparadigma des Reinforcement-Lernens beschreibt eine sehr allgemeine Art des Lernens. Die Umgebung erteilt in Abhängigkeit des aktuellen Zustandes x^k und der gewählten Aktion $a(x^k)$ positive oder negative Signale $r^k(x^k, a(x^k))$ (Rückmeldung (rewards) in Form von Belohnungen oder Bestrafungen).
  Beispiel: Evolutionäre Entwicklung neuronaler Neutzwerke

- *Unüberwachtes Lernen* (**unsupervised learning**): Erfordert nur eine Menge von Stichprobenelementen x^k. Ziel ist das Finden von Strukturen (z.B. Clustern). Dabei ist das Lernziel in den Lernregeln implizit enthalten. (Es muss ein Fixpunkt für die Netzwerkdynamik existieren.)
  Beispiel: Self-Organizing Maps (SOM)

Ferner werden zwei verschiedene Ansätze des Lernens unterschieden:

Offline-Lernen:	globale Sicht auf die Daten Es werden mehrere (meistens alle) Datenpunkte für einen Lern-(Iterations-)Schritt verwendet.
Online-Lernen:	lokale Sicht auf die Daten Für jeden betrachteten Datenpunkt wird unmittelbar ein Lern-(Iterations-)Schritt vollzogen.

Das Trainieren eines Netzes ist nicht streng formalisierbar, denn das Netz weiß nicht, was es lernen soll. Deshalb kommt Lehrer und Stichprobenangebot eine große Bedeutung zu. Dennoch werden in der **Lerntheorie** zunehmend systematische Erkenntnisse über das Lernbare erarbeitet.

In der **statistischen Entscheidungstheorie** (siehe Kapitel 3) wird der Zusammenhang zwischen der Größe des Datenvektors und der Größe der Stichprobe herausgearbeitet. Gelernt werden prinzipiell Invarianten der Stichproben, also Eigenschaften, die in allen Datenvektoren enthalten sind. Beispiel: Gesichtserkennung.

2 Einfache Neuronenmodelle

2.1 McCulloch-Pitts-Zelle und logische Operationen

McCulloch und Pitts (1943): Definition eines logischen Neuronenmodells

Eine MP-Zelle berechnet für einen binären Inputvektor x einen binären Output y. Seien $x_e = (x_1, x_2, ..., x_n)^T$ die n **erregende Leitungen** und $x_h = (x_{n+1}, ..., x_{n+m})^T$ die m **hemmende Leitungen** eines Inputs[1]. Der Input wird nicht gewichtet ver-

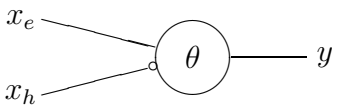


Abbildung 2.1: Veranschaulichung einer einfachen MP-Zelle.

arbeitet. Eine MP-Zelle vergleicht den Input mit einem Schwellwert $\theta \in \mathbb{R}$ nach folgenden Regeln:

a) Falls hemmende Leitungen existieren ($m \geq 1$) und an einer dieser wenigstens ein Eins-Signal/Hemmung anliegt, wird die MP-Zelle gehemmt und gibt eine Null aus (absolute Hemmung).

b) Liegen keine hemmenden Signale an, wird die Summe der erregenden Eingabesignale $x_1, ..., x_n$ berechnet und mit dem Schwellwert θ verglichen.

c) Falls die Erregung größer oder gleich dem Schwellwert ist, gibt das Neuron eine Eins, andernfalls eine Null aus.

$$y := \begin{cases} 0, & \text{falls} \quad \sum_{j=1}^{m} x_{n+j} \geq 1 \\ 0, & \text{falls} \quad \sum_{i=1}^{n} x_i < \theta \\ 1, & \text{sonst} \end{cases}$$

Liegen keine hemmenden Signale an, verhält sich die MP-Zelle wie ein Mehrheitsgatter (mächtiger als AND- und OR-Gatter).

Ein **MP-Netz** ist ein gerichteter Graph, der aus einer Menge von Knoten (MP-Zellen) und zwei Klassen gerichteter Kanten (hemmende und erregende Leitungen) besteht. Dieser Graph kann vorwärtsgerichtet oder rekursiv sein.

[1] Hemmende Leitungen werden in einer graphischen Darstellung mit einem Negationszeichen an der Zelle notiert, siehe Abbildung 2.1.

2.1.1 Konstruktion logischer Funktionen mittels MP-Zellen

a) **Monotone logische Funktionen** $f : \mathbb{B}^n \to \mathbb{B}$ erfordern nur erregende Leitungen.
Beispiel für $n = 2$:

x_1	0	1	0	1	
x_2	0	0	1	1	
y_{AND}	0	0	0	1	$\theta_{\text{AND}} = 2$
y_{OR}	0	1	1	1	$\theta_{\text{OR}} = 1$

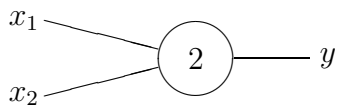


Abbildung 2.2: MP-Zelle für f_{AND}.

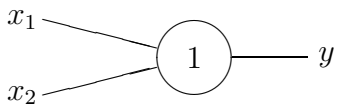


Abbildung 2.3: MP-Zelle für f_{OR}.

Beispiel für $n = 3$: siehe Beispiel für $n = 2$ oben mit $\theta_{\text{OR}} = 1$, $\theta_{\text{AND}} = 3$
Im Gegensatz hierzu vergleiche die bekannte Schaltalgebra für drei Variable:

Konjugation: 2 AND-Gatter mit $n = 2$ Eingängen
Disjunktion: 2 OR-Gatter mit $n = 2$ Eingängen

Eine MP-Zelle wirkt als **Mehrheitsgatter**, liefert also einen mächtigen logischen Baustein. Aber die Schaltalgebra erfordert auch Negation!

b) **Nichtmonotone logische Funktionen** $f : \mathbb{B}^n \to \mathbb{B}$ erfordern auch hemmende Leitungen.

Beispiel NOT: Abbildung 2.4 zeigt eine Zelle, welche nur dann gehemmt ($y_{\text{NOT}} = 0$) wird, wenn das hemmende Signal aktiv ist ($x = 1$).

x	0	1	
y_{NOT}	1	0	$\theta_{\text{NOT}} = 0$

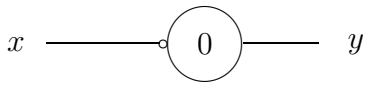


Abbildung 2.4: MP-Zelle für f_{NOT}.

Beispiel NOR: $y_{\mathrm{NOR}} = \overline{y}_{\mathrm{OR}} = \overline{x_1 \vee x_2} = \overline{x}_1 \wedge \overline{x}_2$

x_1	0	1	0	1	
x_2	0	0	1	1	
y_{NOR}	1	0	0	0	$\theta_{\mathrm{NOR}} = 0$

Die NOR-Funktion kann kanonisch durch ein Zweischicht-Netz aus MP-Zellen realisiert werden (siehe Abbildung 2.5).

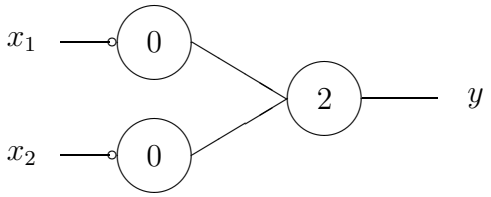


Abbildung 2.5: Kanonische MP-Zelle für f_{NOR}.

Aber auch die spezielle NOR-MP-Zelle in Abbildung 2.6 kann aus der Wertetabelle entworfen werden.

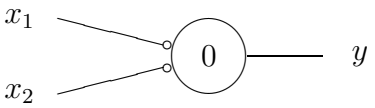


Abbildung 2.6: Spezielle MP-Zelle für f_{NOR}.

Beispiel INH (Inhibition): $y_{\mathrm{INH}} = x_1 \wedge \overline{x}_2$

x_1	0	1	0	1	
x_2	0	0	1	1	
$\overline{x}_2$	1	1	0	0	
y_{INH}	0	1	0	0	$\theta_{\mathrm{INH}} = 1$

Die INH-Funktion kann kanonisch durch ein Zweischicht-Netz aus MP-Zellen realisiert werden (siehe Abbildung 2.7).

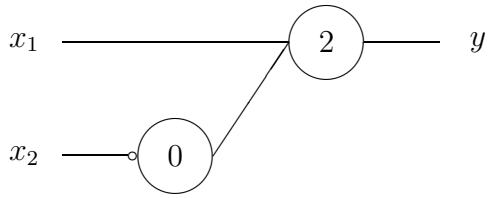


Abbildung 2.7: Kanonische MP-Zelle für f_{INH}.

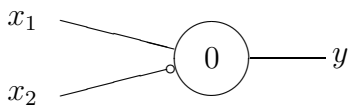


Abbildung 2.8: Spezielle MP-Zelle für f_{INH}.

Aber auch die spezielle INH-MP-Zelle in Abbildung 2.9 kann aus der Wertetabelle entworfen werden.

Jedoch nicht jede der zweistelligen logischen Funktionen kann durch eine einzige (ungewichtete) MP-Zelle repräsentiert werden.

Beispiel NAND: $y_{\text{NAND}} = \overline{y}_{\text{AND}} = \overline{x_1 \wedge x_2} = \overline{x}_1 \vee \overline{x}_2$
Hierfür wird ein Netz von MP-Zellen (zwei NOT und ein OR) benötigt.

x_1	0	1	0	1
x_2	0	0	1	1
y_{NAND}	1	1	1	0

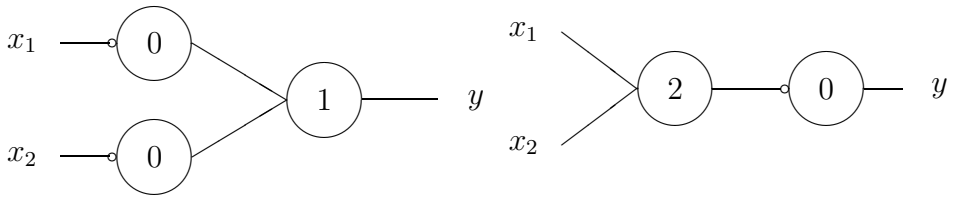


Abbildung 2.9: Zwei Variationen eines MP-Netzes für die NAND-Funktion.

Jede logische Funktion von n binären Variablen kann also mit einem Netz berechnet werden, das nur die Knoten AND, OR und NOT benutzt.

2.1.2 MP-Netztypen

- **Mehrstufigie MP-Netze** erzeugen einfachere Realisierungen (weniger Zellen) als einstufige MP-Netze.
  Beispiel für $n = 3$:

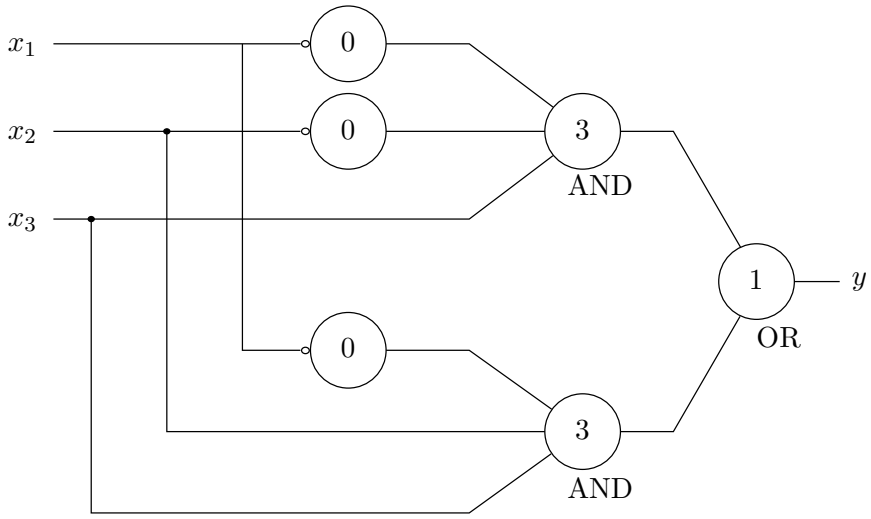


Abbildung 2.10: MP-Netz für $y := (\overline{x}_1 \wedge \overline{x}_2 \wedge x_3) \vee (\overline{x}_1 \wedge x_2 \wedge x_3)$.
(Negation: $x_i = 0 \rightarrow \overline{x}_i = 1$)

x_1	x_2	x_3	y
0	0	1	1
0	1	1	1
sonst			0

Tabelle 2.1: Eingabebelegungen für x_1, x_2 und x_3.

Die Funktion aus Abbildung 2.10 kann als Detektor für die Belegungen der Eingabevariablen aus Tabelle 2.1 angesehen werden. Nur wenn eine dieser beiden Belegungsvarianten auftritt, wird das Netz feuern ($y = 1$), andernfalls gilt $y = 0$. Genauer, die Funktion stellt die Disjunktion zweier Detektoren dar. Nur wenn einer dieser Detektoren feuert, wird die Funktion erfüllt.
Ein Detektor definiert Bedingungen, die konjunktiv sind. Hieraus resultiert ein kompakteres Konstruktionsprinzip eines MP-Netzes. Aber auch die MP-Zellen für NOR und INH sind Beispiele hierfür. Das Konstruktionsprinzip versagt jedoch für NAND wegen der disjunktiven Vertauschung der Argumente.

Konstruktionsprinzip

- Jedem Eingabevektor mit Funktionswert 1 wird eine Zelle zugeordnet.
- Input 0 $\rightsquigarrow$ hemmende Verbindung
- Input 1 $\rightsquigarrow$ erregende Verbindung
- Schwellenwert θ wird gesetzt als die Anzahl der Einsen im Inputvektor.
- Verknüpfung dieser Zellen (der ersten Schicht) über erregende Verbindungen zu Zellen der zweiten Schicht.

– Die Schwellenwerte θ der zweiten Schicht werden auf 1 gesetzt.

Die Anwendung des Konstuktionsprinzips auf die Funktion aus Abbildung 2.10 ergibt das in Abbildung 2.11 dargestellte Netz.

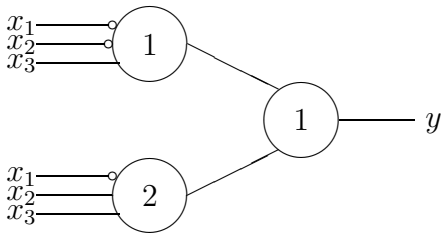


Abbildung 2.11: MP-Netz aus Abbildung 2.10 nach angegebenem Konstruktionsprinzip.

Die Gesetze der Schaltalgebra sind hierauf übertragbar.
Insbesondere gilt: Jede Boolesche Funktion $f : \mathbb{B}^n \to \mathbb{B}$ kann mit einem zweischichtigen MP-Netz berechnet werden (z.B. mittels ihrer disjunktiven Normalform).

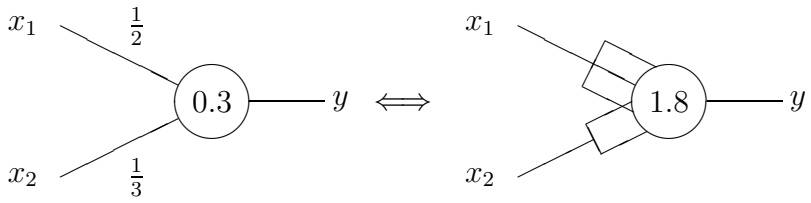
- **Rekursive MP-Netze**
  Vorwärts gerichtete MP-Netze sind gedächtnislos. Sie sind geeignet, beliebige logische Funktionen zu berechnen, wenn deren Eingabe mit konstanter Länge komplett anliegt. Dies ist die Eigenschaft eines Schaltnetzes. Beispielsweise ist hiermit aber nicht die Konstruktion eines Addierers möglich, da das Übertragsbit nicht dem Input zuzuordnen ist.
  Ein Schaltwerk erfordert eine Möglichkeit der Speicherung von Information. **Rekursive MP-Netze besitzen ein Gedächtnis** in Gestalt von Verzögerungselementen!
  Ferner ist jeder endliche Automat einem rekursiven MP-Netz äquivalent.

2.1.2.1 Gewichtete MP-Netze

Eine geradlinige Erweiterung der MP-Zelle besteht in der Zulassung positiver rationaler Gewicht. Zu jedem **positiv, rational gewichteten** MP-Netz kann ein äquivalentes ungewichtetes MP-Netz angegeben werden. Die beiden Netze unterscheiden sich dabei in Topologie und Schwellwerten, wie Abbildung 2.12 veranschaulicht.
Als eine abermals strigente Erweiterung des Konzeptes der positiv rational gewichteten MP-Zelle können auch negative Gewichte zugelassen werden.
Es sind also folgende drei Typen der MP-Zelle von einander zu unterscheiden:

1. MP-Zelle nur mit hemmenden und nicht-hemmenden Leitungen.
2. MP-Zelle mit positiv rationalen gewichteten Leitungen.
3. MP-Zelle mit rationalen gewichteten Leitungen.

Die zuletzt genannte Erweiterung der MP-Zelle stellt den fließenden Übergang zu den derzeit am häufigsten betrachteten Zellen (Neuronen) dar, den Perzeptronen (siehe Kapitel 2.2).



$$\frac{1}{2}x_1 + \frac{1}{3}x_2 \geq 0.3 \iff 3x_1 + 2x_2 \geq 1.8$$

Abbildung 2.12: Beispiel eines gewichteten MP-Netzes.

2.1.2.2 Nachteilige Eigenschaften von MP-Netzen

- Das MP-Netz hat den Nachteil eines "unbegrenzten fan-in", dass heißt unendlich vieler Eingabeleitungen für nicht-rationale Gewichte.
- Die logischen Schaltungen sind nicht biologisch relevant.
- Die Berechnungseinheiten müssen vollständig spezifiert sein, um angewendet werden zu können. Es gibt keine freien Parameter, um ein MP-Netz an ähnliche Problemstellungen anpassen zu können. Lernen existiert für ein MP-Netz nicht.

2.2 Perzeptron und lineare Entscheidungsprobleme

Frank Rosenblatt (Rosenblatt (1958)):

- Vorschlag des "Perceptron" als Neuronenmodell aus Sicht der kognitiven Psychologie
- reellwertige Eingabe und lineare Ausgabe mit reellwertigen Gewichten/Schwellenwert
- **Gewichtete Verknüpfung** mehrerer Neuronen zu einem **vorwärtsgerichteten** Netz
- beschränkte Anzahl von Verbindungen pro Neuron
- Verbindung des Perzeptron-Netzes mit extensiver Vorverarbeitung der Daten Ziel: ein beliebig komplexes Problem wird durch die Vorverarbeitung in ein **linear entscheidbares Problem** überführt.

- Ein **(überwachtes) Lernen** führt zu Änderungen der Gewichte der Kanten des Graphen, läßt aber dessen Topologie ungeändert.

Marvin Lee Minsky und Seymour Papert (Minsky und Papert (1969)):

- zeigten, dass Verbundenheitsproblem und Paritätsproblem (XOR) damit nicht gelöst werden können.

Sei $\theta \in \mathbb{R}$ eine Schwelle, sei $x := (x_1, ..., x_n)^T \in \mathbb{R}^n$ eine Eingabe und sei $w := (w_1, ..., w_n)^T \in \mathbb{R}^n$ ein Vektor indizierter Gewichte.

Ein **Perzeptron** ist eine Berechnungseinheit bzw. Zelle mit Schwellenwert θ, Eingabe x und Gewichtsvektor w sowie Ausgabe

$$y = \begin{cases} 1 & \text{wenn} \quad \sum_{i=1}^{n} w_i x_i \geq \theta \\ 0 & \text{sonst} \end{cases}$$

Die **Propagierungsfunktion** ist linear und wird durch das Skalarprodukt

$$u = \langle w, x \rangle = w^T x = \sum_{i=1}^{n} w_i x_i$$

dargestellt. Diesen Teil des Perzeptrons nennt man auch **linearen Assoziator**.

Die **Aktivierungsfunktion** ist eine nichtlineare Funktion, z.B. die Stufenfunktion

$$y = \text{ step}(u) = \begin{cases} 1 & \text{, wenn} \quad u \geq \theta \\ 0 & \text{, wenn} \quad u < \theta \end{cases}$$

oder die Signumfunktion

$$y = \text{ sgn}(u) = \begin{cases} 1 & \text{, wenn} \quad u \geq \theta \\ -1 & \text{, wenn} \quad u < \theta. \end{cases}$$

kann als Aktivierungsfunktion verwendet werden. Damit wird das Perzeptron zu einem nichtlinearen Berechnungselement.

In diesem **nicht erweiterten** Perzeptron-Modell wirkt die Schwelle als "Bias" (lat. Leben) der Propagierungsfunktion. Das **erweiterte Modell** nutzt die um den Betrag der Schwelle θ reduzierte Propagierungsfunktion (vgl. Abbildung 2.13)

$$u := \sum_{i=1}^{n} w_i x_i - \theta = \sum_{i=0}^{n} w_i x_i$$

mit konstanter Eingabe $x_0 = 1$ und Gewicht $w_0 = -\theta$. Im erweiterten Modell vergleicht also die Aktivierungsfunktion die reduzierte Propagierungsfunktion mit Null.

Das Lernen eines Perzeptrons bezieht sich auf die Gewichte und den Schwellenwert. Für ein Perzeptron stehen folgende äquivalente Interpretationen bereit:

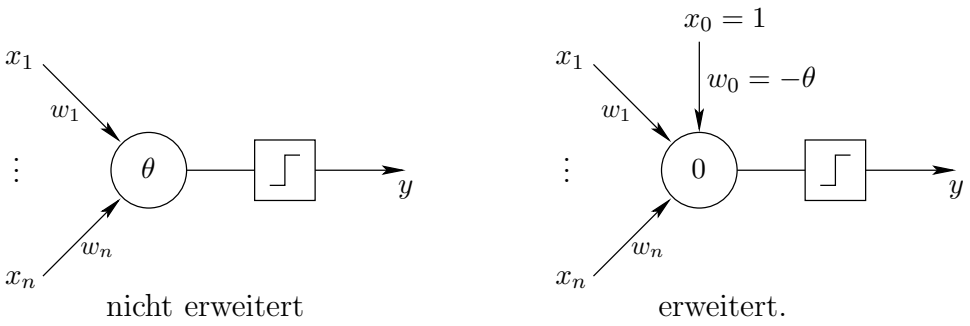


Abbildung 2.13: Einfaches und erweitertes Perzeptron.

- **binäre logische Entscheidungseinheit**
- **lineare Diskriminanzfunktion** für ein 2-Klassen-Problem

Ein Perzeptron **heteroassoziert** im trainierten Zustand den Input mit dem geforderten Output. Beim Lernen (siehe Abschnitt 2.3) werden die Gewichte (und Schwelle) so angepaßt, dass sie das Problem repräsentieren.

Setzt man mehrere Perzeptrone zu einem Perzeptron-Netz zusammen, können komplexere logische Regeln bzw. komplexere Entscheidungsprobleme repräsentiert werden, siehe Abbildung 2.14.

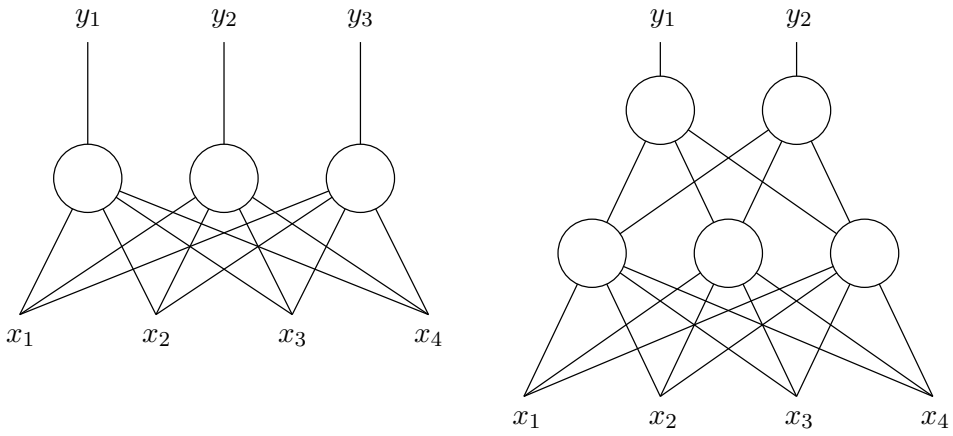


Abbildung 2.14: Perzeptron-Netze.

2.2.1 Repräsentation von Entscheidungsfunktionen

Eine Entscheidungsfunktion liefert ein binäres Ergebnis, das also mit TRUE oder FALSE interpretiert werden kann. Die logischen Funktionen $f : \mathbb{B}^n \mapsto \mathbb{B}$ und die

x_1	x_2	f_{AND}	f_{OR}
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

Tabelle 2.2: Wertetabelle für f_{AND} und f_{OR}.

Diskriminanzfunktionen $f : \mathbb{R}^n \mapsto \mathbb{B}$ gehören hierzu.

Sei $f : \mathbb{B}^n \to \mathbb{B}$ eine logische Funktion.
Problem: Wie müssen die Gewichte und die Schwelle gewählt werden, um die Wahrheitstabelle der Funktion f zu erfüllen? Wann ist also für den Output y des Perzeptrons die Gleichung $y = f(x)$ erfüllt?

Beispiel für $n = 2$: Tabelle 2.2 zeigt die benutzten Funktionen
Gelöst wird ein System von Ungleichungen, das sich durch die Wertebelegung der Eingaben ergibt:

$$\begin{aligned} w_1x_1 + w_2x_2 \geq \theta &\iff \text{TRUE} \quad (y = 1) \\ w_1x_1 + w_2x_2 < \theta &\iff \text{FALSE} \quad (y = 0) \end{aligned}$$

Hier wird als Aktivierungsfunktion eine Stufenfunktion angenommen.

Im Falle von f_{AND} ergibt sich (siehe Abbildung 2.15):

$$\begin{aligned} w_1 \cdot 0 + w_2 \cdot 0 &< \theta \\ w_1 \cdot 0 + w_2 \cdot 1 &< \theta \\ w_1 \cdot 1 + w_2 \cdot 0 &< \theta \\ w_1 \cdot 1 + w_2 \cdot 1 &\geq \theta \end{aligned}$$

Diese vier Ungleichungen repräsentieren die Funktion f_{AND}. Der Zwang dieser Funktionsdefinition ist relativ schwach, wie an der Lage der Trennfunktion zwischen Stichprobenelementen $x \in P$ oder $xinN$ in Abbildung 2.15 zu erkennen ist.
Hierbei wird der durch den Eingabevektor aufgespannte Eingaberaum in zwei Halbräume getrennt.
Die Problemdarstellung im EIngaberaum gibt einen intuitiven Zugang zur Erweiterung der behandelbaren Funktionen entsprechend $f : \mathbb{R}^n \mapsto \mathbb{B}$. Damit wird der Raum der logischen Funktionen verlassen und es werden Diskriminanzfunktionen beschrieben.
Bei geeignet gewähltem Gewichtsvektor führt eine Gruppe der Eingabedaten zur

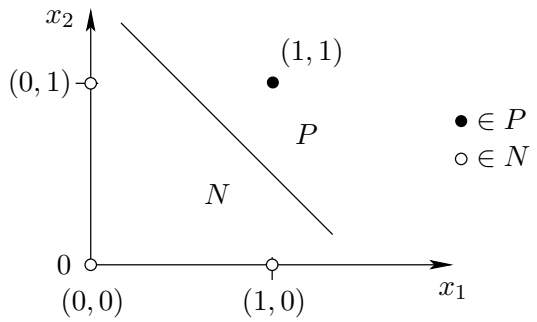


Abbildung 2.15: Lineare Trennbarkeit von f_{AND}.

Entscheidung TRUE ($y = 1$) und eine andere zur Entscheidung FALSE ($y = 0$). Das entspricht einem 2**-Klassen-Entscheidungsproblem**. Das Perzeptron ist ein also **linearer Klassifikator**.

Satz 2.1 (Lineare Trennbarkeit) *Zwei n-dimensionale Mengen P und N von Punkten im Eingaberaum ($\Omega := P \cup N$) heißen **linear trennbar**, falls $n+1$ reelle Zahlen $\theta, w_1, \ldots, w_n$ existieren, so dass für jeden Punkt $x \in P$ gilt: $\sum_{i=1}^{n} w_i x_i \geq \theta$, und für jeden Punkt $x \in N$ gilt: $\sum_{i=1}^{n} w_i x_i < \theta$.*

Der Eingaberaum heißt **absolut linear trennbar**, falls gilt:

$$\begin{aligned} x \in P &\iff w^T x > \theta \\ x \in N &\iff w^T x < \theta \end{aligned}$$

Die **Trennfunktion** $w_1 x_1 + w_2 x_2 = \theta$ entspricht einer Geradengleichung:

$$x_2 = -\frac{w_1}{w_2} x_1 + \frac{\theta}{w_2}$$

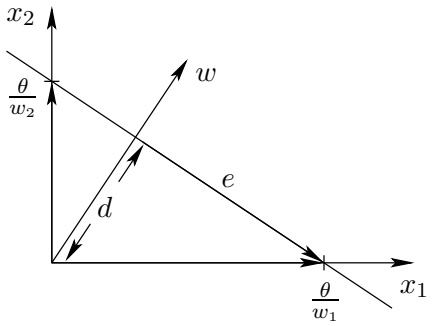


Abbildung 2.16: Veranschaulichung einer Trenngeraden e.

Die Schnittpunkte mit den Koordinaten des Eingaberaumes sind:

$$x_2 = 0 \;\Rightarrow\; x_1 = \frac{\theta}{w_1}$$
$$x_1 = 0 \;\Rightarrow\; x_2 = \frac{\theta}{w_2}$$

Der Vektor der Trenngeraden ergibt sich hier wie folgt:

$$e = \begin{pmatrix} \frac{\theta}{w_1} \\ 0 \end{pmatrix} - \begin{pmatrix} 0 \\ \frac{\theta}{w_2} \end{pmatrix} = \begin{pmatrix} \frac{\theta}{w_1} \\ -\frac{\theta}{w_2} \end{pmatrix}$$

Der Abstand zum Ursprung mit $\|\cdot\|$ als der Euklidischen Norm eines Vektors ist:

$$d = \frac{\theta}{||w||}$$

Satz 2.2 *Die Trenngerade steht senkrecht zum Gewichtsvektor:*

$$e \perp w$$

Es gilt:

$$\langle e, w \rangle = \frac{\theta}{w_1} \cdot w_1 + (-)\frac{\theta}{w_2} \cdot w_2 = \theta - \theta = 0$$

Im Fall der logischen Funktionen f_{AND} und f_{OR} aus Tabelle 2.2 folgen aus den wenigen Eingabedaten nur schwache Zwänge für die Lage der Trennfunktion. Diese logischen Funktionen können deshalb durch viele Varianten der Gewichtsvektoren realisiert werden, z.B.:

- **Realisierung von f_{AND}:**
  Beispiel: $w = (w_0, w_1, w_2)^T = (-1, 0.5, 0.7)^T$

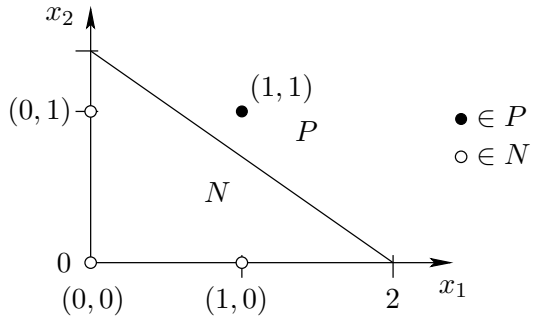


Abbildung 2.17: Trenngerade für f_{AND}.

Für $x_1 = 0$ folgt $x_2 \approx 1.4$.
Für $x_2 = 0$ folgt $x_1 = 2$.

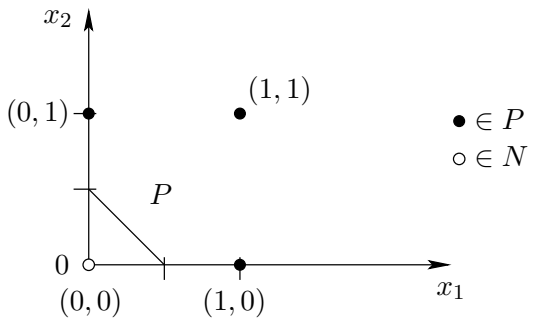


Abbildung 2.18: Trenngerade für f_{OR}.

- **Realisierung von $f_{\mathbf{OR}}$:**
  Beispiel: $w = (w_0, w_1, w_2)^T = (-1, 2, 2)^T$

  Für $x_1 = 0$ folgt $x_2 = 0.5$.
  Für $x_2 = 0$ folgt $x_2 = 0.5$.

2.2.2 Interpretation des Skalarproduktes

Alle Punkte x auf der Trennfunktion (Gerade im zweidimensionalen Fall) erfüllen die Gleichung:

$$\langle w, x \rangle = w^T x = \theta$$

Wegen der Kommutativität des Skalarproduktes gilt aber auch:

$$\langle x, w \rangle = x^T w = \theta$$

Hierbei treten **Eingaberaum** und **Gewichtsraum** als duale / syntaktisch gleichwertige Räume auf.
Das Skalarprodukt kann als Projektion eines Vektors auf einen anderen interpretiert werden:

a) Im Eingaberaum gilt $w^T x = \|w\| \cdot \|x\| \cos\varphi =: \|w\| \cdot \|x_w\|$. Somit kann $x_w = \|x\| \cos\varphi$ als Projektion von x auf w interpretiert werden, siehe Abbildung 2.19.

b) Im Gewichtsraum gilt $w^T x = \|x\| \cdot \|w\| \cos\varphi =: \|x\| \cdot \|w_x\|$. Somit kann $w_x = \|w\| \cos\varphi$ als Projektion von w auf x interpretiert werden.

Unabhängig vom Koordinatensystem wird das Skalarprodukt nur durch die Norm der Vektoren und durch den eingeschlossenen Winkel bestimmt.
Nach Normierung der Vektoren x und w ist nur noch der eingeschlossene Winkel φ maßgebend:

$$\langle w, x \rangle = \cos\varphi = \theta$$

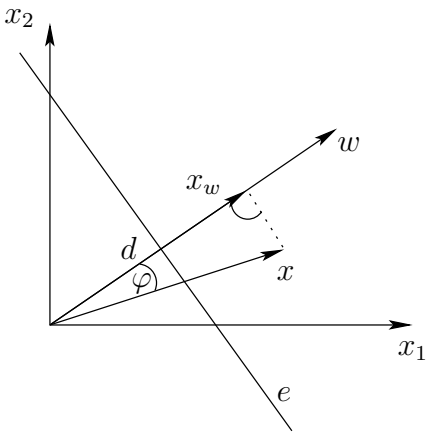


Abbildung 2.19: Darstellung des Skalarproduktes im Eingaberaum.

Die Bedingung der **linearen Trennbarkeit** der Eingabedaten stellt sich nun dar als:

$$\begin{aligned} x \in P &: -\tfrac{\pi}{2} \leq \varphi \leq \tfrac{\pi}{2} \\ x \in N &: \tfrac{\pi}{2} < \varphi < \tfrac{3}{2}\pi. \end{aligned}$$

Es wird festgelegt, dass der Gewichtsvektor stets in den positiven Halbraum zeigt! Wenn also Gewichtsvektor und Eingabevektor in den gleichen Halbraum zeigen, muß $x \in P$ gelten. Hieraus leitet sich das Lernproblem für die Suche eines geeigneten Gewichtsvektors ab. Wie oben beschrieben, ist die Lösung aber nicht eindeutig. Vielmehr müssen nur die Bedingungen der linearen Trennbarkeit erfüllt werden.

Allgemein gilt für den Abstand d der Trennfunktion vom Koordinatenursprung des Eingaberaums:

$$d = \frac{\theta}{\|w\|}$$

Hieraus folgen die Äquivalenzen für die Formulierung einer Entscheidung:

$$\begin{aligned} & w^T x \geq \theta \quad \rightarrow \quad y = 1 \\ \iff \quad & \|w\| \cdot \|x_w\| \geq \|w\| d \quad \rightarrow \quad y = 1 \\ \iff \quad & \|x_w\| \geq d \quad \rightarrow \quad y = 1 \end{aligned}$$

Beispiel: f_{OR} (siehe auch Seite 32 und Abbildung 2.18)

a) $x := (1,1)$, $w := (2,2)$, $\theta := 1$, siehe Abbildung 2.20
$\|x\| = \sqrt{2}, \|w\| = 2\sqrt{2}, d = \frac{1}{2\sqrt{2}}$
$\cos\varphi = \frac{w^T x}{\|w\|\|x\|} = \frac{w_1 x_1 + w_2 x_2}{\|w\|\|x\|} = 1 \rightarrow \varphi = 0^o$: x und w haben dieselbe Richtung

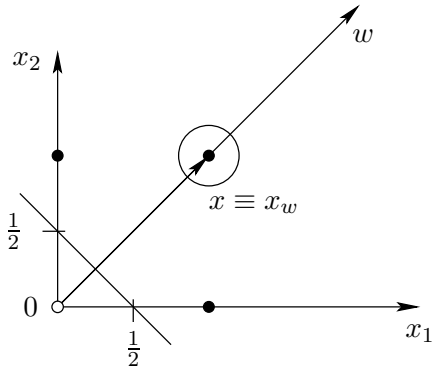


Abbildung 2.20: Lineare Trennbarkeit - Beispiel 1.

$x_w = \|x\| \cos\varphi = \sqrt{2} \rightarrow x_w > d : x \in P$

b) $x := (0,1)$, $w := (2,2)$, $\theta := 1$, siehe Abbildung 2.21

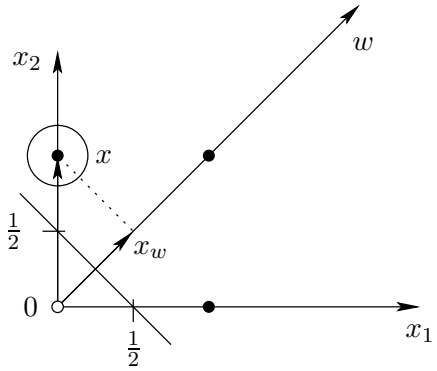


Abbildung 2.21: Lineare Trennbarkeit - Beispiel 2.

$\|x\| = 1$, $\|w\| = 2\sqrt{2}$, $d = \frac{1}{2\sqrt{2}}$

$\cos\varphi > \frac{1}{\sqrt{2}} \rightarrow \varphi = 45^o : x$ liegt im gleichen Halbraum wie w

$x_w = \frac{1}{\sqrt{2}} \rightarrow x_w > d : x \in P$

Im allgemeinen Fall eines n-dimensionalen Eingaberaums stellt die Trennfunktion eine $(n-1)$-dimensionale Hyperfläche dar.

Im Fall des erweiterten Perzeptronmodells (im $(n+1)$-dimensionalen Eingaberaum mit $x_0 = 1$ und $w_0 = -\theta$) gilt für die Trennfunktion $w^T x = 0$. Also geht die Trennfunktion wegen $d = 0$ durch den Ursprung dieses um eine Dimension erhöhten Eingaberaumes.

2.2.3 Alternatives Klassifizierungsproblem im Eingaberaum

Sei $\Omega = P \cup N$ n-dimensional und sei die Aktivierungsfunktion durch die Signumfunktion gegeben, d.h. $y \in \{-1, +1\}$. Sei $k \in \mathbb{N}$. Es werden signierte Stichprobenelemente (x^k, y^k) betrachtet, wobei y^k die Zugehörigkeit von x^k zu P oder N wie folgt angibt:

$$\begin{array}{lcl} x^k \in P & : & w^T x^k > 0 \;\leftrightarrow\; y^k := +1 \;\leftrightarrow\; y^k w^T x^k > 0 \\ x^k \in N & : & w^T x^k < 0 \;\leftrightarrow\; y^k := -1 \;\leftrightarrow\; y^k w^T x^k > 0 \end{array}$$

Weiter wird folgender alternativer Stichprobenraum formuliert: $\Omega' := P' \cup N'$ mit $P' := \left\{x^k | x^k \in P\right\}$ und $N' := \left\{-x^k | x^k \in N\right\}$. Sei $\xi^k \in \Omega'$, dann gilt $\xi^k = y^k x^k$.
Im ξ-Eingaberaum müssen für die lineare Trennbarkeit eines Problems alle Eingabevektoren auf derselben Seite der Trennfunktion liegen:

$$w^T \xi^k > 0$$

Beispiel: $P := \{A, B, C\}$, $N := \{H, I, J\}$ → siehe Abbildung 2.22

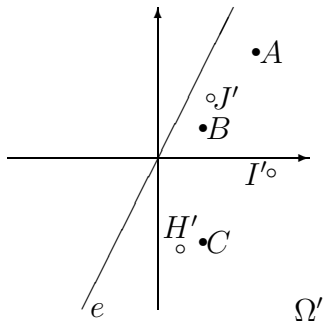


Abbildung 2.22: Lineare Trennbarkeit im alternativen Eingaberaum.

Beispiel: f_{AND}

	x^1	x^2	x^3	x^4
x_0^k	1	1	1	1
x_1^k	0	0	1	1
x_2^k	0	1	0	1
y	-1	-1	-1	1

$\rightsquigarrow$

	ξ^1	ξ^2	ξ^3	ξ^4
ξ_0^k	-1	-1	-1	1
ξ_1^k	0	0	-1	1
ξ_2^k	0	-1	0	1
y	-1	-1	-1	1

$$\begin{array}{lllll} \text{1.Stichprobe:} & w_0\xi_0^1 + w_1\xi_1^1 + w_2\xi_2^1 > 0 & \rightarrow & -w_0 > 0 & \rightarrow \quad \theta > 0 \\ \text{2.Stichprobe:} & w_0\xi_0^2 + w_1\xi_1^2 + w_2\xi_2^2 > 0 & \rightarrow & -w_0 - w_2 > 0 & \rightarrow \quad w_2 < \theta \\ \text{3.Stichprobe:} & w_0\xi_0^3 + w_1\xi_1^3 + w_2\xi_2^3 > 0 & \rightarrow & -w_0 - w_1 > 0 & \rightarrow \quad w_1 < \theta \\ \text{4.Stichprobe:} & w_0\xi_0^4 + w_1\xi_1^4 + w_2\xi_2^4 > 0 & \rightarrow & w_0 + w_1 + w_2 > 0 & \rightarrow \quad w_1 + w_2 > \theta \end{array}$$

In den Ungleichungen tritt nur „>“ auf, da alle Eingaben im positiven Halbraum liegen.

2.2.4 Alternatives Klassifikationsproblem im Gewichtsraum

Sei $\Omega = P \cup N$ n-dimensional und sei die Aktivierungsfunktion durch die Signumfunktion gegeben, d.h. $y \in \{-1, +1\}$. Sei $k \in \mathbb{N}$. Weiter sei der alternative Stichprobenraum formuliert durch: $\Omega' := P' \cup N'$ mit $P' := \left\{x^k | x^k \in P\right\}$ und $N' := \left\{-x^k | x^k \in N\right\}$.
Das Lernproblem lautet: Finde für die Stichprobenelemente $\xi^k \in \Omega'$ den Gewichtsvektor w, so dass $w^T \xi^k > 0$ gilt.

Für jedes Stichprobenelement ξ^k gilt aber auch wegen der Kommutativität des Skalarproduktes die gleichwertige Bedingung $\xi^{kT} w > 0$. Für die gesamte **Stichprobe** $z := (\xi^1, \xi^2, ..., \xi^{|\Omega'|})^T$ kann dies im Gewichtsraum geschrieben werden als

$$zw > \mathbf{0},$$

wobei $\mathbf{0}$ der $|\Omega'|$-dimensionale Nullvektor und z eine $((n+1) \times |\Omega'|)$-Matrix ist.

Im Beispiel $f_{\mathbf{AND}}$ erhält man ein System von Ungleichungen, welches den Durchschnitt von vier Halbräumen im Gewichtsraum repräsentiert:

$$\begin{bmatrix} -1 & 0 & 0 \\ -1 & 0 & -1 \\ -1 & -1 & 0 \\ 1 & 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix} > \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Die Problemdarstellung im Gewichtsraum ist beim Lernen von Bedeutung, da hier die Fehler bei der Suche nach den unbekannten Gewichten explizit berechnet und visualisiert werden können. Zwischen Eingaberaum und Gewichtsraum besteht eine Dualität. Der gesuchte Gewichtsvektor w (einschließlich Schwelle θ) ist ein Punkt im $(n+1)$-dimensionalen Gewichtsraum. Dieser entspricht im $(n+1)$-dimensionalen Eingaberaum einer n-dimensionalen Hyperebene. Wegen der Bedingungsgleichung

$$\langle w, x \rangle = \langle x, w \rangle = 0$$

gilt dies natürlich auch in umgekehrter Weise. Jeder erweiterte Eingabevektor x definiert im Gewichtsraum eine durch den Ursprung gehende Hyperebene, siehe Abbildung 2.23.
Aus den Ungleichungen

$$\langle w, x \rangle = \langle x, w \rangle > 0$$

folgt die Beziehung zwischen Punkten in den beiden dualen Räumen.
Im Gewichtsraum kann eine **Fehlerfunktion** $E(w)$ in folgender Weise definiert werden. Sei $e_{x^k}(w)$ eine binäre Funktion, die vom aktuellen Gewichtsvektor w abhängt und den Zuordnungsfehler von x^k anzeigt:

$$e_{x^k}(w) := \begin{cases} 1 & \text{wenn FALSE} \\ 0 & \text{wenn TRUE} \end{cases}$$

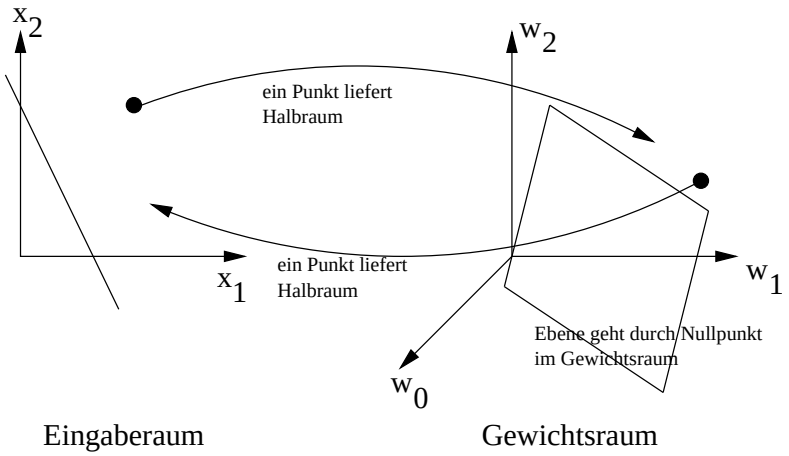


Abbildung 2.23: Dualität von Eingabe- und Gewichtsraum.

mit

$$\begin{array}{lllll} \text{FALSE} & :\Leftrightarrow & y = 0 \quad \text{für } x^k \in P & \text{oder } y = 1 & \text{für } x^k \in N \\ \text{TRUE} & :\Leftrightarrow & y = 1 \quad \text{für } x^k \in P & \text{oder } y = 0 & \text{für } x^k \in N \end{array}$$

Damit ist

$$E(w) = \sum_{k=1}^{|\Omega|} e_{x^k}(w)$$

eine Fehlerfunktion der Stichprobe Ω. Für diese den gesamten Gewichtsraum abdeckende Funktion gilt $E(w) \geq 0$.

Lernen ist die Suche nach dem **globalen Minimum** der Fehlerfunktion!

Ausgehend von einem zufälligen Gewichtsvektor w werden im Gewichtsraum hierzu bessere Lösungen gesucht.

Beispiel $f_{\mathbf{AND}}$:
Die Fehlerfunktion summiert für einen vorliegenden Gewichtsvektor $(w_0, w_1, w_2)^T$ die Anzahl fehlerhafter Resultate, wenn jedesmal die komplette Stichprobe durchgetestet wird.

f_{AND}: $w_0 = -\theta = -1$, $N := \{x^1, x^2, x^3\}$, $P := \{x^4\}$

$$\begin{array}{rcl} x^1 = (0,0)^T & : & -1 + w_1 0 + w_2 0 < 0 \\ x^2 = (0,1)^T & : & -1 + w_1 0 + w_2 1 < 0 \\ x^3 = (1,0)^T & : & -1 + w_1 1 + w_2 0 < 0 \\ x^4 = (1,1)^T & : & -1 + w_1 1 + w_2 1 \geq 0 \end{array}$$

In diesem Beispiel (siehe Abbildung 2.24) wird die Region mit minimalem Fehler ($E(w) = 0$) durch das Innere des Dreiecks $\{(0,1), (1,0), (1,1)\}$ des durch (w_1, w_2)

$w \setminus x$	x^1	x^2	x^3	x^4	$E(w)$
$(0,0)^T$	T	T	T	F	1
$(0,1)^T$	T	F	T	T	1
$(1,0)^T$	T	T	F	T	1
$(1,1)^T$	T	F	F	T	2
$(0.5,0.7)^T$	T	T	T	T	0

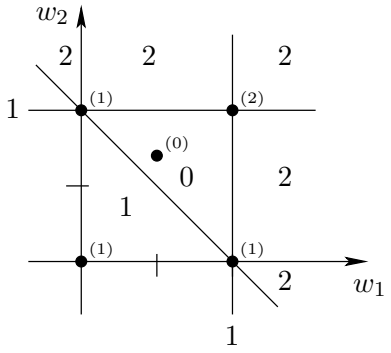


Abbildung 2.24: Veranschaulichte Verwendung einer Fehlerfunktion $E(w)$.

aufgespannten Gewichtsraumes (bei $w_0 = -1$) repräsentiert. Abbildung 2.25 zeigt, wie eine Veränderung der Schwelle w_0 zu einer anderen Lösungsregion führt.

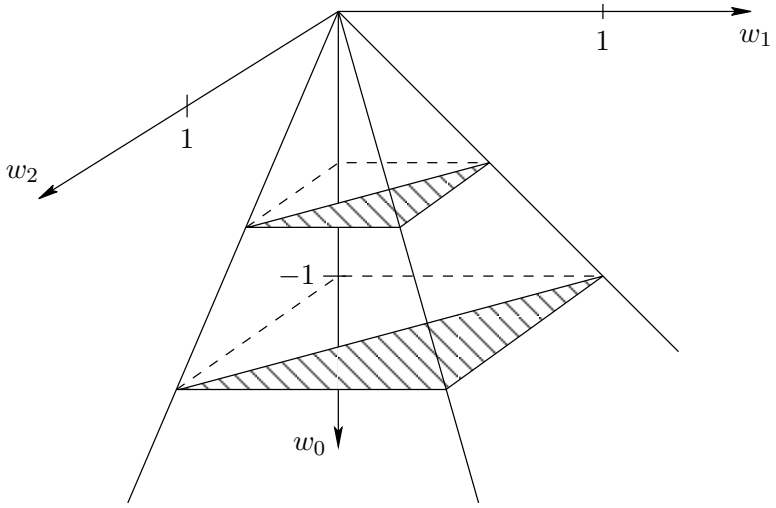


Abbildung 2.25: Fehlerfunktion mit Berücksichtigung des Schwellwertes w_0.

2.3 Perzeptron-Lernen

In diesem Abschnitt werden verschiedene Algorithmen zum Trainieren eines Perzeptrons vorgestellt. Dabei wird darauf verzichtet, explizit eine Fehlerfunktion auszuwerten. Es wird überwachtes Lernen im erweiterten Perzeptronmodell angenommen.
Sei Ω ein Stichprobenraum, $\Omega = P \cup N$. Die Menge der **indizierten Stichproben** wird durch die Paare (x^k, r^k), $\mu = 1, 2, ..., |\Omega|$, gebildet. Hierbei bezeichnet r^k die Zuordnung von x^k zu den Teilmengen P oder N. Also ist r^k die geforderte (requested) Antwort des Perzeptrons auf das Datum x^k. Die Stichprobenmenge Ω wird

aufgeteilt in eine **Trainingsmenge** Ω_L und eine **Testmenge** Ω_T, wobei in etwa $|\Omega_L| \sim 0.2 \cdot |\Omega_T|$ gelten sollte.

Hier soll nur das Training behandelt werden. Ziel des Tests ist die Überprüfung der Leistungsfähigkeit des Klassifikators an einer nicht bekannten Stichprobe. Ist diese nicht ausreichend, spricht man von schlechter **Generalisierung**. Dann muß die Zusammenstellung der Trainingsstichprobe Ω_L geändert und das Training wiederholt werden.

Annahme:
$\Omega_L = P_L \cup N_L$ Trainingsstichprobenmenge
(x^k, r^k), $k = 1, 2, ..., K = |\Omega_L|$ indizierte Stichprobe

Gesucht:
Gewichtsvektor w, der die Mengen P_L und N_L absolut linear trennt, d.h. $w^T x^k > 0$ für $x^k \in P_L$ und $w^T x^k < 0$ für $x^k \in N_L$.

2.3.1 Perzeptron-Lernregel nach Rosenblatt (1958)

Annahme:	$y := f(u) := \text{sgn}(u)$ Initialisierung mit zufälligem Gewichtsvektor $w(0)$.
Gesucht:	Gewichtsvektor $w(t)$ im Iterationsschritt t, der für alle $x^k \in \Omega_L$ keine Korrekturen mehr erfordert.

Folgende vier Fälle sind (bei absolut linearer Trennung) zu unterscheiden:

1. Wenn $x^k \in P_L$ und $y^k = r^k$, also $w^T x^k > 0$, dann OK

2. Wenn $x^k \in P_L$ und $y^k = -r^k$, also $w^T x^k \leq 0$, dann Korrektur

3. Wenn $x^k \in N_L$ und $y^k = r^k$, also $w^T x^k < 0$, dann OK

4. Wenn $x^k \in N_L$ und $y^k = -r^k$, also $w^T x^k \geq 0$, dann Korrektur

Algorithmus (ohne Abbruchbedingung):

start: wähle $w(0)$ zufällig
setze $t := 0$

test: wähle $x \in \Omega_L$ zufällig aber indiziert
if $x \in P$ and $y = r$ then goto test
if $x \in P$ and $y \neq r$ then goto add
if $x \in N$ and $y = r$ then goto test
if $x \in N$ and $y \neq r$ then goto sub

add: $w(t+1) := w(t) + x$
$t := t + 1$
goto test

sub: $w(t+1) := w(t) - x$
$t := t + 1$
goto test

Der Abbruch erfolgt, wenn für keines der $x^k \in \Omega_L$ mehr eine Korrektur erforderlich ist. Der Algorithmus ist ein **lokaler Greedy-Algorithmus**, d.h. für jedes Datum erfolgt unmittelbar die optimale Gewichtskorrektur (es wird ausschließlich für das aktuelle Datum optimiert):
Bei der Korrektur add wird der Gewichtsvektor in Richtung Eingabevektor gezogen, bei der Korrektur sub wird w von x weggezogen. Beide Korrekturschritte können zusammengefaßt werden:

$$w(t+1) := w(t) + ayx$$

mit

$$a = \begin{cases} -1 & \text{falls} \quad y \neq r \\ 0 & \text{sonst} \end{cases}$$

Die Einführung einer Lernkonstante $\gamma \in]0, 1]$ modifiziert die Konvergenzgeschwindigkeit des Algorithmus (siehe Abbildung 2.26 und 2.27):

$$w(t+1) := w(t) + \gamma ayx$$

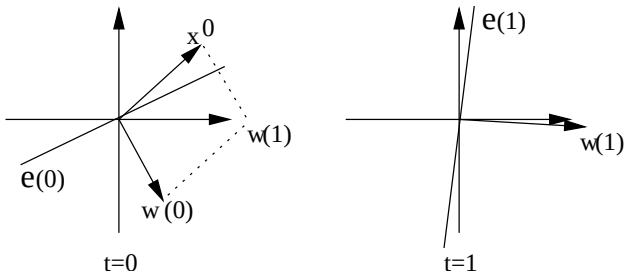


Abbildung 2.26: Beispiel eines Korrekturschrittes im Perzeptron-Lernalgorithmus.

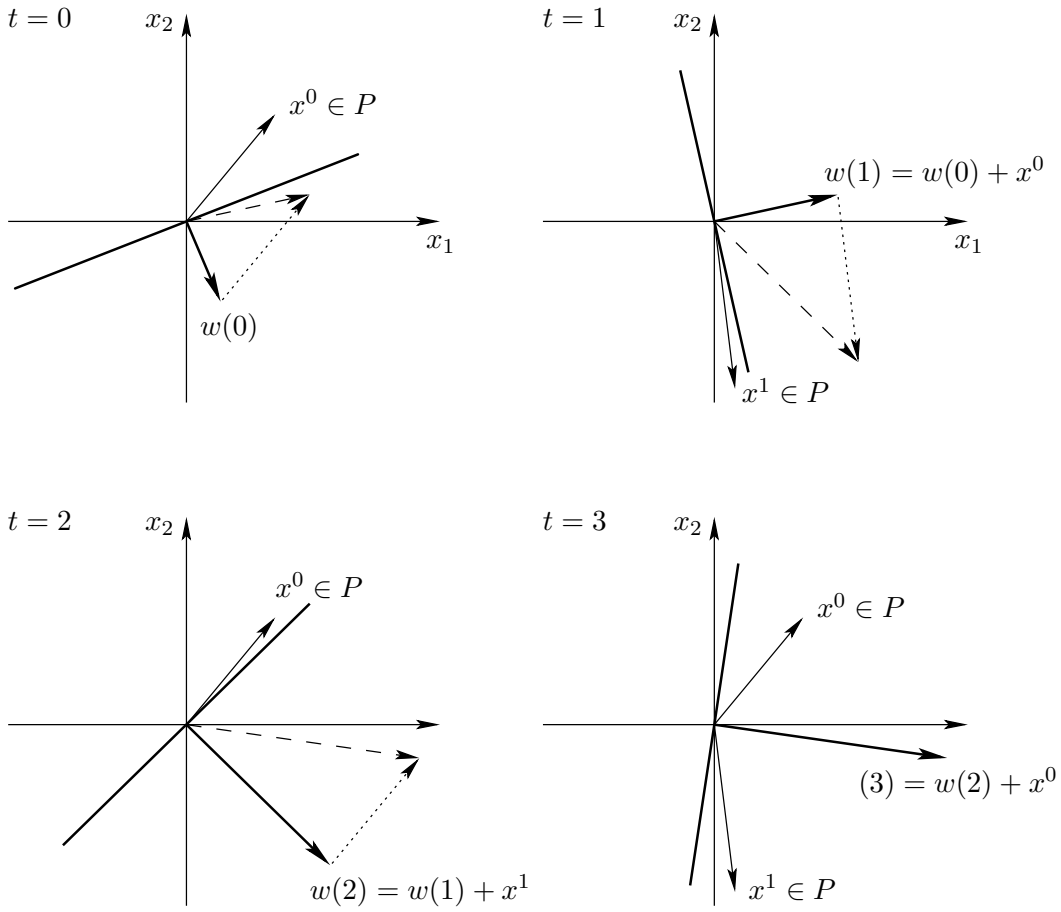


Abbildung 2.27: Beispiel für die Lernkonvergenz des Perzeptrons für $|\Omega| = |P| = 2$.

2.3.2 Konvergenz des Lernalgorithmus

Das Konvergenzverhalten des Algorithmus wird durch zwei Möglichkeiten wirksam modifiziert. Dies sind die geschickte Wahl des initialen Gewichtsvektors $w(0)$ und die Normierung der Vektoren.

Wahl des Startvektors $w(0)$:
Seien $p := |P_L|$, $n := |N_L|$, $i \in [1, p]$, $j \in [1, n]$, $x^i \in P_L$ und $x^{p+j} \in N_L$. Dann wird durch

$$w(0) := \frac{1}{p} \sum_{i=1}^{p} x^i$$

der Startvektor bereits in die richtige Richtung gezogen. Aber auch die Menge N_L kann einbezogen werden:

$$w(0) := \frac{1}{p} \sum_{i=1}^{p} x^i - \frac{1}{n} \sum_{j=1}^{n} x^{p+j}$$

Wie in Abbildung 2.28 veranschaulicht, ergibt sich in diesem Fall $w(0)$ als Differenzvektor der Schwerpunkte von P_L und N_L als $w(0) = w^P(0) - w^N(0)$.

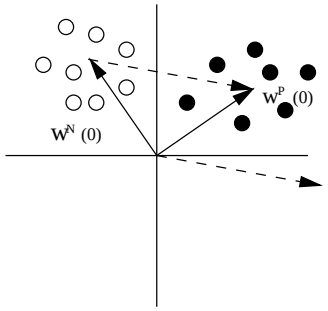


Abbildung 2.28: Differenzvektor der Schwerpunkte von P_L und N_L.

Normierung von Eingabe- und Gewichtsvektoren:
Die Gewichtskorrekturen werden durch die Länge der Eingabevektoren beeinflußt:

- Längere Eingabevektoren ziehen Gewichtsvektoren sehr stark an.
  Für $x(t) \in P$ und $\|x(t)\| \gg \|w(t)\|$ wäre $w(t+1) := w(t) + x(t) \approx x(t)$.
- Kleinere Eingabevektoren haben geringen Einfluß. Falls $\|x(t)\| \ll \|w(t)\|$ wäre $w(t+1) := w(t) + x(t) \approx w(t)$.

Außerdem steigt mit der Anzahl der Korrekturschritte die Länge des Gewichtsvektors. Die damit verbundenen Korrekturwinkel werden immer kleiner. Das heißt, die Lernrate geht gegen Null. Das entspricht einer impliziten Lernkonstante, die die Plastizität des Netzes mit der Zeit ändert, so dass $\gamma(t+1) < \gamma(t)$.

Deshalb werden beide Vektoren normiert verwendet.

Beweis der Konvergenz des Lernalgorithmus:
Falls eine absolute lineare Trennung des Stichprobenraumes Ω_L möglich ist, wird nach endlich vielen Schritten ein Lösungsvektor gefunden.

Widerspruchsannahme: Es tritt niemals der Fall ein, dass durch ein $w(t)$ alle Stichprobenelemente richtig klassifiziert werden.

Vereinbahrung: $\Omega' := P \cup N'$, $\xi^k \in \Omega'$, w^* Lösungsvektor, $\|\xi^k\| = \|w\| = \|w^*\| = 1$.

Adaption: $w(t+1) := w(t) + \xi$ falls ξ durch $w(t)$ nicht richtig klassifiziert wurde.

$$\begin{aligned} \langle w^*, w(t+1)\rangle &= \|w^*\|\|w(t+1)\| \cdot \cos\alpha \\ \Longrightarrow \cos\alpha &= \frac{w^{*T} w(t+1)}{\|w^*\|\|w(t+1)\|} \end{aligned} \tag{2.1}$$

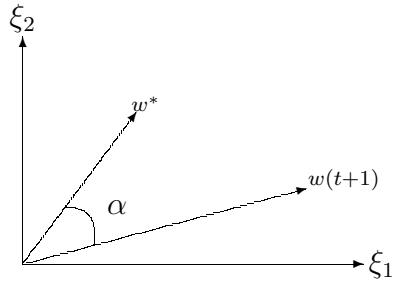


Abbildung 2.29: Beziehung zwischen w^* und $w(t+1)$.

Wir zeigen: Falls $w(t)$ unendlich oft adaptiert werden muß, und somit $t \to \infty$, dann geht auch (2.1) $\to \infty$. **Widerspruch!**

Betrachte den Zähler von (2.1):

$$\begin{aligned} w^{*T}w(t+1) &= w^{*T}(w(t)+\xi) \\ &= w^{*T}w(t) + w^{*T}\xi \\ &\geq w^{*T}w(t) + \delta \end{aligned}$$

mit $0 < \delta = \min\{w^{*T}\xi^k | \forall \xi^k \in \Omega'\}$ wegen absoluter linearer Trennbarkeit von P_L und N_L durch w^*.

Induktion liefert:

$$w^{*T}w(t+1) \geq w^{*T}w(0) + (t+1)\cdot\delta \tag{2.2}$$

Betrachte Nenner von (2.1):

$$\|w(t+1)\|^2 = (w(t)+\xi)^T(w(t)+\xi) = \|w(t)\|^2 + 2w^T(t)\xi + \|\xi\|^2$$

Es gilt $w^T(t)\xi \leq 0$, da andernfalls die Korrektur nicht nötig gewesen wäre. Somit folgt

$$\|w(t+1)\|^2 \leq \|w(t)\|^2 + \|\xi\|^2 \leq \|w(t)\|^2 + 1$$

wegen Normierung der Eingabedaten.

Induktion liefert:

$$\|w(t+1)\|^2 \leq \|w(0)\|^2 + (t+1) \tag{2.3}$$

Aus (2.2) und (2.3) folgt nun;

$$\cos\alpha \geq \frac{w^{*T}w(0) + (t+1)\delta}{\|w^*\|\sqrt{\|w(0)\|^2 + (t+1)}}$$

Für $t \to \infty$ gilt wegen $\delta > 0$ auch (2.1) $\to \infty$, also folgt der Widerspruch, denn da $\cos\alpha \leq 1$, muß t endlich sein.

2.3.3 Delta-Lernregel für beschleunigtes Lernen

Im Allgemeinen ist eine Adaption nicht fehlerkorrigierend, d.h. der vorliegende Fehler wird nicht vollständig behoben. Deshalb ist folgende alternative Lernregel möglich.

Annahme: Sei $x \in P$ (normiert) und $y = -1$, da $w^T(t)x < 0$ (absolute lineare Trennbarkeit). x wurde also falsch klassifiziert mit Fehler $\delta = -w^T(t)x$.

Somit lautet die neue (Delta-) Lernregel:

$$w(t+1) = w(t) + (\delta + \epsilon)x \quad \epsilon : \text{kleine pos. Zahl}$$

Die richtige Klassifizierung von x erfolgt aufgrund von:

$$\begin{aligned} w^T(t+1)x &= (w(t) + (\delta+\epsilon)x)^T x \\ &= w^T(t)x + (\delta+\epsilon)\underbrace{\|x\|^2}_{=1} \\ &= \delta - \delta + \epsilon \\ &= \epsilon > 0 \end{aligned}$$

ϵ garantiert, dass im Gewichtsraum $w(t+1)$ über eine Kante der Fehlerfunktion auf eine niedrigere Stufe absteigt.

Für $x \in N$ und $y = 1$ definiere $w(t+1) = w(t) + (\delta - \epsilon)x$.

Da bei der Delta-Lernregel der Fehler des Gewichtsvektors nicht nur reduziert, sondern behoben wird, handelt es sich um **Lernen mit Fehlerbehebung**.

2.4 Adaline als adaptives lineares System

Das Adaline (adaptive linear element, siehe Abbildung 2.30) wurde eingeführt durch Widrow und Hoff (1960) (α-LMS-Algorithmus) und Widrow (1962) (μ-LMS-Algorithmus). Weiterführende Literatur: Principe u. a. (2000); Widrow und Stearns (1985); Widrow und Winter (1988)
Prinzipiell ist das Adaline dem Perzeptron ähnlich, allerdings verhalten sich die Lernregeln anders.

Eigenschaften des Perzeptrons:

- binärer Output → lineares Entscheidungsproblem (lineare Trennfkt.)
- nichtlineare Lernregel, da binärer Output rückgekoppelt wird
- Richtung des Gewichtsvektors (nicht Betrag) bestimmt das Entscheidungsproblem

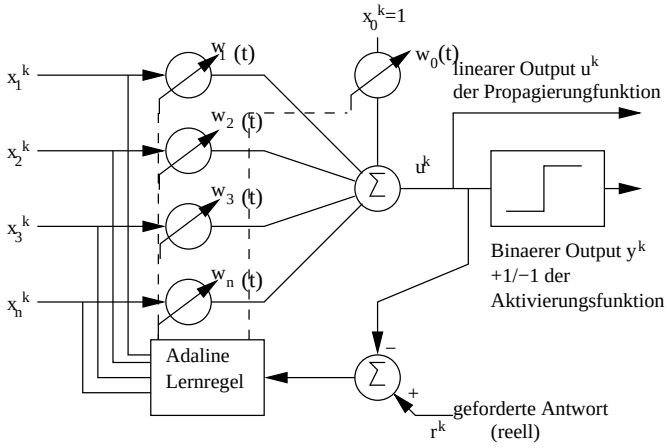


Abbildung 2.30: Erweitertes Adaline mit 4 Eingabeleitungen.

- **Offline-Lernen** möglich → Batchmode: alle Trainingsdaten müssen vorliegen → Trainingsphase + Arbeitsphase

Eigenschaften des Adaline:

- wenn binärer Output verwendet → lineares Entscheidungsproblem
- wenn reellwertiger linearer Output verwendet → **adaptives Filter**
- **Lineare Lernregel**, da Propagierungsfunktion rückgekoppelt wird
  → garantiert nicht Trennung eines linear trennbaren Problems
  → **Lernziel: MMSE-optimale Fehlerminimierung** der Propagierungsfunktion
- **Online-Lernen** möglich → Adaption

Wie bei allen Lernregeln wird auch beim Adaline das **Prinzip der minimalen Störung** realisiert: Das Erlernte soll bei der Korrektur nur minimal gestört werden.

Es muss über den zugrunde liegenden, Daten generierenden Prozess (DGP) in Ω eine der folgenden Annahmen gemacht werden:

a) Ω ist **statistisch stationär → optimales System**
Alle $K = |\Omega|$ Stichprobenelemente werden durch die gleichen Systemparameter (w, θ) repräsentiert. Möglichkeiten zur Suche der Systemparameter:

1. Lineare Regression über gesamten Stichprobenraum (Offline-Lernen bzw. Batchmode): r^k nicht erforderlich, da die Ordnung des Modells vorgegeben ist.
2. Online-Lernen der Systemparameter (stichprobenweise): r^k erforderlich.

b) Ω ist **statistisch instationär → adaptives System**
In verschiedenen Gebieten von Ω wirken unterschiedliche Systemparameter (stetige oder unstetige Übergänge möglich). Im Training durchwandert das System den Stichprobenraum, wobei die Ähnlichkeit (Korrelation) benachbarter Daten genutzt wird. Möglichkeiten zur Suche der Systemparameter: Online-Lernen

Es sind zwei Online-Lernregeln bekannt:

a) **Fehlerkorrekturlernen**: α-LMS-Algorithmus
Verändere die Gewichte so, dass der Fehler des Outputs für das präsentierte Stichprobenelement (x^k, r^k) minimal wird (ähnlich Perzeptron).

b) **Gradientenabstiegslernen**: μ-LMS-Algorithmus (steepest descent)
Verändere die Gewichte während jeder Präsentation einer Stichprobe so, dass der mittlere quadratische Fehler für den gesamten Stichprobenraum minimiert wird.

2.4.1 α-LMS-Algorithmus

Der Index k der zum Training verwendeten Stichprobe (x^k, r^k) wird nicht notiert. Es werden Eingabevektoren zugelassen. Der aktuelle Fehler im Iterationsschritt t sei:

$$e(t) = r - w^T(t)x = r - u(t),\ u(t) \in \mathbb{R}$$

Hieraus folgt die Korrektur der Gewichte (siehe Abbildung 2.31):

$$w(t+1) = w(t) + \alpha \frac{e(t)x}{\|x\|^2}$$

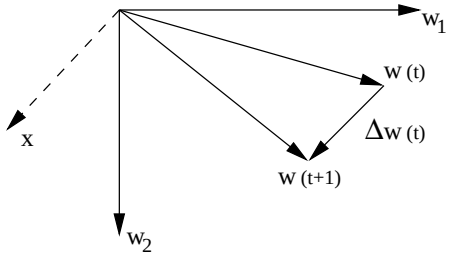


Abbildung 2.31: Beispiel einer Gewichtskorrektur im α-LMS-Algorithmus.

Der Korrekturterm $\Delta w(t) = w(t+1) - w(t) = \alpha \frac{e(t)x}{\|x\|^2}$ ist proportional $\|x\|^{-1}$. Die Eingabe wird deshalb normiert verwendet.

Der Korrekturschritt verändert den Fehler um:

$$\Delta e(t) = \Delta(r - w^T(t)x) = -\Delta w^T(t)x$$

Hieraus folgt wegen

$$\Delta e(t) = -\alpha \frac{e(t)x^T x}{\|x\|^2} = -\alpha e(t)$$

eine zum Fehler proportionale Korrektur desselben um den Faktor α (lineare Fehlerkorrektur).

Eigenschaften:

- Initialisierung des Gewichtsvektors als Nullvektor
- $\alpha \in [0.1, 1.0[$
- $\Delta w(t)$ parallel zu x: geringst mögliche Gewichtsänderung wird erzeugt (minimale Störung des Trainingserfolges der vorherigen Stichproben)
- $|\Delta w(t)| \sim \frac{1}{\|x\|}$

2.4.2 Offline-Lernen der Parameter eines linearen Systems

In der Folge wird das Adaline mit der aus der Numerik bekannten **linearen Regression** (vergleiche Abbildung 2.32) über einer Menge von Daten in Verbindung gebracht.

Seien $(x^k, u^k) \in \Omega$, $k \in \mathbb{N}$, Stichproben gegeben mit der erklärenden Variablen $x^k = (x_1^k, \ldots, x_n^k)^T \in \mathbb{R}^n$ und zu erklärender Variable $u^k \in \mathbb{R}$.
Annahme: Zwischen Output u^k und Input x^k bestehe ein **linearer Zusammenhang**:

$$u^k = wx^k + b$$

Einschränkend wird zudem die Annahme gemacht, dass nur die Fehler der Funktion u betrachtet werden (least square error).

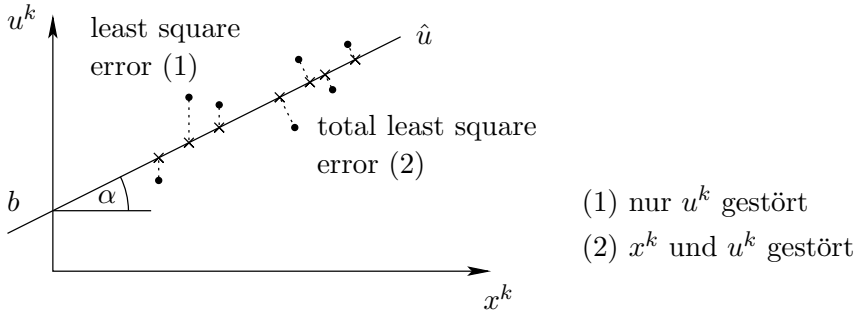


Abbildung 2.32: Veranschaulichung einer linearen Regression.

2.4.2.1 Lineare Regression über K Stichproben

Sei zunächst $x^k \in \mathbb{R}$.
MMSE-Verfahren: Gaußsches Fehlerminimierungsverfahren
Für alle $x^k \in \Omega$ gilt das Modell der gestörten Daten:

$$u^k(x^k) := wx^k + b + \delta_n\sqrt{\sigma_n^2} \quad \begin{array}{l}\text{mit } \delta_n \sim \text{unkorreliertes weißes Rauschen} \\ \text{mit } Var(\delta_n) = 1 \text{ und } E(\delta_n) = 0\end{array}$$

Somit ist $\delta_n\sqrt{\sigma_n^2}$ unkorreliertes weißes Rauschen mit Varianz $\sigma_n^2 \in \mathbb{R}^+$.

Gesucht werden die Regressionskoeffizienten, d.h. die optimalen Systemparameter $(\widehat{w}, \widehat{b})$. Diese werden nicht mit den tatsächlichen Parametern (w, b) übereinstimmen, sondern diese approxmieren. Es wird also die Funktion $\hat{u}(\widehat{w}, \widehat{b}) \equiv \hat{u}$ mit minimalem mittlerem quadratischem Fehler (MMSE) $E_{\min}$ gesucht.

Bemerkung: In der Folge werden Schätzwerte oder geschätzte Funktionen immer mit einem Dach versehen notiert.

$\hat{u}^k := \hat{w}x^k + \hat{b}$	$\sim$	Regressionsgerade/Regressions-Hyperebene
$E := \frac{1}{K}\sum_{k=1}^{K}(e^k)^2$	$\sim$	mittlerer quadrat. Fehler (MSE) mit $e^k := u^k - \hat{u}^k$
$E_{\min} := \min\{E\}$	$\sim$	MMSE-Kriterium

E ist eine konvexe Funktion, somit genügt als Minimierungansatz die Betrachtung der **notwendigen Bedingung erster Ordnung (FOC):**

Einsetzen des Modells $\hat{u}^k$ in E und Ableiten nach den Modellparametern $(\hat{w}, \hat{b})$:

$$\begin{aligned}\frac{\partial\frac{1}{K}\sum_{k=1}^{K}(u^k - \hat{u}^k)^2}{\partial\hat{b}} &\stackrel{!}{=} 0 \quad \Rightarrow \quad \sum_{k=1}^{K}(u^k - \hat{w}x^k - \hat{b}) = 0 \\ \frac{\partial\frac{1}{K}\sum_{k=1}^{K}(u^k - \hat{u}^k)^2}{\partial\hat{w}} &\stackrel{!}{=} 0 \quad \Rightarrow \quad \sum_{k=1}^{K}(u^k - \hat{w}x^k - \hat{b})x^k = 0\end{aligned}$$

Normalengleichungen:

$$\begin{aligned}\sum_k u^k &= K\hat{b} + \hat{w}\sum_k x^k \\ \sum_k x^k u^k &= \hat{b}\sum_k x^k + \hat{w}\sum_k (x^k)^2\end{aligned}$$

Regressionskoeffizienten der Geraden $\hat{u}$:

$$\hat{w} = \frac{K\sum_k x^k u^k - \sum_k x^k \sum_k u^k}{K\sum_k (x^k)^2 - \left(\sum_k x^k\right)^2}$$

$$\hat{b} = \frac{K\sum_k \left(x^k\right)^2 \sum_k u^k - \sum_k x^k \sum_k x^k u^k}{K\sum_k \left(x^k\right)^2 - \left(\sum_k x^k\right)^2}$$

Die geschätzte Gerade $\widehat{u}$ verläuft durch den Punkt $(\bar{x}, \bar{u})$ mit $\bar{x} = \frac{1}{K}\sum_{k=1}^{K} x^k$ und $\bar{u} = \frac{1}{K}\sum_{k=1}^{K} u^k$. Deshalb lassen sich die Regressionskoeffizienten auch angeben als:

$$\hat{w} = \frac{\sum_k (x^k - \bar{x})(u^k - \bar{u})}{K\sum_k (x^k - \bar{x})^2}$$

$$\hat{b} = \frac{\sum_k \left(x^k\right)^2 \sum_k u^k - \sum_k x^k \sum_k x^k u^k}{K\sum_k (x^k - \bar{x})^2}$$

Frage: Wie gut ist die Schätzung der Regressionsgeraden bzw. wie gut spiegeln die Daten ein lineares Modell wieder?
Antwort: Die Berechnung des Korrelationskoeffizienten zwischen Eingabe und Ausgabe liefert hierfür Hinweise.

Korrelationskoeffizient (Die überstrichenen Größen $\overline{x}, \overline{u}$ sind die Mittelwerte.):

$$\rho := \frac{\mathrm{Cov}(u,x)}{\sqrt{\mathrm{Var}(x)\mathrm{Var}(u)}} = \frac{\frac{1}{K}\sum_k (x^k - \overline{x})(u^k - \overline{u})}{\sqrt{\frac{1}{K}\sum_k (u^k - \overline{u})^2}\sqrt{\frac{1}{K}\sum_k (x^k - \overline{x})^2}} \in [-1, 1]$$

$\rho \in \{-1, +1\}$: perfekter linearer Zusammenhang zwischen x und u
$\rho = 0$: Daten unkorreliert

- Leastsquare-Schätzung erlaubt, Output für unbekannte (neue) Daten $x \in \Omega$ zu schätzen (interpoliert und extrapoliert).
- Ist **bester linearer Schätzer** ohne Bias (hat minimale Varianz unter allen linearen Schätzern und ist ohne systematischen Fehler, BLUE).
- Kann für polynominale Kurven höherer Ordnung verallgemeinert werden (verallg. Leastsquare-Schätzer): **nichtlineare Regressionsmodelle**
- Gültig auch für vektorielle Variable → Regressionslinie wird **Hyperebene**

2.4.2.2 Multivariate Regression

Sei $n \in \mathbb{N}$. Es werden nun vektorwertige bzw. multidimensionale Eingaben $x^k \in \mathbb{R}^n$ betrachtet (die Komponenten seien unabhängige Variable).

Beispiel für $n = 2$:
Anpassung einer Ebene (siehe Abbildung 2.33):

$$u = b + w_1 x_1 + w_2 x_2 = w_0 1 + w_1 x_1 + w_2 x_2$$

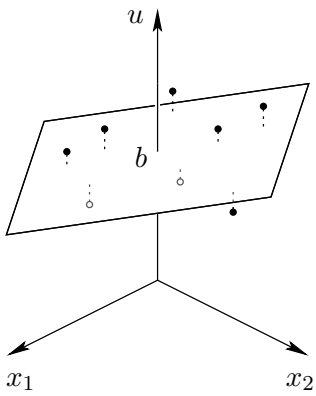


Abbildung 2.33: Veranschaulichung einer multivariaten Regression.

Um die Ähnlichkeit zum Modell des Adaline herzustellen, wird der Bias b in den Gewichtsvektor w aufgenommen. Es gilt also $w = (w_0, \ldots, w_n)^T \in \mathbb{R}^{n+1}$.

Die Einzelfehler e^k und der Stichprobenfehler E sind (vergleiche Abschnitt 2.4.2.1):

$$\begin{aligned} e^k &= u^k - \hat{u}^k = u^k - \sum_{i=0}^{n} \hat{w}_i x_i^k \\ E &= \frac{1}{K} \sum_{k=1}^{K} (e^k)^2 \end{aligned}$$

Da E eine konvexe Funktion ist, genügt das Nullsetzen der ersten Ableitung, um die Bedingung für das (eindeutige) Minimum von E zu erhalten:

$$\frac{\partial E}{\partial \hat{w}_i} \stackrel{!}{=} 0 \quad \Rightarrow \quad \sum_k x_i^k (u^k - \sum_i \hat{w}_i x_i^k) = 0$$

Hieraus folgen die $n + 1$ sogenannten **Normalengleichungen**

$$\sum_k x_i^k u^k = \sum_i \hat{w}_i \sum_k x_i^k x_j^k \qquad j = 0, 1, ..., n \ ,$$

die linear in der Unbekannten $\hat{w}$ sind, womit sie sich durch Matrixinversion oder besser mittels Cholesky- oder QR-Verfahren lösen lassen.

Zur kompakteren Schreibweise der Normalengleichungen wird weiter definiert:

Autokorrelationsmatrix der Inputdaten:

$$R = (R_{ij}) \text{ mit dem Autokorrelationskoeffizienten } R_{ij} = \frac{1}{K} \sum_k x_i^k x_j^k$$

Kreuzkorrelationsvektor von Input und Output:

$$p = (p_0, ..., p_n)^T \text{ mit } p_j = \frac{1}{K} \sum_k x_j^k u^k$$

Somit können die Normalengleichungen notiert werden als:

$$p = R\hat{w}$$

Die optimale Lösung

$$\hat{w} = R^{-1}p$$

mit minimalem Anpassungsfehler $E_{\min}$ wird als **Wiener-Lösung** bezeichnet.

Die **Lösung des multivariaten Regressionsproblems** ist demnach das Produkt von inverser AK-Matrix und KK-Vektor und lässt sich wie folgt interpretieren: Der Kreuzkorrelationsvektor modelliert die Beziehung zwischen Eingabe und Ausgabe, normiert auf die Struktur in den Eingabedaten (Autokorrelationsmatrix).

2.4.3 Lernen der Parameter eines linearen Systems

Betrachtet wird nun das **Lernen** der Modellparameter aus einzelnen Stichprobendaten. Dieses Paradigma ist anwendbar für Online- und Offline-Lernen. Im Fall des Offline-Lernens wird das Modell iterativ über den gesamten Stichprobenraum Ω verfeinert. Die Fehlerkorrektur wird erst nach dem Durchlaufen der gesamten Stichprobenmenge vorgenommen. Das ist typisch für **Batchmode-Lernen**. Beim Online-Lernen würde jede einzelne Stichprobe zu einer Korrektur der Systemparameter führen, wie in Abbildung 2.34 dargestellt. Damit kann sich das System an eine Drift der Daten anpassen.

Grundlage: Separierung eines Teilsystems, das die aktuelle Systemperformance auswertet und die Systemparameter korrigiert. Ein solches System wird auch als **lineares System mit Fehlerrückkopplung** (error feedback) bezeichnet.

Es wird zunächst das Offline-Lernen bzw. das Batchmode-Lernen behandelt.
Voraussetzung: Die geforderten Antworten r^k sind bekannt; (x^k, r^k) verfügbar.

Dieses System realisiert **überwachtes Lernen**:

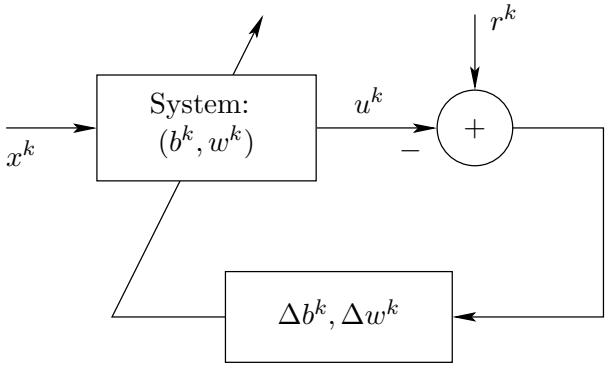


Abbildung 2.34: Lineares System mit Fehlerkückkopplung.

- Lernregel ist bekannt.
- Die Stichprobenelemente sind indiziert, enthalten also auch die Bedeutungszuordnung der Daten. Sie werden zufällig gezogen. Es kann also aus der Reihenfolge der Daten keine Information gelernt werden.

Fehlerraum der Parameter:
Annahmen:

- mehrdimensionales Datem $x^k \in \mathbb{R}^(n+1)$
- erweitertes Modell des Perzeptrons/Adalines $\rightarrow u^k = w^T x^k$

Jedes einzelne Stichprobendatum $(x^k, r^k) \in \Omega$ bewirkt den Stichprobenfehler

$$e^k = r^k - w^T x^k \ .$$

Der Gesamtfehler (das konvexe Fehlerfunktional) für Ω lautet

$$E = \frac{1}{2K} \sum_k (r^k - w^T x^k)^2 = \frac{1}{2K} \sum_k \left((w^T x^k)^2 - 2 r^k w^T x^k + (r^k)^2 \right) \geq 0$$

und ist quadratisch in w. Der Vorfaktor $\frac{1}{2}$ wird aus Bequemlichkeit eingeführt.

Die Verbesserung der Systemparameter wird dann aus dessen Gradienten

$$\nabla E = \left(\frac{\partial E}{\partial w_0}, \ldots, \frac{\partial E}{\partial w_n} \right)^T$$

berechnet.

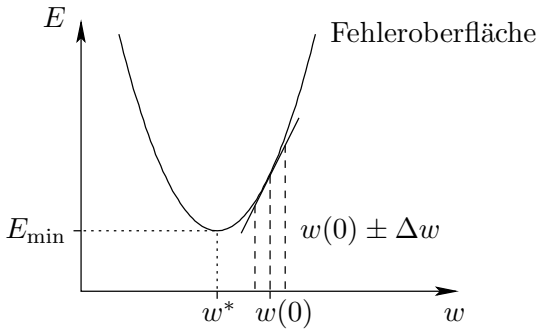


Abbildung 2.35: Beispiel einer konvexen Fehlerfunktion.

Hierfür existiert immer ein globales Minimum (siehe Abbildung 2.35):

$$E_{\min} = E(w^*) = E(\nabla E = 0)$$

Aus Gründen der Anschaulichkeit wird nun auf eindimensionale Daten $x^k \in \mathbb{R}$ zurückgegriffen. Entsprechend gelte $w \in \mathbb{R}$ und $w_0 = 0$. Dies bedeutet $\bar{x} = \bar{r} = 0$. Die Suche nach dem Minimum liefert für E jetzt:

$$\nabla E = 0 : \frac{1}{K}\left(-\sum_k r^k x^k + w\sum_k (x^k)^2\right) = 0$$

Der optimale Gewichtsvektor w^* ergibt sich somit zu:

$$w^* = \frac{\sum_k x^k r^k}{\sum_k (x^k)^2}$$

Hieraus folgt für den optimalen Gewichtsvektor das Minimum der Fehlerfunktionals:

$$E_{\min} = \frac{1}{2K}\sum_k \left(r^k - \frac{\sum_k r^k (x^k)^2}{\sum_k (x^k)^2}\right)^2$$

Und für die Gestalt der Fehleroberfläche folgt:

$$E = E_{\min} + \frac{1}{2K}(w - w^*)\sum_k (x^k)^2 (w - w^*)$$

Merke:

- Sowohl der minimale Fehler $E_{\min}$ als auch der optimale Gewichtsvektor w^* hängen vom Input und gefordertem Output ab.
- Die Form der Fehleroberfläche ist nur vom Input abhängig, ist also lediglich eine Funktion der Struktur der Daten (d.h. der Autokorrelationsmatrix).

- Somit können die optimalen Systemparameter im Vektor w anstatt mit Hilfe der analytischen Regression auch mit einer Suche auf der Fehlerfunktion bestimmt werden.

Suche nach dem Prinzip des steilsten Abstiegs:

- Möglich, da **konvexe Fehlerfunktion** mit globalem Minimum
- Gradient kann lokal berechnet werden (für gegebenes Datum (x^k, r^k))
- Gradient weist immer in Richtung max. Änderung (vergleiche Abbildung 2.36)

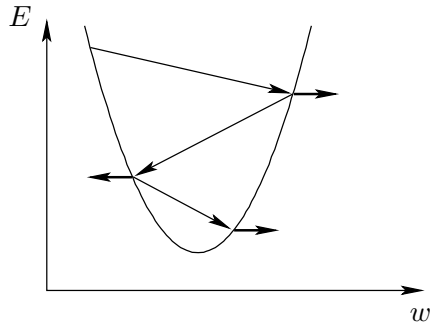


Abbildung 2.36: Darstellung eines Gradientenabstiegs.

- Iteration für endliches oder unbegrenztes K

Methode des steilsten Gradientenabstiegs:
Starte mit willkürlichem initialem Gewicht $w(0)$. In den Klammern steht die laufende Iterationsschrittnummer. Berechne das Fehlerfunktional für $w(0)$ und modifiziere die Gewichte proportional zum negativen Gradienten. Dadurch wandert das System an die Position $w(1)$ auf der Fehlerkurve. Allgemein gilt $w(t+1) = w(t) - \mu \cdot \nabla E(t)$. Dabei ist μ eine Konstante, die sicher stellt, dass die Suche stabil konvergiert.

- Greedy-Algorithmus (lokale Berechnung)
- Steepest-Descent-Verfahren

2.4.4 μ-LMS-Algorithmus

Die Methode des Gradientsabstiegs kann auch für Online-Lernen eingesetzt werden. Dieses Verfahren heißt μ-LMS-Algorithmus.

Problem: Wie soll der unbekannte Gradient geschätzt werden?

Vorschlag: Verwende den momentanen Gradienten als Schätzung des wahren Gradienten, vergleiche Widrow und Hoff (1960).

Das heißt, anders als im Batchmode-Verfahren wird keine Schätzung über einem Ensemble von Daten (der gesamten Stichprobe) durchgeführt. Vielmehr erfolgt der nächste Iterationsschritt mit dem nächsten Datum. Das bedeutet, dass ein solches System einige **Einschwingzeit** benötigt, bis es die Struktur der Daten erlernt hat (Autokorrelation, Kreuzkorrelation).
Er kann anstelle des Zeitparameters t auch der Stichprobenindex k gesetzt werden.

$$\nabla E^k = \frac{\partial}{\partial w^k} E^k \approx \frac{1}{2} \frac{\partial}{\partial w^k} \left(e^k\right)^2 = -e^k x^k$$

- Es werden nur lokale Fehler begangen. Diese werden im Verlaufe der Anpassung an die Struktur der Daten herausgefiltert.
- μ-LMS-Algorithmus:
$$w^{k+1} = w^k + \mu e^k x^k$$

Wiener-Lösung:

$$\begin{aligned} \nabla E^k = -e^k x^k &= -(r^k - w^k x^k) x^k \\ &= -r^k x^k + w^k (x^k)^2 \quad (= -p - R w^k) \end{aligned}$$

mit $p \sim$ Kreuzkorrelationsvektor
und $R \sim$ Autokorrelationsmatrix

Im Optimum gilt somit: $p = Rw^* \Rightarrow w^* = R^{-1} p$
Dies gilt streng genommen nur, wenn eine Erwartungswertschätzung $E\left\{(e^k)^2\right\}$ über alle Stichproben (x^k, r^k) erfolgt.
Hier: Annahme der Verwendung eines Ensembles von identischen adaptiven Linearkombinationen. Daraus folgt ein subobtimales Verhalten, aber auch Effizienz.

2.4.5 Bewertung der vorgestellten Verfahren

Es wurden sowohl die analytische Lösung (Regression) als auch die iterative Lösung (Lernen) dargestellt. Für das Lernen wurde ferner die Iteration über den einzelnen Stichprobenelementen oder über allen Daten des Stichprobenraums betrachtet.

2.4.5.1 Offline-Training versus Online-Training

Das oben beschriebene Vorgehen für Online-Lernen hat zur Folge, dass der Weg zum Minimum um die **Gradientenrichtung** gestört ist (Zickzack), denn die Iteration des Trainings erfolgt über den einzelnen Stichprobenelementen.

Eine Alternative wäre: Berechne die Gewichtskorrekturen für jede Probe und speichere diese ohne sie auszuführen während eines Passes durch die Trainingsmenge (Epoche). Die Iteration des Trainings erfolgt in diesem Fall über den Epochen.
Am Ende jeder Epoche: Führe eine Gewichtskorrektur mit der Summe der Korrekturen aus. Das entspricht für einen Iterationsschritt dem Batchmode-Training. Man erhält einen glatteren Schätzer für den Gradienten. Für Anwendungen, in denen Online-Lernen von Interesse ist, verbietet sich aber gewöhnlich dieses Vorgehen.

Für beide Verfahren gelten die folgenden Aussagen zur Robustheit und Stabilität:

Robustheit
Unabhängig von den Startwerten $w(0)$ konvergiert der Algorithmus zum selben Wert w^*. Der Grund hierfür liegt in der konvexen Form der Fehlerfunktion, das nur von der Struktur der Daten abhängt, aber nicht von der Art des Trainings. Das System **extrapoliert** (**interpoliert)** Antworten für bisher im Training nicht enthaltene Daten (z.B. der Teststichprobe), aber die Performance ist etwas schlechter.

Stabilität
Einziger freier Parameter für beide Verfahren ist die **Schrittweite** μ, $\rightarrow$ **Lernkurve** siehe Abbildung 2.37.

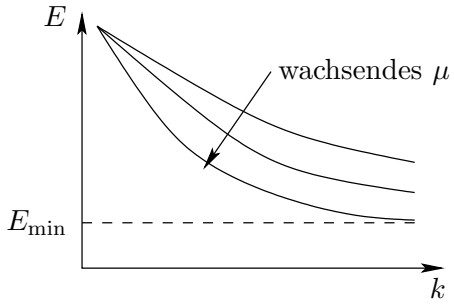


Abbildung 2.37: Lernkurven in Abhängigkeit der Schrittweite.

- wenn $w^k \rightarrow w^*$, dann $E^k \rightarrow E_{\min}$
- wenn μ zu groß, kann der iterative Prozeß divergieren

Gewichtskurve:
Kann man aus den Struktureigenschaften der Daten einen optimalen Konvergenzparameter ableiten?

- geeignete Anpassung der Konvergenz

$$w^{k+1} := w^* + (1 - \mu\lambda)^{k+1}(w^0 - w^*)$$

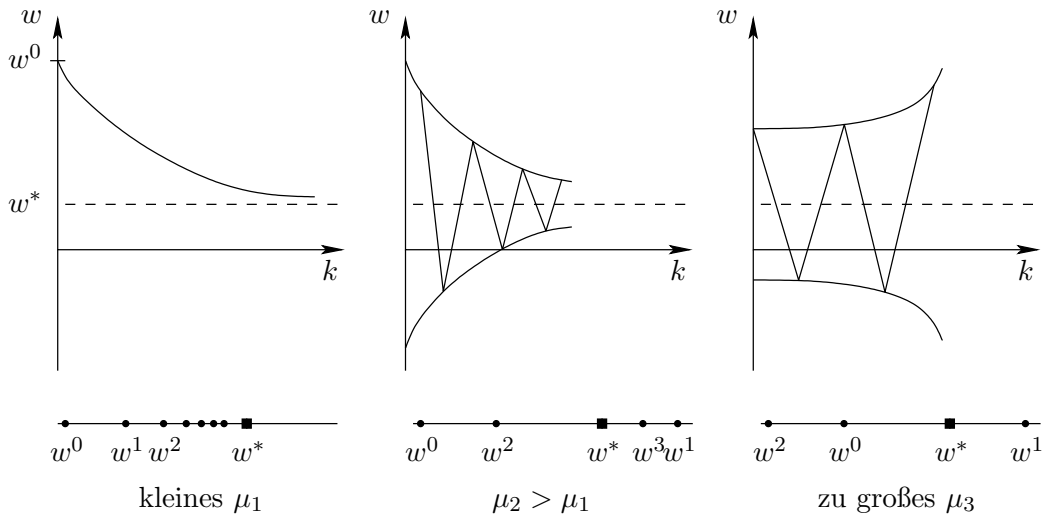


Abbildung 2.38: Gewichtskurven in Abhängigkeit der Schrittweite.

mit

$$\lambda = \frac{1}{K} \sum_k (x^k)^2$$

konvergiert, wenn $(1 - \mu\lambda)^{k+1} \leq 1$ gilt und Konvergenz zu w^*, wenn $(1 - \mu\lambda)^{k+1} < 1$ gilt (vergleiche Abbildung 2.38).
Dies entspricht der Wahl einer Lernkonstante $\mu < \mu_{\max} = \frac{2}{\lambda}$.

- Zeitkonstante der Adaption:
  Annahme: eine Zeiteinheit entspricht einer Epoche oder einer Stichprobe
  Exponentielle Annäherung von w^k an w^* mit $\exp\left(-\frac{t}{\tau}\right)$, $\tau = \frac{1}{\mu\lambda}$ $\rightarrow$ schnelle Adaption $\sim$ kleine Zeitkonstante $\sim$ große Schrittweite

2.4.5.2 Analytische Lösung versus iterative Lösung

Obwohl die analytische Lösung und die iterative Lösung äquivalente Verfahren sind, ist in lernenden Systemen oft die **iterative Lösung zu bevorzugen**:

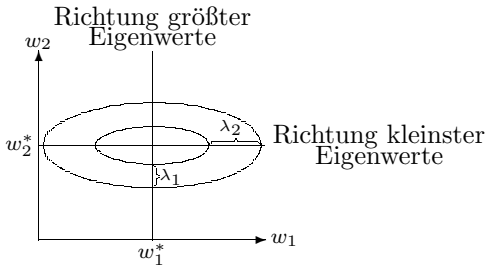
- Die analytische Lösung benötigt den gesamten(!) Datensatz für die Berechnung von **Autokorrelationsmatrix** R und **Kreuzkorrelationsvektor** p.
- Die Invertierung von R ist rechenaufwendig (quadratisch in der Anzahl der Parameter) und existiert u.U. nicht.
  ($\rightarrow$ **Pseudoinverse**, siehe Abschnitt 4.2)
  Wenn R **schlecht konditioniert** ist, wird R^{-1} u.U. sehr ungenau!
- Der LMS-Algorithmus ist sehr effizient (linear in der Anzahl der Gewichte).
- Die analytische Methode kann nur für (praktisch wenig relevante) nichtlineare Probleme erweitert werden (z.B. rein quadratische Probleme). Die iterative Methode kann sich prinzipiell an alle nichtlinearen Funktionen anpassen.

Die iterative Lösung kann aber auch einige Probleme aufwerfen:

- Es besteht keine Garantie, dass w gegen w^* konvergiert (hängt von Daten und Schrittweite ab).
- Die Matrizen R mit großem **Spektrum** (Bandbreite der Eigenwertverteilung) verursachen eine **langsame Konvergenz**, da die Fehlerkurve aus Abbildung 2.35 sehr flach ist.

$$E(w) = E_{\min} + \frac{1}{2}(w - w^*)^T R(w - w^*)$$

  Die Fehlerkurve ist ein Paraboloid mit elliptischem Querschnitt, siehe auch Abbildung 2.39.
  Die **Eigenvektoren** von R bestimmen die Hauptachsen von $E(w)$. Die **Eigenwerte** von R bestimmen die Gradienten von $E(w)$ entlang der Hauptachsen. Hieraus ergibt sich, dass die kleinsten Eigenwerte die Konvergenzgeschwindigkeit und die größten Eigenwerte die Schrittweite der Korrekturen nach oben begrenzen.



größter Eigenwert: $\lambda_1 = \frac{1}{\nabla w_2}$
kleinster Eigenwert: $\lambda_2 = \frac{1}{\nabla w_1}$

Abbildung 2.39: Veranschaulichung des Querschnitts einer Fehlerkurve.

2.5 Adaptive Signalverarbeitung mittels Adaline

Ein Adaline (siehe Abbildung 2.40) kann als digitaler Filter (Transversalfilter) interpretiert werden (siehe Abbildung 2.41).
Es ist vom Typ **IIR-Filter** (infinite impulse response), d.h. ein **rekursiver Filter**.

- Der gefilterte Output ist eine Linearkombination des momentanen Signals x^k und der vorhergehenden Signale $x^{k-1}, ..., x^{k-n}$.
- Die Impulsantwort wird durch die Adaption der Gewichte variiert, so dass das Output-Signal an das geforderte (aktuelle) Signal im Sinne des MMSE

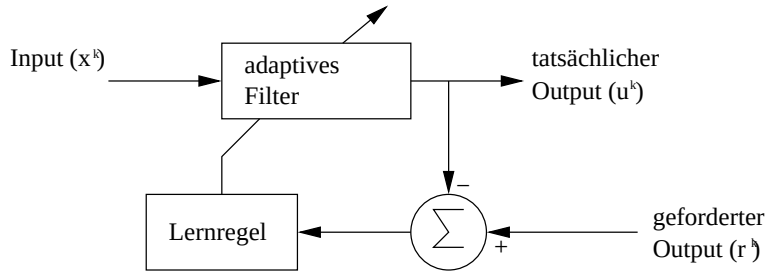


Abbildung 2.40: Darstellung eines Adaline.

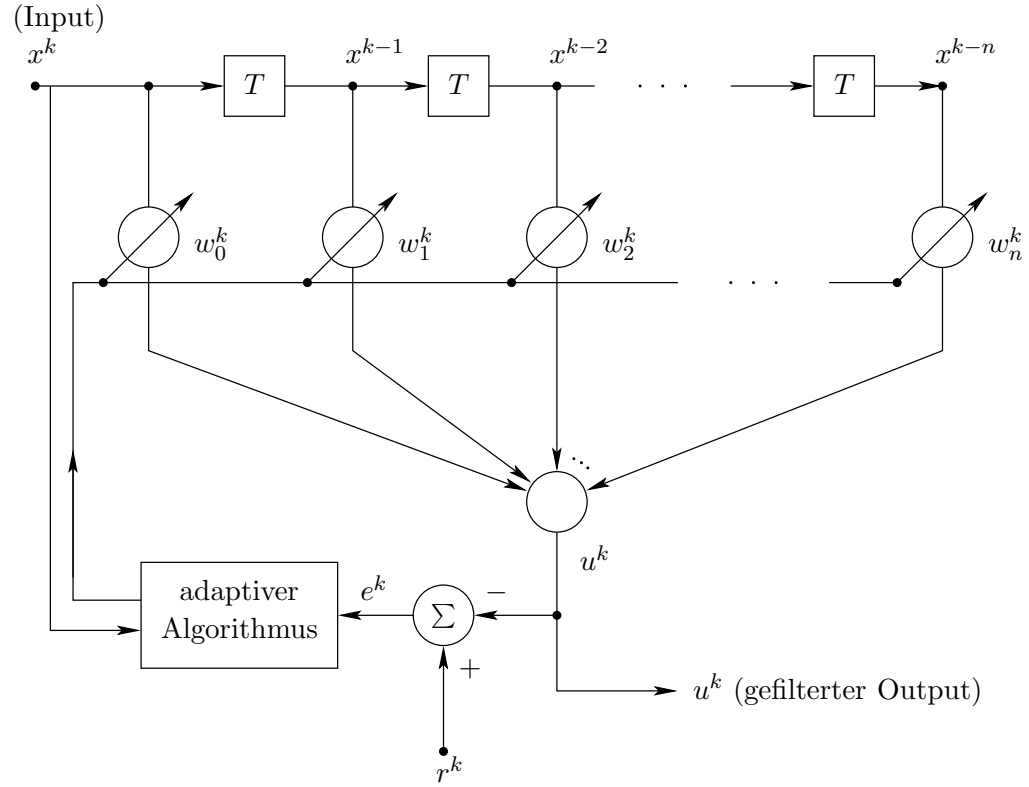


Abbildung 2.41: Adaline als digitaler Filter.

angepaßt wird.
Achtung: Bei IIR-Filtern sind die Koeffizienten der Inputsantwort nicht die Koeffizienten der Linearkombination (Gewichte)!

2.5.1 System-Identifikation

Annahme: Ein System unbekannter Struktur habe eine beobachtbare Input-Output-Relation.

Aufgabe: Suche einer geeigneten Systembeschreibung durch eine Menge von

Parametern.

Lösung: Modellierung des Systems (siehe Abbildung 2.42), so dass

- gemeinsamer Input für unbekanntes System und Adaline
- gefordeter Output des Systemmodells = tatsächlicher Output des unbekannten Systems

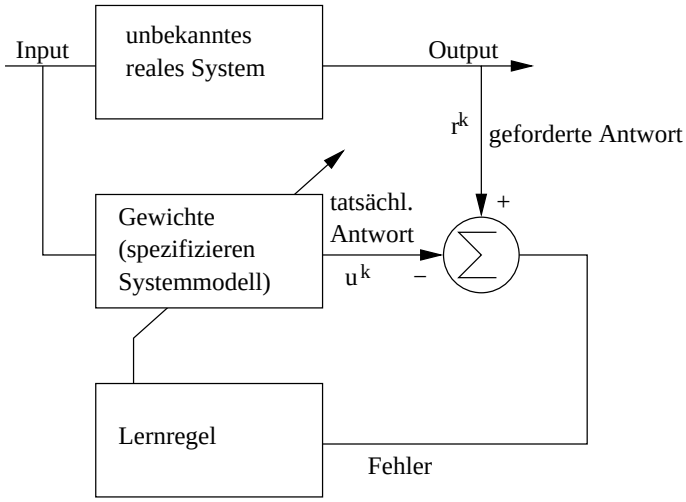


Abbildung 2.42: Adaline im Zusammenspiel mit einen unbekannten realen System.

- Ergebnis: Die Filterparameter (Gewichte) beschreiben das Systemmodell

2.5.2 Prädiktion

Aufgabe: Schätzung künftiger Signalwerte aus gegenwärtigen und zurückliegenden Signalwerten.

Wiener-Filter: Bei bekannter Autokorrelation des Signals (und des Rauschens) ist eine optimale Schätzung möglich.

Adaline: Autokorrelation wird gelernt.
Annahme: Schwach stationäres Signal (Mittelwert und Kovarianz bzw. Korrelation beschreiben das Signal vollständig.)
Lösung: Aus einer n Schritte zurückliegenden Abbildung $I \rightarrow O$ wird ein Systemmodell gelernt, das die aktuelle Abbildung $I \rightarrow O$ zu berechnen gestattet:

1. Nichtverzögertes Signal dient als gefordertes Signal.
2. Verzögerter Input wird zur Gewichtsadaption im Sklavenfilter verwendet.

3. Kopie der optimierten Gewichte wird im adaptiven Filter zur Prädiktion verwendet.

Abbildung 2.43 zeigt einen Adalineprädiktor, während Abbildung 2.44 das Ergebnis einer solchen Vorhersage präsentiert. In Abbildung 2.45 ist die Prädiktion eines Lorenz-Attraktors mittels anderer Neuronentypen dargestellt.

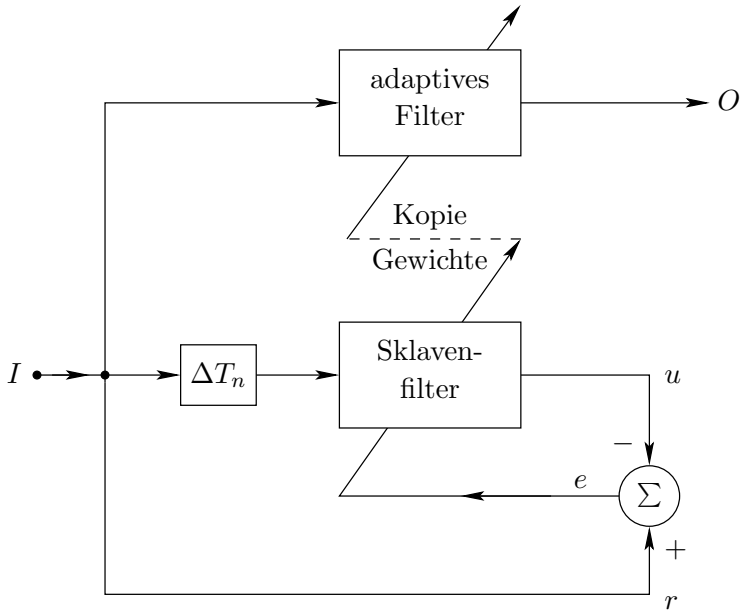


Abbildung 2.43: Vorhersagefilter mittels Adaline.

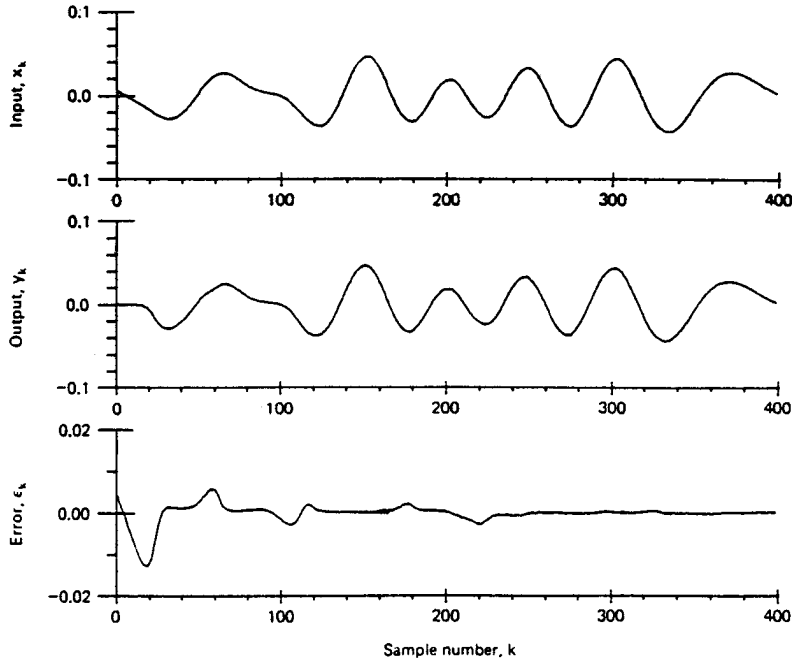


Figure 1.7 Signals in the adaptive predictor in Figure 1.6. The error (ϵ) becomes nearly zero as the system learns to predict the input (x) and the output (y) approaches x.

Abbildung 2.44: Beispiel einer Adalineprädiktion.

2.5.3 Rauschunterdrückung (noise cancelling)

Annahme: Additives Rauschen $\sim f = s + n$

Ziel: Unterdrückung des Rauschens n ohne das Signal s zu stören

- Ist nur annähernd erreichbar, da Eigenschaften von Signal und Rauschen nur stochastisch beschreibbar
- Beispiele klassischer Optimalfilter: Wiener-Filter, Kalman-Filter

Ansatz: Verwendung eines Referenz-Rauschsignals n_1, das mit dem zu beseitigenden Rauschen n_0 korreliert ist (siehe Abbildung 2.46).

- s ist die geforderte Antwort des adaptiven Filters
- Adaptive Rauschunterdrückung subtrahiert das gefilterte Referenzrauschen n_1 vom primären Input f.
- Anstelle Vorwissen hier Lernen der Signaleigenschaften

Annahme:

- s, n_0, n_1, u statistisch stationär, Mittelwert Null

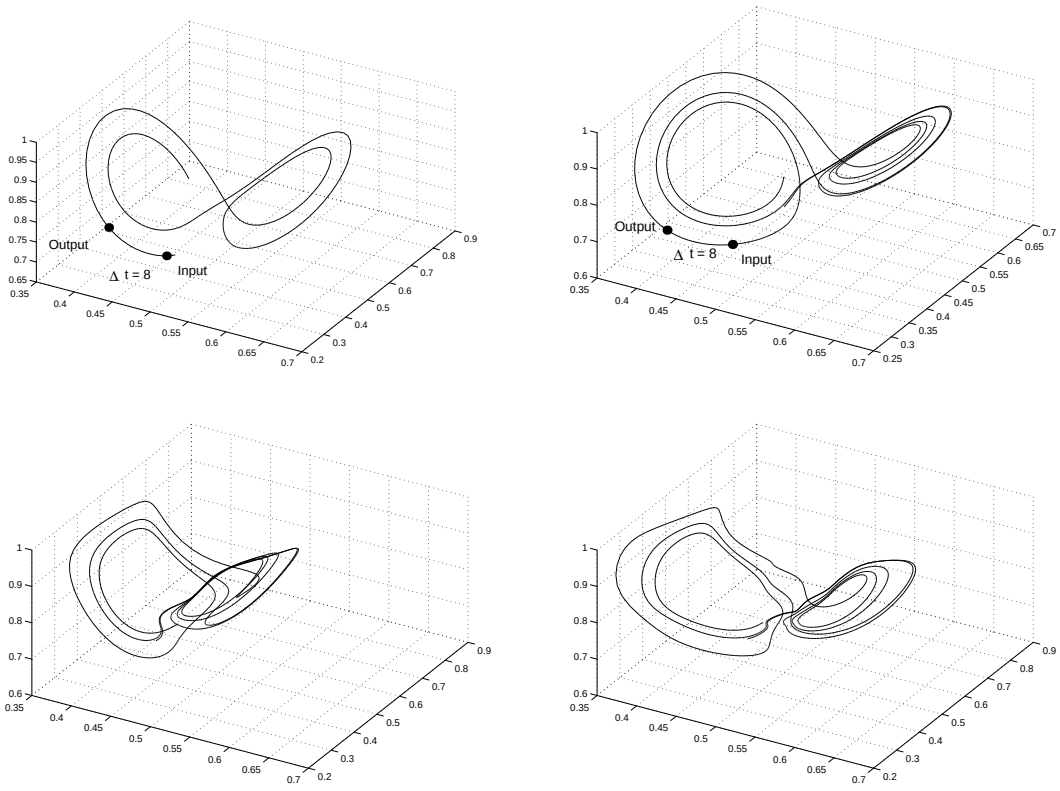


Abbildung 2.45: Prädiktion eines Lorenz-Attraktors mittels quaternionischen Clifford-Neuronen. Oben: links Trainingsmenge, rechts Testmenge. Unten: links MLP (13 verdeckte Knoten), rechts quaternionisches Spinor-CMLP (4 verdeckte Knoten).

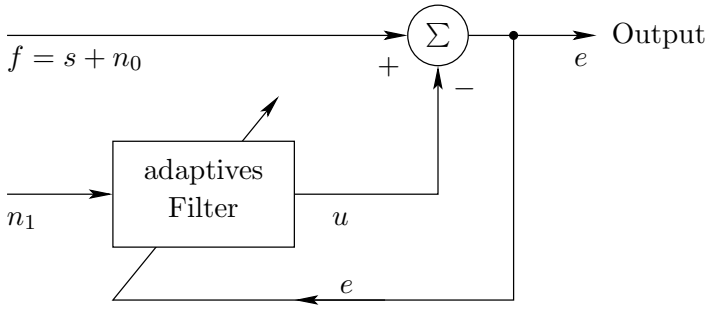


Abbildung 2.46: Rauschunterdrückung mittels adaptivem Filter.

- s unkorreliert zu n_0 und n_1
- n_1 korreliert zu n_0
- Der Output (Signal s) soll durch das Fehlersignal repräsentiert sein.

Lösung: Wegen $e = s + n_0 - u$ wird $e - s = n_0 - u \overset{!}{\approx} 0$ gefordert.

Es gilt:

$$e^2 = s^2 + 2s(n_0 - u) + (n_0 - u)^2$$
$$\Rightarrow E\left\{e^2\right\} = E\left\{s^2\right\} + 2E\left\{s(n_0 - u)\right\} + E\left\{(n_0 - u)^2\right\}$$

Gilt weiterhin $E\left\{s^2\right\} =$ const und $E\left\{s(n_0 - u)\right\} = 0$ (wegen Unkorreliertheit), so ist die Minimierung von $E\left\{e^2\right\}$ durch Filteranpassung möglich, **ohne** die Signalleistung $E\left\{s^2\right\}$ zu beeinflussen:

$$E_{\min}\left\{e^2\right\} = E\left\{s^2\right\} + E_{min}\left\{(n_0 - u)^2\right\}$$

Hieraus folgt, dass u die MMSE-Schätzung von n_0 ist.
Wegen $(e - s) = (n_0 - u)$ folgt außerdem, dass e die MMSE-Schätzung von s ist.
Das wird erreicht durch Minimierung der Outputleistung $E\left\{e^2\right\}$ des Systems!

Anwendung: Messung eines vom EKG der Mutter überlagerten fötalen EKGs (Veranschaulichung in den Abbildungen 2.47 und 2.48)

- Abdominale Ableitung: primärer Input $s + n_0$
- Brustableitung: Referenzinput n_1

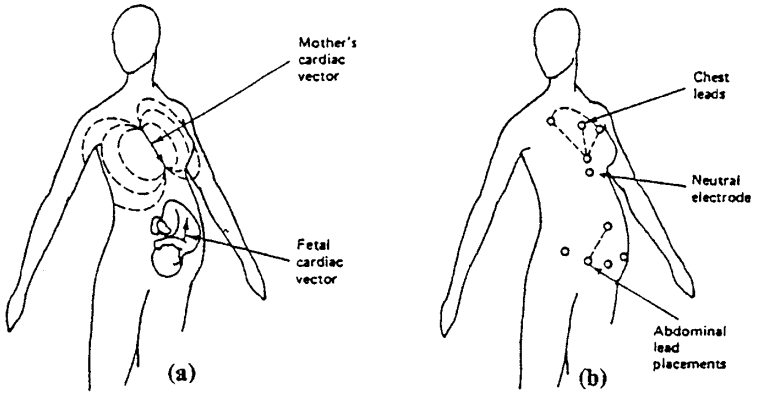


Figure 7. Canceling maternal heartbeat in fetal electrocardiography: (a) cardiac electric field vectors of mother and fetus; (b) placement of leads.

Abbildung 2.47: Problemstellung Schwangerschafts-EKG.

2.5.4 Modellierung eines inversen Systems

Gesucht : Inverse Systemübertragungsfunktion MTF
Lösung : Hintereinanderschalten des Systems und eines adaptiven Filters, das die Wirkung des Systems auslöscht.
Das Ziel besteht also darin, den Input auf den Output identisch abzubilden.

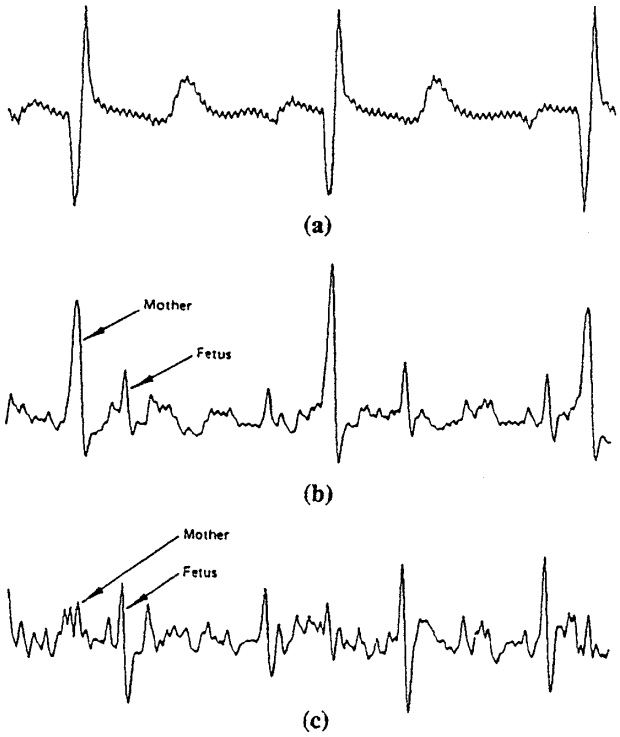


Figure 8. Result of fetal ECG experiment (bandwidth, 3-35 Hz; sampling rate, 256 Hz): (a) reference input (chest lead); (b) primary input (abdominal lead); (c) noise canceler output.

Abbildung 2.48: Angewandte Rauschunterdrückung für ein Schwangerschafts-EKG.

Grundlage der Methode:

- Systemübertragungsfunktion

$$STF = \mathcal{F}\{PSF\}$$

- Punktverbreitungsfunktion (Impulsantwort) PSF
  Jedes physikalische System besitzt eine PSF (deterministische Störung bzw. Verschmierung eines Impulses).

- Modulationsübertragungsfunktion (siehe Abbildung 2.49)

$$MTF = |STF|$$

  Beschreibt die verminderte Kontrastübertragung für höhere Frequenzen.

Gesucht: Funktion MTF^{-1}, die im Frequenzraum die Wirkung der MTF des Systems auslöscht (vergleiche Abbildung 2.50).
Annahme: Fehlersignal e sei nur klein

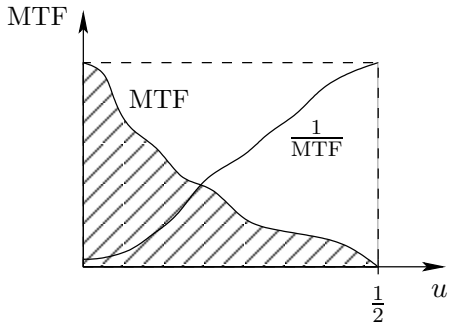


Abbildung 2.49: MTF.

Dann folgt $y^k \approx s^k$: Die Wirkung des unbekannten Systems P und des gelernten Modells H des inversen Systems heben sich nahezu auf.

$$\text{System } S(z) \approx Y(z) = H(z)P(z)S(z)$$

Hieraus folgt:

$$H(z) \approx \frac{1}{P(z)}$$

Bemerkung: Die Z-Transformierte einer diskreten Funktion f_k ist $F(z)$ mit:

$$F(z) := \sum_{k=-\infty}^{\infty} f_k z^{-k},\ z := e^{j\omega k}$$

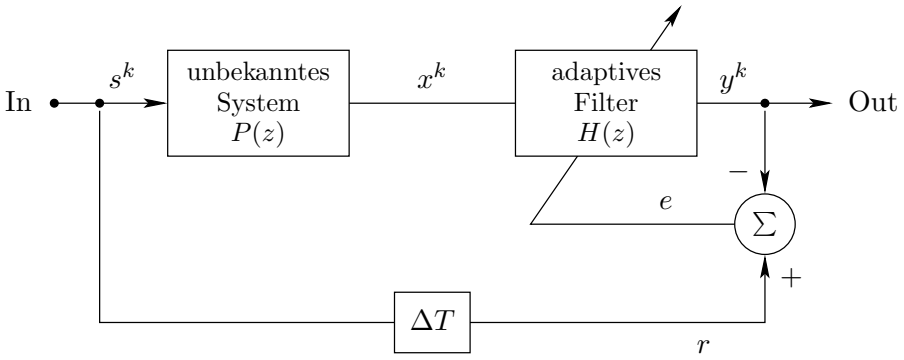


Abbildung 2.50: Schaltung zum Erlernen inverser Systeme.

Anwendungen:

- Korrektur von Meßgerätefehlern
- Kanalegalisierung (ISDN-Telefon: Beseitigung von Interferenzen benachbarter Übertragungskanäle)
- Adaptive Kontrolle: Stabbalance, Rückwärtsparken
- Auslöschung von Motorgeräuschen im PKW

2.6 Nichtlineare Entscheidungsprobleme

In diesem Abschnitt werden zwei Möglichkeiten vorgestellt, ein nichtlineares Entscheidungsproblem zu lösen. Tatsächlich stellen nichtlineare Entscheidungsprobleme die am häufigsten in der Praxis anzutreffenden Probleme dar.

1. Zurückführen auf eine gewisse Menge linearer Probleme: Hierzu werden in 2.6.1 das MLP und in 2.6.2 das SLN eingeführt. In Kapitel 4 wird das MLP eingehender behandelt.

2. Erweiterung des Perzeptrons durch eine Entscheidungsfunktion, die ein Polynom im Eingaberaum darstellt. In 2.6.3 wird das HON eingeführt und in 2.7 werden mit Hilfe der HON Invarianten bezüglich geometrischer Transformationsgruppen modelliert.

Wir lernten, dass das **Adaline** im **Offline-Verfahren** (Batchmode) **stationäre Signale** lernen kann. Nur in diesem Fall liefert das Adaline als lineare Berechnungseinheit sehr gute Ergebnisse.
Im Fall der **Nichtstationarität** des Stichprobenraumes gibt es zwei Möglichkeiten:

a) Trainieren einer Menge N von Adalines A_i für die Regionen Ω_i, $i = 1, ..., N$ des Stichprobenraumes $\Omega = \bigcup_i \Omega_i$ und bei der Anwendung

   1. Test der Zugehörigkeit der Daten $x \in \Omega_i$ als nichtlinearer erster Schritt
   2. Anwendung des Adalines A_i als lineare Berechnungseinheit für Ω_i

   Insgesamt ist das Verfahren **nicht-linear** wegen des erforderlichen Tests.

b) Anwendung des Adalines als **Online-Verfahren**. Hierbei erfolgt eine Adaption der Berechnungsvorschrift. Offensichtlich ist dieses Verfahren lokal linear (aber suboptimal) und eignet sich besonders wenn die Regionengrenzen zwischen den Ω_i nicht scharf sind.

Das **Perzeptron** setzt die **lineare Trennbarkeit** (Anwendung einer Hyperebene als Trennfunktion) einer Stichprobenpartitionierung $\Omega = \Omega_P \cup \Omega_N$ voraus. Ist das Problem aber **nicht linear trennbar**, so gibt es ebenfalls zwei Möglichkeiten:

a) **Vorverarbeitung der Daten**, so dass das Problem linear entscheidbar wird. z.B. durch **Koordinatentransformation**:
   Beispiel 1: Kartesische Koordinaten $\Longrightarrow$ Polarkoordinaten (siehe Abbildung 2.51)
   Dieses Verfahren stellt nicht immer eine Lösung dar.

   Beispiel 2: Nichtkonvexe Mengen $\Omega_i \rightarrow$ nichtlineare Eigenwerttransformation
   Durch eine geeignete nichtlineare Koordinatentransformation kann ein nicht

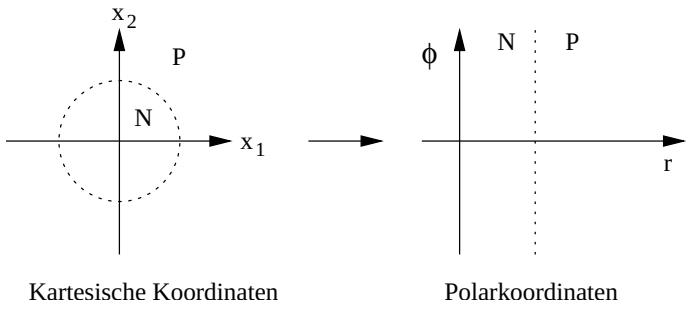


Abbildung 2.51: Koordinatentransformation.

linear trennbares Problem linear trennbar werden, siehe Abbildung 2.52. Abbildung 2.53 demonstriert eine lineare Eigenwerttransformation (PCA, siehe Abschnitt 3.10.1) zur Vorverarbeitung, die aber selbst noch nicht zur linearen Trennung führt.

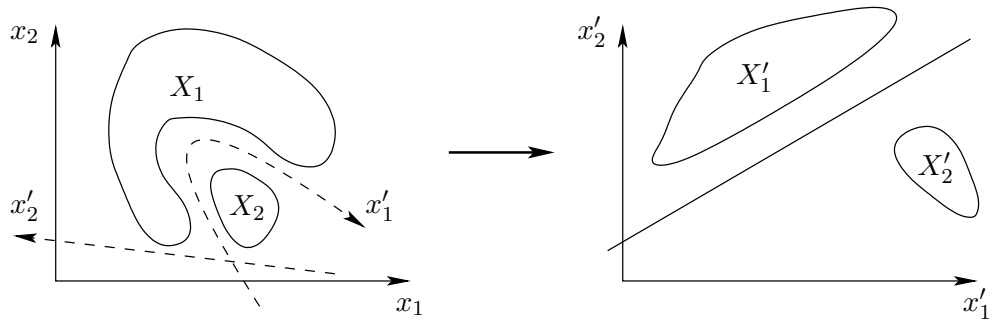


Abbildung 2.52: Nichtlineare Eigenwerttransformation.

b) **Anpassung des Klassifikators** an das Problem:

- **Hypersphären-Neuron** (entwickelt in unserer Arbeitsgruppe)

  Verallgemeinerung des Perzeptrons: Das Perzeptron wird in die konforme Geometrie angebettet. Hypersphären sind geometrische Entitäten, die im konformen Raum linear transformiert werden können. Eine Hypersphäre im Euklidischen Raum bleibt auch eine Hypersphäre im konformen Raum. Es gibt aber einen zum konformen Raum ähnlichen Raum, in dem sich die Hypersphäre als Hyperebene darstellt. Es erfolgt eine nichtlineare Transformation der Daten, die der Transformation des Problems in diesen „konformen“ Raum entspricht.

- **Stückweise lineare Approximation** einer nichtlinearen Entscheidungsgrenze durch ein SLN (Aufgabenteilung)

- **Umkodierung des Problems** in mehreren Verarbeitungsschichten eines kNN (MLP)

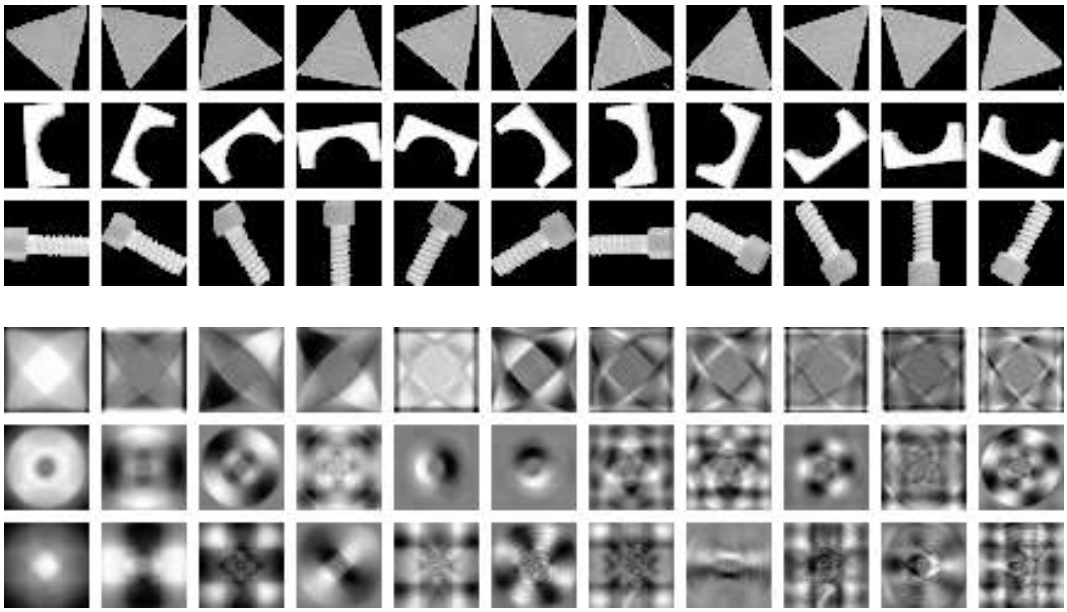


Abbildung 2.53: Daten (oben) und „Eigenbilder“ der drei Klassen (unten).

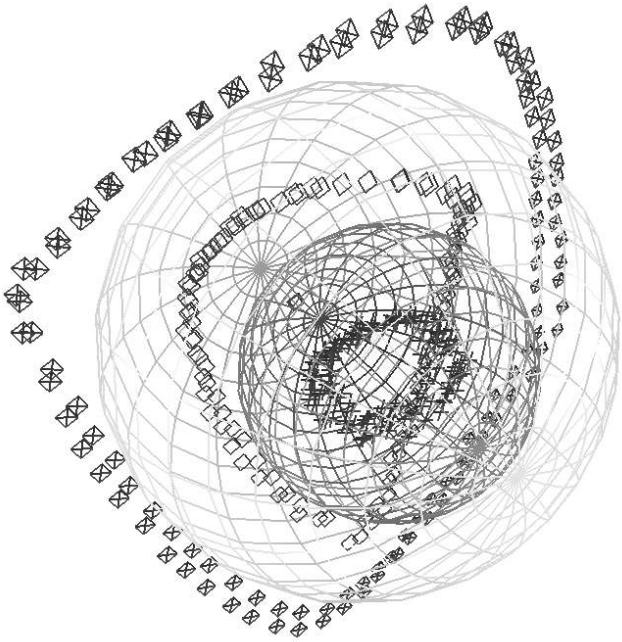


Abbildung 2.54: Trennung der Klassen aus Abbildung 2.53 mittels Hypersphären.

- Umkodierung des Problems durch **monomiale Erweiterung** des Eingaberaumes (HON)

2.6.1 XOR-Problem und Multilayer-Perzeptron (MLP)

Das **XOR-Problem** ist identisch mit dem von Minsky und Papert identifizierten **Paritätsproblem**. Nur für ungerade Parität der Eingabedaten antwortet die XOR-Funktion mit $y = 1$, sonst mit $y = 0$:

x_1	0	1	0	1
x_2	0	0	1	1
y_{XOR}	0	1	1	0

Es gibt keine lineare Trennfunktion, die Datenvektoren mit gerader Parität von denen mit ungerader Parität trennt.

1. $x = (0,0)$: $w_1x_1 + w_2x_2 = 0$ $< \theta$, da FALSE
2. $x = (1,0)$: $w_1x_1 + w_2x_2 = w_1$ $\geq \theta$, da TRUE
3. $x = (0,1)$: $w_1x_1 + w_2x_2 = w_2$ $\geq \theta$, da TRUE
4. $x = (1,1)$: $w_1x_1 + w_2x_2 = w_1 + w_2$ $< \theta$, da FALSE

Offensichtlich kann Bedingung 4 nicht erfüllt werden, wenn die anderen Bedingungen als gültig angesehen werden.

Für $f : \mathbb{B}^2 \to \mathbb{B}$ gibt es 16 verschiedene Boolesche Funktionen:

x_1	x_2	f_0	f_1	f_2	f_3	f_4	f_5	f_{XOR}	$\cdots$	f_{NXOR}	$\cdots$	f_{14}	f_{15}
0	0	0	1	0	1	0	1	0		1		0	1
0	1	0	0	1	1	0	0	1		0		1	1
1	0	0	0	0	0	1	1	1		0		1	1
1	1	0	0	0	0	0	0	0		1		1	1

Davon sind $f_6 \equiv$ XOR und $f_9 \equiv$ NXOR die einzigen Funktionen, für die ein Perzeptron kein geeignetes Modell darstellt.

Für $f : \mathbb{B}^n \to \mathbb{B}$ existieren $2^{2^n} n-$stellige Boolesche Funktionen (2^n Kombinationen des Inputs ergeben 2^{2^n} Kombinationen des Outputs).

Die Zahl n_{lt} der linear trennbaren Funktionen geht im Verhältnis zur Zahl der nicht-linear trennbaren Funktionen mit wachsendem n stark zurück:

$$2^{\frac{n(n-1)}{2}} < n_{lt}(n) \leq 2\sum_{i=0}^{n} \binom{2^n - 1}{i} < 2^{2^n}$$

Bemerkung: $\binom{2^n-1}{i} = \frac{(2^n-1)!}{(2^n-1-i)!i!}$

Für $n = 2$ existieren 14 lineare Trennfunktionen, siehe Abbildung 2.55.

Das XOR-Problem läßt sich aber mit 3 linearen Trennfunktionen lösen (siehe Abbildung 2.56), die in zwei Schichten angeordnet sind. Dies ist eine einfache Variante eines **Multilayer-Perzeptronnetzes** (MLP). Offensichtlich repräsentiert diese

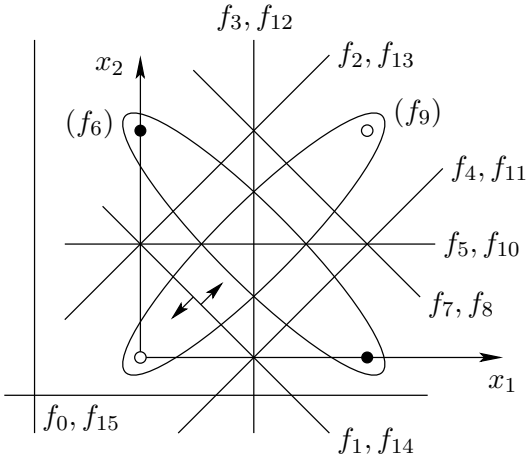


Abbildung 2.55: Mit einem Perzeptron realisierbare und nicht realisierbare Trennfunktionen.

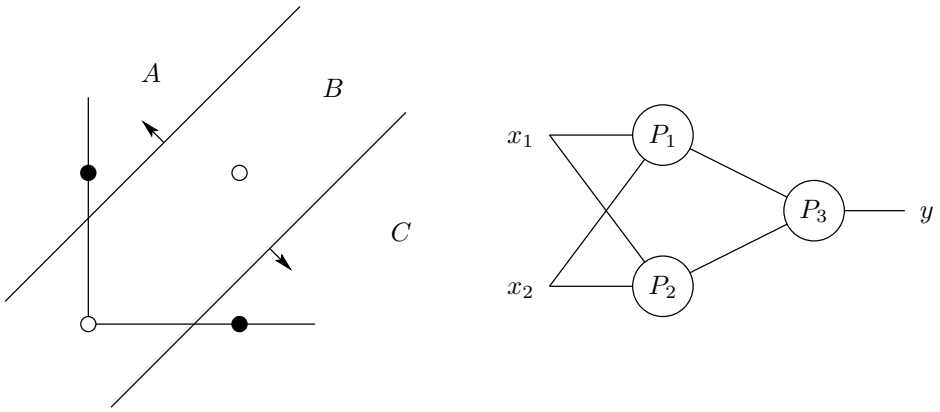


Abbildung 2.56: Berechnung von f_{XOR} mittels Perzeptronen.

Architektur eine **nichtlineare Trennfunktion**.

Frage: Welche Funktionen können die drei Perzeptrone repräsentieren?
Antwort: Hierfür existieren 16 mögliche Kombinationen aus den 14 linear trennbaren Booleschen Funktionen.

Beispiel 1: $f_{\mathrm{XOR}} = f_{14}(f_4, f_2)(x_1, x_2) = (x_1 \wedge \overline{x}_2) \vee (\overline{x}_1 \wedge x_2)$ (siehe Abb. 2.57)
Beispiel 2: $f_{\mathrm{XOR}} = f_8(f_7, f_{14})(x_1, x_2) = (\overline{x_1 \wedge x_2}) \wedge (x_1 \vee x_2)$

Erklärung der Lösbarkeit durch drei Neuronen:
Der Eingaberaum wird in drei Regionen A, B, C unterteilt:

pos. Halbräume	:	A, C	(y=1)
neg. Halbraum	:	B	(y=0)

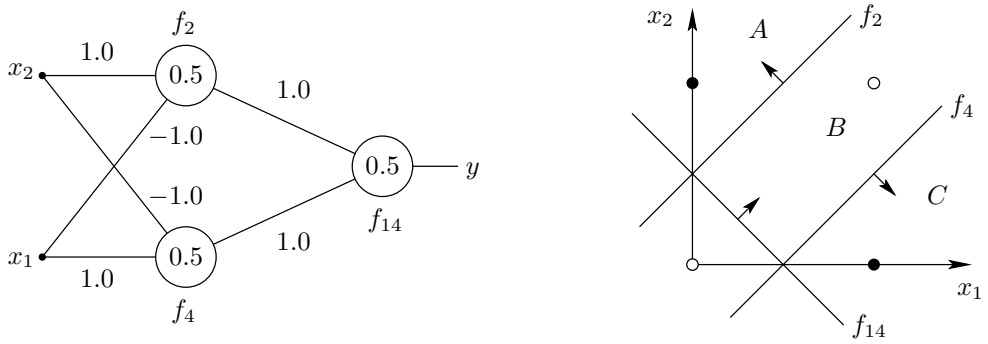


Abbildung 2.57: Beispiel eines Multilayer-Perzeptrons.

Die beiden Neuronen der ersten Schicht kodieren die Regionen, in der sich die Eingabe befindet. Das einzelne Neuron in der zweiten Schicht dekodiert die Regionen.

Hierbei erfolgt eine Projektion aus dem 4D-Stichprobenraum in einen 3D-Regionenraum!

- Für die Kodierung von drei Regionen werden zwei Bits benötigt. Hieraus ergeben sich vier Lösungsvarianten.

A	B	C
01	00	10
00	01	11
10	11	01
11	10	00

- Da die zwei Neuronen der ersten Schicht vertauscht werden dürfen, ergeben sich $2 \cdot 4 = 8$ Lösungen für f_{XOR}.
- Weitere 8 Lösungen folgen aus einer anderen Aufteilung des Eingaberaumes, vgl. Abbildung 2.58.

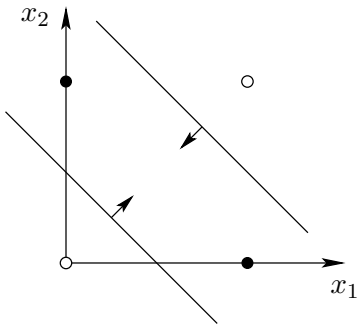


Abbildung 2.58: Alternative Aufteilung des Eingaberaums.

- Also existieren insgesamt 16 Lösungen.

2.6.2 Singlelayer-Perzeptron (SLN) und Lerntheorie

Offensichtlich existiert auch eine weniger effiziente Lösung des XOR-Problems durch ein SLN in einer einzigen Schicht von Perzeptronen. Dabei wird eine **verteilte Kodierung** des Outputs y_{XOR} verwendet. XOR ist erfüllt, wenn jedes´der Ausgabeneuronen feuert. Auch hiervon existieren verschiedene Varianten, siehe Abildung 2.59.

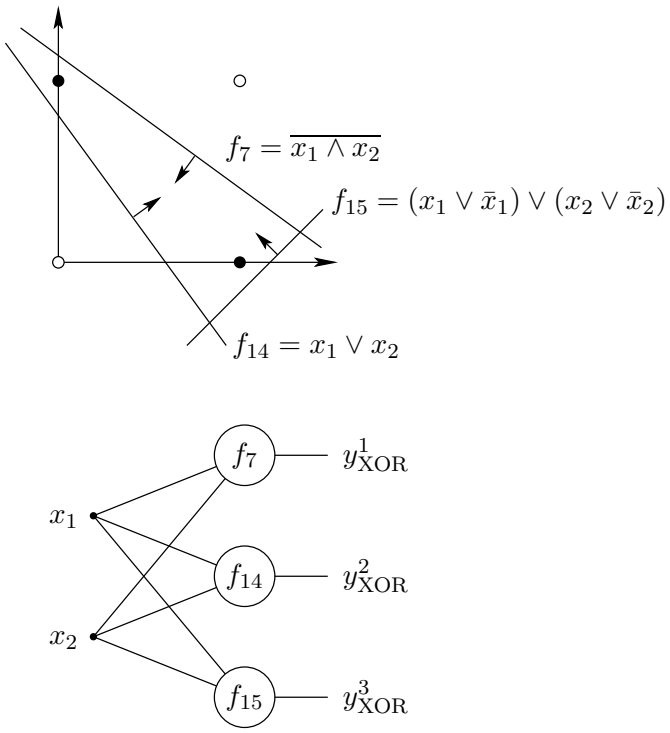


Abbildung 2.59: Berechnung von f_{XOR} mittels SLN.

Computational Lerntheorie: Welche Elemente des Eingaberaumes können in Teilräumen bzw. Klassen zusammengefaßt werden?

Frage 1: Welches Konzept klassifiziert die Teilmenge $S \subset X$ korrekt?
Hierbei sei X der Eingaberaum ($X \equiv \mathbb{R}^n$).
Im Fall des Perzeptrons wird der Eingaberaum in Halbräume zerteilt.

Hieraus folgt: Ein Element $x \in S$ ist lernbar ($y = 1$), da das Halbraumkonzept x richtig zuordnet.

Beispiel 1: Ein Cluster S von Elementen x wird in $\mathbb{R}^2$ durch drei Perzeptrone kodiert.
Generell gilt, um einen konvexen Cluster im $\mathbb{R}^n$ zu definieren, sind mindestens $n+1$ Neuronen erforderlich, vergleiche Abbildung 2.60, links. Hier werden die drei Trennlinien jeweils durch ein Bit in einem dreistufigen Binärcode dargestellt. Der Wert Eins zeigt in Richtung des positiven Halbraums, der Wert Null zeigt in Richtung des negativen Halbraums.

Beispiel 2: Identifikation der Vereinigungsmenge zweier Cluster im $\mathbb{R}^2$, vergleiche Abbildung 2.60, rechts.

1. Schicht : 2 x 3 Perzeptrone
2. Schicht : 2 x 1 Perzeptron (AND) definiert die einzelnen Cluster
3. Schicht : 1 x 1 Perzeptron (OR) definiert die Vereinigungsmenge der Cluster

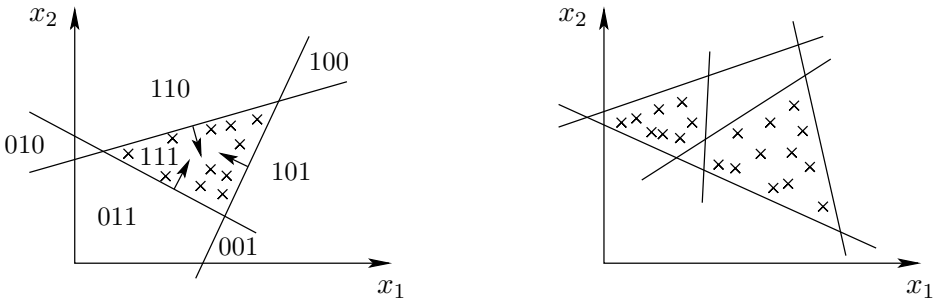


Abbildung 2.60: Identifikation von Clustern im $\mathbb{R}^2$ mittels Perzeptron.

Auf diese Art können beliebige konvexe Polytope klassifiziert werden.

Frage 2: Welche maximale Anzahl von Elementen im Eingaberaum kann durch ein gegebenes Konzept korrekt klassifiziert werden?

Annahme: Perzeptrone, $X \equiv \mathbb{R}^2$

Drei Punkte in beliebiger Position können durch das Halbraumkonzept von drei Perzeptronen linear getrennt werden, siehe Abbildung 2.61. Vier Punkte in allgemeiner Lage sind hingegen mit 3 Perzeptronen nicht mehr linear trennbar (vergleiche f_{XOR} in 2.6.1).
.
Eine Boolesche Funktion $f : \mathbb{B}^2 \to \mathbb{B}$ benötigt zu ihrer Definition aber minimal $2^2 = 4$ unabhängige Stichprobenelemente. Also muß es unter diesen Funktionen solche geben, die sich nicht linear trennen lassen.
Allgemein gilt: Sind K Punkte in einem n-dimensionalen Raum gegeben, so können mit Hilfe $(n-1)$-dimensionaler Hyperebenen $c(K, n)$ verschiedene Gruppierungen

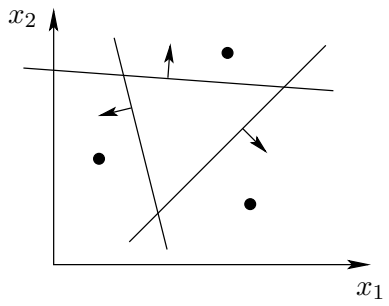


Abbildung 2.61: Trennbarkeit von 3 Punkten mittles Perzeptronen.

unterschieden werden, wobei gilt:

$$c(K,n) := 2\sum_{i=0}^{n}\binom{K-1}{i}$$

Vapnik-Chervonenskis-Dimension (VCD):
Die VCD gibt die maximale Anzahl von Stichprobenelementen im Eingaberaum an, die mit einer gegebenen Klasse von Konzepten getrennt werden kann. Damit ist die VCD ein Maß der Systembefähigung, ein Entscheidungsproblem zu lösen. Ein Problem ist bezüglich eines Konzeptes lernbar, wenn die VCD endlich ist.
Im Falle Boolescher Funktionen erfolgt durch die Projektion des Eingaberaumes auf den Regionenraum eine Umkodierung des Entscheidungsproblems bzw. eine Anpassung an die VCD ($n = 2 : \text{VCD} = 3$), so dass das Problem lösbar wird.

2.6.3 Polynomiale Klassifikatoren und HON

In diesem Abschnitt werden **higher order nets** (HON) bzw. Neuronen höherer Ordnung als **polynomiale Klassifikatoren** einführt. Das sind Systeme, die eine polynomiale Trennfunktion zur Klasseneinteilung verwenden.

Wir bedienen uns wieder des Beispieles der XOR-Funktion. Hierfür war typisch, dass eine der Regionen im Eingaberaum (entweder Ω_N oder Ω_P) nicht zusammenhängend war.

Ein Polynom der Ordnung m

$$y = p_m(x) = \sum_{i=0}^{m} a_i x^i$$

mit $x, y \in \mathbb{R}$ ist eine (nichtlineare) Funktion, die eine topologische Abbildung des Eingaberaumes auf dem Ausgaberaum in der Weise erzeugt, dass nicht zusammenhängende Gebiete des Eingaberaumes einem zusammenhängenden Gebiet des Ausgaberaumes zugeordnet werden können.

Beispiel 1: $n = 1$ ($x \in \mathbb{R}$)
Ein Beispiel für die Trennbarkeit verschiedener Gebiete ($R_1, R_2 \subset \mathbb{R}$) mittels einer Polynomfunktion gibt Abbildung 2.62.

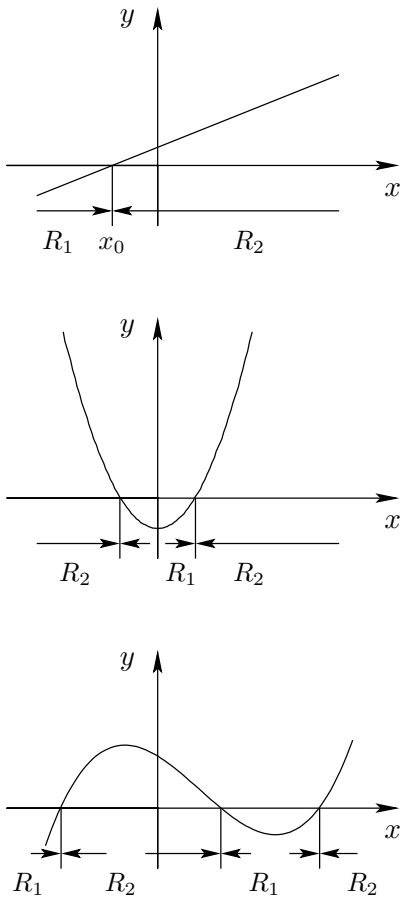


Abbildung 2.62: Beispiel einer Gebietstrennung mittles Polynomfunktion.

Erweitert man diese Funktion auf einen n-dimensionalen Eingaberaum, $x \in \mathbb{R}^n$, so treten auch Mischterme der Koeffizienten des Eingabevektors bis zur Ordnung m auf:

$$x_1^{\mu_1} x_2^{\mu_2} \cdots x_n^{\mu_n} \text{ mit } \mu_1 + \mu_2 + \cdots \mu_n \leq m$$

Beispiel 2: n beliebig, $m = 2$

$$y = a_0 + \sum_{i=1}^{n} a_i x_i + \sum_{i=1}^{n-1} \sum_{k=i+1}^{n} a_{ik} x_i x_k + \sum_{i=1}^{n} a_{ii} x_i^2$$

Beispiel 3: $n = 2$, $m = 2$

$$y = a_0 + a_1 x_1 + a_2 x_2 + a_{12} x_1 x_2 + a_{11} x_1^2 + a_{22} x_2^2$$

Offensichtlich kann eine nichtlineare Abbildung $f : \mathbb{R}^n \to \mathbb{R}$ durch ein Polynom $p_m(x)$ auch als lineare Abbildung $g : \mathbb{R}^N \to \mathbb{R}$ interpretiert werden. Der neue Eingaberaum wird durch die aus den Eingabekomponenten gebildeten **Monome** $\xi_j = x_1^{\mu_1} x_2^{\mu_2} \cdots x_n^{\mu_n}, \sum_i \mu_i \leq m$ gebildet, von denen es N Stück gibt:

$$N := \frac{(n+m)!}{n!m!}$$

Also läßt sich auch schreiben:

$$y = \sum_{i=0}^{N-1} a_i \xi_i$$

Die Nicht-Linearität wird in die Bildung der Monome verschoben.

Beispiel 4: $n = 2$, $m = 2$

$$\xi = (\xi_0 = 1, \xi_1 = x_1, \xi_2 = x_2, \xi_3 = x_1 x_2, \xi_4 = x_1^2, \xi_5 = x_2^2)$$

Leider wächst die Anzahl N der freien Parameter a_i sehr schnell an ($n = m = 10 : N = 184756$).

Beispiel 5: XOR-Problem
Im Fall des XOR-Problems ($n = 2$) genügt ein Polynom der Ordnung $m = 2$, um die nichtlineare Trennung von Ω_P und Ω_N zu realisieren.

$$\begin{aligned} u \equiv u(x) &= \sum_{i=0}^{2} \sum_{j=0}^{2} w_{ij} x_i x_j \\ &= w_0 + 2w_1 x_1 + 2w_2 x_2 + w_{11} x_1^2 + w_{22} x_2^2 + 2w_{12} x_1 x_2 \\ &\text{mit} \quad w_{ij} = w_{ji}, w_{0i} = w_{i0} = w_i \text{ und } x_0 = 1 \end{aligned}$$

Hierbei werden in der verwendeten Summendarstellung die Terme der Ordnung $< m$ absorbiert.

Im ursprünglichen Eingaberaum stellt das Polynom für $u = 0$ die Ortskurve einer Ellipse oder einer Hyperbel in allgemeiner Lage dar, je nach dem, wie die Regionenbildung erfolgt, bzw. welche Vorzeichen die Gewichte haben und ob nur Terme x_i^2 oder auch Terme $x_i x_j$ verwendet werden, vergleiche Abbildung 2.63.

Annahme: Ellipse in Hauptachsenrichtung

$$\frac{(z_1 - c)^2}{a^2} + \frac{(z_2 - d)^2}{b^2} = 1$$

mit $a \geq \frac{1}{2}\sqrt{2}$, $b < \frac{1}{\sqrt{2}}$, $c = d = 0$ (Ellipse geht durch Ursprung)

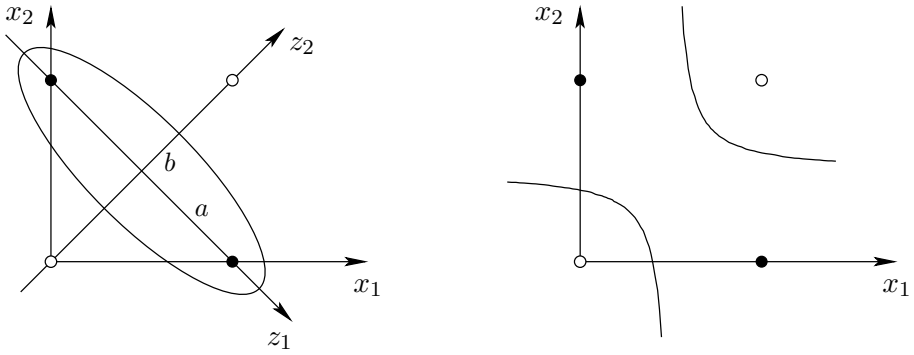


Abbildung 2.63: Ellipse und Hyperbel in allgemeiner Lage.

Die Ellipse im 2-dimensioanlen Eingaberaum entspricht der Projektion einer linearen Trennfunktion im 6-dimensionalen Raum der Monome. Im Falle des XOR-Problems wird aber nicht der gesamte 6-dimensionalen Raum benötigt, wie gleich zu sehen sein wird.

Sei $\xi = (x_1, x_2, x_1x_2)$. Dann kann in diesem 3D-Raum das XOR-Problem linear getrennt werden (siehe Abbildung 2.64), wobei folgende Umkodierung der Stichprobenelemente erfolgt:

$$(x_1, x_2) : (0,0) \to (0,0,0), (1,1) \to (1,1,1), (1,0) \to (1,0,0), (0,1) \to (0,1,0)$$

Hier definieren $(1,0,0)$ und $(0,1,0)$ die positiven Stichprobenelemente.

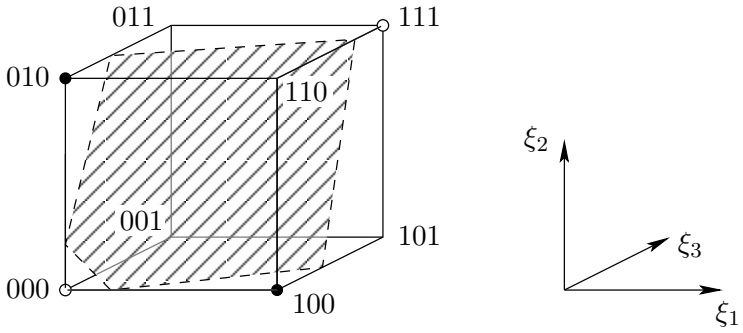


Abbildung 2.64: XOR-Problem im 3D-Raum.

Beispiel einer Trennebene:
Die Trennebene

$$\xi_1 + \xi_2 - 2\xi_3 - \frac{1}{4} = 0$$

entspricht der Propagierungsfunktion

$$u(x) = -\frac{1}{4} + x_1 + x_2 - 2x_1x_2 \begin{cases} > 0 & x \in \Omega_P \\ < 0 & x \in \Omega_N \end{cases} .$$

Für den Test auf Erfüllung der XOR-Funktion $\xi_1 + \xi_2 - 2\xi_3 \geq \frac{1}{4}$ ergibt sich:

$$\begin{array}{lrcll} (1,0,0): & 1 \geq \frac{1}{4} & \rightarrow & \text{TRUE:} & y = 1 \\ (0,1,0): & 1 \geq \frac{1}{4} & \rightarrow & \text{TRUE:} & y = 1 \\ (0,0,0): & 0 \geq \frac{1}{4} & \rightarrow & \text{FALSE:} & y = 0 \\ (1,1,1): & 0 \geq \frac{1}{4} & \rightarrow & \text{FALSE:} & y = 0 \end{array}$$

Lernen des XOR-Problems mittels HON:
Sei $\Omega = \{x^1, x^2, x^3, x^4\}$, und sei $x^k = (x_1^k, x_2^k)$.
Annahme: $\xi = (\xi_1,\ \xi_2,\ \xi_3)$ genügt zur Kodierung
Freiheitsgrade: Wahl der Aktivierungsfunktion und Kodierung der Eingabedaten

a) Stufenfunktion

step	x^1	x^2	x^3	x^4
ξ_1^k	0	0	1	1
ξ_2^k	0	1	0	1
ξ_3^k	0	0	0	1
r_{XOR}	0	1	1	0

$\xi_3 = x_1 x_2$ kodiert AND $\rightarrow$ Kodierung des Problems deshalb nicht günstig

b) Signumfunktion
$x_i^k \in \{-1, 1\} \Longrightarrow$ Transformation des Koordinatenursprunges nach $(0.5, 0.5)$

sgn	x^1	x^2	x^3	x^4
ξ_1^k	-1	-1	1	1
ξ_2^k	-1	1	-1	1
ξ_3^k	1	-1	-1	1
r_{XOR}	-1	1	1	-1

Da $\xi_3^k = -r_{\text{XOR}}^k$, kodiert ξ_3 die Funktion NXOR. ξ_3 repräsentiert also die **Korrelation** der Eingabe, während XOR die **Antikorrelation** repräsentiert. Diesen Unterschied im Vorzeichen zu lernen, ist eine einfache Aufgabe. Deshalb kann bei dieser Kodierung des XOR-Problems auch auf die Beiträge von ξ_1 und ξ_2 in der Propagierungsfunktion verzichtet werden.

Ausgabefunktion:

$$y = \text{sgn}(w_{12} x_1 x_2 - \theta)$$

Gewichtskorrektur:

$$w_{12}(t+1) = w_{12}(t) + \left(r^k - y^k(t)\right) x_1^k x_2^k$$

Annahme: $w_{12}(0) = 0.5$, $\theta = 0$

Lernzyklus:

$$
\begin{aligned}
t=0 \;:\; & x^1=(-1,-1), r^1=-1 \\
& y^1(0)=1 \approx \text{ FALSE, da } y^1(0)\neq r^1 \\
& w_{12}(1)=0.5-2(-1)(-1)=-1.5 \\
t=1 \;:\; & x^1=(-1,-1), r^1=-1 \\
& y^1(1)=-1 \approx \text{ TRUE, da } y^1(1)=r^1 \\
t=1 \;:\; & x^2=(-1,1), r^2=1 \\
& y^2(1)=1 \approx \text{ TRUE, da } y^2(1)=r^2 \\
t=1 \;:\; & x^3=(1,-1), r^3=1 \\
& y^3(1)=1 \approx \text{ TRUE, da } y^3(1)=r^3 \\
t=1 \;:\; & x^4=(1,1), r^4=-1 \\
& y^4(1)=-1 \approx \text{ TRUE, da } y^4(1)=r^4
\end{aligned}
$$

Die XOR-Funktion wird mittels **eines** Higher-Order-Neurons in einem Zyklus gelernt. Im Gegensatz hierzu brauchen 2-Schicht-Multi-Layer-Perzeptronen (2 Kodierer + 1 Dekodierer) u.U. tausende Zyklen!

2.7 Kodierung von Invarianzen in HON

Lernen mittels kNN bedeutet, eine in der Topologie und den Gewichten eines kNN gegebene implizite Repräsentation eines Problems zu finden, das ebenfalls nur implizit durch die Beispiele aus dem Stichprobenraum definiert ist. Es wird gefordert, dass die neuronale Repräsentation nur die Eigenschaften des Problems erfaßt und unabhängig von der Darstellung im Stichprobenraum ist.

Im Allgemeinen kann dies nur dadurch antrainiert werden, dass der Stichprobenraum alle möglichen Varianten der Präsentation des Problems enthält, so dass diese Varianten herausgemittelt werden und das eigentliche Problem als Invariante übrig bleibt.

Beispiel: Robot Vision
Probleme: Objekterkennung, Bewegungserkennung, Visual Servoing,...

Präsentations-/Umwelt-/Meßeinflüsse:

- Beleuchtungsgeometrie und -intensität (Luminanzvarianten)
- Rauschen und Verschmierung durch die Kamera
- Beobachtungsgeometrie (projektive Störungen)

- "Eigenbewegungen" der Objekte (geometrische Transformationsgruppen: Translation, Rotation, Skalierung, ...)
  Annahme: Bewegung starrer Körper (Gruppe $SE(3)$)

Es ist sehr aufwendig (praktisch unmöglich) alle Einflüsse im Stichprobenraum zu erfassen. Deshalb ist es wünschenswert, das kNN so zu entwerfen, dass es durch seine Architektur die gewünschte Invarianz leistet. Dies ist nur in speziellen Fällen möglich:

1) In HON kann Invarianz der Ausgabe bezüglich willkürlicher endlicher Transformationsgruppen kodiert werden.

2) In Clifford-MLPs können ebenfalls solche Invarianzen repräsentiert werden. Clifford-MLPs sind HON überlegen (sind eine algebraische Verallgemeinerung).

In dieser Vorlesung wird aber nur das HON behandelt, weil es einfacher darzustellen ist.

Beispiel: Lernen einer 2D-Ähnlichkeitstransformation mittels Clifford-Neuron
Problem: Durchführung einer 2-dimensionalen Ähnlichkeitstransformation, z.B. (siehe Abbildung 2.65):

$$f : \mathbb{R}^2 \to \mathbb{R}^2; x \mapsto \frac{3}{2} x \begin{pmatrix} \cos(-55°) & -\sin(-55°) \\ \sin(-55°) & \cos(-55°) \end{pmatrix} + [1, 0.8]$$

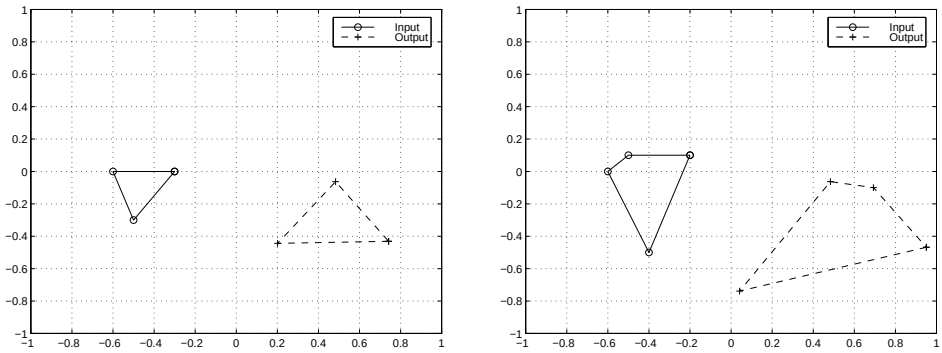


Abbildung 2.65: Euklidische 2D-Ähnlichkeitstransformation: Trainingdaten (links) und Testdaten (rechts).

Das Lernen der Aufgabe erfolgt mittels eines reellwertigen SLNs mit zwei Neuronen bzw. eines einzigen komplexwertigen oder spinorwertigen Neurons. Lernkonvergenz und Resultate dieser drei Möglichkeiten finden sich in den Abbildungen 2.66 und 2.67.

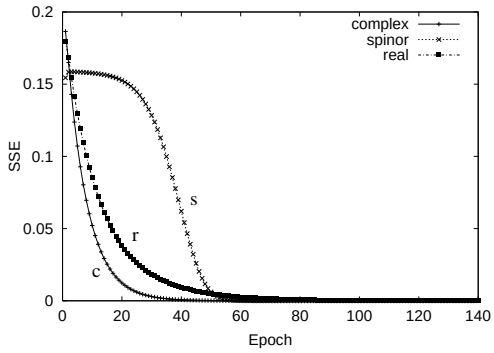


Abbildung 2.66: Euklidische 2D-Ähnlichkeitstransformation: Konvergenz des Lernens.

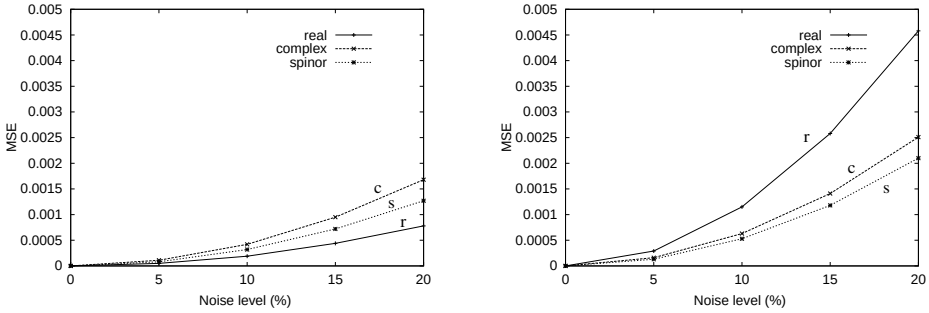


Abbildung 2.67: Euklidische 2D-Ähnlichkeitstransformation: Rauschabhängigkeit der Resultate.

2.7.1 Invarianz durch Mittelwertbildung über eine Gruppe

Zunächst soll der Begriff Invarianz formalisiert werden.

Definition: Gruppe
Eine Menge G, für die folgende vier Bedingungen erfüllt sind, heißt *Gruppe*:

(1) In G ist eine Verknüpfung (z.B. Multiplikation) erklärt.

(2) Die Verknüpfung ist assoziativ.

(3) Die Menge G enthält ein Einselement e (neutrales Element).

(4) Zu jedem Element $a \in G$ existiert ein Inverses $a^{-1} \in G$.

■

Diese Definition hat folgende Interpretation:

(1) Die Verknüpfung zweier Element $a, b \in G$ liefert ein Element $c \in G$:

$$c = a \circ b$$

Ist die Verknüpfung außerdem noch kommutativ, so heißt die Gruppe **kommutativ** oder **Abelsch** (nach dem norwegischen Mathematiker Niels Henrik Abel, 1802-1829).

(2) $a, b, c \in G$:

$$a \circ (b \circ c) = (a \circ b) \circ c$$

(3) $a, e \in G$:

$$a = a \circ e$$

(4) Wenn $a, b, c \in G$, dann auch $b^{-1} \in G$:

$$c = a \circ b$$

$$\Longrightarrow c \circ b^{-1} = a \circ b \circ b^{-1} = a \circ e = a$$

Die Anwendung eines Gruppenelements $g \in G$ auf ein Datum $x \in X$ erfolgt entsprechen $x' := gx$ mit $x' \in X$.

Definition (Invarianz bezüglich Gruppe):
Sei G eine endliche Gruppe. Dann bedeutet Invarianz einer Abbildung $f : x \longrightarrow y$:

$$y(gx) = y(x) \qquad \forall g \in G$$

■

Invarianz durch Summenbildung in kNN:
Annahme: σ sei die Aktivierungsfunktion und u sei die Propagierungsfunktion

$$y(x) = \sigma \left[\sum_{g \in G} u(gx) \right]$$

Zu zeigen: Invarianz bzgl. G
Für $g, h \in G, \quad f = g \circ h \in G$ gilt:

$$y(hx) = \sigma \left[\sum_{g \in G} u(g \circ hx) \right] = \sigma \left[\sum_{f \in G} u(fx) \right] = y(x)$$

Die Multiplikation aller Terme einer Summe über einer endlichen Gruppe mit einem Element dieser Gruppe bewirkt eine **Permutation der Summenterme**. Daraus folgt

$$\sum_{g = f \circ h^{-1} \in G} u(gx) = \sum_{f \in G} u(fx)$$

und die Invarianz von $y(x)$ nach Voraussetzung.

Beispiel: XOR-Problem
Das XOR-Problem wird durch die **Gruppe der Vorzeichenumkehr** definiert und ist folglich auch invariant bezüglich dieser Gruppe:

$$\begin{aligned} x \in \Omega_P &: x_1 = -x_2 \\ x \in \Omega_N &: x_1 = x_2 \end{aligned}$$

$G_s := \{s_0, s_1\}$ Gruppe der Vorzeichenumkehr mit

$$s_0 \{x_i\} = x_i \ , \quad s_1 \{x_i\} = -x_i$$

Umformung der Gleichung $y = \sigma(u)$ als Summe (Mittelwert) über die Gruppe:

$$\begin{aligned} y &= \sigma \left[\sum_{j=0}^{1} u(s_j x) \right] \\ &= \sigma \left((w_1(s_0 x_1 + s_1 x_1) + w_2(s_0 x_2 + s_1 x_2) + w_{12}(s_0 x_1 s_0 x_2 + s_1 x_1 s_1 x_2) \right) \\ &= \sigma \left(w_1(x_1 - x_1) + w_2(x_2 - x_2) + w_{12}(x_1 x_2 + x_1 x_2) \right) \\ &= \sigma(2 w_{12} x_1 x_2) \end{aligned}$$

Die linearen Anteile $w_i x_i$ tragen nichts nur Definition von XOR bei!
Damit für $x = (x_1, x_2) \in \Omega_P$ folgt $y = +1$, muss wegen $x_1 x_2 < 0$ (Antikorrelation) $w_{12} < 0$ gelten. Dieser Zwang der Gewichte ist also eine Folge der Invarianz der Funktion bzgl. der Gruppe der Vorzeichenumkehr.

2.7.2 Gruppenmittel für geometrische Transformationen

Das durch Gruppenmitteilung abgeleitete **Perzeptron zweiter Ordnung** ist speziell angepaßt an die XOR-Funktion. Es ist damit nicht mehr universell wie ein allgemeiner Ansatz zweiter Ordnung. Es hat aber gegenüber einem MLP-Netz mit Neuronen erster Ordnung einige Vorteile, zum Beispiel schnellere Konvergenz. Dies hat ihre Ursache darin, dass die für das XOR-Problem erforderliche Kodierung nicht mehr während des Trainings gesucht werden muß, sondern die spezifische Kodierung ist in der Architektur bereits vorgegeben.

Beispiel: Lernen komplexer Multiplikation
Seien $x, y, z \in \mathbb{C}$. Zu lernen sind die Symmetrien einer 2×2-Matrix W, so dass die Abbildung $z := Wx$ einer komplexen Multiplikation $z = xy$ entspricht.

Das Prinzip der Gruppenmitteilung zur Berechnung von Invarianten geht auf die große deutsche Mathematikerin Emmy Noether (1882 - 1935) zurück.

Einschub: Emmy Noether
- Tochter des berühmten Mathematikers Max Noether (1844 - 1921, algebraische Geometrie).

- Mitbegründerin der modernen abstrakten Algebra nach herausragenden Arbeiten (Dissertation, Erlangen 1908) zur Theorie algebraischer Invarianter.
- Habilitation, Göttingen, 1919 bei D. Hilbert
- Eine der ersten Frauen in Deutschland mit Hochschullehrer-Laufbahn.
- Wurde jedoch nie zum ordentlichen Professor berufen.
- Zitat: „Meine Herren, eine Universität ist doch keine öffentliche Badeanstalt!“ (Spektrum der Wissenschaft, August 2004, S.70-77)
- Emmy-Noether-Programm der DFG[2]

Satz 2.3 Noethers Invariantentheorem (1916):

Gegeben sei eine endliche Gruppe G, die über einem endlich dimensionalen Vektorraum X wirkt ($x \in X : (x_1, ..., x_n)$). Es sei $I_1, ..., I_e \in \mathbb{C}[X]^G$ eine endliche Menge von Invarianten, aus denen die Menge aller Invarianter $\mathbb{C}[X]^G$ aller möglicher Funktionen des Vektorraumes X mit Invarianz bezüglich der Gruppe G gebildet werden können. Die Menge $\{I_1, ...I_e\}$ bildet eine Basis von $\mathbb{C}[X]^G$. Das Gruppenmittel eines beliebigen **Polynoms** *$f \in \mathbb{C}[X]$ sei definiert durch:*

$$\tilde{f}(x) = \sum_{g \in G} f(gx)$$

Es ist möglich, eine Basis von $\mathbb{C}[X]^G$ durch Berechnung der Gruppenmittel über alle **Monome** *$x_1^{b_1} x_2^{b_2} ... x_n^{b_n}$ mit $\sum_{j=1}^{n} b_j \leq |G|$ zu berechnen. Solche Basis hat höchstens $\binom{|G|+n}{n}$ Elemente.* ∎

Das Gruppenmittel eines Polynoms aus X stellt eine Abbildung des Polynoms in die bezüglich der Gruppe G invariante Äquivalenzklasse dar. Ein bezüglich des Gruppenmittels erweitertes HON operiert deshalb über der Äquivalenzklasse der repräsentierten Funktion bezüglich der Gruppe G. Dies wirkt sich dadurch auf die Architektur des Netzes/Neurons aus, dass seine Freiheitsgrade (die Gewichte) gewissen Zwängen unterworfen werden.

Wir zeigen diese Zusammenhänge für geometrische Transformationen wie

Translation, **Skalierung** und **Rotation**,

sowie für die Gruppen, die mehrere derartige Transformationen enthalten.

Annahmen: siehe Giles, Griffin und Maxwell (1988)
Sei $p(x)$ eine Signaleigenschaft am Ort x, z.B. Grauwert oder Ort selbst oder ...

[2] `http://www.dfg.de/forschungsfoerderung/nachwuchsfoerderung/emmy_noether`

Sei $n \leq 2$ und seien $x_j \in \mathbb{R}^n$ Orte, die durch Gruppenwirkung in Beziehung stehen.

Betrachte ein Neuron zweiter Ordnung: $m = 2$ sei maximale Ordnung

$$y = \sigma(u) = \sigma\left[\sum_{j=0}^{m} u^{(j)}\right]$$

$$\begin{aligned} u^{(0)} &= w^{(0)} \\ u^{(1)} &= \int w^{(1)}(u; x_1) p(x_1) dx_1 \\ u^{(2)} &= \iint w^{(2)}(u; x_1, x_2) p(x_1) p(x_2) dx_1 dx_2 \end{aligned}$$

Die Integrationsgrenzen werden je nach Problem gewählt, z.B.: $x \in [-\infty, \infty]$.
Allgemein:

$$\begin{aligned} w &= \left(w^{(0)}, w^{(1)}, ..., w^{(m)}\right)^T \\ w^{(j)} &= w^{(j)}(u; x_1, ..., x_j),\ j \in [0, m] \end{aligned}$$

1. 1D-Translationsinvarianz:
Sei $x_j \in \mathbb{R}$. Sei T die **Translationsgruppe** und sei $t \in T$ ein Translationsvektor:

$$t\left(p(x)\right) = p_0(x) = p(x + x_0)$$

Translationsinvarianz:

$$t\left(y(w, p(x))\right) = y\left(w, p_0(x)\right) = y\left(w, p(x + x_0)\right) \stackrel{!}{=} y\left(w, p(x)\right)$$

Können die Orte x durch Translation ineinander überführt werden, so ist die Antwort eines Knotens (und damit des Netzes) auf das Merkmal $p(x)$ unabhängig vom Ort.

Beispiel: Auftreten eines gelben Farbwertes auf einer Bildzeile

Hieraus folgt Translationsinvarianz für jedes $u^{(j)}$: (zunächst gilt die Gleichheit)

$$\begin{aligned} u^{(1)} &= \int w^{(1)}(u; x_1) p(x_1) dx_1 = \int w^{(1)}(u; x_1) p(x_1 + x_0) dx_1 \\ u^{(2)} &= \iint w^{(2)}(u; x_1, x_2) p(x_1) p(x_2) dx_1 dx_2 \\ &= \iint w^{(2)}(u; x_1, x_2) p(x_1 + x_0) p(x_2 + x_0) dx_1 dx_2 \end{aligned}$$

Aus der geforderten Invarianz folgt, dass die Gewichte $w^{(j)}$ für die translatierten Merkmale (rechts) und die nicht translatierten Merkmale (links) gleich sind. Deshalb werden auf der rechten Seite folgende Substitutionen durchgeführt:

$$x_1 \longrightarrow x_1 - x_0 \ , \ x_2 \longrightarrow x_2 - x_0 \quad (\text{auch } t^{-1} \in T!)$$

$$\begin{aligned} u^{(1)} &= \int w^{(1)}(u; x_1)p(x_1)dx_1 = \int w^{(1)}(u; x_1 - x_0)p(x_1)d(x_1 - x_0) \\ u^{(2)} &= \iint w^{(2)}(u; x_1, x_2)p(x_1)p(x_2)dx_1dx_2 \\ &= \iint w^{(2)}(u; x_1 - x_0)p(x_1)p(x_2))d(x_1 - x_0)d(x_2 - x_0) \end{aligned}$$

Wegen $x \in [-\infty, \infty]$ gilt: $d(x_1 - x_0) = dx_1$ und $d(x_2 - x_0) = dx_2$.
Hieraus folgt im Kontinuierlichen ein Zwang für die funktionelle Form der Gewichte, bzw. im Diskreten eine Begrenzung erlaubter Verbindungen und damit der Zahl der Gewichte.

$$\begin{aligned} w^{(1)}(u; x_1) &= w^{(1)}(u; x_1 - x_0) \\ w^{(2)}(u; x_1, x_2) &= w^{(2)}(u; x_1 - x_0, x_2 - x_0) \end{aligned}$$

- Die **Gewichte erster Ordnung** sind unabhängig von den Eingabekoordinaten, sie hängen nur von der Ausgabe (u) ab:

  $$w^{(1)}(u; x_1) = w^{(1)}(u)$$

  Ein Netz erster Ordnung würde unter diesem Zwang der Gewichte den **Mittelwert der Merkmale** $p(x)$ lernen. Dieser ist zwar translationsinvariant aber auch konstant und trägt somit **keine Strukturinformation**. Das Neuron könnte keine Strukturen lernen!

- Die **Gewichte zweiter Ordnung** hängen nur vom Differenzvektor der Eingabe ab:

  $$w^{(2)}(u; x_1, x_2) = w^{(2)}(u; x_2 - x_1) = w^{(2)}(u; \Delta)$$

  Die Berechnung des Gruppenmittels der Translation bedeutet hier, alle möglichen Differenzvektoren $\Delta \in \mathbb{R}$ zu berücksichtigen.

  $$y = \sigma \left[\iint w^{(2)}(u; \Delta)p(x_1)p(x_1 + \Delta)dx_1d\Delta \right]$$

  In der eckigen Klammer steht die Autokorrelationsfunktion des Merkmals $p(x)$, die bekanntermaßen translationsinvariant ist. Sie ist die inverse Fouriertransformierte der spektralen Leistungsdichte, $|P(u)|^2$, des Merkmals $p(x)$. Das Gewicht zweiter Ordnung $w^{(2)}$ wichtet hierbei die Autokorrelation.

- Für eine diskrete Implementierung mit $x \in \mathbb{R}^n$ (n Eingabeknoten, die mit der Ausgabe voll verbunden sind) reduziert sich die Anzahl der Gewichte zweiter Ordnung von n^2 auf n. Nur Gewichte der Differenzkoordinaten sind erforderlich.

2. 2D-Translationsinvarianz:
Sei $x_j \in \mathbb{R}^2$, $x_j = (x_{1,j}, x_{2,j}$. Die Differenzvektoren $\Delta = x_2 - x_1 = (\Delta_1, \Delta_2)$ sind durch deren **Länge**

$$|\Delta| = \|x_2 - x_1\|$$

und **Orientierung**

$$\alpha = \operatorname{atan}\left(\frac{\Delta_2}{\Delta_1}\right) = \operatorname{atan}\left(\frac{x_{2,2} - x_{1,2}}{x_{2,1} - x_{1,1}}\right)$$

bestimmt. Die Translationsinvarianz von Strecken $\Delta := x_2 - x_1$ bedeutet:

$$\begin{aligned} w^{(1)}(u; x_1) &= w^{(1)}(u) \\ w^{(2)}(u; x_1, x_2) &= w^{(2)}(u; |\Delta|, \alpha) \end{aligned}$$

Die Berechnung des Gruppenmittels der Translation erfolgt nach:

$$y = \sigma\left[\iiint w^{(2)}(u; |\Delta|, \alpha)\, p(x_1)\, p(x_1 + \Delta)\, dx_1\, d|\Delta|\, d\alpha\right]$$

3. 2D-Skalierungsinvarianz:
Sei $x_k \in \mathbb{R}^2$. Sei S die Gruppe der Skalierung. Dann gelte folgender Ansatz für ein $s \in S$:

$$s\,(p(x)) = a^n p(ax)$$

mit a als Skalenfaktor und n als Dimension des Eingabraumes (hier: $n = 2$).

Dieser Ansatz der Skalierung gewährleistet, dass das Integral über das Merkmal (die Energie des Merkmals) unabhängig von der Skala ist. Ohne den Vorfaktor würde sich die Struktur stark ändern:

$a > 1$: Stauchung des Definitionsbereiches $\rightarrow$ Funktion wird steiler
$a < 1$: Streckung des Definitionsbereiches $\rightarrow$ Funktion wird flacher

Skalierungsinvarianz erfordert:

$$\begin{aligned} w^{(1)}(u; x_1) &= w^{(1)}(u; \frac{x_1}{a}) \\ w^{(2)}(u; x_1, x_2) &= w^{(2)}(u; \frac{x_1}{a}, \frac{x_2}{a}) \end{aligned}$$

a) **Annahme: Polarkoordinaten**
Sei $x_k = (\rho_k, \varphi_k)$; $\rho = 0$ sei das Skalierungszentrum.

$$\begin{aligned} w^{(1)}(u; \rho_1, \varphi_1) &= w^{(1)}(u; \varphi_1) \approx \text{ Unabhängigkeit von } \rho \\ w^{(2)}(u; \rho_1, \rho_2, \varphi_1, \varphi_2) &= w^{(2)}(u; \frac{\rho_2}{\rho_1}, \frac{\varphi_1}{\varphi_2}) \end{aligned}$$

Die Gewichte hängen explizit nur vom Winkel ab.

b) **Annahme: Kartesische Koordinaten**
Sei $x^k = (x_{k,1}, x_{k,2})$.

$$w^{(1)}(u; x_1) = w^{(1)}(u; \frac{x_{1,2}}{x_{1,1}})$$

Für die Gewichte zweiter Ordnung existieren verschiedene Möglichkeiten, den Zwang zu formulieren:

$$\begin{aligned} w^{(2)}(u; x_1, x_2) &= w^{(2)}(u; \frac{x_{1,2}}{x_{1,1}}, \frac{x_{2,2}}{x_{2,1}}) \\ w^{(2)}(u; x_1, x_2) &= w^{(2)}(u; \frac{x_{2,1}}{x_{1,1}}, \frac{x_{2,2}}{x_{1,2}}) \\ w^{(2)}(u; x_1, x_2) &= w^{(2)}(u; \alpha) \end{aligned}$$

mit $\alpha =$ atan $\left(\frac{x_{2,2}-x_{1,2}}{x_{2,1}-x_{1,1}}\right)$.

Im Gegensatz zur 2D-Translationsinvarianz ist bei der 2D-Skalierungsinvarianz der einzige Freiheitsgrad für einen Zwang durch die **Orientierung der Strecke** $\Delta = x_2 - x_1$ gegeben.

4. 2D-Rotationsinvarianz
Sei $x_k \in \mathbb{R}^2$. Sei R die Gruppe der Rotationen. Dann gilt für ein $r \in \mathbb{R}$:

$$r\,(p(x)) = p(rx)$$

Die Rotation ist eine lineare Operation (es existiert eine Matrixdarstellung).

a) **Annahme: Polarkoordinaten**
Sei $x_k = (\rho_k, \varphi_k); \rho = 0$ sei das Rotationszentrum.

$$\begin{aligned} w^{(1)}(u; x_1) &= w^{(1)}(u; \rho_1) \approx \text{ Unabhängigkeit von } \varphi \\ w^{(2)}(u; x_1, x_2) &= w^{(2)}(u; \rho_1, \rho_2, \varphi_2 - \varphi_1) \end{aligned}$$

Die Gewichte hängen explizit nur von der radialen Distanz ab.

b) **Annahme: Kartesische Koordinaten**
Sei $x_k = (x_{k,1}, x_{k,2})$.

$$\begin{aligned} w^{(1)}(u; x_1) &= w^{(1)}(u; |x_1|) \\ w^{(2)}(u; x_1, x_2) &= w^{(2)}(u; |\Delta|) \end{aligned}$$

Der einzige im Fall der 2D-Rotation trainierbare Zwang ist durch die **Länge der Strecke** $\Delta = x_2 - x_1$ gegeben.

5. 2D-Translations und Skalierungsinvarianz
Translation um $x_0 \in \mathbb{R}^2$ und Skalierung um $\alpha \in \mathbb{R}$ sind nicht kommutativ:

$$s \circ t \neq t \circ s$$

a) Zuerst Translation, dann Skalierung:

$$\begin{aligned} w^{(1)}(u;x_1) &= w^{(1)}(u;\frac{x_1-x_0}{a}) \\ w^{(2)}(u;x_1,x_2) &= w^{(2)}(u;\frac{x_1-x_0}{a},\frac{x_2-x_0}{a}) \end{aligned}$$

b) Zuerst Skalierung, dann Translation:

$$\begin{aligned} w^{(1)}(u;x_1) &= w^{(1)}(u;\frac{x_1}{a}-x_0) \\ w^{(2)}(u;x_1,x_2) &= w^{(2)}(u;\frac{x_1}{a}-x_0,\frac{x_2}{a}-x_0) \end{aligned}$$

Sei $x_k = (x_{k,1}, x_{k,2})$. Für $w^{(2)}$ ist die folgende Lösung invariant bzgl. **Skala und Translation um die Strecke** $\Delta = x_2 - x_1$ und die Lösung ist unabhängig von der Reihenfolge der Transformationen:

$$\begin{aligned} w^{(2)}(u;x_1,x_2) &= w^{(2)}(u;\alpha) \\ &\quad \text{mit } \alpha := \operatorname{atan}\left(\frac{x_{2,2}-x_{1,2}}{x_{2,1}-x_{1,1}}\right) \end{aligned}$$

6. 2D-Rotations- und Skaleninvarianz
Sei $x_k = (x_{k,1}, x_{k,2})$.

$$\begin{aligned} w^{(1)}(u;x_1) &= w^{(1)}(u;\frac{\rho_1}{a},\varphi_1) \\ w^{(2)}(u;x_1,x_2) &= w^{(2)}(u;\frac{\rho_2}{\rho_1},\varphi_2-\varphi_1) \end{aligned}$$

7. 2D-Invarianz bezüglich Translation, Skala und Rotation
Bei einem HON zweiter Ordnung gäbe es keinen Freiheitsgrad, der zur Formulierung dieses Zwanges (Integration über die Freiheitsgrade) dienen könnte.
Aber bei einem HON dritter Ordnung werden die Beziehungen dreier Punkte betrachtet. Diese bilden unter der geforderten Invarianz **ähnliche Dreiecke**. Das heißt, die eingeschlossenen Winkel bleiben γ_j erhalten (siehe Abbildung 2.68):

$$w^{(3)}(u;x_1,x_2,x_3) = w^{(3)}(u;\gamma_1,\gamma_2,\gamma_3)$$

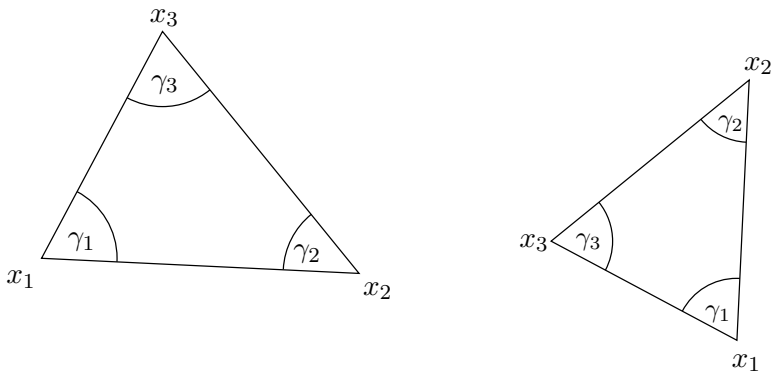


Abbildung 2.68: Ähnliche Dreiecke.

3 Grundlagen der statistischen Entscheidungstheorie

3.1 Einführung und Übersicht

In diesem Kapitel werden die wichtigsten Berechnungsparadigmen und Methoden der statistischen Entscheidungstheorie eingeführt, um in der Folge die Problemlösung mittels kNN auf dieser Grundlage motivieren und verstehen zu können. Die statistische Entscheidungstheorie beschäftigt sich mit dem Treffen von in gewisser Hinsicht optimalen Entscheidungen auf der Grundlage von

- **Wissen** über den Problembereich
  (statistisch beschreibbares Wissen)
- **Daten** als Ergebnis von Messungen zum Problem
  (statistisch auswertbare Messungen, empirisches Wissen).

Oft werden Optimierungsaufgaben unter Randbedingungen formuliert. Diese **Randbedingungen** betreffen z.B. die Konsequenzen (Kosten, Risiken) einer Entscheidung im Kontext der Aufgabe.
Allgemein klafft zwischen der **Realität** und der **Theorie** eine Lücke, die kühn übersprungen werden muß, um einen theoretischen Apparat wie die Wahrscheinlichkeitstheorie anwenden zu können:

- Wahrscheinlichkeiten $\Longrightarrow$ Häufigkeiten
- Dichtefunktionen $\Longrightarrow$ Häufigkeitsverteilungen

Wesentlich ist die Verwendung **bedingter Wahrscheinlichkeiten** und **bedingter Dichtefunktionen**.

Ein **Mustererkennungssystem** hat meist eine der folgenden typischen Aufgaben zu lösen:

1. **Identifizieren** eines Musters als Vertreter einer **bekannten Klasse:**
   Dies führt zu Klassifizierung durch **überwachtes Training**.
   Beispiel: Finden einer Entscheidungsgrenze
2. **Zuordnung** eines Musters zu einer bis dato **unbekannten Klasse**:
   Dies führt zu Klassifizierung durch **unüberwachtes Training**:
   Beispiel: Clusterbildung bzw. Clusteranalyse der Daten.

In der Vorlesung interessieren besonders lernende Systeme. Mustererkennung in diesem Sinn bedeutet, ein Muster einer Äquivalenzklasse zuzuordnen. Dieser Zugang des Lernens wird gewählt, um eine Beschreibung der Äquivalenzklasse durch implizite Repräsentation in den Gewichten eines neuronalen Netzes zu erkennen.

Für ein Mustererkennungssystem unterscheidet man gewöhnlich zwei **Operationsmodi**:

1. **Trainingsphase**: Sammeln empirischen Wissens (Daten) und Trainieren der Aufgabe.

2. **Arbeitsphase**: Anwenden der trainierten (impliziten) Regeln auf neue Daten, um z.B. optimale Entscheidungen zu treffen.

Bei **adaptiven Systemen** kann diese Unterteilung nicht mehr getroffen werden. Man spricht in diesem Fall auch von **Online-Lernen**.

Es gibt aber auch typische Problemstellungen der Mustererkennung, bei denen anstelle der Zuordnung eines Datums zu einer Äquivalenzklasse der Vergleich eines Datums mit einem oder mehreren anderen Mustern im Vordergrund steht:

1. **Verifikation**: Vergleich eines Datums (aus mehreren) mit einem vorhandenen Muster.

2. **Identifikation**: Vergleich eines Datums mit mehreren (vielen) vorhandenen Mustern.

Die vorhandenen Muster stellen ebenfalls empirisches Wissen dar. Man nennt sie auch **Templates** (Masken). Templates reduzieren das Konzept der Klassen auf einen Repräsentanten (den **Prototypen**). Folglich wird bei diesem Konzept nicht die Separierung der Struktureigenschaften einer Klasse in die invarianten und varianten Anteile durchgeführt. Vielmehr interessiert das Wiedererkennen eines bzw. mehrerer Templates in den experimentellen Daten.

3.1.1 Begriffe der statistischen Entscheidungstheorie

Ω:	**Problembereich**: umfaßt alle möglichen Situationen, Muster etc. **sicheres Ereignis**: Stichprobenraum (Raum aller möglichen Ereignisse)
$\Psi \subset \Omega$:	**stochastisches Ereignis** (Ergebnis eines zufälligen Versuchs): Es existieren unendlich viele versch. Partitionierungen $\Omega = \cup_j \Psi_j$. Existiert eine Zerlegung $\Omega = \bigcup_j \Psi_j$ mit $\Psi_i \neq \Psi_j$ für $i \neq j$, so heißt Ω auch vollständiges System unvereinbarer Ereignisse.
$\emptyset$:	**leere Menge**:

	Ist $\Psi = \emptyset$, dann heißt Ψ auch **unmögliches Ereignis**.
$\omega \in \Omega$:	**Elementarereignis**: Im Ergebnis eines zufälligen Versuches ist Ψ genau dann eingetreten, wenn $\omega \in \Psi$.
$P(\Psi)$:	**Wahrscheinlichkeit** für das Eintreten des Ereignisses Ψ
$P(\Psi_i, \Psi_j)$:	**Verbundwahrscheinlichkeit** (joint probability) für das gleichzeitige Eintreten der Ereignisse Ψ_i und Ψ_j. Es gilt: $P(\Psi_i, \Psi_j) = P(\Psi_i \cap \Psi_j)$
$P(\Psi_i\|\Psi_j)$:	**Bedingte Wahrscheinlichkeit**, d.h. Wahrscheinlichkeit für das Eintreten des Ereignisses Ψ_i unter der Bedingung, dass auch Ψ_j eingetreten ist, also gilt: $P(\Psi_i\|\Psi_j) = \frac{P(\Psi_i,\Psi_j)}{P(\Psi_j)}$ für $P(\Psi_j) > 0$ Hier heißt $P(\Psi_j)$ auch **a priori Wahrscheinlichkeit**.
$X : \Omega \to \mathbb{R}$:	**Stochastische Variable**, Zufallsgröße: Mit dieser Abbildung können Ereignissen aus Ω Zahlen zugeordnet werden.
$x = X(\omega)$:	**Realisierung** einer stochastischen Variablen
$P(X)$:	**Wahrscheinlichkeitsverteilung** der stochastischen Variablen X: Aus $\Psi \longrightarrow P(\Psi)$ folgt $X \longrightarrow P(X)$.
$p(x)$:	**Dichtefunktion** der Realisierungen x

Es gilt:

$$P(X) := \int p(x)\, dx$$

Im Fall **diskreter stochastischer Variabler** $X : \Omega \to \mathbb{Z}/\mathbb{N}$ geht die Dichtefunktion der Realisierungen über in deren **Einzelwahrscheinlichkeiten** $P(x_i)$.

$E\{X\}$: **Erwartungswert** einer stochastischen Variablen:

$$E\{X\} = \int_{-\infty}^{\infty} x\, p(x) dx$$

Der Erwartungswert $E\{X\}$ ist das erste **gewöhnliche Moment** von X. Höhere Momente von X werden notiert als $E\{X^i\}$:

$$E\left\{X^i\right\} = \int_{-\infty}^{\infty} x^i p(x) dx$$

Die auf den Erwartungswert $E\{X\}$ bezogenen Momente heißen **zentrale Momente**. Von großer Bedeutung ist das zweite zentrale Moment, die **Varianz**:

$$\begin{aligned} \mathrm{Var}\{X\} &= E\left\{(X - E\{X\})^2\right\} \\ &= \int_{-\infty}^{\infty} (x - E\{X\})^2 p(x) dx \end{aligned}$$

Die Schätzung des Erwartungswertes hängt von der Dichteverteilung der Realisierungen, $p(x)$, ab.

Liegen M Realisierungen x_i der diskreten stochastischen Variablen X vor und sind diese durch eine Dichtefunktion der **Gleichverteilung** beschrieben, dann wird $p(x)$ ersetzt durch die Einzelwahrscheinlichkeiten $P(x_i)$ für alle Realisierungen x_i:

$$P(x_i) = \frac{1}{M}$$

Folglich wird der Erwartungswert geschätzt als arithmetischer Mittelwert m:

$$E\{X\} \stackrel{!}{=} m(x_i) = \frac{1}{M} \sum_{i=1}^{M} x_i$$

Totale Wahrscheinlichkeit:

Sei $\Omega = \{\Psi_k\}_1^n$ ein vollständiges System unvereinbarer Ereignisse und sei Ψ_j ein Ereignis aus einer anderen Partitionierung von Ω. Dann gilt:

$$P(\Psi_j) = \sum_{k=1}^{n} P(\Psi_j|\Psi_k)P(\Psi_k)$$

Multiplikationsregel:

$$P(\Psi_j \cap \Psi_k) = P(\Psi_j|\Psi_k)P(\Psi_k) = P(\Psi_k|\Psi_j)P(\Psi_j)$$

Mit der Multiplikationsregel folgt eine für uns sehr wichtige Beziehung: Kennt man die Wahrscheinlichkeiten $P(\Psi_j)$ und $P(\Psi_k)$ und die bedingten Wahrscheinlichkeiten $P(\Psi_k|\Psi_j)$, so lassen sich auch die bedingten Wahrscheinlichkeiten $P(\Psi_j|\Psi_k)$ angeben.

Bayes-Formel:

$$P(\Psi_j|\Psi_k) = \frac{P(\Psi_k|\Psi_j)P(\Psi_j)}{P(\Psi_k)}$$

(Thomas Bayes (1702-1761) engl. Theologe mit großem mathematischem Interesse)

Beispiel: siehe Tabelle 3.1.1
Eine **Urne** ist ein Modell der stochastischen Variablen, in der auf alle Realisierungen $x = X(\omega)$ zugegriffen werden kann. Aus der Menge aller Realisierungen können statistische Aussagen über die zugrunde liegende stochastische Variable abgeleitet werden.
Doch oft liegen die Realisierungen als Messungen über einem **Parameterraum** T (Ort, Zeit,...) vor. Das heißt, die stochastische Variable ist auch eine Funktion $X(t), t \in T$ (z.B. eine Zeitreihe):

$$X(\omega, t) \mid \Omega \times T \to \mathbb{R}^{|T|\cdot|\Omega|} \quad \textbf{stochastischer Prozess}$$

$P(\text{Fieber}\|\text{Grippe})$	Schätzung ist einfach
$P(\text{Grippe}\|\text{Fieber})$	schwierig schätzbar
$P(\text{Grippe})$	Existiert z.Zt. eine Grippewelle?
$P(\text{Fieber})$	Temperaturmessung

$$\Rightarrow P(\text{Grippe}|\text{Fieber}) = \frac{P(\text{Fieber}|\text{Grippe}) \cdot P(\text{Grippe})}{P(\text{Fieber})}$$

Tabelle 3.2: Beispiel zur Bayes-Formel.

Um einen stochastischen Prozess besser erfassen zu können, sind folgende Projektionen vorteilhaft (vergleiche Abbildung 3.1):

1. Realisierungen sind reelle Funktionen bezüglich des Freiheitsgrades $t \in T$:
   Für $\omega = \omega^* \in \Omega$:
   $$X : \Omega \to \mathbb{R}^{|T|}, \quad \omega \mapsto x_{\omega^*}(t) := X(\omega^*, t)$$

2. Realisierungen einer stochastischen Variablen bezüglich des Freiheitsgrades $\omega \in \Omega$:
   Für $t = t^* \in T$:
   $$X : T \to \mathbb{R}^{|\Omega|}, \quad t \mapsto x_{t^*} := X(\omega, t^*)$$

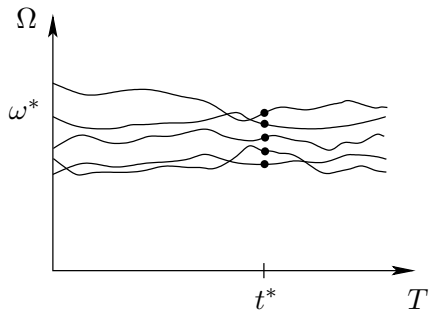


Abbildung 3.1: Beispiel mehrerer Realisierungen einen stochastischen Prozesses.

Es wird gefordert, dass beide Projektionen zur gleichen Prozesscharakterisierung führen. Hierbei entstehen oft Probleme in der Praxis (Forderung der Erfüllung der **Ergodenhypothese → ergodischer Prozess**).
Diskrete Parameterräume führen dazu, dass ein Prozess X auch als stochastische Vektorvariable behandelt werden kann. An jeder Koodinate t_i wirkt die stochastische Variable $X(t_i)$ mit $X = (X(t_1), X(t_2), ..., X(t_n))$.
Stochastische Prozesse in ihrer allgemeinsten Formulierung sind sehr schwer auf praktische Probleme anwendbar. Deshalb werden oft gravierende Einschränkungen getroffen. Dann erweist sich das Konzept aber als äußerst tragfähig.

Sind die Abtastpositionen t_i beliebig verschiebbar, $t_i \mapsto t_i + \tau$, so heißt ein stochastischer Prozess **stationär**, wenn alle seine Momente hiervon unberührt bleiben.
Ein **schwach stationärer Prozess** ist vollständig durch seine ersten beiden Momente (Mittelwert und Kovarianz) bestimmt. Diese Prozessklasse wird häufig gewählt.

3.1.2 Problembeschreibung eines Entscheidungsprozesses

Gegeben sei ein **Stichprobenraum** $\Omega_S := \{(x^s, y^{s_k}) \mid s = 1, ..., S\}$, der für unsere Aufgabe implizit die Sicht auf die Welt repräsentiert. Die Problemlösung kann höchstens so gut sein, wie die durch den Stichprobenraum gegebene Problembeschreibung! Es kommt also in erster Linie auf gute Stichproben an. Sie sollten den Raum Ω aller möglicher Stichproben gut abtasten ($\Omega_S \subset \Omega$!).
Die **Stichprobe** (x^s, y^{s_k}) ordnet das Datum (**Merkmal**) x^s einer Klasse c_k durch den Referenzwert y^{s_k} zu. Wird y^{s_k} in dieser Weise dem Merkmal x^s zugeordnet, heißt die Stichprobe **indiziert**. Sei $\{c_k \mid k = 1, ..., K\}$ die Menge aller **Klassen/-Zustände**, die im Stichprobenraum Ω_S unterschieden werden sollen. Offenbar bildet $\{c_k\}$ eine Partitionierung von Ω_S in ein System unvereinbarer Ereignisse, d.h. $c_i \neq c_j$ für $i \neq j$.
Es werden Datenvektoren/Merkmalsvektoren zugelassen, d.h. $x^s \in \mathbb{R}^n$ wird repräsentiert durch den Spaltenvektor $x^s = (x_1^s, x_2^s, ..., x_n^s)^T$. Offensichtlich spannt $\mathbb{R}^n$ einen n-dimensionalen Vektorraum auf, in dem x^s einen Punkt markiert. Dieser Raum heißt auch **Merkmalsraum**.
Für die Zuordnungsvorschrift bzw. den Klassenindex y^{s_k} einer Klasse c_k zu einem Merkmalsvektor x^s können unterschiedliche Kodierungen verwendet werden.

1. **Skalare Kodierung** des Klassenindexes

   $$y^{s_k} = k \quad \text{mit } k \in \mathbb{N}$$

   In manchen Fällen führt diese Konvention zu Problemen, da der Ordinalität und Metrik der natürlichen Zahlen keine entsprechende Beziehung der Klassen im Stichprobenraum gegenüber steht.

2. **Vektorielle Kodierung** des Klassenindexes
   Sei $y^s = (y_1^s, \cdots, y_K^s)^T$ der Interpretationsvektor des Merkmals x^s mit $y_k^s \in \{0, 1\}$ zur Indizierung der Klassenzugehörigkeit zu c_k. Dann gilt:

   $$c_k: \quad y_j = \begin{cases} 1 & \text{für } j = k \\ 0 & \text{sonst} \end{cases}$$

   Der Zielvektor y^s spannt eine K-dimensionale Basis für das Stichprobenelement (x^s, y_k^s) auf und $y^{s_k} = (0, 0, \cdots, 0, 1, 0, \cdots, 0)^T$ wählt hieraus den k-ten Basisvektor aus.
   Diese Kodierung hat den Vorzug, dass alle Basisvektoren numerisch völlig

gleichberechtigt sind und dass demzufolge keine unerwünschten Nachbarschaftsbeziehungen zwischen den Klassen c_k repräsentiert werden. Die **Hamming-Distanz** aller Klassen untereinander ist vom Betrag 2, weil sich jeweils zwei Klassen an zwei Positionen der Kodierung unterscheiden.

Die Menge der Stichproben wird als Realisierungsmenge eines Zufallsprozesses betrachtet. Obwohl die Wertepaare (x^s, y^{s_k}) **zufällig** sind, entstanden sie aber **nicht regellos!**

Die Regelhaftigkeiten drücken sich in den **Verteilungsdichten** der Merkmale $p(x, c_k)$ aus. Es werden zwei grundlegend verschiedene Situationen für

$$p(x, c_k) = p(x|c_k)P(c_k)$$

unterschieden, die auch zu verschiedenen Aufgabenstellungen führen:

1. $p(x, c_k)$ ist bekannt für alle $k \in \{1, ..., K\}$.
   Damit ist das durch Ω_S gegebene Problem vollständig statistisch charakterisiert. Irgend ein Merkmal $x^t \in \Omega$, $x^t \notin \Omega_S$ kann interpretiert werden, das heißt, ihm kann ein Klassenindex y^{t_k} zugeordnet werden. Vorausgesetzt sei eine gute Abdeckung der Ω durch Ω_S.

2. $p(x, c_k)$ ist nicht oder nur partiell bekannt.
   Je nach vorliegender Kenntnis über $p(x, c_k)$ lassen sich unterschiedliche Verfahren zum
   - Lernen ihrer Gestalt aus empirischen Daten oder
   - Entscheidungsfinden unter heuristischen Annahmen für a priori gegebene Dichtefunktionen

   formulieren.

Eine ideale Situation für die Klassenzuordnung eines Datums x bestünde, wenn $p(x|c_k)$ eine Delta-Funktion

$$p(x|c_k) = \begin{cases} 1 & \text{für } x \to c_k \\ 0 & \text{sonst} \end{cases}$$

wäre. Dann könnten die Merkmale eindeutig zur Klasseneinteilung führen. Die Merkmale würden selbst sehr scharf die Klassen repräsentieren. Die Realität sieht anders aus. Die Merkmale x beschreiben nur sehr unscharf und nicht eindeutig die Klassenzugehörigkeit. Dies hat folgende Ursachen:

- Variabilität der Individuen einer Population, die alle derselben Klasse zuzuordnen sind
- Unvermeidbare Meßfehler

Folglich beinhaltet jede Entscheidung ein **Risiko**, **Fehler** zu begehen. Die bedingten Dichtefunktionen $\{p(x|c_k) \mid k = 1, ..., K\}$ drücken diese Unsicherheit aus. Andererseits ist die

Generelle Aufgabe eines statistischen Klassifikators:
Bestimme die Klassenzugehörigkeit eines Merkmalsvektors unter der Randbedingung der **Minimierung des Fehlerrisikos**.

Hieraus folgt, dass, beginnend mit a priori gemachten Annahmen über die bedingten Dichten (dies ist Teil des Vorwissens über den Problembereich), aus empirischen Daten weiteres Wissen (empirisches Wissen) gewonnen wird, das zu einer Schärfung der bedingten Dichtefunktionen führt und so das Fehlentscheidungsrisiko reduziert.

3.1.3 Beispielszenarien eines Entscheidungsprozesses

Mit Hilfe von drei beispielhaften Szenarien werden unterschiedliche Situationen erfassen, die in den folgenden Abschnitten mehr im Detail behandelt werden.

Szenario 1: Ausschließliche Anwendung von a priori Wissen
Gegeben: $P(c_1), P(c_2)$ (d.h. Zweiklassen-Problem)
Annahme: $P(c_1) > P(c_2)$
Hieraus kann offensichtlich die Entscheidung e abgeleitet werden:

$$e : j = 1$$

Im Fall $P(c_1) >> P(c_2)$ ist diese Entscheidung sehr sicher. Dennoch wird aber stets ein Fehler ϵ mit der **Fehlerwahrscheinlichkeit**

$$P(\epsilon) = P(c_2)$$

zu beachten sein. Aus diesem Szenario leitet sich keine Lernaufgabe ab.

Szenario 2: Bayes-optimale Strategie
Gegeben: $P(c_k)$, $p(x|c_k)$
Gesucht: $e : j = \arg\max_k P(c_k|x)$
Aus den a priori Wahrscheinlichkeiten der Klassenzugehörigkeit $P(c_k)$ und aus den empirisch zu ermittelnden bedingten Dichten $p(x|c_k)$ wird nach der Bayes-Formel die **a posteriori Wahrscheinlichkeit** $P(c_k|x)$ berechnet. Die Entscheidung e wird zugunsten der maximalen a posteriori Wahrscheinlichkeit gefällt, siehe Abschnitt 3.2.

Szenario 3: Risikominimierende Strategie
Jede Entscheidung, auch die Bayes-optimale Entscheidung, trägt das Risiko von Fehlentscheidungen in sich. Jede Fehlentscheidung hat negative Konsequenzen bzw. Kosten zur Folge. Kennt man alle Kosten jedweder Entscheidung a priori, kann eine

risikominimierende Strategie realisiert werden. Dazu werden die a posteriori Wahrscheinlichkeiten mit den Kosten gewichtet. Seien g_{ij} die Kosten einer Entscheidung e_i, wenn tatsächlich c_j vorlag. Dann lassen sich zwei Situationen unterscheiden (siehe Abschnitt 3.3.):

a) $P(c_1|x) \approx P(c_2|x)$
Die hiermit verbundene Unsicherheit kann nur über die Entscheidungskosten aufgelöst werden.

b) $P(c_1|x) >> P(c_2|x)$
Die Entscheidung $e : j = 1$ wird unter Umständen durch die Kosten modifiziert.

3.1.4 Partitionierung des Stichprobenraumes

Es werden zwei Phasen bei der Arbeit mit einem Klassifikator unterschieden: **Trainingsphase** und **Arbeitsphase**. Sei $\Omega_S := \Omega_T \cup \Omega_A$. Dabei vermittelt Ω_T eine unter Umständen eingeschränkte Sicht auf die Eigenschaften von $\Omega \subset \Omega$. Es muß aber sichergestellt werden, dass das im Training aquirierte empirische Wissen auch auf Ω_A angewandt werden kann. Allgemein muß davon ausgegangen werden, dass $\Omega_T \cap \Omega_A << \Omega_A$.

a) **Trainingsphase** des Klassifikators
Partitionierung

$$\Omega_T = \Omega_L \cup \Omega_E$$

mit Ω_L als **Lernstichprobe** und Ω_E als **Teststichprobe**.
Faustregel:

$$|\Omega_L| \approx 0.2|\Omega_E|$$

- **Lernphase**: Trainiere den Klassifikator mit Hilfe der Lernstichprobe Ω_L so lange, bis die Entscheidung $e : x \rightarrow c_k$ mit einem Minimum an Fehlentscheidungen erfolgt.

- **Testphase (Evaluation)**: Teste mit Hilfe der Teststichprobe Ω_E die **Stabilität** der Fehlerrate des Klassifikators. Wenn der Klassifikator in der Testphase ähnlich gute Ergebnisse liefert wie in der Lernphase, hat der Klassifikator die inneren Strukturen von Ω_T aus Ω_L gelernt. Damit kann in die Arbeitsphase übergegangen werden (im Vertrauen, dass auch die Eigenschaften von Ω_A gelernt wurden). Wenn der Klassifikator in der Testphase schlechtere Leistung zeigt als in der Lernphase, setze das Lernen fort mit u.U. geänderter bzw. erweiterter Lernstichprobe.
Die Eigenschaft eines Klassifikators, stabile Entscheidungen in der Testphase zu zeigen, heißt in der Neuroinformatik **Generalisierung**. Hierunter wird verstanden, dass das System lernte, die **Invarianten** von den **Varianten** der Problemstellung zu unterscheiden.

Folgende Lernstrategien führen zu unterschiedlichen Trainingsstichproben:

1. **Überwachtes Lernen**:
   Hierbei werden die klassenbedingten Dichtefunktionen geschätzt und z.B. in einer Bayes-optimalen Entscheidungsregel angewendet. Voraussetzung ist eine **indizierte Stichprobe**:

$$\Omega_T = \left\{(x^t, y^t) \mid t \in \{1, ..., T\}\right\}$$

2. **Unüberwachtes Lernen**:
   In einer **Clusteranalyse** werden Dichtefunktionen geschätzt, ohne dass die Klassenzuordnung bekannt ist. Verwendet wird also eine **nicht-indizierte Stichprobe**:

$$\Omega_T = \left\{x^t \mid t \in \{1, ..., T\}\right\}$$

b) **Arbeitsphase** des Klassifikators
Für eine nichtindizierte Datenmenge $\Omega_A = \{x^a \mid a \in \{1, ..., A\}\}$ werden die Klassenzuordnungen y^{a_k} geschätzt.
Offensichtlich wird die in der Testphase verwendete Stichprobe ebenfalls nicht-indiziert angewendet und die Aufgabe des Klassifikators während des Tests besteht in einer Simulation der Arbeitsphase, siehe Abbildung 3.2.

3.2 Bayes-optimale Entscheidungen

In diesem Abschnitt wird eine Methode angegeben, wie Vorwissen in Form der a priori Wahrscheinlichkeiten $P(c_k)$ und klassenbedingt zugeordnete empirische Daten in Form der bedingten Dichtefunktionen $p(x|c_k)$ zu optimalen Entscheidungen führen. Die **Bayes-Formel**

$$P(c_k|x) = \frac{p(x|c_k)P(c_k)}{p(x)} \tag{3.1}$$

beschreibt, wie durch empirische Daten x mit der klassenbedingten Dichtefunktion $p(x|c_k)$, auch **Likelihood** genannt, die **a priori Wahrscheinlichkeiten** $P(c_k)$ in die (unbekannten) **a posteriori Wahrscheinlichkeiten** $P(c_k|x)$ umgewandelt werden.
Hierbei ist $p(x)$ die Dichteverteilung der Daten,

$$p(x) = \sum_{k=1}^{K} p(x, c_k) = \sum_{k=1}^{K} p(x|c_k)P(c_k)$$

also ein Normierungsfaktor, der sicherstellt, dass $\sum_{k=1}^{K} P(c_k|x) = 1$ gilt.
Die a posteriori Wahrscheinlichkeit gibt die Wahrscheinlichkeit für das Vorliegen

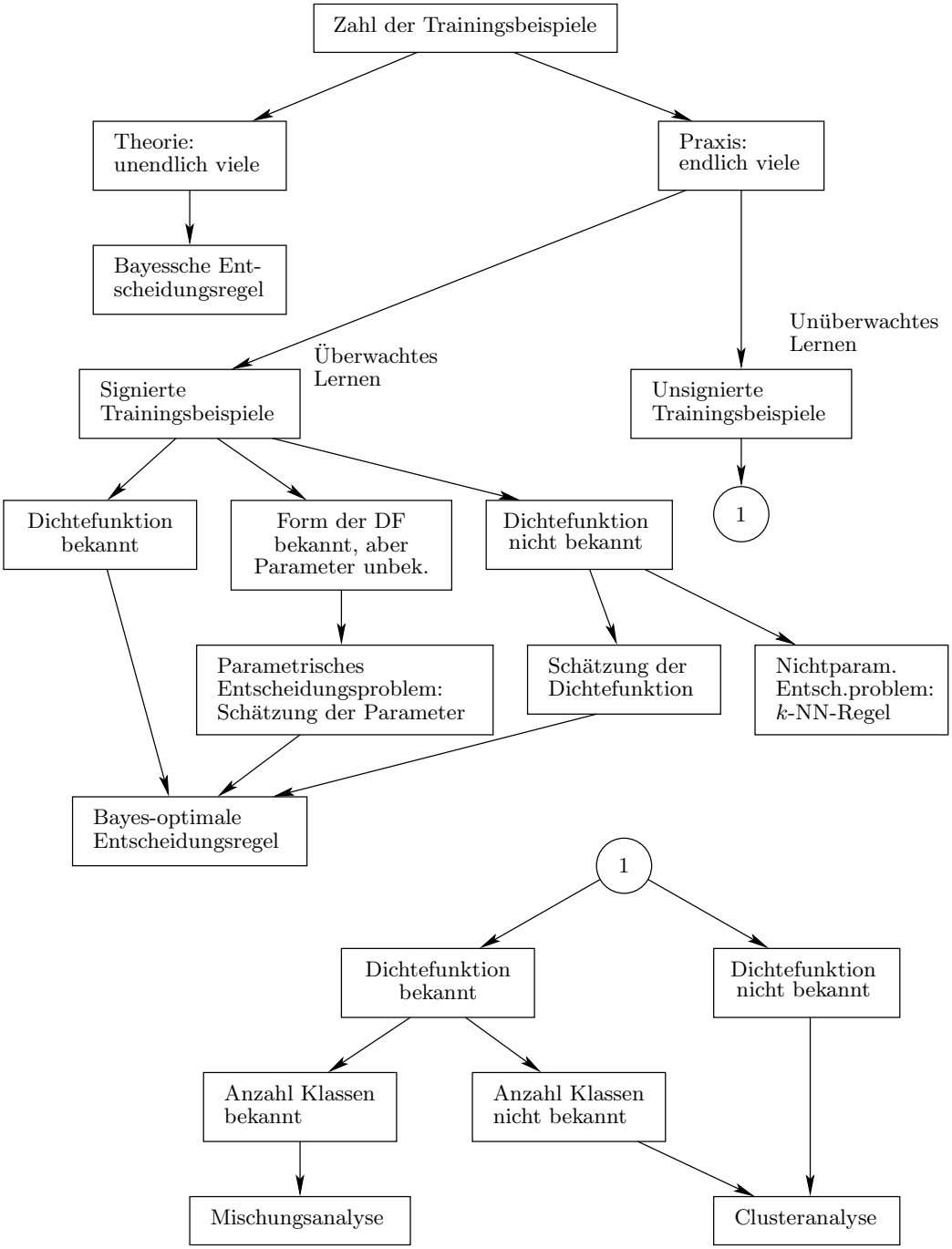


Abbildung 3.2: Synopsis der statistischen Entscheidungsfindung.

der Klasse c_k im Falle der vorliegenden Daten x an.

MAP-Strategie:
Die Beobachtung x ist derjenigen Klasse c_j zuzuordnen, deren a posteriori Wahr-

scheinlichkeit maximal ist:

$$e : j = \arg\max_k P(c_k|x)$$

Es wird gezeigt: Die MAP-Strategie minimiert die mittlere Wahrscheinlichkeit, einen Fehler bei dieser Entscheidung zu begehen.

Sei $P(\epsilon|x)$ die bedingte Wahrscheinlichkeit für die gegebenen Daten x, einen Fehler ϵ zu begehen.
Annahme: $K = 2$ ("Zwei-Klassen-Problem")

$$P(\epsilon|x) = \begin{cases} P(c_1|x), \text{ wenn}, e : x \to c_2 \\ P(c_2|x), \text{ wenn}, e : x \to c_1 \end{cases}$$

Annahme: K allgemein, Entscheidung für Klasse c_j

$$P(\epsilon|x) = \sum_{\substack{k=1 \\ k \neq j}}^{K} P(c_k|x) \; = \; 1 - P(c_j|x)$$

Die Fehlerwahrscheinlichkeit hängt also von den Daten ab.
Die **mittlere Fehlerwahrscheinlichkeit** $P(\epsilon)$ lautet:

$$P(\epsilon) = \int_{-\infty}^{\infty} P(\epsilon, x)dx = \int_{-\infty}^{\infty} P(\epsilon|x)p(x)dx$$

Wenn für jedes x die bedingte Wahrscheinlichkeit $P(\epsilon|x)$ so klein als möglich ist, muß auch $P(\epsilon)$ minimal sein.
Annahme 1: Für ein bestimmtes x^* gelte $p(x^*|c_1) = p(x^*|c_2)$
Dann liefert x^* keine Aussage über die Klassenzugehörigkeit. Nur die a priori Wahrscheinlichkeiten $P(c_1)$ und $P(c_2)$ geben einen Anhaltspunkt für die Klassenzugehörigkeit:

$$e : \begin{cases} j = 1, \text{ wenn } P(c_1) > P(c_2) \\ j = 2, \text{ wenn } P(c_1) \leq P(c_2) \end{cases}$$

Annahme 2: Gleiche a priori Wahrscheinlichkeiten
Wenn $P(c_1) = P(c_2)$, beruht die Entscheidung völlig auf den Dichtefunktionen $p(x|c_k)$. Diese Dichtefunktion nennt man auch **Likelihood** von c_k in Bezug auf x.

Maximum-Likelihood-Schätzer:
Gilt $P(c_k) = \frac{1}{K}$ für jedes $k \in \{1, ..., K\}$, so nennt man die Entscheidungsstrategie $e : j = \arg\max_k p(x|c_k)$ eine Maximum-Likelihood-Schätzung.

Tabelle 3.2 gibt eine Hierarchie von Schätzern an, die sich aus der Bayes-Formel ableiten lassen, wenn zusätzliche Annahmen gemacht werden.

Name	Ausdruck	Annahme
MAP-Schätzer (max. a posteriori)	$P(c_k\|x) \rightarrow \max$	keine
Bayes-Schätzer	$p(x\|c_k) \cdot P(c_k) \rightarrow \max$	Bayes-Formel $p(x)$ irrelevant
ML-Schätzer (max. likelihood)	$p(x\|c_k) \rightarrow \max$	wie oben und zusätzlich $\forall k = 1 \ldots K, P(c_k)$ identisch
MMSE-Schätzer (min. mean squared error)	$\ln(p(x\|c_k)) \rightarrow \max$	wie oben und zusätzlich Gaußsche klassenbedingte Dichtefunktion

Tabelle 3.3: Verschiedene Schätzer.

3.3 Risikominimierende Entscheidungen

In diesem Abschnitt soll die zusätzliche Kenntnis der **Kosten** einer Entscheidung genutzt werden, um Entscheidungen zu treffen, die das damit verbundene **Risiko** minimieren.
Seien $g_{ij} = g(e_i|c_j)$ die **Kosten einer Entscheidung** $e_i : k = i$ wenn tatsächlich die Klasse c_j vorlag. Die Entscheidungen hängen offensichtlich zunächst von den Daten ab, d.h. $e(x)$ ist eine **Funktion der Daten**. Sind außerdem die Kosten aller möglichen Entscheidungen bekannt, so können diese in der **Kostenmatrix**

$$G = (g_{ij})$$

zusammengefaßt werden.

Das **bedingte Risiko** einer Entscheidung lautet

$$r(e_i|x) = \sum_{j=1}^{K} g_{ij} P(c_j|x)$$

mit den a posteriori Wahrscheinlichkeiten $P(c_j|x)$.

Das **Gesamtrisiko** über alle möglichen Entscheidungen $e(x)$ lautet:

$$r = \int r(e(x)|x)p(x)dx$$

Wenn $e(x)$ so gewählt wird, dass das bedingte Risiko für jedes x minimal wird, so wird auch das Gesamtrisiko minimal.

Folgende Strategie liefert die besten theoretisch zu erwartenden Entscheidungsergebnisse (**Bayessche Entscheidungsregel**):

1. Berechne das bedingte Risiko $r(e_k|x)$ für alle $k \in \{1, ..., K\}$
2. Wähle die Entscheidung e_j, die für die gegebenen Daten x das bedingte Risiko minimiert: $e : j = \arg\min_k r(e_k|x)$

In der Folge werden drei Standardbeispiele betrachtet, die auch in künftigen Abschnitten immer wieder herangezogen werden:

Beispiel 1: 2-Klassen-Entscheidung
Beispiel 2: Klassifikation mit minimaler Fehlerrate
Beispiel 3: Klassifikation mit umgekehrt proportionaler Kostenfunktion

Beispiel 1: 2-Klassen-Entscheidung
Sei $g_{ij} = g(e_i|c_j)$ bekannt. Dann sind die bedingten Risiken:

$$\begin{aligned} r(e_1|x) &= g_{11}P(c_1|x) + g_{12}P(c_2|x) \\ r(e_2|x) &= g_{21}P(c_1|x) + g_{22}P(c_2|x) \end{aligned}$$

Hieraus folgt e_1, wenn $r(e_1|x) < r(e_2|x)$:

$$(g_{21} - g_{11})P(c_1|x) > (g_{12} - g_{22})P(c_2|x)$$

bzw.

$$\frac{P(c_1|x)}{P(c_2|x)} > \frac{g_{12} - g_{22}}{g_{21} - g_{11}}$$

Annahme 1: Korrekten Entscheidungen werden keine Kosten zugeordnet, d.h. $g_{11} = g_{22} = 0$.
Annahme 2: Falschen Entscheidungen werden gleiche positive Kosten zugeordnet, d.h. $g_{ij} = g_{ji} > 0$ für $i \neq j$.
Dann wird die Entscheidung durch diejenige Klasse getragen, die bei gegebenem x wahrscheinlicher ist; aus obiger Beziehung folgt $P(c_1|x) > P(c_2|x)$.
Werden Fehler teurer bewertet als richtige Entscheidungen, so folgt aus der Bayes-Formel (Ersetzen der a posteriori Wahrscheinlichkeiten):

$$\frac{p(x|c_1)}{p(x|c_2)} > \frac{g_{12} - g_{22}}{g_{21} - g_{11}} \frac{P(c_2)}{P(c_1)} \equiv T$$

Die Bayessche Entscheidungsregel kann so interpretiert werden, dass für eine Entscheidung e_1 das Likelihood-Verhältnis einen Schwellwert T überschreiten muß, der unabhängig von den Beobachtungen x ist.

Beispiel 2: Klassifikation mit minimaler Fehlerrate

$$\text{Sei definiert:} \quad e(x) \begin{cases} \text{korrekt}, & \text{wenn } e_i : c_j \quad \text{und } i = j \\ \text{fehlerhaft}, & \text{wenn } e_i : c_j \quad \text{und } i \neq j \end{cases}$$

Symmetrische Kostenfunktion: alle Fehler sind gleich teuer

$$g_{ij} = \begin{cases} 0 & i = j \\ 1 & i \neq j \end{cases} \quad \text{für alle } i, j = 1, ..., K$$

Hier fallen mögliche Kosten aus der Risokoabschätzung heraus.
Bedingtes Risiko:

$$\begin{aligned} r(e_i|x) &= \sum_{j=1}^{K} g_{ij} P(c_j|x) \\ &= \sum_{j \neq i} P(c_j|x) \\ &= 1 - P(c_i|x) \end{aligned}$$

Die Bayessche Entscheidungsregel für die Strategie der Klassifikation mit minimaler Fehlerrate wählt diejenige Entscheidung aus, die das bedingte Risiko minimiert, d.h. die a posterio Wahrscheinlichkeit maximiert.

Allgemeiner Fall: $j \in \{0, 1, ..., K\}$ mit c_0 sei die **Rückweisungsklasse**

$$g_{ij} = \begin{cases} g_R & \text{für} \quad i = 0 & \text{Rückweisung} \\ 0 & \text{für} \quad i = j, i \neq 0 & \text{korrekte Entscheidung} \\ g_F & \text{für} \quad i \neq j, i \neq 0 & \text{falsche Entscheidung} \end{cases}$$

Rückweisung kann eingesetzt werden, wenn z.B. die maximale a posteriori Wahrscheinlichkeit kleiner als eine Schwelle T_R ist, z.B. $T_R = 1 - \frac{g_R}{g_F}$.

Hieraus folgt eine Entscheidung für die Klasse c_j, wenn

$$\begin{aligned} &1. \quad P(c_j|x) = \max_k \{P(c_k|x)\} \\ &2. \quad P(c_j|x) \geq T_R. \end{aligned}$$

Mit der Rückweisungsschwelle T_R kann zwischen Fehlerrate und Rückweisungsrate balanciert werden. Unter der Annahme gleicher a priori Wahrscheinlichkeiten für alle Klassen folgt:

$$\frac{1}{K} < T_R \leq 1 \quad \text{entspricht} \quad 0 \leq \frac{g_R}{g_F} \leq \frac{K-1}{K}$$

Beispiel 3: Klassifikation mit umgekehrt proportionaler Kostenfunktion
Gegeben seien Ereignisse mit stark unterschiedlichen a priori Wahrscheinlichkeiten. Dann hätte die Strategie minimaler Fehlerrate (BSP2) zur Folge, dass häufig eintretende Ereignisse sicherer klassifiziert werden als seltene Ereignisse (wegen $P(c_k|x) \sim P(c_k)$).

Beispiel: $P(c_1) = 0.01, P(c_2) = 0.99$
Die Entscheidung e_2, alle Ereignisse der Klasse c_2 zuzuordnen, garantiert eine Fehlerrate von 1%. Fast alle Ereignisse der Klasse c_2 werden damit richtig eingeordnet, aber fast alle Ereignisse der Klasse c_1 werden gleichzeitig falsch eingeordnet, da hier die Fehlrate bei 99% liegt.

Lösung des Problems: Die Kosten der Falscherkennung seltener Ereignisse seien viel größer als die Kosten der Falscherkennung häufiger Ereignisse.

Beispiel: Screening-Strategie (siehe auch Abschnitt 3.5)
Eine positive Entscheidung bedeutet, dass eine Krankheit diagnostiziert wurde.

1. Selten auftretendes Ereignis (Krankheit) darf auf keinen Fall übersehen werden (falsch negative Entscheidung).

2. Häufig auftretendes Ereignis (Gesundheit) darf falsch klassifiziert werden (falsch positive Entscheidung).

Ansatz:
Sei $g_F(i) := \frac{g_F}{K \cdot P(c_i)}$ und definiere:

$$g_{ij} = \begin{cases} g_R & \text{für } i = 0 & \text{Rückweisung} \\ 0 & \text{für } i = j, i \neq 0 & \text{korrekte Entscheidung} \\ g_F(i) & \text{für } i \neq j, i \neq 0 & \text{falsche Entscheidung} \end{cases}$$

Damit ist die Kostenmatrix nicht mehr symmetrisch.
Hieraus folgt die Entscheidung für die Klasse c_j, wenn gilt:

$$\begin{aligned} 1. \quad p(x|c_j) &= \max_k \{p(x|c_k)\} \\ 2. \quad p(x|c_j) &> \sum_{k=1}^{K} \left(1 - \frac{g_R}{g_F} K \cdot P(c_k)\right) p(x|c_k) \end{aligned}$$

In der zweiten Bedingung ist die klassenabhängige Rückweisungsschwelle $T_R(k)$ enthalten. Dies ist ein Maximum-Likelihood-Klassifikator.

3.4 Diskriminanzfunktionen und Entscheidungsgrenzen

Ein Klassifikator repräsentiert K Diskriminanzfunktionen $d_i(x), i = 1, ..., K$, und wählt in der Arbeitsphase für ein Datum x diejenige Klasse c_j aus, für die $d_i(x)$ maximal ist:

$$e_j : x \rightarrow c_j \ , \text{ wenn } d_j(x) > d_i(x) \quad i = 1, ..., K, \ i \neq j$$

Die Wahl der Diskriminanzfunktion ist nicht eindeutig. Sie hängt u.a. von der gewählten Entscheidungsstrategie ab.

Beispiele:

- Bayes-Klassifikator (bedingtes Risiko): $d_i(x) = -r(e_i|x)$
- Beispiel 2 (minimale Fehlerrate): $d_i(x) = P(c_i|x)$

Allgemein gilt: Wenn f eine monoton wachsende Funktion ist, bleibt der Klassifikator durch die Ersetzung $d_i(x) \rightarrow f(d_i(x))$ unbeeinflußt.

Beispiele für Diskriminanzfunktionen: (minimale Fehlerrate)

$$\begin{aligned} d_i(x) &= P(c_i|x) = \frac{p(x|c_i)P(c_i)}{p(x)} \\ d_i(x) &= p(x|c_i)P(c_i) \\ d_i(x) &= \log p(x|c_i) + \log P(c_i) \end{aligned}$$

Die Entscheidungsregel unterteilt den Merkmalsraum in **Entscheidungsregionen** $\mathcal{R}_1, ..., \mathcal{R}_K$. Wenn $d_j(x) > d_i(x)$ für alle $i \neq j$, dann:

$$x \in \mathcal{R}_j \rightarrow e_j : x \rightarrow c_j$$

Die Regionen sind benachbart. Es existieren also **Entscheidungsgrenzen**, wo $d_k(x) = d_i(x)$ gilt. Im Falle des Perzeptrons sind diese Entscheidungsgrenzen Hyperebenen. Entscheidungsgrenzen und Diskriminanzfunktionen sind komplementäre Beschreibungen der Problemlösung durch einen Klassifikator.
Äquivalent zur Betrachtung einer Ungleichung $d_j(x) > d_i(x)$ für alle $i \neq j$ ist die Verwendung einer Differenz-Diskriminanzfunktion $d(x) = d_j(x) - d_i(x)$.

Beispiel 1: (2-Klassen-Problem)

$$d(x) = d_1(x) - d_2(x)$$

$$e(x) : \begin{cases} x \rightarrow c_1, & \text{wenn} \quad d(x) > 0 \\ x \rightarrow c_2, & \text{sonst} \end{cases}$$

Dies entspricht der Diskrimination mittels Schwellenelement (Stufenfunktion. bzw. Signumfunktion) beim Perzeptron.

Beispiel 2: (minimale Fehlerrate)

$$\begin{aligned} d(x) &= P(c_1|x) - P(c_2|x) \\ d(x) &= \log \frac{p(x|c_1)}{p(x|c_2)} + \log \frac{P(c_1)}{P(c_2)} \end{aligned}$$

3.5 Fehlerwahrscheinlichkeiten und Entscheidungsfindung

Dieser Abschnitt setzt sich mit der Tatsache auseinander, dass die durch einen Klassifikator begangenen **Fehler unvermeidlich** sind (im Allgemeinen). Sie müssen also zugelassen werden. Es besteht lediglich die Chance, die Auswirkungen dieser Fehler bezüglich der Aufgabenstellung zu minimieren.
Hieraus folgt, dass die gesuchte Entscheidungsgrenze das Ergebnis einer **Optimierung** ist.

3.5.1 Entscheidungsmatrix

Annahme: Beispiel 2 von Seite 111, siehe Abbildung 3.3
Die Diskriminanzfunktionen werden durch a posteriori Wahrscheinlichkeiten repräsentiert.

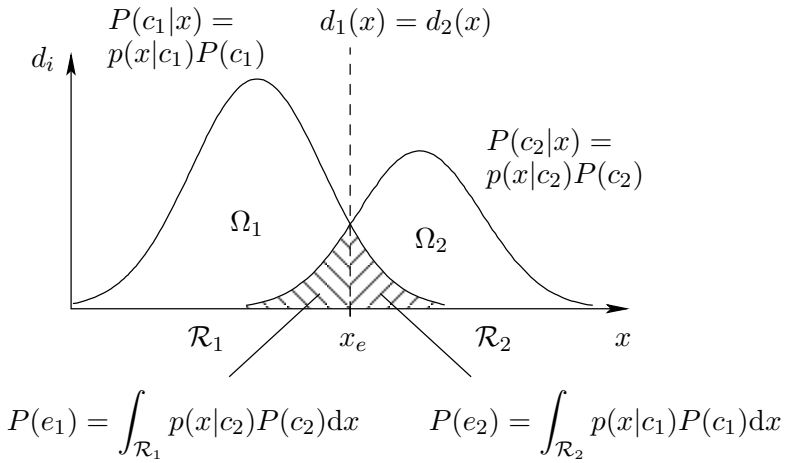


Abbildung 3.3: Diskriminanzfunktionen bei einer 2-Klassen-Trennung.

Die Bedingung $d_1(x) = d_2(x)$ definiert eine Entscheidungsgrenze bei x_e, die den Markmalsraum für x in die Regionen $\mathcal{R}_1$ und $\mathcal{R}_2$ partitioniert. Überlappen sich die beiden Diskriminanzfunktionen, so entstehen bei der Zuordnung $e_j : x \mapsto c_j$ in Abhängigkeit von der Situation $x_i \in \mathcal{R}_i$ zwei Arten von Fehlern:

a) $P(\epsilon_1) = P(x \in \mathcal{R}_2, c_1) = \int_{\mathcal{R}_2} p(x|c_1)P(c_1)dx$

b) $P(\epsilon_2) = P(x \in \mathcal{R}_1, c_2) = \int_{\mathcal{R}_1} p(x|c_2)P(c_2)dx$

Der Gesamtfehler des Systems ist also $P(\epsilon) = P(\epsilon_1) + P(\epsilon_2)$.
Je stärker die Überlappung der Diskriminanzfunktionen ist, umso größer sind diese Fehler. Es gibt zwei Strategien der optimierten Entscheidungsfindung, die den Einfluss der Fehler reduzieren.

1. Verringerung der Überlappung der Diskriminanzfunktion
   Je weiter die Verteilungen voneinander getrennt sind und je kleiner ihre Varianzen sind, umso so kleiner sind die Entscheidungsfehler. Sei

$$D = \frac{|\mu_1 - \mu_2|}{\sqrt{\frac{1}{2}(\sigma_1^2 + \sigma_2^2)}}$$

   eine Diskriminanzfunktion. Dann führt deren Maximierung zu einer geeigneten Strategie.

2. Verschieben des Gewichts zwischen beiden Fehlern
   Durch Verschiebung des Entscheidungsgrenze x_e nach kleineren oder größeren Werten werden die Fehler $P(\epsilon_1)$ und $P(\epsilon_2)$ in zueinander umgekehrtem Verhältnis verändert. Diese Strategie ist bedeutsam, wenn den Fehlerarten unterschiedliche Kosten zugerechnet werden können.

		Struktur wird erkannt	
		nein: e_1	ja: e_2
Struktur ist vorhanden	nein: c_1	$TNF = e_{11}$	$FPF = e_{21}$
	ja: c_2	$FNF = e_{12}$	$TPF = e_{22}$

Abbildung 3.4: Entscheidungstabelle.

Beispiel: Kantendetektion (siehe Abbildung 3.5)
Unter Kantendetektion versteht man das Auffinden von Grauwertkanten in einem Bild. Ein Verfahren zur Kantendetektion besteht wenigstens aus zwei Schritten (siehe Abschnitt 3.5 des CV-Skripts).

1. Anwendung eines Ableitungsoperators (LSI-Filter, siehe Abschnitt 1.4 des CV-Skripts) durch Faltung des Bildes mit dem Operator. Der Operator repräsentiert eine Maske (Template) der gesuchten Struktur. Die Maske einer Kante könnte durch eine Ableitung ihrer Bildfunktion repräsentiert werden.

2. Anwendung eines Schwellwertoperators auf die Ausgabe des LSI-Filters. Alle Filterantworten, die kleiner als die gewählte Schwelle sind, werden unterdrückt. Das Ergebnis ist ein Binärbild, wie in Abbildung 3.5.

Dieses einfache Verfahren liefert Konturbilder, die „dick“ sind. Durch Dünnen der binären Objektstrukturen (siehe Kapitel 7 des CV-Skripts) in einem 3. Schritt kann das Ergebnis verbessert werden. Dennoch liefert das Verfahren Ergebnisse,

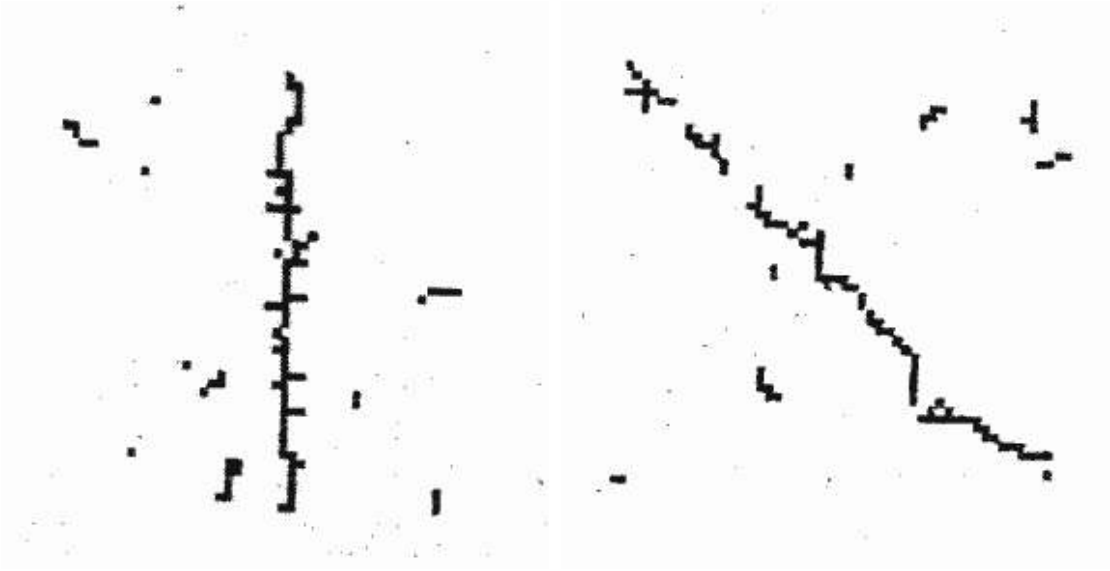


Abbildung 3.5: Konturbildbeispiele.

die sämtliche vier Klassen von Entscheidungen der Entscheidungstabelle wiederspiegeln.

Die Einträge der Entscheidungstabelle (siehe Abbildung 3.4) lassen sich zu einer Entscheidungsmatrix zusammenfassen. Sei E eine **Entscheidungsmatrix**:

$$E = (e_{ij}) = ((e_i|c_j)) = \begin{pmatrix} TNF & FPF \\ FNF & TPF \end{pmatrix}$$

Hierbei bedeuten: TNF : true negative fraction
TPF : true positive fraction
FNF : false negative fraction
FPF : false positive fraction

Flächennormierung der Funktionen $P(c_1|x)$ und $P(c_2|x)$:

$$\begin{aligned} TPF + FNF &= 1 \quad \text{für } x \in \Omega_+ \\ TNF + FPF &= 1 \quad \text{für } x \in \Omega_- \end{aligned}$$

3.5.2 ROC-Kurven und Neyman-Pearson-Strategie

Ein wichtiges Hilfsmittel zur Bewertung eines Klassifikators für ein 2-Klassen-Problem sind die sogenannten ROC-Kurven, siehe Abbildung 3.6. ROC steht für Receiver Operating Characteristics. ROC-Kurven stellen den Anteil richtig positiver Entscheidungen (TPF) gegen den Anteil falsch positiver Entscheidungen (FPF) dar. In Abbildung 3.6 sind verschiedene ROC-Kurven eingetragen. Die

Form einer ROC-Kurve gibt die Qualität eines Klassifikators bzw. Detektors an. Die Kurve 1 steht für eine Strategie des Ratens. Kurve 2 zeigt eine Leistung, die schlechter ist als ein Zufallsergebnis (Kurve 1). Kurve 3 spiegelt etwa einen realistischen Klassifikator bzw. Detektor wieder, und Kurve 4 den nahezu idealen Verlauf einer ROC-Kurve, bei welcher für ein sehr kleines FPF ein sehr hohes TPF erreicht wird.
Es ist das Ziel, einen Klassifikator zu entwickeln, der einen maximalen Abstand D' zur Diagonalen (Kurve 1) besitzt. Dieser Abstand korreliert mit dem oben genannten Diskriminanzkriterium D, denn offensichtlich gilt für $|D'| > 0$, dass $TPF > FPF$ und $TNF > FNF$.

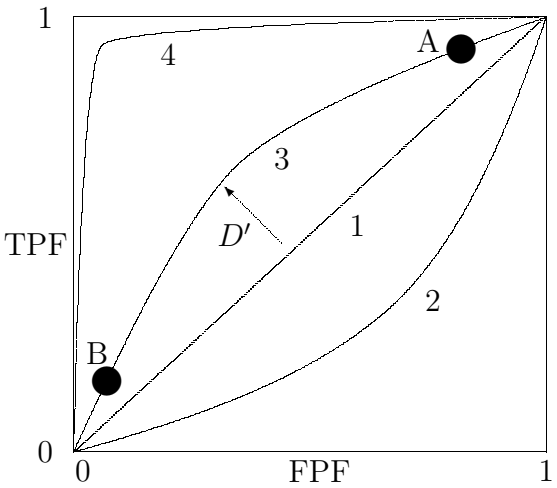


Abbildung 3.6: ROC-Kurven.

Neyman-Pearson-Strategie (Neyman und Pearson (1933)):
Wähle einen akzeptablen (niedrigen) Wert für FPF und maximiere TPF. Dadurch wird die ROC-Kurve in Richtung des Kurventyps 4 aus Abbildung 3.6 getrieben. Wie dies erreicht wird, hat nicht notwendigerweise etwas mit Statistik zu tun, sondern betrifft die Aufbereitung der Daten (Vorverarbeitung), die einer statistischen Analyse oder Interpretation zugeführt werden kann.
Zum Beispiel könnte im oben beschriebenen Verfahren zur Kantendetektion ein kontextsensitiver, d.h. ein adaptiver Schwellwertoperator, verwendet werden.

Die Verfolgung der Neyman-Pearson-Strategie liefert also Verteilungsfunktionen, die eine bestimmte Gestalt und Distanz zueinander haben. Ausgehend von dieser Situation besteht nun noch die Möglichkeit, den Arbeitspunkt des Klassifikators bzw. Detektors auf einer ROC-Kurve festzulegen. Die Verschiebung des Arbeitspunkts auf der ROC-Kurve entspricht der Verschiebung der Balance zwischen den Fehlern $P(\epsilon_1) = FPF$ und $P(\epsilon_2) = FNF$. In Abhängigkeit von den Kosten dieser Fehler für die konkrete Aufgabe ist der entsprechende Arbeitspunkt zu finden. Der Arbeitspunkt A auf der ROC-Kurve 3 in Abbildung 3.6 entspricht einer

eher liberalen Entscheidung, weil mehr falsch positive Entscheidungen zugelassen werden. Man erreicht dies durch eine Verschiebung der Entscheidungsschwelle x_e zu niedrigeren Werten.
Der Arbeitspunkt B auf derselben ROC-Kurve entspricht einer eher konservativen Entscheidung, weil weniger falsch positive Entscheidungen zugelassen werden. Man erreicht dies durch eine Verschiebung der Entscheidungsschwelle x_e zu höheren Werten.
Folgende Strategien sind in der Praxis üblich.

Sensitivität: $TPF = 1 - FNF$ (Arbeitspunkt A)
Strategie: Maximierung der Sensitivität bedeutet **Minimierung der falsch negativen Entscheidungen**. Das bedeutet die Minimierung des Anteiles nicht erkannter aber vorhandener Befunde. Diese Strategie entspricht dem Vorgehen bei einem Screening-Test in der Medizin.

Spezifität: $TNF = 1 - FPF$ (Arbeitspunkt B)
Strategie: Maximierung der Spezifität bedeutet **Minimierung der falsch positiven Entscheidungen**. Das bedeutet die Minimierung des Anteils fehlerhaft festgestellter (tatsächlich nicht vorhandener) positiver Befunde. Diese Strategie entspricht dem Vorgehen bei einer Differentialdiagnose in der Medizin.

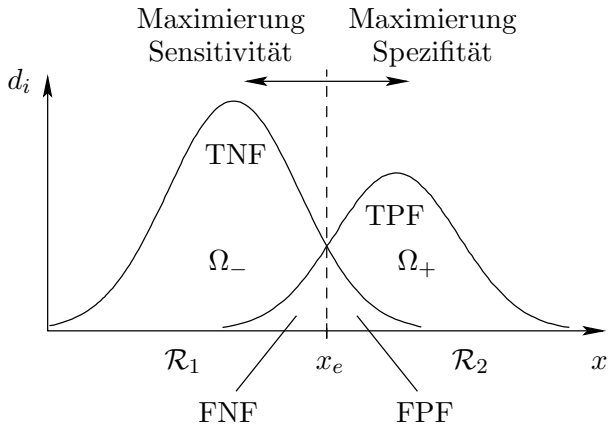


Abbildung 3.7: Trade-off zwischen Sensitivität und Spezifität.

Abbildung 3.7 veranschaulicht die partielle Komplementarität der Strategien von Sensitivität und Spezifität.

Merke: Im Fall von mehr als 2 Klassen ($K > 2$) existieren mehr Möglichkeiten, falsch zu entscheiden, als korrekt zu entscheiden.

Die Wahrscheinlichkeit für eine korrekte Entscheidung lautet:

$$\begin{aligned} P(\text{TRUE}) &= \sum_{i=1}^{K} P(x \in \mathcal{R}_i, c_i) \\ &= \sum_{i=1}^{K} P(x \in \mathcal{R}_i | c_i) P(c_i) \\ &= \sum_{i=1}^{K} \int_{\mathcal{R}_i} p(x|c_i) P(c_i) dx \end{aligned}$$

Die Bayes-Strategie maximiert diese Wahrscheinlichkeit gegenüber allen anderen Strategien.

3.5.3 Spezielle Probleme

Hier sollen noch zwei weitere für den Entwurf von Klassifikatorn bzw. Detektoren interessante Sachverhalte geklärt werden, siehe auch Daugman (2000).

3.5.3.1 Kombinationen von Entscheidungen

Es sei die Kombination zweier Systeme angenommen, von denen jedes Entscheidungen mit den in Abschnitt 3.5.1 angeführten Fehlern $P(\epsilon_1)$ und $P(\epsilon_2)$ trifft. Wie sehen die Fehler des kombinierten Systems aus?
Solche Probleme sind z.B. auch dort anzutreffen, wo neuronale Netze als Experten für gewisse Aspekte des Problems kombiniert werden.
Zwei intuitive Annahmen über das Resultat, die in sich jedoch widersprüchlich sind, können gemacht werden:

- Die Kombination unterschiedlicher Verfahren müsste das Ergebnis verbessern, da mehr Information besser ist als weniger.

- Sind die Verfahren von unterschiedlicher Leistungsfähigkeit, müsste die Leistung ihrer Kombination zwischen derjenigen der einzelnen Verfahren liegen.

Beide Intuitionen sind teilweise wahr: In Abhängigkeit der Kombinationsart der Verfahren wird eine der Fehlerraten (FPF oder FNF) besser als die beste der einzelnen Verfahren, während die andere Fehlerart schlechter wird als die schlechteste der einzelnen Verfahren.

Seien die einzelnen Verfahren mit dem Label 1 bzw. 2 gekennzeichnet. Dann sind zwei Methoden der Verknüpfung der statistischen Situationen zu einer Entscheidung denkbar:

A) Disjunktive Verknüpfung: Entscheidung für die Klasse c_2, wenn entweder Verfahren 1 oder Verfahren 2 diese Entscheidung unterstützt.

B) Konjunktive Verknüpfung: Entscheidung für die Klasse c_2, nur wenn beide Verfahren diese Entscheidung unterstützen.

Demnach sind folgende Fehlerraten zu unterscheiden:

$$FPF_1, FPF_2, FPF_D, FPF_K$$

$$FNF_1, FNF_2, FNF_D, FNF_K$$

Regel A: Disjunktion
Eine falsch negative Entscheidung der Kombination tritt nur ein, wenn sowohl Verfahren 1 als auch Verfahren 2 falsch negativ entscheiden.

$$FNF_D = FNF_1 \, FNF_2$$

Also gilt $FNF_D < FNF_1$ und $FNF_D < FNF_2$.
Eine falsch positive Entscheidung der Kombination entspricht dem Komplement der Wahrscheinlichkeit, dass weder Verfahren 1 noch Verfahren 2 falsch positiv entscheiden.

$$\begin{aligned} FPF_D &= 1 - (1 - FPF_1)(1 - FPF_2) \\ &= FPF_1 + FPF_2 - FPF_1 \, FPF_2 \end{aligned}$$

Also ist die falsch positive Rate von Entscheidungen bei disjunktiver Verknüpfung der Verfahren höher als jede falsch positive Rate der Einzelentscheidungen.

Regel B: Konjunktion
Eine falsch positive Entscheidung der Kombination tritt nur ein, wenn beide Verfahren eine falsch positive Entscheidung treffen.

$$FPF_K = FPF_1 \, FPF_2$$

Folglich reduziert sich die Wahrscheinlichkeit für falsch positive Entscheidungen. Für jede falsch negative Entscheidung des kombinierten Verfahrens gilt:

$$\begin{aligned} FNF_K &= 1 - (1 - FNF_1)(1 - FNF_2) \\ &= FNF_1 + FNF_2 - FNF_1 \, FNF_2 \end{aligned}$$

Also ist die falsch negative Rate von Entscheidungen bei konjunktiver Verknüpfung der Verfahren höher als jede falsch negative Rate der Einzelentscheidungen.

Beispiel:

	$P(\epsilon_1) = P(\epsilon_2)$			$P(\epsilon_1) \neq P(\epsilon_2)$		
	FPF	FNF	$P(\epsilon)$	FPF	FNF	$P(\epsilon)$
starkes Verfahren	0.001	0.001	0.002	0.001	0.001	0.002
schwaches Verfahren	0.01	0.01	0.02	0.002	0.002	
disjunktive Verknüpfung	0.01099	0.00001	0.011	0.002	0.000001	0.002
konjunktive Verknüpfung	0.00001	0.01099	0.011	0.000001	0.002	0.002

Schlussfolgerungen des Beispiels:

1. Wenn für beide Verfahren $P(\epsilon_1) = P(\epsilon_2)$ gilt, d.h. der Arbeitspunkt x_e liegt im Gleichgewicht bei $d_1(x) = d_2(x)$ für den Fall gleicher Diskriminanzfunktionen für Ω_- und Ω_+, dann ist die Verwendung eines einzelnen starken Verfahrens (D groß) besser als dessen Kombination mit einem leistungsschwachen Verfahren (D klein).

2. Wenn die disjunktive Verknüpfung gewählt wird, sollte der Arbeitspunkt x_e des schwachen Verfahrens so verschoben werden, dass FPF_{schwach} höchstens der Gleichgewichtsfehlerrate $P_{stark}(\epsilon)$ des starken Verfahrens entspricht. Dann gilt auch $FPF_D \approx FPF_{schwach} \approx P_{stark}(\epsilon)$.

3. Wenn die konjunktive Verknüpfung gewählt wird, sollte der Arbeitspunkt x_e des schwachen Verfahrens so verschoben werden, dass FNF_{schwach} höchstens der Gleichgewichtsfehlerrate $P_{stark}(\epsilon)$ des starken Verfahrens entspricht. Dann gilt auch $FNF_K \approx FNF_{schwach} \approx P_{stark}(\epsilon)$.

3.5.3.2 Identifikation und Verifikation

In diesem Abschnitt sollen die bei Identifikation und Verifikation (siehe Abschnitt 3.1) zu beachtenden statistischen Sachverhalte skizziert werden.
Zur Erinnerung:

> **Verifikation** ist ein Vergleich der Daten mit einem einzigen Template (prototypische Repräsentation einer Klasse).
> Beispiel: Suche eines bestimmten Musters in einer Stichprobe.
>
> **Identifikation** ist ein Vergleich der Daten mit einer Menge von Templates.
> Beispiel: Zuordnung eines Elements der Stichprobe zu einem Referenzmuster aus einer Menge von Templates.

Im Fall der Identifikation ist die Wahrscheinlichkeit für eine falsche Rückweisung (false reject) eines speziellen Templates, FNF, die gleiche wie im Fall der Verifikation. Hier werden nun die Forderungen an die Wahrscheinlichkeit einer einzelnen falschen Zuweisung (false accept, FPF) betrachtet, wenn ein Datum mit insgesamt N Templates verglichen werden muss.

Sei N die Kardinalität der Menge von Templates. Sei P_1 die Wahrscheinlichkeit einer falschen Zuweisung (FPF) bei Verifikation (ein Template) und sei P_N die Wahrscheinlichkeit für eine falsche Zuweisung bei Identifikation (N Tempates). Bei jedem Vergleich mit einem Template ist die Wahrscheinlichkeit, nicht eine falsche Zuweisung zu treffen, $1 - P_1$. Da N unabhängige Vergleiche bei der Identifikation erforderlich sind, ist die Wahrscheinlichkeit, in N Vergleichen keine falsche Zuweisung zu haben, $(1 - P_1)^N$. Also ist die Wahrscheinlichkeit, unter N Vergleichen wenigstens eine falsche Zuweisung zu beobachten, gegeben durch $P_N = 1-(1-P_1)^N$.

Beispiel:
Ein biometrischer Verifikator erreiche $TNF = 99.9\%$ richtige Rückweisungen bei $1:1$-Vergleichen; also $P_1 = 0.001$. Wie wird dieser Verifikator bei einer Identifikationsaufgabe von N unabhängigen Templates operieren?

Größe der Datendank N	Wahrscheinlichkeit einer falschen Zuweisung P_N
200	18%
2000	86%
10000	99.995%

Bereits bei etwa 7000 Templates übertrifft die Wahrscheinlichkeit einer falschen Zuweisung ($P_N = 99.91\%$) in einem Identifikationsversuch die Wahrscheinlichkeit für eine richtige Rückweisung ($TNF = 99.9\%$) in einem Verifikationsversuch.
Identifikation stellt also höhere Anforderungen an ein Mustererkennungsproblem als Verifikation. Bedenkt man, wie begrenzt die Güte biometrischer Klassifikationsverfahren ist, leitet sich aus dieser Einsicht ein Horrorszenario für biometrische Identifikation (mit Millionen von Templates bei Grenzkontrollen) ab.

Aus der Näherung $P_N \approx NP_1$ für kleine $P_1 << \frac{1}{N} << 1$ ist erkennbar, dass ein Identifikator für eine Datenbasis von N Templates ungefähr N-Mal genauer klassifizieren muss, als ein Verifikator, um vergleichbare Ergebnisse bzgl. falscher Zuweisungen zu erreichen. Der einzige Ausweg zur Beherrschung dieses Problems ist, die Wahrscheinlichkeit P_1 für einzelne falsche Zuweisungen signifikant kleiner als $\frac{1}{N}$ zu machen.

Beispiel:
Für $N = 10^7$ leistet $P_1 = 10^{-9}$ eine Sicherheit von nur 99%, nicht fälschlicherweise mit einer der Referenzen identifiziert (falsch identifiziert) zu werden.

3.6 Normalverteilung der Daten

Sehr häufig nimmt man eine Normalverteilung der klassenbedingten Dichtefunktionen $p(x|c_i)$ an. Hierfür gibt es mehrere Gründe:

- Die Normalverteilung ist analytisch gut handhabbar.
- Die Normalverteilung ist ein wichtiges Modell für den Fall, dass der **Prototypvektor** μ_i als Repräsentant der Klasse c_i Störungen unterworfen ist, die zu einer Streuung in einem Intervall führen. Das entspricht der Intuition:
  1. Merkmale unterscheiden sich erheblich für verschiedene Klassen bei großen **Interklassendistanzen**: $|\mu_i - \mu_k| \rightarrow \max$
  2. Merkmale unterscheiden sich nur wenig innerhalb einer Klasse bei kleinen **Intraklassendistanzen**: $|x - \mu_i| \rightarrow \min$

  Hieraus folgt das Optimierungsziel:

  $$|\mu_i - \mu_k| >> |x - \mu_i| \quad \text{für} \quad e(x) : x \rightarrow c_i$$

  Siehe Realisierung mittels Neyman-Pearson-Strategie, Abschnitt 3.5.2, oder Fisher's Diskriminanzanalyse, Abschnitt 3.10.2.

- Unterschiedliche Fehlereinflüsse bewirken nach dem zentralen Grenzwertsatz der Wahrscheinlichkeitsrechnung eine Gaußverteilung (Normalverteilung) der Merkmale um den Repräsentanten (Mittelwert der Daten).

Im Folgenden werden einige interessante Eigenschaften der Normalverteilung angeben, die als Ergänzung zu den Eigenschaften der (mehrdimensionalen) Gaußfunktion aus dem Skript Computer Vision I betrachtet werden können.

3.6.1 Univariate Normalverteilung

Es seien $x, \mu, \sigma \in \mathbb{R}$.
Die univariate[0] Dichtefunktion der Daten x nach Flächennormierung lautet:

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right)$$

mit erstem gewöhnlichen Moment (**Mittelwert**):

$$\mathrm{E}\{X\} = \int_{-\infty}^{\infty} x p(x) dx = \mu$$

[0]Funktionsbezeichnungen ($m, n \in \mathbb{N}$):

skalar und univariat	:	$\mathbb{R}^1 \mapsto \mathbb{R}^1$	vektoriell und univariat	:	$\mathbb{R}^1 \mapsto \mathbb{R}^m$
skalar und multivariat	:	$\mathbb{R}^n \mapsto \mathbb{R}^1$	vektoriell und multivariat	:	$\mathbb{R}^n \mapsto \mathbb{R}^m$

und zweitem zentralen Moment (**Varianz**):

$$\mathrm{E}\left\{(X-\mu)^2\right\} = \int_{-\infty}^{\infty}(x-\mu)^2 p(x)dx = \sigma^2$$

Ist $p(x)$ normalverteilt, dann ist die Dichtefunktion vollständig durch die beiden Parameter μ, σ^2 bestimmt und man schreibt[1]:

$$p(x) \sim N(\mu, \sigma^2)$$

Man bezeichnet σ auch als Standardabweichung, als Maß der Breite der Normalverteilung. Bei $|x-\mu| = \sigma$ ist die (auf $p(\mu) = 1$ normierte) Normalverteilung auf eine Höhe von $e^{-\frac{1}{2}} = 0.606$ abgefallen. Außerdem liegen 95% einer normalverteilten Population von Daten innerhalb $|x-\mu| = 2\sigma$.

3.6.2 Multivariate Normalverteilung

Es seien $x, \mu \in \mathbb{R}^n$ und es sei $\Sigma \in \mathbb{R}^{n\times n}$.
Die multivariate[1] Dichtefunktion der Daten x nach Flächennormierung lautet[2]:

$$p(x) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$

Notation:

$$p(x) \sim N(\mu, \Sigma)$$

mit **Mittelwertvektor**:

$$\mu = \mathrm{E}\{X\} \qquad , \quad \mu_i = \mathrm{E}\{X_i\}$$

und **Kovarianzmatrix**:

$$\Sigma_{n\times n} = \mathrm{E}\left\{(X-\mu)(X-\mu)^T\right\} = (\sigma_{ij})$$

sowie **Kovarianz** (Cov):

$$\sigma_{ij} = \mathrm{E}\left\{(X_i-\mu_i)(X_j-\mu_j)^T\right\}$$

und **Varianz** (Var):

$$\sigma_{ii} = \mathrm{E}\left\{(X_i-\mu_i)^2\right\}$$

Eigenschaften der Kovarianzmatrix Σ:

- symmetrisch ($\Sigma = \Sigma^T$)

[1] Nur bei der univariaten Normalverteilung wird historisch bedingt die Varianz σ^2 als Parameter verwendet, bei anderen Dichtefunktionen wird die Standardabweichung σ notiert.
[2] Es wird notiert: $|\Sigma| = \det\Sigma$

- positiv semi-definit ($\sigma_{ij} \geq 0$)
  Eine gängige Annahme ist, dass Σ positiv definit ($|\Sigma| > 0$, streng positiv) ist, so dass Entscheidungen nicht „entarten“ können.

- Seien $i \neq j$, dann heißen x_i und x_j **statistisch unabhängig**, falls $\sigma_{ij} = 0$ gilt:

$$\Sigma = \begin{pmatrix} \sigma_{11} & 0 & \dots & 0 \\ 0 & \sigma_{22} & & \vdots \\ \vdots & & \ddots & \\ 0 & \dots & & \sigma_{nn} \end{pmatrix} \sim p(x) = p(x_1)p(x_2)\dots p(x_n)$$

3.6.3 Transformationen normalverteilter Daten

Satz 3.1 *Sei $z = Ax + y$ eine lineare Abbildung der Daten $x \in \mathbb{R}^n$ und $A \in \mathbb{R}^{n \times n}$. Dann gilt unabhängig von der Verteilung von x (aufgrund der Linearität des Erwartungswertoperators):*

$$\begin{aligned} \mathrm{E}\{Ax + y\} &= A(\mathrm{E}\{x\}) + y \\ \mathrm{Cov}\ \{Ax + y\} &= A(\mathrm{Cov}\{x\})A^T \end{aligned}$$

Satz 3.2 *Die Verteilung einer Linearkombination unabhängiger normalverteilter stochastischer Variabler ist wieder normalverteilt. Für $p(x) \sim N(\mu, \Sigma)$ folgt mit Satz 3.1:*

$$\begin{aligned} p(x) \sim N(\mu, \Sigma) \rightarrow &\quad p(z) \sim N(A\mu + y, A\Sigma A^T) \\ p(x) \sim N(\mu, \Sigma) \rightarrow &\quad p\left(\Sigma^{-1/2}(x - \mu)\right) \sim N(\mathbf{0}, I) \\ &\quad \mathbf{0}: \text{ Nullvektor }, \ I: \text{ Identitätsmatrix} \\ p(x) \sim N(\mu, \Sigma) \rightarrow &\quad p\left((x - \mu)^T\Sigma^{-1}(x - \mu)\right) \sim \chi^2_n \\ &\quad \chi^2 \sim \text{Chi-Quadrat-Verteilung mit } n \text{ Freiheitsgraden} \end{aligned}$$

Satz 3.3 *Die Multiplikation zweier unabhängiger Normalverteilungen ergibt wieder eine (nicht normierte) Normalverteilung:*

$$N(a, A) \cdot N(b, B) \simeq N(c, C) \text{ mit}$$

$$\begin{aligned} C &= (A^{-1} + B^{-1})^{-1} \\ c &= CA^{-1}a + CB^{-1}b \end{aligned}$$

3.6.4 Marginale und bedingte Verteilungen (Randverteilungen)

Sei der Vektor $z = (x^T, y^T)^T$ normalverteilt

$$p\left(z = \begin{pmatrix} x \\ y \end{pmatrix}\right) \sim N\left(\begin{pmatrix} a \\ b \end{pmatrix}, \begin{pmatrix} A & C \\ C^T & B \end{pmatrix}\right)$$

mit A, B die **Autokovarianzmatrizen** von x bzw. y und a, b deren Mittelwerte. C ist die (nicht-symmetrische) **Kreuzkovarianzmatrix** zwischen x und y.
Dann sind die **marginalen Normalverteilungen**:

$$p(x) \sim N(a, A) \text{ und}$$

$$p(y) \sim N(b, B)$$

sowie die **bedingten Normalverteilungen**:

$$p(x|y) \sim N(a + CB^{-1}(y - b), A - CB^{-1}C^T) \text{ und}$$

$$p(y|x) \sim N(b + C^T A^{-1}(x - a), B - C^T A^{-1}C)$$

3.6.5 Projektion der Daten

Sei $y = a^T xa$ eine Projektion von x auf den Einheitsvektor a mit $\|y\| = a^T x$.
Ist Σ die Autokovarianz von x, so gibt $a^T \Sigma a$ die Autokovarianz der projizierten Daten an.
Bei **Fisher's linearer Diskriminanzanalyse** (siehe Kapitel 3.10.2) wird ein mehrdimensionaler Merkmalsvektor auf ein zu berechnendes 1-dimensionales Merkmal projiziert.

3.6.6 Eigenwertdarstellung von Σ

Die multidimensionale Normalverteilung ist vollständig bestimmt durch $n + \frac{n(n+1)}{2}$ unabhängige Parameter, wobei der erste Anteil auf μ und der zweite Anteil auf Σ zurückzuführen ist.
Dabei beschreibt μ die Schwerpunktlage des Datenclusters und Σ beschreibt dessen Gestalt. Bildet man die **Isodensiten** der Normalverteilung, so wird deren Ortskurve durch die **Hyperellipsoide**

$$(x - \mu)^T \Sigma^{-1}(x - \mu) = \text{const}$$

bestimmt. Dabei sind die **Eigenvektoren** von Σ die Hauptachsen der Hyperellipsoide und die **Eigenwerte** von Σ geben die (inversen) Längen der Hauptachsen an. Die **Determinante** von Σ gibt die Fläche des durch die Matrixzeilen ausgespannten **Parallelotops** an.

3.6.7 Mahalanobis-Distanz

Die Mahalanobis-Distanz liefert eine an die Struktur des Datenclusters angepaßte Metrik des Merkmalsraumes. Die Mahalanobis-Distanz d_M zwischen dem Datum x und dem Repräsentanten des Clusters μ,

$$d_M := \left((x-\mu)^T \Sigma^{-1} (x-\mu)\right)^{1/2}$$

hat zur Folge, dass die Isodensiten der Daten konstanten Abstand zu μ haben. Die Mahalanobis-Distanz nutzt eine Σ^{-1}-Norm. Für $\Sigma = I$ fällt die Mahalanobisdistanz mit der Euklidischen Distanz zusammen.
Sei V_n das Volumen der n-dimensionalen Einheits-Hyperkugel im Falle $\Sigma = I$, so ist das Volumen der Hyperellipsoide

$$V = V_n |\Sigma|^{1/2} d_M^n$$

ein Maß für die Streuung der Daten um den Mittelwert μ.
Für gegebenes n variiert diese Streuung also proportional $|\Sigma|^{1/2}$.

3.7 Diskriminanzfunktionen für Normalverteilungs-Dichten

Für den wichtigen Fall von Normalverteilungs-Dichten werden die Diskriminanzfunktionen angeben, wobei drei Situationen unterschieden werden. Für die Klassifikation mit minimaler Fehlerrate (siehe Beispiel 2 auf Seite 111), d.h. alle Entscheidungen sind gleich teuer, wurde die Diskriminanzfunktion

$$d_i(x) = \log\ p(x|c_i) + \log\ P(c_i)$$

angegeben. Unter der Annahme der multivariaten Normalverteilung

$$p(x|c_i) \sim N(\mu_i, \Sigma_i)$$

läßt sich die Diskriminanzfunktion in vier additive Terme aufspalten:

$$d_i(x) = -\frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1} (x-\mu_i) - \frac{n}{2}\log 2\pi - \frac{1}{2}\log|\Sigma_i| + \log P(c_i)$$

Es wird sich herausstellen, dass der erste und der vierte Term wichtig sind. In der Folge werden die Konsequenzen für unterschiedliche Annahmen bzgl. der Kovarianz untersucht.

Fall 1: $\forall i : \Sigma_i = \sigma^2 I$
Die Merkmale sind statistisch unabhängig und haben gleiche Varianz. Das Problem zerfällt also in n univariate und gleiche Normalverteilungen. Dies ist der einfachste,

aber auch gleichzeitig am wenigsten realistische Fall. Die Cluster bilden gleich große Hyperkugeln um μ_i. Das heißt, die Euklidische Norm ist anwendbar. Es gilt:

$$|\Sigma_i| = \sigma^{2n} \quad , \; \Sigma_i^{-1} = \frac{1}{\sigma^2} I$$
$$d_i(x) = -\frac{\|x - \mu_i\|^2}{2\sigma^2} + \log P(c_i)$$

Da die Terme (2) und (3) für alle i gleich sind, können sie für die Distanzfunktion unberücksichtigt bleiben.
Es werden zwei Situationen unterschieden, um weitere Aussagen über die Distanzfunktion machen zu können.

A) $P(c_i) = \text{const}$ für alle i
In diesem Fall wird x dem nächsten Repräsentanten μ_i zugeordnet.
Dies entspricht dem **Minimum-Distanz-Klassifikator**, auch **NN (Nächster Nachbar)-Klassifikator** genannt.
Wenn μ_i als idealer Prototyp von x angesehen werden kann, ist der NN-Klassifikator mit einem **Template-Matching-Filter** vergleichbar, d.h. $d_i = \max$ für $x = \mu_i$.

B) $P(c_i) \neq P(c_j)$ für alle $i \neq j$
Wenn $\|x - \mu_i\| = \|x - \mu_j\|$, wird der a priori wahrscheinlichsten Klasse der Vorzug gegeben. Andernfalls, wegen

$$\|x - \mu_i\|^2 = x^T x - 2\mu_i^T x + \mu_i^T \mu_i$$

folgt:

$$d_i(x) = -\frac{1}{2\sigma^2}(x^T x - 2\mu_i^T x + \mu_i^T \mu_i) + \log P(c_i)$$

Der Term $x^T x$ ist für alle i gleich und trägt nicht zur Klassenunterscheidung bei. Man erhält die **lineare Diskriminanzfunktion**:

$$d_i(x) = w_i^T x + w_{i0} \quad \text{mit } w_i = \frac{\mu_i}{\sigma^2} \text{ und } w_{i0} = -\frac{1}{2\sigma^2}\mu_i^T \mu_i + \log P(c_i)$$

Eine **lineare Maschine** ist ein Klassifikator mit linearer Diskriminanzfunktion. Das Konzept ist aber nicht auf den Fall $\Sigma \approx I$ beschränkt.

Die Entscheidungsgrenzen sind **stückweise Hyperebenen**, die durch lineare Gleichungen der Art $d_i(x) = d_j(x)$ definiert werden.

Annahme: $x \in \mathbb{R}^2$, $\mathcal{R}_i$ benachbart $\mathcal{R}_j$
Dann beschreibt $w^T(x - x_0) = 0$ eine Gerade durch den Punkt x_0 und orthogonal zu w:

$$w = \mu_i - \mu_j$$

$$x_0 = \frac{1}{2}(\mu_i + \mu_j) - \frac{\sigma^2}{\|\mu_i - \mu_j\|^2} \log \frac{P(c_i)}{P(c_j)}(\mu_i - \mu_j)$$

Im Fall A ($P(c_i) = P(c_j)$) liegt x_0 auf halbem Weg zwischen μ_i und μ_j auf w. Es liegt der **Minimum-Distanz-Klassifikator** vor (siehe Abbildung 3.8).

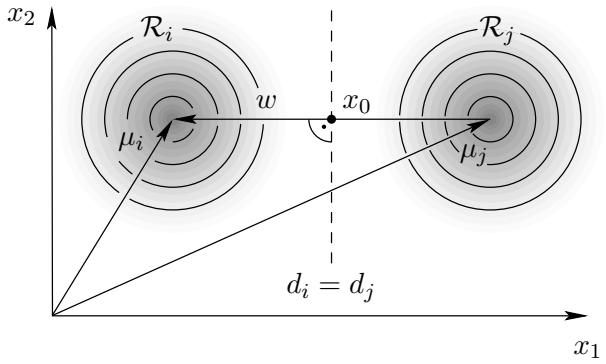


Abbildung 3.8: Veranschaulichung des Minimum-Distanz-Klassifikators.

Der Merkmalsraum wird durch **Voronoi-Zerlegung** in Zellen aufgeteilt, deren Grenzen durch die Punkte $x_{0,i,j}$ gehen (siehe Abbildung 3.9).

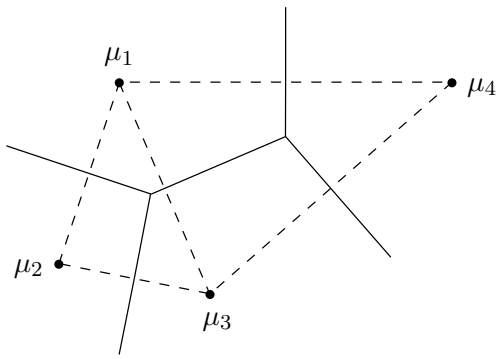


Abbildung 3.9: Veranschaulichung der Voroni-Zerlegung eines Merkmalsraums.

Im Fall B ($P(c_i) \neq P(c_j)$) liegt x_0 weiter entfernt vom a priori wahrscheinlicheren Mittelwert. Ist aber $\sigma^2 << \|\mu_i - \mu_j\|^2$, d.h. der Intraklassenabstand ist klein im Vergleich zum Interklassenabstand, dann wird die Lage der Entscheidungsgrenze wenig durch die a priori Wahrscheinlichkeiten bestimmt.

Fall 2: $\forall i : \Sigma_i = \Sigma$
Die Kovarianzen aller Klassen werden als gleich angenommen. Die Äquidensiten bilden deshalb Hyperellipsoide gleicher Größe, Form und Orientierung.
Die Diskriminanzfunktionen sind:

$$d_i(x) = -\frac{1}{2}(x - \mu_i)^T \Sigma^{-1}(x - \mu_i) + \ \ln P(c_i)$$

A) $P(c_i) = P(c_j)$ für alle $i \neq j$
Die Diskriminanzfunktionen entsprechen der quadrierten Mahalanobisdistanz.

$$d_i(x) = d_M^2(x) = (x - \mu_i)^T \Sigma^{-1} (x - \mu_i) = \|x - \mu_i\|_{\Sigma^{-1}}^2$$

Der Klassifikator ist ein **Minimum-Distanz-Klassifikator** bzgl. der Σ^{-1}-Norm.

B) $P(c_i) \neq P(c_j)$ für alle $i \neq j$

$$\|x - \mu_i\|_{\Sigma^{-1}}^2 \sim x^T \Sigma^{-1} x + \ldots$$

Der quadratische Term wird wieder, da er unabhängig von i ist, nicht berücksichtigt.
Hieraus ergibt sich die **lineare Diskriminanzfunktion**:

$$d_i(x) = w_i^T x + w_{i0} \quad \text{mit} \quad w_i = \Sigma^{-1}\mu_i \text{ und}$$
$$w_{i0} = -\tfrac{1}{2}\mu_i^T \Sigma^{-1} \mu_i + \log P(c_i)$$

Die Entscheidungsgrenzen sind abermals durch Hyperebenen repräsentiert.
Betrachtet wird wieder $\mathcal{R}_i$ benachbart zu $\mathcal{R}_j$ für $x \in \mathbb{R}^2$.
Wegen

$$w^T(x - x_0) = 0 \quad \text{mit } w = \Sigma^{-1}(\mu_i - \mu_j),\ x_0 = \frac{1}{2}(\mu_i + \mu_j) - \ldots$$

liegt nun aber die Hyperebene (Trennlinie) nicht mehr orthogonal zur Verbindungslinie zwischen μ_i und μ_j (siehe Abbildung 3.10).

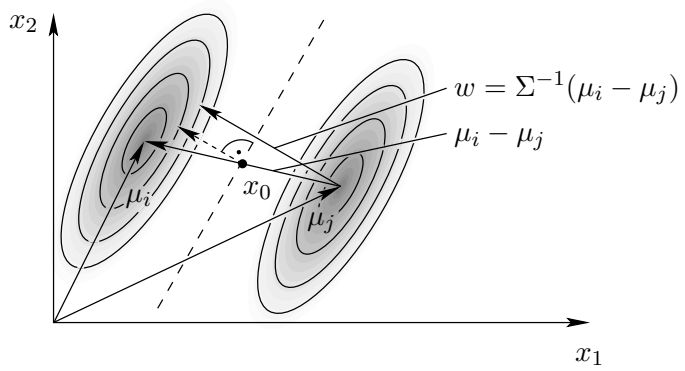
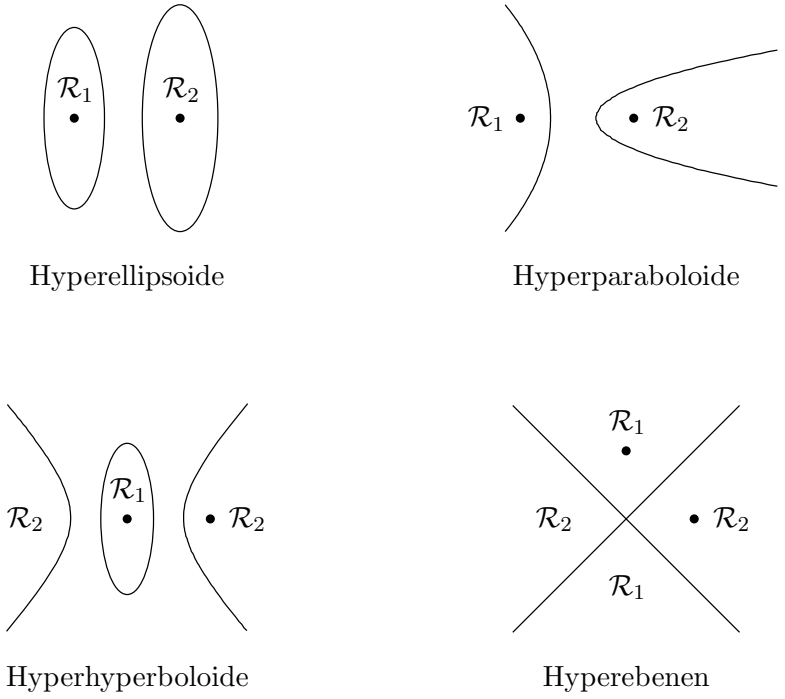


Abbildung 3.10: Nicht-Orthogonalität von Hyperebene und Verbindungslinie.

Fall 3: $\forall i : \Sigma_i$ beliebig
Es liegt eine **quadratische Diskriminanzfunktion** vor:

$$d_i(x) = x^T W_i x + w_i^T x + w_{i0} \quad \text{mit} \quad W_i = -\tfrac{1}{2}\Sigma_i^{-1},$$
$$w_i = \Sigma_i^{-1}\mu_i \text{ und}$$
$$w_{i0} = -\tfrac{1}{2}\mu_i^T\Sigma_i^{-1}\mu_i - \tfrac{1}{2}\ln|\Sigma_i| + \ln P(c_1)$$

Die Entscheidungsoberflächen sind **Hyperquadrics**, siehe Abbildung 3.11. Wegen der Komplexität der Analyse wird dieser Fall nur selten verwendet.



Hyperellipsoide | Hyperparaboloide

Hyperhyperboloide | Hyperebenen

Abbildung 3.11: Verschiedene Formen von Trennhyperebenen.

3.8 Parameterschätzung und überwachtes Lernen

Ein **optimaler Klassifikator** könnte entworfen werden, wenn $P(c_i)$ und $p(x|c_i)$ bekannt wären. In der Praxis ist dies aber nicht der Fall. Vielmehr ist vorhanden

- Generelles und vages Wissen über den Problembereich
- Stichprobe Ω_T.

Man geht deshalb in folgender Weise vor:

1. Schätzung der unbekannten Wahrscheinlichkeiten und Wahrscheinlichkeitsdichten
2. Verwendung der Schätzungen als **wahre Werte**, die nicht von der verwendeten Stichprobe abhängen.

Dabei treten insbesondere bei der **Schätzung der klassenbedingten Dichten** folgende Probleme auf:

- Die Stichprobe ist zu klein, um eine unbekannte Funktion gut anpassen zu können.
- Die Dimension des Merkmalsraums ist häufig zu groß.

Beide Sachverhalte führen dazu, dass der Merkmalsraum recht „leer“ ist.
Der Ausweg besteht häufig darin, die Dimension des Merkmalsraumes zu reduzieren (z.B. mittels statistischer Hauptachsentransformation - siehe Abschnitt 3.10).

Eine Standardmethode, die den Inhalt dieses Abschnittes bildet, ist die Annahme einer parametrisierten Verteilungsform der klassenbedingten Dichten und die Schätzung der Parameter dieser Funktion, siehe Tabelle 3.2.
Es wird also ein **Ersatzproblem** formuliert:

$$\text{Schätzung von } p(x|c_i) \quad \overset{\text{Annahme}}{\underset{\text{z.B.} N(\mu_i, \Sigma_i)}{\longrightarrow}} \quad \text{Schätzung von } p(x|c_i, \theta_i)$$

Hierbei bezieht sich die Bedingung auf das Paar (c_i, θ_i).
Die θ_i stellen einen Parametervektor dar, z.B. im Falle der Normalverteilung:

$$\theta_i = (\mu_i, \Sigma_i)$$

Es werden drei wichtige Methoden der Parameterschätzung erläutert.

A) **Maximum-Likelihood-Schätzung** (Abschnitt 3.8.1)
Annahme: Die Parameter sind Größen, deren Werte fest sind aber unbekannt. Die beste Schätzung wird dadurch definiert, dass sie die Wahrscheinlichkeit der in der Stichprobe enthaltenen Beispiele maximiert.

B) **Bayes-Schätzung** (Abschnitt 3.8.2)
Annahme: Die Parameter sind **stochastische Variable** mit bekannter **a priori Verteilung**. Die Breite der a priori Verteilung drückt die vorhandene Unsicherheit über das Problem der Klassentrennung aus. Durch Beobachtung dieser Variablen wird die a priori Verteilung in eine **a posteriori Verteilung** konvertiert. Dies führt zu einer Korrektur der “wahren” Werte der Parameter. Bayes-Schätzung führt zu einer **Schärfung** der a posteriori Dichtefunktionen. Dies entspricht dem Informationsgewinn aus den Beobachtungen.

C) **Sequentielle Parameterschätzung** (Abschnitt 3.8.3)
Während die Verfahren A und B das Lernen der Parameter der klassenbedingten Dichtefunktion im Batch-Mode (offline) ermöglichen, ist das Verfahren C ein Online-Verfahren.

Bei überwachtem Lernen werden Stichprobenelemente der Art $(x^t, y^t) \in \Omega_T$ verwendet. Die **Grundwahrheit** y^t für jedes Datum x^t sei also bekannt.

3.8.1 Maximum-Likelihood-Schätzung

Es existiere eine Partitionierung der Stichprobenmenge in K Teilmengen, wobei K die Anzahl der Klassen angibt:

$$\Omega_T = \Omega_{T_1} \cup \Omega_{T_2} \cup ... \cup \Omega_{T_K}$$

D.h., für die Stichprobenelemente gelte:

$$(x^t, y^t) \equiv (x^t, c_i) \in \Omega_{T_i} \equiv X_i \quad , i \in \{1, 2, ..., K\}$$

Annahme: Die klassenbedingte Dichte $p(x|c_i)$ habe die bekannte Parameterform:

$$p(x|c_i) = p(x|c_i, \theta_i) \quad \text{mit Parametervektor } \theta_i := (\theta_{i_1}, \theta_{i_2}, ..., \theta_{i_r})^T$$

Neues Problem: Schätzung des unbekannten Parametervektors aus der Probeninformation.

Annahme: Die Parameter verschiedener Klassen sind funktional unabhängig, d.h. θ_i ist unabhängig von X_j für $i \neq j$. Deshalb kann mit jeder Klasse unabhängig gearbeitet werden.

Folglich existieren K **separate Probleme** der Art:
Schätze mit Hilfe einer unabhängigen Stichprobe X_i den Parametervektor θ_i.
Wegen der Unabhängigkeit kann in der Folge der Klassenindex weggelassen werden.

Annahme: Der Stichprobenvorrat **einer** Klasse sei $X = \left\{x^1, ..., x^T\right\}$. Aus dieser Menge werden **unabhängig voneinander** die Merkmalsvektoren x^t entnommen. Deshalb wird die bedingte Dichtefunktion des Stichprobenvektors dargestellt durch:

$$p(X|\theta) = \prod_{t=1}^{T} p(x^t|\theta) \equiv L(\theta)$$

Man bezeichnet $L(\theta)$ als die **Likelihood** von θ bezüglich X.

Maximum-Likelihood-Schätzung

$$\widehat{\theta} = \max_\theta p(X|\theta)$$

$\widehat{\theta}$ stimmt dann am besten mit den beobachteten Daten überein.

Annahme: $p(X|\theta)$ sei eine bezüglich θ differenzierbare Funktion
Gradientenvektor bezüglich θ:

$$\nabla_\theta = \left(\frac{\partial}{\partial\theta_1}, \frac{\partial}{\partial\theta_2}, \ldots, \frac{\partial}{\partial\theta_r}\right)^T$$

Logarithmische Likelihood (Merke, log ist eine monotone Funktion!):

$$\begin{aligned} l(\theta) &= \log\ p(X|\theta) \\ l(\theta) &= \Sigma_{t=1}^{T} \log\ p(x^t|\theta) \\ \nabla_\theta l(\theta) &= \Sigma_{t=1}^{T} \nabla_\theta \log p(x^t|\theta) \end{aligned}$$

Hieraus folgen r Gleichungen der Art $\nabla_\theta l(\widehat{\theta}) \stackrel{!}{=} 0$.
Diese ML-Schätzung ist nicht für alle Verteilungsfunktionen möglich (z.B. nicht für die Gleichverteilung). Sie wird am häufigsten im Fall der Normalverteilung angewendet.

Beispiel 1:
Annahme: $p(x) \sim N(\mu, \Sigma)$, Σ bekannt
Gesucht: ML-Schätzung $\widehat{\mu}$
Für jedes Stichprobenelement x^t berechne:

$$\ln\ p(x^t|\mu, \Sigma) = -\frac{1}{2}\ln\left((2\pi)^n|\Sigma|\right) - \frac{1}{2}(x^t - \mu)^T\Sigma^{-1}(x^t - \mu)$$

$$\nabla_\mu \ln p(x^t|\mu, \Sigma) = \Sigma^{-1}(x^t - \mu)$$

Bemerkung: $\frac{d}{dz}(x^T A z) = (A + A^T)z$ für z Vektor und A Matrix (hier: $\Sigma = \Sigma^T$)
Für die gesamte Stichprobe erhält man die Normalengleichung:

$$\sum_{t=1}^{T} \Sigma^{-1}(x^t - \widehat{\mu}) \stackrel{!}{=} 0$$

Nach Multiplikation mit Σ erhält man hieraus die Schätzung $\widehat{\mu}$ des Mittelwertes der Normalverteilung, welche die Likelihood maximiert:

$$\widehat{\mu} = \frac{1}{T}\sum_{t=1}^{T} x^t$$

Die ML-Schätzung des unbekannten Populationsmittelwertes ist der Mittelwert aller Trainingsbeispiele, der Schwerpunkt des Datenclusters.

Beispiel 2:
Annahme: $p(x) \sim N(\mu, \Sigma)$
Gesucht: μ und Σ
a) Univariate Lösung: $\theta_1 = \mu, \theta_2 = \sigma^2$
Für jedes Stichprobenelement x^t berechne:

$$\ln p(x^t|\theta) = -\frac{1}{2}\log 2\pi\theta_2 - \frac{1}{2\theta_2}(x^t - \theta_1)^2$$

$$\nabla_\theta \log p(x^t, \theta) = \begin{pmatrix} \frac{1}{\theta_2}(x^t - \theta_1) \\ -\frac{1}{2\theta_2} + \frac{(x^t - \theta_1)^2}{2\theta_2^2} \end{pmatrix}$$

Für die gesamte Stichprobe:

$$\sum_{t=1}^{T} \frac{1}{\widehat{\theta}_2}(x^t - \widehat{\theta}_1) \stackrel{!}{=} 0$$

$$-\sum_{t=1}^{T} \frac{1}{\widehat{\theta}_2} + \sum_{t=1}^{T} \frac{(x^t - \widehat{\theta}_1)^2}{\widehat{\theta}_2^2} \stackrel{!}{=} 0$$

Hieraus erhält man die ML-Schätzungen $\widehat{\theta}_1 = \widehat{\mu}, \widehat{\theta}_2 = \widehat{\sigma^2}$,

$$\begin{aligned} \widehat{\mu} &= \frac{1}{T}\sum_{t=1}^{T} x^t \\ \widehat{\sigma^2} &= \frac{1}{T}\sum_{t=1}^{T}(x^t - \widehat{\mu})^2 \end{aligned}$$

b) Multivariate Lösung: $\theta_1 = \mu = (\mu_1, ..., \mu_n)$, $\theta_2 = \Sigma$ $(n \times n)$-Matrix
Für die Datenvektoren $x^t = (x_1^t, \ldots, x_n^t)^T$ erhält man:

$$\begin{aligned} \widehat{\mu} &= \frac{1}{T}\sum_{t=1}^{T} x^t \\ \widehat{\Sigma} &= \frac{1}{T}\sum_{t=1}^{T}(x^t - \widehat{\mu})(x^t - \widehat{\mu})^T \end{aligned}$$

- $\widehat{\mu}$ ist wieder der Stichprobenmittelwert.
- $\widehat{\Sigma}$ ist das arithmetische Mittel von T Matrizen $(x^t - \widehat{\mu})(x^t - \widehat{\mu})^T$.

Die wahre Kovarianz, Σ, ist durch den Erwartungswert definiert:

$$\Sigma = E\left\{(X - \mu)(X - \mu)^T\right\}$$

Wegen $\Sigma \neq \widehat{\Sigma}$ ergibt sich eine als **Bias** bezeichnete systematische Abweichung der Schätzung. Folgender Ansatz liefert eine Schätzung der Kovarianz, die frei von Bias ist.

Sei die **Stichproben-Kovarianzmatrix** gegeben durch:

$$C = \frac{1}{T-1}\sum_{t=1}^{T}(x^t - \widehat{\mu})(x^t - \widehat{\mu})^T$$

Da

$$C = \frac{T}{T-1}\widehat{\Sigma}$$

gilt, liefert $\widehat{\Sigma}$ insbesondere für große Stichproben eine gute Schätzung der Bias-freien Stichproben-Kovarianzmatrix.
Beide Schätzungen können nicht als gut oder schlecht bezeichnet werden. Sie sind nur verschieden und haben unterschiedliche positive Eigenschaften.

3.8.2 Bayes-Schätzung

Im Gegensatz zur ML-Schätzung ist die Bayes-Schätzung frei von irgend einem Bias.
Es wird zunächst die Bayes-Formel auf das beabsichtigte Trainingsszenario angepasst. Anschließend wird das Teilproblem der Parameterschätzung der Dichtefunktion allgemein formuliert. Dieses Teilproblem muß natürlich gelöst werden, um die a posteriori Wahrscheinlichkeiten der Klassen angeben zu können. Schließlich wird das Bayes-Verfahren der Parameterschätzung im Fall normalverteilter Dichtefunktionen vorgestellt.

Gegeben sei eine partitionierte Stichprobenmenge $X = \bigcup_i X_i$ mit $i = 1, ..., K$. Gesucht sind die auf X bezogenen a posteriori Wahrscheinlichkeiten $P(c_i|x, X)$. In der Stichprobe ist implizites Wissen über die Dichtefunktionen $p(x|c_i, X)$ enthalten. Es werden außerdem die a priori Wahrscheinlichkeiten $P(c_i|X)$ als gegeben vorausgesetzt. Ziel wird es also sein, diese a priori Wahrscheinlichkeiten mit Hilfe der Beobachtungen in a posteriori Wahrscheinlichkeiten umzuwandeln:

$$P(c_i|x, X) = \frac{p(x|c_i, X)P(c_i|X)}{\sum_{j=1}^{K} p(x|c_j, X)P(c_j|X)}$$

Zunächst werden einige Annahmen gemacht, die sich auf die Formulierung der Bayes-Formel auswirken werden.

1. Es wird überwachtes Training mit indizierten Stichprobenelementen vorausgesetzt. Das bedeutet:

   $$p(x|c_i, X) \rightarrow p(x|c_i, X_i)$$

   Weil $x \in X_j$ keinen Einfluß auf $p(x|c_i, X_i)$ für $i \neq j$ hat, kann das Training separat für jede Klasse, repräsentiert durch die Teilmenge X_i durchgeführt werden, also:

   $$p(x|c_i, X_i) \rightarrow p(x|X_i)$$

2. Es wird außerdem angenommen, dass die a priori Wahrscheinlichkeiten der Klassen c_i unabhängig von der Auswahl der betrachteten Trainingsmenge X_i gelten, also:

   $$P(c_i|X) = P(c_i|X_i) = P(c_i)$$

Danach stellt sich die Bayes-Formel so dar:

$$P(c_i|x, X_i) = \frac{p(x|X_i)P(c_i)}{\sum_{j=1}^{K} p(x|X_j)P(c_j)}$$

Zunächst muß also $p(x|X_i)$ für $i = 1, ..., K$ geschätzt werden. Das sind K separat zu lösende Probleme. Es wird angenommen, dass die funktionelle Form von $p(x|X_i)$

bekannt ist. Diese wird spezifiziert durch die unbekannten Parametervektoren θ_i. Damit kann $p(x|X_i)$ durch $p(x|\theta_i)$ repräsentiert werden, wie in Gleichung (3.2) zu sehen sein wird. Anders als bei der ML-Schätzung werden die Parametervektoren θ_i hier als **Zufallsvektoren** angenommen. Für jede Teilmenge X_i ergibt sich die

Problemumformulierung: Schätze den Parametervektor θ_i aus der Stichprobe X_i unter der Annahme, dass seine Dichteverteilung $p(\theta_i)$ a priori bekannt sei. Man nennt dieses Verfahren **Bayes-Lernen** einer parametrisch spezifizierten Dichtefunktion.

Gegeben die Teilmenge X_i besteht also die Aufgabe darin, aus der a priori Dichte $p(\theta_i)$ eine a posteriori Schätzung der Dichtefunktion der Parameter $p(\theta_i|X_i)$ zu bestimmen:

$$p(\theta_i) \xrightarrow{X_i} p(\theta_i|X_i)$$

Idealerweise ist $p(\theta_i)$ eine Deltafunktion $\delta(\widehat{\theta}_i)$ am Ort $\widehat{\theta}_i$. Tatsächlich drückt die Breite der a priori Dichtefunktion $p(\theta_i)$ den Grad an Unsicherheit aus, der mittels der Daten aus X_i reduziert werden soll. Mit anderen Worten: die a posteriori Dichtefunktion der Parameter soll eine Deltafunktion gut annähern. Die Dichtefunktion soll also geschärft werden.
Die in der Bayes-Formel gesuchte Dichtefunktion $p(x|X_i)$ läßt sich als marginale Dichtefunktion der Verbunddichte $p(x, \theta_i|X_i)$ formulieren:

$$p(x|X_i) = \int_{\theta_i} p(x, \theta_i|X_i) d\theta_i$$

Der Integrant wird umformuliert:

$$p(x, \theta_i|X_i) = p(x|\theta_i, X_i)p(\theta_i|X_i)$$

Unter der Annahme, dass der zu klassifizierende Merkmalsvektor x unabhängig von der speziellen Wahl der X_i sei (diese Annahme stellt keine Beschränkung dar), notieren wir:

$$p(x|\theta_i, X_i) = p(x|\theta_i)$$

Damit erhalten wir

$$p(x|X_i) = \int_{\theta_i} p(x|\theta_i)p(\theta_i|X_i) d\theta_i \tag{3.2}$$

Wenn außerdem $p(\theta_i|X_i)$ ersetzt wird durch $\delta(\widehat{\theta}_i)$, dann folgt wegen der Ausblendeigenschaft[3] der Deltafunktion:

$$p(x|X_i) \approx p(x|\widehat{\theta}_i)$$

[3] Ausblendeigenschaft der Deltafunktion:

$$p(x|\widehat{\theta}_i) = \int p(x|\theta_i)\, \delta(\widehat{\theta}_i)\, d\theta_i \quad \text{mit} \quad \delta(\theta_i) = \begin{cases} 1 & \text{für } \theta_i = \widehat{\theta}_i \\ 0 & \text{sonst} \end{cases}$$

Dies ist tatsächlich aber nur eine Idealisierung. Deshalb muß die Gleichung (3.2) in folgender Weise interpretiert werden: $p(x|X_i)$ ist das mit $p(\theta_i|X_i)$ gewichtete Mittel von $p(x|\theta_i)$.

Dies soll nun am Beispiel des **Bayes-Lernens einer normalverteilten Dichtefunktion** behandelt werden.

a) Univariater Fall

Für die Verteilung der Daten gelte: $p(x|\theta_i) = p(x|\mu) \sim N(\mu, \sigma^2)$
Es wird also angenommen, dass μ eine unbekannte stochastische Variable sei und σ^2 als bekannt vorausgesetzt werden darf. Die Varianz drückt die Unsicherheit von x aus. Diese ist besser a priori zu schätzen als der Mittelwert μ, wenn $p(x)$ eine breite Verteilung annimmt.
Das Verfahren zur Berechnung von $p(x|X_i)$ nach Gleichung (3.2) unterteilt sich in zwei Schritte.

Schritt 1: Berechnung der Gewichte $p(\mu|X_i)$

Als bekannt angenommen werden:

1. σ^2
2. $p(\mu) \sim N(\mu_0, \sigma_0^2)$ mit
   $\mu_0 \sim$ beste a priori Vermutung für μ
   $\sigma_0^2 \sim$ Unsicherheit dieser Vermutung

Durch die Entnahme von T unabhängigen Stichprobenelementen $x^t \in X_i$ erhält man für das Gewicht die Verteilungsfunktion $p(\mu|X_i)$:

$$p(\mu|X_i) = \alpha \prod_{t=1}^{T} p(x^t|\mu)p(\mu) \tag{3.3}$$

mit dem Skalierungsfaktor $\alpha = \alpha(X)$ aber $\alpha \neq \alpha(\mu)$.
Andererseits gilt:

$$p(\mu|X_i) \sim N(\mu_T, \sigma_T^2) = \frac{1}{\sqrt{2\pi}\sigma_T} \exp\left(-\frac{1}{2}\frac{(\mu - \mu_T)^2}{\sigma^2}\right) \tag{3.4}$$

wegen $p(\mu) \sim N(\mu_0, \sigma_0^2)$ und $p(x|\mu) \sim N(\mu, \sigma^2)$, da das Produkt von unabhängigen Normalverteilungen eine Normalverteilung ist. Es sind also drei Verteilungen der Parameter zu unterscheiden.
Die Parameter μ_T und σ_T^2 sind die Unbekannten der Verteilung der Gewichte. Gleichsetzen von (3.3) und (3.4) liefert die beste Schätzung für μ nach T Beobachtungen:

$$\mu_T = \frac{T\sigma_0^2}{T\sigma_0^2 + \sigma^2} m_T + \frac{\sigma^2}{T\sigma_0^2 + \sigma^2}\mu_0$$

mit dem Stichprobenmittelwert (ML-Schätzung): $m_T = \frac{1}{T}\sum_{t=1}^{T} x^t$
und der Unsicherheit dieser Schätzung: $\sigma_T^2 = \frac{\sigma_0^2\sigma^2}{T\sigma_0^2+\sigma^2}$

Für den gesuchten Mittelwert kann man auch notieren:

$$\mu_T = \beta_1 m_T + \beta_2 \mu_0 \text{ mit } \beta_1 + \beta_2 = 1$$

Während $\beta_2\mu_0$ die a priori Information ausdrückt, steht $\beta_1 m_T$ für die empirische Information. Folglich liegt μ_T zwischen m_T und μ_0 und es gilt $\lim_{T\to\infty} \mu_T = m_T$ für $\sigma_0^2 \neq 0$.

Bei großer Stichprobe ($T \to \infty$) wird die a priori Information unwichtig. Die Varianz der Schätzung von μ, d.h. σ_T^2, geht für $T \to \infty$ gegen Null. Die Information über den Mittelwert wird scharf.
Diese Strategie der Schärfung von a posteriori Information durch Beobachtungen heißt **Bayes-Lernen**.

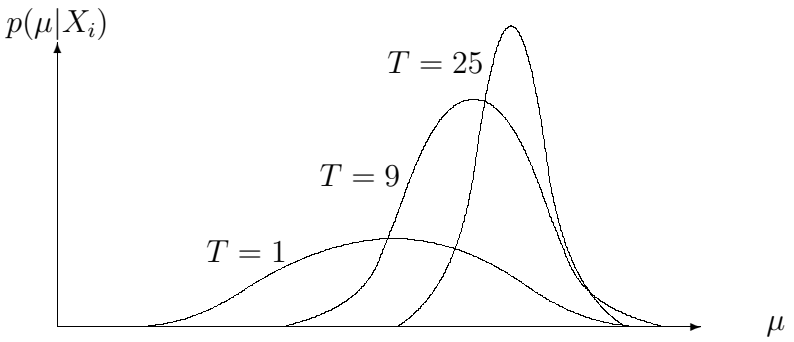


Abbildung 3.12: Schärfung der Gewichtsfunktion durch Bayes-Lernen.

Schritt 2: In diesem Schritt wird entsprechend (3.2) die parametrisch spezifizerte Dichteverteilung $p(x|X_i)$ berechnet:

$$p(x|X_i) = \int p(x|\mu)p(\mu|X_i)d\mu$$
$$\text{mit } p(x|\mu) \sim N(\mu, \sigma^2) \text{ und } p(\mu|X_i) \sim N(\mu_T, \sigma_T^2)$$

Allerdings muß dieser Schritt nicht explizit ausgeführt werden, weil das

Ergebnis unmittelbar folgt:

$$\begin{aligned} p(x|X_i) &= \int \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right) \frac{1}{\sqrt{2\pi}\sigma_T} \exp\left(-\frac{1}{2}\frac{(\mu-\mu_T)^2}{\sigma_T^2}\right) d\mu \\ &= \frac{1}{2\pi\sigma\sigma_T} \exp\left(-\frac{1}{2}\frac{(x-\mu_T)^2}{\sigma^2+\sigma_T^2}\right) f(\sigma,\sigma_T) \\ &\quad \text{mit } f(\sigma,\sigma_T) = \int \exp\left(-\frac{1}{2}\frac{\sigma^2+\sigma_T^2}{\sigma^2\sigma_T^2}\left(\mu - \frac{\sigma_T^2 x + \sigma^2\mu_T}{\sigma^2+\sigma_T^2}\right)^2\right) d\mu \end{aligned}$$

Es gilt offensichtlich:

$$p(x|X_i) \sim N(\mu_T, \sigma^2 + \sigma_T^2)$$

Um die klassenbedingte Dichte $p(x|X_i)$ zu erhalten, über dessen parametrische Form $p(x|\mu) \sim N(\mu, \sigma^2)$ bekannt ist, braucht man nur folgende Ersetzung nach T Beobachtungen durchzuführen:

$$\begin{aligned} \mu &\longrightarrow \mu_T \\ \sigma^2 &\longrightarrow \sigma^2 + \sigma_T^2 \end{aligned}$$

Da nun $p(x|X_i) = p(x|c_i, X_i)$ ermittelt ist, kann auch der Bayes-Klassifikator entworfen werden.

b) Multivariater Fall
Die Erweiterung auf den multivariaten Fall ist einfach durchzuführen.

Gegeben: $p(x|\mu) \sim N(\mu, \Sigma), \Sigma$ bekannt
$p(\mu) \sim N(\mu_0, \Sigma_0)$, μ_0 und Σ_0 bekannt
$X = \{x^1, ..., x^T\}$

Gesucht: $p(x)$ durch Approximation mittels $p(x|X)$
Lösung:
1. Schritt: Gewichte $p(\mu|X) \sim N(\mu_T, \Sigma_T)$
2. Schritt: Verteilung der Daten $p(x|X) \sim N(\mu_T, \Sigma + \Sigma_T)$

Das Ergebnis kann in folgender Weise interpretiert werden. Die Realisierungen x ergeben sich als Summe zweier stochastischer Variabler:

1. Zufallsvektor μ mit $p(\mu|X) \sim N(\mu_T, \Sigma_T)$
2. unabhängiger Zufallsvektor y mit $p(y) \sim N(0, \Sigma)$

Die Summe zweier unabhängiger normalverteilter Variabler ist wieder eine normalverteilte Variable (siehe Abschnitt 3.6.3). Da sich für eine große Zahl von Beobachtungen ($T \to \infty$) die a posteriori Wahrscheinlichkeitsdichte

$p(\theta_i|X_i)$ einer Deltafunktion nähert, nähert sich die Bayes-Schätzung von $p(x)$ der ML-Schätzung, welche auf der Annahme fester Parameterwerte beruht. Die Bayes-Schätzung kann für eine begrenzte Anzahl von Beobachtungen aber bessere Schätzungen liefern als die ML-Schätzung, da sie zusätzliche Information über die a priori Verteilung der Parameter nutzt.

3.8.3 Sequentielle Parameterschätzung

In diesem Abschnitt wird eine **iterative Technik zur Parameterschätzung** eingeführt, die das aktuelle Schätzergebnis mittels **eines neuen Datums** aktualisiert. Diese Verfahrensweise hat mehrere Vorzüge:

1. Vermeidung des Speicherns großer Datenmengen
2. Online-Lernen der Parameter in Echtzeitsystemen wird möglich.
3. Adaptives Systemverhalten, das sich Änderungen der Bedingungen anpaßt, wird möglich.

Beispiel: ML-Schätzung des Mittelwertes
Batchmode: $\widehat{\mu} = \frac{1}{T}\sum_{t=1}^{T} x^t$
Online-Mode: $\widehat{\mu}_{T+1} = \widehat{\mu}_T + \frac{1}{T+1}(x^{T+1} - \widehat{\mu}_T)$
Hier sind lediglich $\widehat{\mu}_T$ und T zu speichern. Die Korrekturen nehmen mit wachsendem T ab. Ist die Konvergenz ausreichend gesichert?

Einschub: Herleitung der rekursiven Schätzung aus der nicht rekursiven Schätzung

$$\begin{aligned}
\hat{\mu}_T &= \frac{1}{T}\sum_{t=1}^{T} x^t \\
&= \frac{1}{T}x^T + \frac{1}{T}\sum_{t=1}^{T-1} x^t \\
&= \frac{1}{T}x^T + \frac{T-1}{T}\hat{\mu}_{T-1} \\
&= \hat{\mu}_{T-1} + \frac{1}{T}(x^T - \hat{\mu}_{T-1})
\end{aligned}$$

Interpretation: Der neue Datenvektor x^T trägt zu $\hat{\mu}_T$ bei als Verschiebung von $\hat{\mu}_{T-1}$ um ein kleines Stück, proportional zu $\frac{1}{T}$, in Richtung des Fehlers $(x^T - \hat{\mu}_{T-1})$.

3.8.3.1 Robbins-Monro-Algorithmus

Literatur: Robbins und Monro (1951)

Der Ansatz zur iterativen Berechnung des Mittelwertes (oben) kann als Spezialfall eines allgemeinen Problems gesehen werden. Dies ist das Finden der **Nullstellen**

einer stochastisch definierten Funktion.
Gegeben: Eine stochastische Funktion $y(x)$, die eine gestörte Repräsentation einer unbekannten unterliegenden Funktion $f(x)$ darstellt. Obwohl die (x, y)-Daten Störungen aufweisen, sind y und x zueinander (nichtlinear) korreliert. Es existiert also eine Funktion $f(x)$:

$$\widehat{y} = f(x) = E\{Y|X\}$$

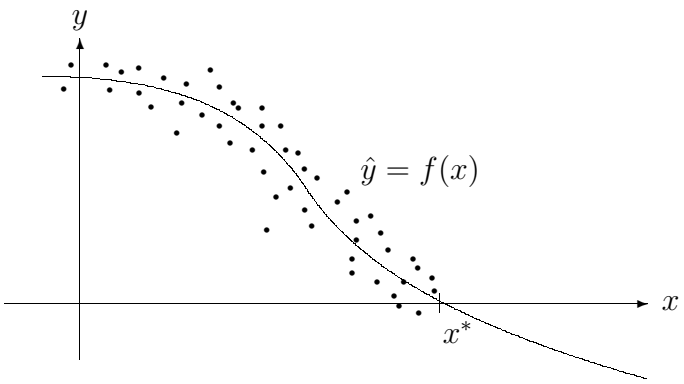


Abbildung 3.13: Stochastisch gestörte Funktion und ihre nichtlineare Regressionsfunktion.

Es ist nicht das Ziel, die komplette (nichtlineare) Regressionsaufgabe zu lösen, also $f(x)$ zu schätzen. Ziel des Robbins-Monro-Algorithmus ist vielmehr die Schätzung der Nullstelle x^*, so dass $f(x^*) = 0$ gilt.
Wenn das obige Beispiel zur iterativen Berechnung des Mittelwertes auf eine Normalverteilung bezogen wird, entspricht der RM-Algorithmus der Berechnung dieses Mittelwertes durch Finden der Nullstelle der Ableitung der Gaußfunktion.

Annahmen:

1. y hat eine endliche Varianz:	$E\{(Y-f)^2\|X\} < \infty$
2. O.B.d.A. gilt:	$f(x) > 0$ für $x < x^*$
	$f(x) < 0$ für $x > x^*$

Dann kann die Nullstelle x^* der gesuchten Funktion $f(x)$ iterativ geschätzt werden:

$$x_{T+1} = x_T + \alpha_T f(x_T)$$

Hierbei ist $\{\alpha_T\}$ eine Folge positiver Zahlen mit gewissen Eigenschaften, die sichern, dass $\lim\limits_{T\to\infty} x_T = x^*$ mit Wahrscheinlichkeit 1 gilt[4]:

1. $\lim\limits_{T\to\infty} \alpha_T = 0$: Abnahme der Korrekturbeiträge mit T (sichert Konvergenz gegen Grenzwert).

[4]Vielfach auch notiert als $\operatorname{plim}\limits_{T\to\infty}(x_T = x^*) = 1$.

2. $\sum_{T=1}^{\infty} \alpha_T = \infty$: Korrekturen sind hinreichend groß.
3. $\sum_{T=1}^{\infty} \alpha_T^2 < \infty$: Das akkumulierte Rauschen hat eine endliche Varianz.

3.8.3.2 Sequentielle ML-Parameterschätzung mittels RM-Algorithmus

Sei die Likelihoodfunktion gegeben durch

$$L(\theta) = p(X|\theta) = \prod_{t=1}^{T} p(x^t|\theta),$$

dann liefert

$$\frac{\partial}{\partial\theta} L(\theta) \Big|_{\widehat{\theta}} \stackrel{!}{=} 0$$

die ML-Schätzung $\widehat{\theta}$ für den Parametervektor θ.
Logarithmierung der Likelihoodfunktion und Einführung eines Faktors $\frac{1}{T}$, um den Grenzwert betrachten zu können, ergibt:

$$\frac{1}{T}\frac{\partial}{\partial\theta}\left(\sum_{t=1}^{T} \log p(x^t|\theta)\right) \Big|_{\widehat{\theta}} \stackrel{!}{=} 0$$

$$\Longrightarrow \lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} \frac{\partial}{\partial\theta} \log p(x^t|\theta) = E\left\{\frac{\partial}{\partial\theta} \log p(X|\theta)\right\} \stackrel{!}{=} 0$$

Iteratives Schema des Findens der Nullstelle als ML-Schätzung:

$$\theta_{T+1} := \theta_T + \alpha_T \frac{\partial}{\partial\theta} \log p(x^{T+1}|\theta) \mid_{\theta_T}$$

Das Verfahren ist gut implementierbar, wenn die funktionelle Form von $p(X|\theta)$ gegeben ist.

Annahme: $p(X|\theta) \sim N(\mu, \sigma^2)$, μ unbekannt
Die Ableitung der Gaußfunktion hat einen Nulldurchgang bei $x = \mu$, ist links davon positiv und rechts davon negativ. Dies entspricht der Annahme des RM-Algorithmus.
Für $\alpha_T = \frac{\sigma^2}{T+1}$ folgt die iterative Schätzung des Mittelwertes μ:

$$\widehat{\mu}_{T+1} = \widehat{\mu}_T + \frac{1}{T+1}(x^{T+1} - \widehat{\mu}_T)$$

Die Regressionsfunktion $f(x) = E\{Y|X\}$ liefert:

$$E\left\{\frac{X - \widehat{\mu}}{\sigma^2}\right\} = \frac{\mu - \widehat{\mu}}{\sigma^2}$$

mit der Nullstelle μ für $\lim_{T\to\infty} \widehat{\mu}_T = \mu$.
In der Abbildung 3.14 ist diese Regressionsfunktion und die jeweilige Verteilung $p(y|\widehat{\mu})$ aufgetragen, die der Ableitung der logarithmierten Likelihoodfunktion entspricht.
Dieses Verfahren findet Anwendung bei adaptiven neuronalen Netzen.

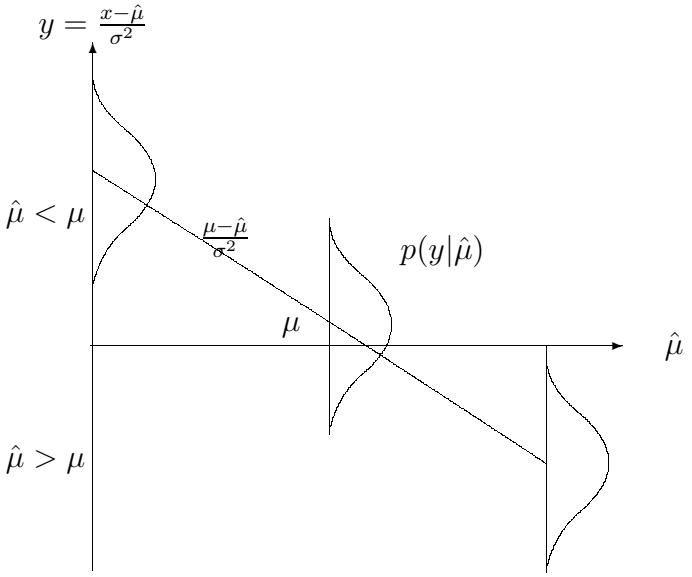


Abbildung 3.14: Iterative ML-Parameterschätzung für eine Normalverteilung.

3.9 Nichtparametrische Schätzverfahren

In Abschnitt 3.1.4 wurde eine Synopsis der Methoden zur statistischen Entscheidungsfindung vorgestellt. Eine zentrale Rolle spielte die Schätzung der Dichtefunktion der Daten $p(x|c_i)$. In Abschnitt 3.8 ging man davon aus, dass die funktionale Gestalt der Dichtefunktion bekannt sei, und es wurden drei Verfahren zur Schätzung ihrer Parametrisierung aufgezeigt. Oft befindet man sich aber nicht in einer derart komfortablen Situation. Vielmehr muß davon ausgegangen werden, dass $p(x|c_i)$ funktional nicht allgemein gültig spezifierbar ist, sondern die Gestalt von den Daten stark abhängt. Entstammen die Daten aus unterschiedlichen Bereichen von Ω, so ändert $p(x)$ seine Gestalt.
In diesem Fall hat man ein nichtparametrisches Entscheidungsproblem vorliegen und $p(x)$ ist nichtparametrisch zu schätzen. Als einzige Annahme über $p(x)$ kann man eine gewisse **Glattheit** der Funktion annehmen.
In diesem Abschnitt werden zwei Standardmethoden nicht-parametrischer Schätzungen von $p(x)$ vorgestellt, die in Variationen in unterschiedlichen neuronalen Architekturen anzutreffen sind. Dies sind

A) Kernbasierte Methode (siehe Abschnitt 3.9.2)

B) K-Nächste-Nachbar-Methode (siehe Abschnitt 3.9.3).

Beide Methoden finden sich z.B. in der Architektur der RBF-Netze wieder. Sie habe eine enge Beziehung zur Histogramm-Methode. Deshalb werden in 3.9.1 die histogrammbasierte Schätzung von $p(x)$ diskutiert.

3.9.1 Histogramme

Literatur: Bishop (2004), Duda, Hart und Stork (2004)

Ein Histogramm $h(x)$ repräsentiert die Häufigkeitsverteilung des diskretisierten Merkmales x. Es stellt demzufolge eine Approximation der Dichtefunktion $p(x)$ dar. Sei die Spanne des Merkmals x in M Intervalle (Urnen) gleichmäßig unterteilt. Dann repräsentiert $h(x_i)$ die Einzelwahrscheinlichkeit dafür, in der i-ten Urne eine gewisse Anzahl von Stichproben bezüglich des Merkmals x anzutreffen. Offensichtlich ist die Anzahl M der Urnen ein frei wählbarer Parameter. Tatsächlich ist dieser Parameter sehr kritisch für die erreichbare Güte der Approximation von $p(x)$ durch $h(x)$. Die **Urnenbreite wirkt als Glättungsparameter** auf die Häufigkeitsverteilung:

- zu schmale Urnen → sehr stark verrauschtes Histogramm
- zu breite Urnen → sehr stark verschmiertes Histogramm

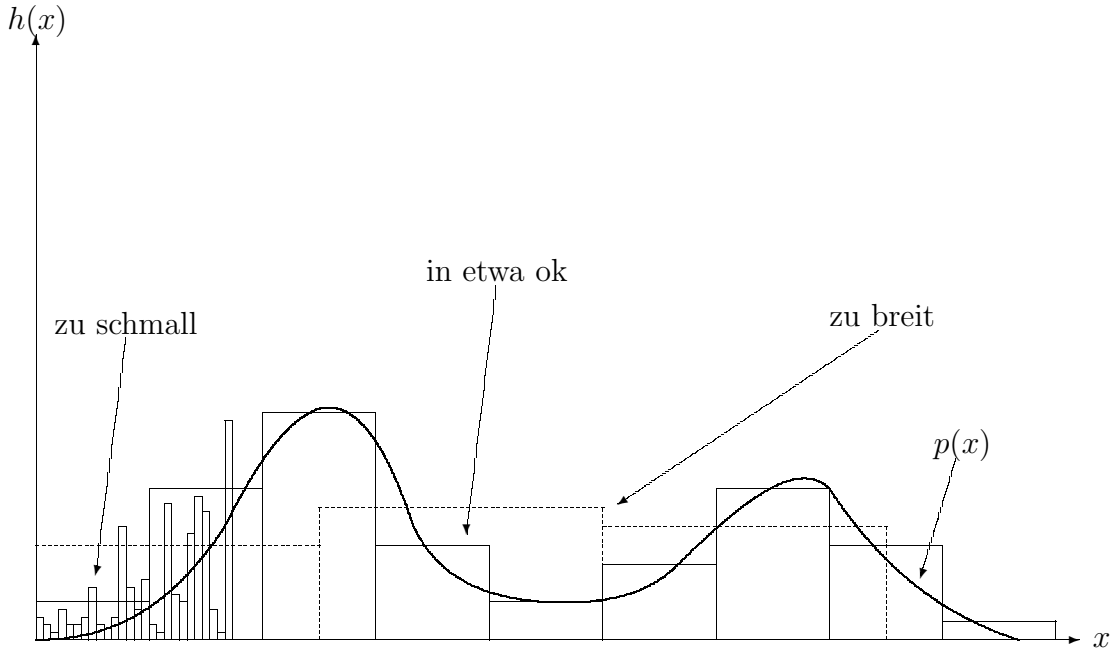


Abbildung 3.15: Zur optimalen Wahl einer Urnengröße.

In beiden Extremfällen geht die Ähnlichkeit von $h(x)$ zu $p(x)$ verloren.
Hieraus entsteht das **Problem**:

> Wie kann die Urnenbreite optimal gewählt werden, wenn die Grundwahrheit (die Gestalt der Dichtefunktion) nicht bekannt ist?

Offensichtlich hat ein gut berechnetes Histogramm den **Vorteil**, die implizit in der Stichprobenmenge enthaltene Struktur der Daten verdichtet zu repräsentieren. Wenn es konstruiert ist, sind die erzeugenden Daten verzichtbar.

Ein histogrammbasiertes Schätzverfahren für $p(x)$ hat aber zwei wesentliche **Nachteile**:

1. Die geschätzte Dichtefunktion ist **nicht glatt**.
   Es ist unwahrscheinlich, dass sie die beobachteten Daten in ihrer wahren Struktur repräsentiert.

2. Mehrdimensionale (vektorielle) Daten erfordern **mehrdimensionale Histogramme**. Mittels **Statistiken höherer Ordnung** werden die Beziehungen zwischen den Komponenten des Merkmalsvektors analysiert (siehe Abschnitt 3.10.1).

Sei n die Dimension des Merkmalsvektors $x = (x_1, ..., x_n)$. Dann kann der gesamte aufgespannte Merkmalsraum in M^n Urnen zerlegt werden. Das Histogramm bildet eine n-dimensionale Hyperfläche im $(n+1)$-dimensionalen Häufigkeitsraum. Während T Stichprobenelemente im 1D-Histogramm auf M Urnen aufzuteilen sind, sind sie im nD-Histogramm auf M^n Urnen aufzuteilen. Das bedeutet:

Der Stichprobenumfang ist auf T^n anzuheben, um eine vergleichbare Dichte der Daten im nD-Histogramm zu erhalten. Da dies praktisch aber nicht realisierbar ist, bleibt die Mehrzahl der Urnen leer und die Struktur von $p(x)$ ist nicht repräsentierbar. Man nennt dies das **Problem der Dimensionalität** (curse of dimensionality) eines Entscheidungsfindungsprozesses.

Das trifft natürlich auch auf neuronale Architekturen voll zu. Es ist also nicht sinnvoll, zur Charakterisierung eines Problems so viel als möglich Merkmale zu Hilfe zu nehmen. Vielmehr ist folgende Vorgehensweise anzuraten:

Beginne mit einer niedrigen Anzahl von Merkmalen und überprüfe, ob das Problem lösbar ist. Im Allgemeinen wird eine bestimmte Fehlerrate der Entscheidungen beobachtet. Ist diese zu hoch, nehme schrittweise **weitere unabhängige Merkmale** hinzu und beobachte die Tendenz der Fehlerrate. Sie sollte abnehmen. Ab einer gewissen Anzahl von Merkmalen steigt sie aber wieder an. Dies hat seine Ursache im Problem der Dimensionalität.

In Abschnitt 3.10 wird eine Methode vorgestellt, wie aus einer Menge von (abhängigen) Merkmalen eine andere Menge unabhängiger Merkmale berechnet werden kann. Diese ist die Karhunen-Loeve-Transformation. Ebenso wie nicht eine allgemeine Aussage über die Dimension des Merkmalsraumes gemacht werden kann, ist auch der erforderliche Stichprobenumfang mehr eine Erfahrungsgröße.

Man kann nur feststellen:
Einfache Probleme zeigen die oben beschriebenen Effekte nicht, weil sie zu ihrer Beschreibung nur einen niedrig dimensionalen Merkmalsraum erfordern. Aber

interessante praktische Probleme sind meist keine einfachen Probleme.

Nun wird die histogramm-basierte Schätzung von $p(x)$ formalisiert.
Die Wahrscheinlichkeit dafür, dass der Merkmalsvektor x mit der unbekannten Dichtefunktion $p(x)$ aus der Region $\mathcal{R}$ des Merkmalsraumes X stammt, ist gegeben durch:

$$P = \int_{\mathcal{R}} p(x')dx',\ P \in [0,1]$$

Die Region $\mathcal{R}$ ist als eine Urne im Histogramm $h(x)$ zu verstehen. Also ist für P eine Einzelwahrscheinlichkeit zu berechnen.
Sei $T = |\mathrm{X}|$ die Gesamtzahl verfügbarer Daten und K von ihnen entstammen der Region $\mathcal{R} \subset \mathrm{X}$. Ob ein Datum $x \in \mathrm{X}$ einer Region $\mathcal{R} \subset \mathrm{X}$ zuzuordnen ist, entspricht dem stochastischen Ereignis $\psi : x \in \mathcal{R}$. Hieraus folgt eine Zerlegung des Stichprobenraums $\Omega = \psi \cup \bar{\psi}$. Mit der Wahrscheinlichkeit $p = P(x \in \mathcal{R})$, $0 \leq p \leq 1$, wird das Ereignis ψ beschrieben. Dies gilt unabhängig von der Nummer des Versuchs.
Sei $Y^{(T)}$ die stochastische Variable für das Eintreten von ψ in einer Serie von T Versuchen. Die Realisierungen von $Y^{(T)}$ sind die Zahlen $K = 0, 1, \ldots, T$, also die Anzahl der günstigen Ausgänge bei T Versuchen.
Die Einzelwahrscheinlichkeiten hierfür, also $P(Y^{(T)} = K)$, wird durch die **Binomialverteilung**

$$P(K) = \binom{T}{K} p^K (1-p)^{T-K}$$

für $K = 0, 1, \ldots, T$ beschrieben. Die Parameter der Binomialverteilung sind T und p. Es existiert eine einfache Rekursionsformel zur Berechnung der $P(K)$:

$$P(K) = P(K-1)\left(1 + \frac{(T+1)p - K}{K(1-p)}\right)$$

Für große Stichproben geht die Binomialverteilung in die Normalverteilung über.

Die Momente der Binomialverteilung sind:

Erwartungswert:	$E\left\{Y^{(T)} = K\right\} = Tp$ bzw. $E\left\{\frac{K}{T}\right\} = p$
Varianz:	$\mathrm{Var}\left\{Y^{(T)} = K\right\} = E\left\{(K - E\{K\})^2\right\} = Tp(1-p)$ bzw. $\mathrm{Var}\left\{\left(\frac{K}{T}\right)\right\} = \frac{p(1-p)}{T}$

Die Varianz drückt die erwartete Unsicherheit, d.h. die Abweichung vom Erwartungswert aus. Für $T \to \infty$ verschwindet diese Unsicherheit und die diskrete Dichtefunktion geht in eine Deltafunktion am Erwartungswert über.

Für hinreichend großes T erwartet man, dass

$$P \approx \frac{K}{T} \tag{3.5}$$

als gute Schätzung der Wahrscheinlichkeit P angesehen werden kann.
Unter der Annahme, dass $p(x)$ kontinuierlich und glatt über $\mathcal{R}$ ist, folgt

$$P = \int_{\mathcal{R}} p(x')dx' \approx p(x) \cdot V \tag{3.6}$$

als gute Approximation für P. Hierbei ist V das Volumen (mehrdimensionale Intervallbreite) der Urne, die der Region $\mathcal{R}$ zugeordnet wird. Als intuitives Ergebnis aus (3.5) und (3.6) folgt für die Dichtefunktion nach der Histogrammmethode:

$$p(x) \approx \frac{P}{V} = \frac{K}{TV} \tag{3.7}$$

Aber in Gleichung (3.5) wurde implizit die Annahme gemacht, dass $\mathcal{R}$ relativ groß ist. Hingegen ist Gleichung (3.6) an die Annahme gebunden, dass $\mathcal{R}$ relativ klein ist. Diese beiden widersprechenden Annahmen führen dazu, dass Gleichung (3.7) durch zwei unterschiedliche Strategien erfüllt werden kann, wenn K und V als Parameter dieser Dichtefunktion interpretiert werden.

1. **Wähle festes** K und bestimme das entsprechende Volumen V aus den Daten. Dies entspricht der K-Nächste-Nachbar-Methode (Abschnitt 3.9.3).

2. **Wähle festes** V und bestimme den Anteil K der Daten, die in Region $\mathcal{R}$ fallen. Dies entspricht den kernbasierten Methoden (Abschnitt 3.9.2).

In beiden Fällen werden jedoch, anders als bei der Histogrammmethode, die Urnen nicht gleichmäßig platziert, sondern an die vorliegenden Daten geheftet. Sie werden also adaptiv verteilt.

Für $T \to \infty$ konvergieren beide Verfahren gegen die wahre Wahrscheinlichkeitsdichte $p(x)$, vorausgesetzt, dass V geeignet mit T schrumpft und K mit T in geeigneter Weise wächst.

3.9.2 Kernbasierte Methode

Ausgehend von der Diskussion der Gleichung (3.7) wird ein festes Volumen der Urnen gewählt und das dazu passende K bestimmt.

Annahme: Die Urnenregion $\mathcal{R}$ sei ein Hyperkubus der Seitenlänge h und der Dimension n, der im n-dimensionalen Merkmalsraum am Ort $x \in \mathcal{R}$ lokalisiert ist, um $p(x)$ zu schätzen. Der Merkmalsraum wird also mit Hyperkuben des Volumens

$$V = h^n$$

abgetastet (nicht unbedingt vollständig).

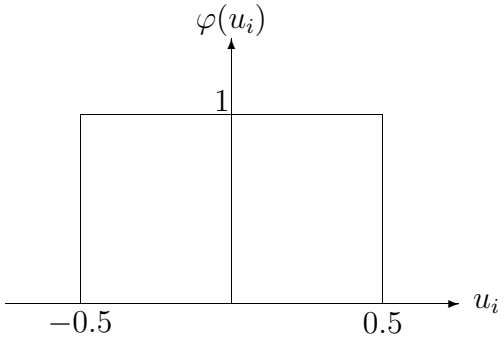


Abbildung 3.16: Beispiel eines Volumens bzw. einer Urne einer Kerndichteschätzung (kernel estimation).

Sei

$$\varphi(u_i) = \begin{cases} 1 & \text{für} \quad |u_i| < \frac{1}{2} \\ 0 & \text{sonst} \end{cases}$$

eine **Kernfunktion**, auch **Parzen-Fenster** genannt, mit den relativen Koordinaten $u = (u_1, ..., u_n)$, die am Ort x lokalisiert ist. Jedes Datum $x^t \in X$ trägt mit dem Inkrement

$$\varphi\left(\frac{x - x^t}{h}\right) = \begin{cases} 1 & \text{für} \quad x^t \in \mathcal{R} \\ 0 & \text{sonst} \end{cases}$$

zur Häufigkeitsverteilung bei. Somit entfallen

$$K(x) = \sum_{t=1}^{T} \varphi\left(\frac{x - x^t}{h}\right)$$

von T Merkmalsvektoren, auf den am Ort x fixierten Hyperkubus. Für festes $V = h^n$ erhält man die geschätzte Dichtefunktion der Merkmale:

$$\widehat{p}(x) = \frac{K}{TV} = \frac{1}{T} \sum_{t=1}^{T} \frac{1}{h^n} \varphi\left(\frac{x - x^t}{h}\right)$$

Damit erinnert die Methode der Kernfunktionen an die Histogrammmethode. Jedes Ereignis wird einmal gezählt mit dem Beitrag der Einheitsbreite der Urnen aber einer **adaptiven Verteilung der Urnen**.
Dabei treten natürlich Diskontinuitäten von $\widehat{p}(x)$ auf. Deshalb werden als Kernfunktion sogenannte **glättende Filterfunktionen** gewählt.

Beispiel: Multivariate Gaußkerne ($\Sigma = \sigma^2 I$)
Man erhält als Schätzung der Dichtefunktion:

$$\widehat{p}(x) = \frac{1}{T} \sum_{t=1}^{T} \frac{1}{(2\pi h^2)^{n/2}} \exp\left(-\frac{\|x - x^t\|^2}{2h^2}\right)$$

Allgemein werden an Kernfunktionen folgende Forderungen gestellt:

$$\varphi(u) \geq 0, \quad \int \varphi(u)du = 1$$

Hieraus folgt:

$$\widehat{p}(x) \geq 0, \quad \int \widehat{p}(x')dx' = 1$$

In der folgenden Abbildung werden für die Fälle h zu klein, h zu groß und h etwa in Ordnung die Auswirkungen auf die geschätzte Dichtefunktion $\widehat{p}(x)$ gezeigt und mit der wahren Dichtefunktion $p(x)$ (fette Kurve) verglichen. Gefordert wird, dass $\widehat{p}(x)$ die in $p(x)$ enthaltene Klassentrennung wiedergibt. Ist h zu groß, bewirkt die zu starke Glättung eine Verwischung dieser Unterschiede. Ist h zu klein, repräsentiert die Kurve $\widehat{p}(x)$ die Stichprobenelemente im Sinne eines Auswendiglernens, einschließlich der darin enthaltenen Schwankungen der Daten (Rauschen). Die Schätzung muß aber hiervon abstrahieren können (**Generalisierung** bzgl. Rauschen).

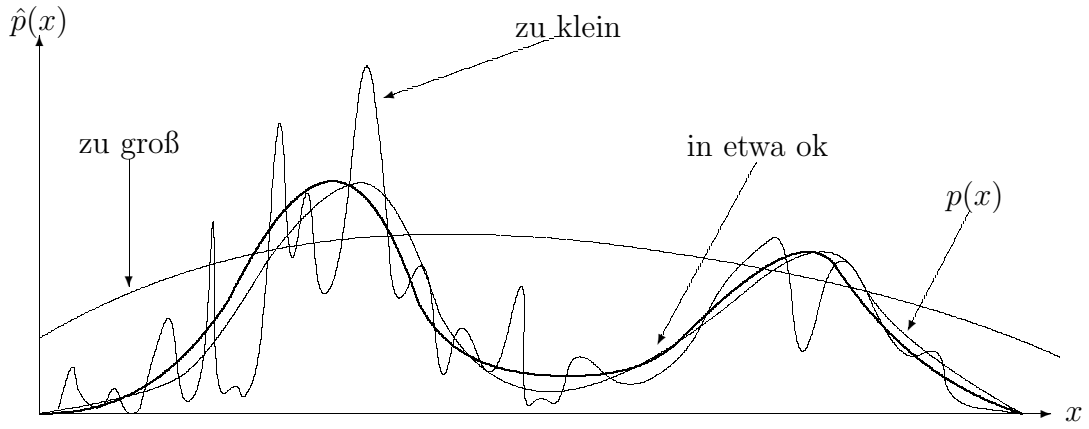


Abbildung 3.17: Kerndichteschätzung in Abhängigkeit von der Urnengröße.

Erwartungswert der geschätzten Dichtefunktion:

$$\begin{aligned} E\left\{\widehat{p}(x)\right\} &= \lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \frac{1}{h^n} \varphi\left(\frac{x - x^t}{h}\right) \\ &= E\left\{\frac{1}{h^n}\varphi\left(\frac{x - x^t}{h}\right)\right\} \\ &= \int \frac{1}{h^n}\varphi\left(\frac{x - x'}{h}\right) p(x')dx' \end{aligned}$$

Unter der Annahme, dass die Stichprobenwerte x^t unabhängig entsprechend der wahren Dichtefunktion $p(x)$ erzeugt wurden, läßt sich der Erwartungswert der geschätzten Dichtefunktion $\widehat{p}(x)$ als **Faltung** der wahren Dichtefunktion mit der

Kernfunktion interpretieren. Man erhält also eine geglättete Variante der wahren Dichtefunktion.
Sind die Daten nicht gestört, sollte

$$\lim_{h \to 0} \widehat{p}(x) = p(x)$$

gelten. In der Praxis ist aber von verrauschten Daten auszugehen. Deshalb kann h nur ein Minimum annehmen, um Auswendiglernen zu vermeiden.
Einige weitere Eigenschaften kernbasierter Schätzverfahren:

- Die Speicherung aller Datenvektoren ist erforderlich (Batch-Mode).
- Als Variante ist die Anpassung von Lage und Breite der Fenster an die Struktur der Daten (Dichtefunktion) möglich. Ist eine Dichtefunktion stark gekrümmt, sollten die Kernfunktionen klein sind und eng platziert werden. Im Fall geringer Krümmung ist es effizienter, mit großen Kernfunktionen und weiten Abständen zu arbeiten. Dies wird z.B. bei Gaußneuronen im Kontext von RBF-Netzen, Wavelet-Netzen und Support-Vektor-Maschinen (SVM) praktiziert.
- Die Schätzung weist einen Bias auf.

3.9.3 K-Nächste-Nachbar-Methode

Das Problem der Kernmethode ist die optimale Wahl der Fensterbreite, die eine Funktion des Ortes im Merkmalsraum X ist. Starke Krümmung der wahren Dichtefunktion erfordert eine kleine Kernbreite und schwache Krümmung ermöglicht große Kernbreite. Das Dilemma liegt darin, dass man die wahre Dichtefunktion nicht kennt. Außerdem führen adaptive Breiten der Kernfunktion u.U. zu variierender statistischer Signifikanz, da in breite Fenster mehr Daten fallen können als in schmale Fenster.
Ausgehend von der Histogrammmethode (Gleichung 3.7) wird bei der KNN-Methode die Anzahl der pro Urne akkumulierten Stichprobenelemente K fixiert und das **Volumen der Urnen angepaßt**.

Das heißt:
Die Breite der Hyperkuben wächst, bis sie K Elemente enthalten.
Hier **wirkt K als Glättungsparameter** für die Schätzung der Dichtefunktion. Es treten ähnliche Phänomene auf wie bei der Kernmethode. Hier gilt für $x \in \mathcal{R}$: Ist $p(x)$ relativ groß, sollte V klein sein und ist $p(x)$ relativ klein, sollte V groß sein. Weil K über alle Urnen konstant ist, werden abgeleitete Entscheidungen mit gleicher stochastischer Signifikanz unabhängig von der Dichte $p(x)$ getroffen. Ein Problem der Methode besteht darin, dass das Integral $\int p(x)dx$ divergieren kann. Deshalb stellt $p(x)$ u.U. keine echte Dichtefunktion dar. Auch bei der KNN-Methode müssen alle Trainingsdaten verfügbar sein.

KNN-Klassifikationsregel:
Ein Bayes-Klassifikator erfordert die a priori Wahrscheinlichkeiten der Klassen $P(c_i),\ i \in \{1, ..., C\}$, und die klassenbedingten Dichtefunktionen der Merkmale $p(x|c_i)$.
Sei $X = \bigcup_i X_i$ mit $\sum_{i=1}^{C} T_i = T$. Für jedes $x \in X$ werde ein Hyperkubus mit variablem Radius gewählt, der unabhängig von der Klasse K Stichprobenelemente einschließt und an x befestigt wird. Damit enthält ein Hyperkubus K_i Elemente der Klasse c_i.

Klassenbedingte Dichtefunktion:

$$p(x|c_i) = \frac{K_i}{T_i V}$$

Totale Dichtefunktion:

$$p(x) = \frac{K}{TV}$$

A priori Wahrscheinlichkeit:

$$P(c_i) = \frac{T_i}{T}$$

A posteriori Wahrscheinlichkeit:

$$P(c_i|x) = \frac{p(x|c_i)P(c_i)}{p(x)} = \frac{K_i}{K}$$

Nach der MAP-Strategie erfolgt die Zuordnung eines Vektors x zu der Klasse c_i, für die das Verhältnis $\frac{K_i}{K}$ maximal wird. Abbildung 3.18 zeigt das Beispiel eines 7NN-Klassifikators, der das aktuelle Datum der Klasse C_2 zuordnet.

Für $K = 1$ geht die KNN-Methode in die **Nächste-Nachbar-Regel** (NN-Regel) über, die einer Voronoi-Zerlegung (siehe auch Seite 127) des Merkmalsraumes entspricht. Hieraus ergibt sich eine stückweise lineare Trennfunktion.

Wie Abbildung 3.18 zeigt, tragen in der konkreten Situation nur die **Marginalen** (auch **Support-Vektoren** genannt) zur Entscheidung bei. Das Verfahren ist deshalb auch sehr sensibel bzgl. Ausreißern. Bei SVM (Support-Vektor-Maschinen) werden nur die Marginalen betrachtet.
Generell gilt auch bei den Verfahren der KNN-Klassifikation, dass die kritische Rolle des Glättungsparameters (K) nicht zu einfach angebbaren Lösungen führt.

Glättungsparameter war im Histrogramm die Urnenbreite, bei den Kernmethoden die Kernbreite und beim KNN-Verfahren der Wert von K. Auch in neuralen Netzen findet man derartige Glättungsparameter:

RBF-Netz:	Breite der radialen Basisfunktionen
MLP:	Anzahl der verdeckten Knoten und Verbindungen

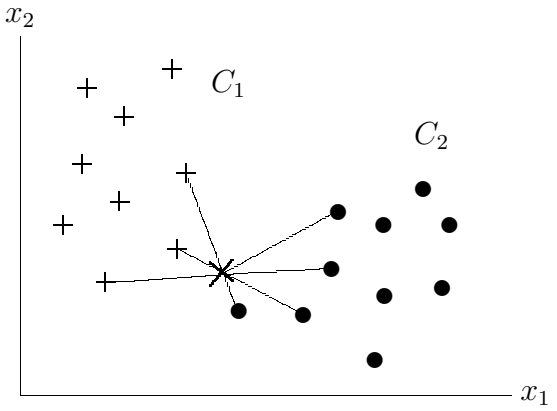


Abbildung 3.18: Beispiel einer Klassifizierung nach 7NN-Regel.

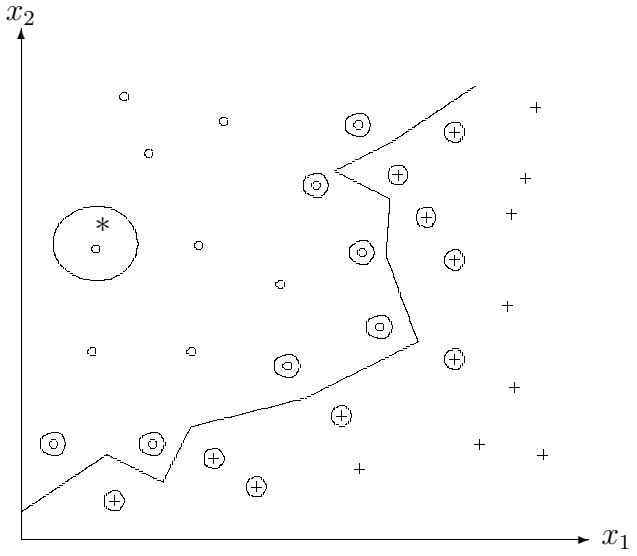


Abbildung 3.19: Support-Vektoren.

Es gilt: Überglättung der Modelldichte führt zu einem Anwachsen des Bias (Fehlers) und damit zu relativ unrealistischen Schätzungen. Andererseits führt eine unzureichende Glättung zu einem Anwachsen der Varianz, so dass die Modelldichte stark verrauscht wird und sehr sensitiv gegenüber neuen Daten ist. Die Suche nach einer Balance zwischen Bias und Varianz wird in Kapitel 4 behandelt.

3.9.3.1 Kullback-Leibler-Distanz

Schließlich soll noch eine Methode angesprochen werden, die es gestattet den Abstand zwischen Likelihoodfunktionen zu bestimmen. Dies würde z.B. der Optimierungsaufgabe dienen:
Wähle den Glättungsparameter so, dass $\|\widehat{p}(x) - p(x)\|$ minimal wird.

Das Modell der Daten ist gegeben durch $p(x, r) = p(r|x)p(x)$. Unter der Annahme von T unabhängigen Daten ist die Likelihood gegeben durch:

$$\begin{aligned}\widehat{L}(x, r) &= \prod_{t=1}^{T} p(x^t, r^t) \\ &= \prod_{t=1}^{T} p(e^t) \text{ mit } e^t := x^t - r^t \\ &= \prod_{t=1}^{T} p(r^t|x^t)p(x^t)\end{aligned}$$

Hieraus folgt der Ausdruck für den negativen Logarithmus der Likelihoodfunktion:

$$-\log \widehat{L} = -\sum_{t=1}^{T} \log p(r^t|x^t) - \sum_{t=1}^{T} \log p(x^t)$$

Von Interesse ist lediglich der zweite Summenausdruck, da dieser die Dichtefunktion $p(x^t)$ enthält. Ein Maß für die Übereinstimmung von praktischer und wahrer Dichtefunktion ist der Erwartungswert:

$$\begin{aligned}\mathrm{E}\left\{-\log \widehat{L}\right\} &= -\lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} \log \widehat{p}(x^t) \\ &= -\int p(x) \log \widehat{p}(x) dx \qquad (3.8)\end{aligned}$$

Im Grenzübergang unendlich großer Stichproben geht die Summe über die Stichproben in ein Integral über und der arithmetische Mittelwert in das mit der wahren Verteilung der Daten gewichtete Mittel.
Wenn $\widehat{p}(x) = p(x)$ gilt, dann gilt ebenfalls:

$$H(x) = \mathrm{E}\left\{-\log \widehat{L}\right\} = -\int p(x) \log p(x) dx$$

Dies ist die **Entropie** $H(x)$ von $p(x)$ als Maß der Verbreiterung (Gestalt). Im Falle der Gleichverteilung ist die Entropie maximal. Wird diese intrinsische Eigenschaft vom Erwartungswert (3.8) subtrahiert, erhält man die **Kullback-Leibler-Distanz** (oder **asymmetrische Divergenz**) d_{KL} der Dichtefunktionen:

$$d_{KL}(p|\hat{p}) = -\int p(x) \log \left(\frac{\widehat{p}(x)}{p(x)}\right) dx$$

Es gilt $d_{KL} \geq 0$ mit $d_{KL} = 0$ für $\widehat{p}(x) = p(x)$.
Da aber $p(x)$ tatsächlich nicht bekannt ist, bleibt die Wahl des optimalen Glättungsparameters (ob K, Kernbreite oder Urnenbreite) prinzipiell ein schwieriges Problem.

Die KL-Distanz kann Anwendung finden, wenn zwei Dichtefunktionen verglichen werden sollen. Eine spezielle Anwendung liefert die gemeinsame (mutual) Information zweier stochastischer Variabler. Seien x und y die Realisierungen der beiden stochastischer Variabler X und Y, und sei $p(x, y)$ die Verbunddichtfunktion. Wenn X und Y stochastisch unab"hangig sind, kann die Verbunddichtefunktion in ihre Marginalen faktorisiert werden. Es gilt dann $p(x, y) = p(x) \cdot p(y)$. Die gemeinsame Information, "$I(x, y)$, zwischen den Realisierungen x und y wird durch die KL-Distanz beschrieben.

$$I(x, y) = d_{KL}(p(x, y)|p(x)p(y)) = -\iint p(x, y) \log \left(\frac{p(x)p(y)}{p(x, y)} \right) dxdy$$

Es gilt $I(x, y) \geq 0$, und es gilt $I(x, y) = 0$ genau dann, wenn x und y unabhängig sind. Die Anwendung von Summen- und Produktregel für Wahrscheinlichkeiten liefert

$$I(x, y) = H(x) - H(x|y) = H(y) - H(y|x),$$

wobei $H(x|y)$ und $H(y|x)$ bedingte Entropien sind. Diese Beziehung hat folgende intuitive Interpretationen:

- Die Unschärfe von x wird durch die Kenntnis von y (und umgekehrt) reduziert.

- Aus einer Bayes-Perspektive: $p(x)$ kann als a priori Verteilung von x angegeben werden. Dann ist $p(x)|y$ die a posteriori Verteilung nachdem die neuen Daten y beobachtet wurden.

Die gemeinsame Information beschreibt somit die Reduktion der Unschärfe von x als Folge der Beobachtung von y.
Anwendungsbeispiel: Vergleich zweier MRI-Bilder, die mit unterschiedlichen Modalitäten genommen wurden, um sie räumlich anzupassen (Registrierung).

3.10 Dimensionsreduktion des Merkmalsraumes

Der Anwender ist in der Praxis oft geneigt, zur Beschreibung der Struktureigenschaften seine Stichprobe einen möglichst umfangreichen Satz von Merkmalen zu verwenden. Er verbindet damit die Hoffnung, gewünschte Klassentrennungen zu erleichtern. Diese naive Vorgehensweise führt gewöhnlich nicht zum Erfolg. Hierfür gibt es drei Gründe:

1. Die verwendeten Merkmale sollten unabhängig sein. Im Fall der Normalverteilung bedeutet dies Orthogonalität der Merkmale. Die Verwendung von abhängigen Merkmalen vergrößert nur die Dimension des Merkmalsraumes, ohne zur Klassentrennung beizutragen. Es gibt statistische Verfahren der **Merkmalsselektion**, um die möglichen Beziehungen zwischen Merkmalen aufzudecken und die Stichprobenbeschreibung auf eine Menge unabhängiger Merkmale zu begrenzen. Dieses Problem wird hier nicht weiter behandelt.

2. Die Chance der linearen Lösbarkeit eines Entscheidungsproblems reduziert sich oft mit zunehmender Dimension des Merkmalsraumes. Erinnert sei an die lineare Lösbarkeit n-dimensionaler Boolescher Funktionen (siehe Abschnitt 2.6.1). In anderen Fällen führt die Erhöhung der Dimension des Merkmalsraumes zu einer linear entscheidbaren Repräsentation eines nichtlinearen Problems. In Abschnitt 2.6 sind zwei neuronale Architekturen vorgestellt worden, das MLP und das HON, welche ein Problem so umkodieren, dass es sich als lineares Entscheidungsproblem darstellt.

3. Es wurde bereits mehrfach auf das Problem der Dimensionalität hingewiesen: Die Zahl notwendiger Stichproben zum Erlernen der Entscheidungsfunktion wächst exponentiell mit der Dimension des Merkmalsraumes.

Das Problem der Begrenzung oder Reduktion der Dimension des Merkmalsvektors stellt sich insbesondere, wenn **Bilddaten** klassifizieren werden sollen.
Bereits ein kleiner Ausschnitt von 30×30 Bildpunkten stellt einen Merkmalsvektor der Grauwerte von der Dimension 900 dar. Es ist deshalb interessant, einige Struktureigenschaften durch Vorverarbeitung (Filterung) herauszuarbeiten, so dass mit Unterabtastung gearbeitet werden kann, z.B. Verwendung jedes zweiten Bildpunktes reduziert die Dimension bereits um den Faktor 4.

In diesem Abschnitt werden zwei **lineare Verfahren** zur **Reduktion der Dimension des Merkmalsraumes** vorgestellt, die eine Transformation $Y = AX$ des Merkmalsraumes X auf den Merkmalsraum Y durchführen. Das heißt, die Merkmale werden dergestalt verändert, dass mit der Transformation eine Reduktion der Dimension des Merkmalsraumes eintritt bzw. durchgeführt werden kann. Unter dem Namen **Projection Pursuit** faßt man eine generelle Strategie für solche (auch nichtlineare) Transformationen zusammen.

A) In Abschnitt 3.10.1 wird die **Karhunen-Loeve-Transformation** vorgestellt. Das ist eine stochastische Eigenwerttransformation der Kovarianzmatrix. Dieses Verfahren stellt eine Rotation des Merkmalsraumes dar, so dass eine gewisse Menge von Merkmalen vernachläßigt werden kann. Das Verfahren besteht in zwei Schritten:

   1. Dekorrelation der Merkmale

   2. Abschneiden der Merkmale mit kleinen Eigenwerten.

B. In Abschnitt 3.10.2 wird die Idee von **Fisher's lineare Diskriminanzanalyse** vorgestellt. Dieses Verfahren beantwortet die Frage nach dem Vorliegen einer Richtung im Merkmalsraum, die zu bester Klassentrennung bei kompaktester Repräsentation der Klassen führt. Die mehrdimensionalen Daten werden auf ein einziges zu berechnendes Merkmal projiziert.

3.10.1 Karhunen-Loeve-Transformation

Die Karhunen-Loeve-Transformation (**KLT**) ist seit langer Zeit in der statistischen Mustererkennung bekannt. Es existieren für leicht unterschiedliche Varianten eine Reihe von Namen: Faktoranalyse, Hauptkomponenten-Transformation (**PCA** - principal component analysis), Hotelling-Transformation.
Es existieren zwei unterschiedliche Definitionen der KLT, die zum gleichen Algorithmus führen:

- Pearson (1901): Die KLT ist eine lineare Projektion eines mehrdimensionalen Merkmalsraumes. Es werden die mittleren Projektionskosten , definiert als den mittleren quadratischen Abstand zwischen den Datenpunkten und ihren Projektionen minimiert.
- Hotelling (1933): Die KLT ist eine orthogonale Projektion der Daten auf einen linearen Teilraum reduzierter Dimension, so dass die Varianz der projezierten Daten maximiert wird.

Offensichtlich besteht eine enge Verwandtschaft zur Regressionsaufgabe nach dem „total least square“-Verfahren, wenn der lineare Teilraum als die gesuchte Regressionsfunktion interpretiert wird.

Die KLT stellt eine **stochastische Eigenwerttransformation** dar, die durch Diagonalisierung der Kovarianzmatrix realisiert wird.
In den Unterlagen zu den Vorlesungen „Computer Vision I“ und „Computer Vision II“ findet sich eine ausführliche Beschreibung des Beispiels einer KLT für einen zweidimensionalen Merkmalsvektor. Dieses Beispiel wird hier kurz zusammengefaßt, um das Verfahren der Diagonalisierung der Kovarianzmatrix einzuführen. Anschließend wird eine Verallgemeinerung auf n Dimensionen vorgenommen.

Maximierung der Varianz:
Sei $X := \{x^1, ...x^T\}$ eine Menge von Datenvektoren $x^t \in \mathbb{R}^n$. Ziel sei die Projektion der Daten auf einen Unterraum der Dimension $k < n$, so dass die Varianz der projezierten Daten maximiert wird.

Annahme: $k = 1$
Die Richtung u_1 des 1-dimensionalen Unterraumes ist gegeben durch $\frac{u_1}{||u_1||}$, $u_1 \in \mathbb{R}^n$. Da hier nur die Richtung von u_1 interessant ist, wurde u_1 als ein Einheitsvektor definiert. Es gilt demnach $u_1^T u_1 = 1$.
Die Projektion des Datums x^t auf diesen Einheitsvektor ist durch das Skalarprodukt $< u_1, x^t >= u_1^T x^t$ betragsmäßig gegeben.

Ist

$$m := \frac{1}{T} \sum_{t=1}^{T} x^t$$

der (unverzerrt geschätzte, empirische) Stichprobenmittelwert, so ist dessen Projektion $< u_1, m >= u_1^T m$. Ist

$$\Sigma := \frac{1}{T-1} \sum_{t=1}^{T} (x^t - m)(x^t - m)^T$$

die (unverzerrt geschätzte, empirische) Kovarianzmatrix, so folgt für die (unverzerrt geschätzte, empirische) Varianz der projezierten Daten X' (da u_1 1-dimensional ist, existiert nur noch die Varianz)

$$\mathrm{Var}\{X'\} = \frac{1}{T-1} \sum_{t=1}^{T} (u_1^T x^t - u_1^T m)^2 = u_1^T \Sigma u_1.$$

Es folgt die Maximierung der projezierten Varianz $u_1^T \Sigma u_1$ in Bezug auf die Wahl einer geeigneten Richtung des Einheitsvektors u_1:

Die Nebenbedingung $u_1^T u_1 = 1$ verhindert $||u_1|| \to \infty$ als Lösung. Hieraus folgt eine Maximierung unter Nebenbedingungen, die mit Hilfe des Lagrange-Multiplikatorsatzes gelöst werden kann.
Sei $\lambda_1 \in \mathbb{R}$ der benötige Lagrange-Multiplikator, dann ist

$$u_1^T \Sigma u_1 - \lambda_1 (1 - u_1^T u_1)$$

die zu maximierende Zielfunktion.
Nullsetzen der ersten Ableitung nach u_1 liefert als stationäre Lösung

$$\Sigma u_1 = \lambda_1 u_1 \ ,$$

d.h., u_1 muss ein Eigenvektor von Σ sein.
Die Multiplikation der Eigenwertgleichung von links mit u_1^T und die Anwendung von $u_1^T u_1 = 1$ ergibt für die maximale Varianz

$$\mathrm{Var}_{max}\{X'\} = u_1^T \Sigma u_1 \ .$$

Die Varianz der projezierten Daten wird also maximiert, wenn u_1 ein Eigenvektor der Kovarianzmatrix Σ ist. Die maximale Varianz ist gleich dem (größten) Eigenwert λ_1. Man bezeichnet diesen Eigenvektor als die erste Hauptkomponente.

Annahme: $k > 1$
Zusätzliche Hauptkomponenten können durch die Annahme, dass jede zusätzliche Richtung diejenige ist, welche die projezierte Varianz im Vergleich zu allen möglichen Richtungen (die zu den bereits betrachteten orthogonal sind) maximiert, berechnet werden.

Wenn $k = n$ gilt, erhält man als Lösung die n Eigenvektoren $u_1, ..., u_n$ der Kovarianzmatrix Σ, welche die entsprechenden n Eigenräume zu den n Eigenwerten $\lambda_1, ..., \lambda_n$ aufspannen (da die Kovarianzmatrix symmetrisch ist, gilt $\lambda_i \in \mathbb{R}^+$).

Hauptkomponentenanalyse bedeutet also:

- Bestimmung des Stichprobenmittels m.
- Bestimmung der Stichprobenkovarianzmatrix Σ.
- Finden der n Eigenvektoren von Σ, die den n größten Eigenwerten entsprechen.

Dekorrelation mittels Mini-KLT
Die Struktur eines Bildsignals wird durch die Menge aller möglichen Wechselwirkungen der Grauwerte $g \in \{0, ..., G-1\}$ an unterschiedlichen Positionen bestimmt. Wenn allein die Dichtefunktion der Merkmale $p(g)$ betrachtet wird, so verzichtet man bewußt auf die Beschreibung solcher Wechselwirkungen.
Verfahren, die auf einer 1-dimensionalen Dichtefunktion $p(g)$ beruhen, werden einer **Statistik 1.Ordnung** zugerechnet. Tatsächlich sind aber auch Vektormerkmale $x = (x_1, ..., x_n)^T$ von Interesse. Eine Dichtefunktion $p(x)$ für den n-dimensionalen Fall erlaubt Verfahren, die einer **Statistik n. Ordnung** zugerechnet werden.

Hier erfolgt eine Beschränkung auf eine Statistik 2.Ordnung, um eine Bildstruktur zu charakterisieren. Es werden die Wechselwirkungen der Grauwerte an zwei Punkten berücksichtigt. Diese Punkte seien der **Aufpunkt** $A = (m, n)$ und der um den Vektor $\tau = (\xi, \eta)$ verschobene **Testpunkt** $T := (m + \xi, n + \eta)$. Die bedingte Dichtefunktion $p(g_A, g_T|\tau)$ wird durch ein **zweidimensionales Histogramm** approximiert. Da die Rolle von Aufpunkt und Testpunkt getauscht werden können, wird das Histogramm symmetrisiert und man nennt $h(g_A, g_T\,;\,\tau)$ eine **Cooccurrence-Matrix**. Je nach Wahl von τ sind die Grauwerte g_A und g_T mehr oder weniger korreliert. Dies drückt sich darin aus, dass sich die Häufigkeitsverteilung um die Diagonale der Matrix ansiedelt. Projiziert man die Cooccurrence-Matrix auf die Achsen g_A oder g_T, erhält man die (identischen) 1D-Histogramme (Marginale):

$$h(g) = \sum_{g_A} h(g_A, g_T; \tau) = \sum_{g_T} h(g_A, g_T; \tau)$$

Seien X_1, X_2 zwei identische stochastische Variable mit den Realisierungen x_1, x_2. Es gelte $h(x_1) = h(x_2)$ und $h(x_1, x_2)$ sei die Cooccurrence-Matrix. Dann sind die ersten beiden Momente der Statistik 1. Ordnung:

Mittelwert:

$$\mu = E\{X_1\} = E\{X_2\} = \int_{-\infty}^{\infty} x_1 p(x_1) dx_1 = \int_{-\infty}^{\infty} x_2 p(x_2) dx_2$$

Varianz:

$$\begin{aligned}\sigma^2 &= \sigma_{11} = \sigma_{22} = E\left\{(X_1 - E\{X_1\})^2\right\} = E\left\{(X_2 - E\{X_2\})^2\right\} \\ &= \int_{-\infty}^{\infty} (x_1 - E\{X_1\})^2 p(x_1) dx_1 = \int_{-\infty}^{\infty} (x_2 - E\{X_2\})^2 p(x_2) dx_2\end{aligned}$$

Sei

$$\mu_{12} := E\{X_1, X_2\} = \int_{-\infty}^{\infty}\int_{\infty}^{\infty} x_1 x_2 p(x_1, x_2) dx_1 dx_2$$

der Erwartungswert der Statistik 2.Ordnung. Dann definiert

$$\begin{aligned}\sigma_{12} = \sigma_{21} &:= E\{(X_1 - E\{X_1\})(X_2 - E\{X_2\})\} \\ &= \mu_{12} - E\{X_1\} E\{X_2\}\end{aligned}$$

die **Kovarianzen** der Beziehungen zwischen X_1 und X_2.
Tatsächlich werden diese Momente aus den Histogrammen $h(x_1) = h(x_2)$ und $h(x_1, x_2)$ geschätzt.

Die **Kovarianzmatrix**

$$\Sigma = \begin{pmatrix} \sigma_{11} & \sigma_{12} \\ \sigma_{21} & \sigma_{22} \end{pmatrix} = \sigma^2 R$$

mit $\sigma = \sigma_{11} = \sigma_{22}$ entspricht der auf die Varianz normierten **Autokorrelationsmatrix**

$$R = \begin{pmatrix} \rho_{11} & \rho_{12} \\ \rho_{21} & \rho_{22} \end{pmatrix}$$

mit $\rho_{11} = \rho_{22} = 1$. Die $\rho_{ij} = \frac{1}{\sigma^2}\sigma_{ij}$ heißen Korrelationskoeffizienten[5].

Eine **Diagonalisierung** der Autokorrelationsmatrix oder der Kovarianzmatrix wird durch Lösung der **Eigenwertgleichung**

$$\Sigma Y = \lambda Y$$

erreicht. Dabei ist $Y = (Y_1, Y_2)^T$ ein neuer Merkmalsvektor, der durch Rotation der Variablen $X_1 - \mu$ und $X_2 - \mu$ um 45^o entsteht. Das neue Basissystem zur Beschreibung der Struktur der Daten wird durch die **Eigenvektoren** der Kovarianzmatrix gebildet. Die Eigenwertgleichung ist so zu lesen, dass die Eigenvektoren unter der Wirkung der Kovarianzmatrix invariant sind bis auf den Vektor der **Eigenwerte** $\lambda = (\lambda_1, \lambda_2)$.
Die Eigenvektoren erhält man aus

$$Y_1 = \begin{pmatrix} \frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} X_1 - \mu \\ X_2 - \mu \end{pmatrix}$$

$$Y_2 = \begin{pmatrix} \frac{1}{\sqrt{2}} & 0 \\ 0 & -\frac{1}{\sqrt{2}} \end{pmatrix} \begin{pmatrix} X_1 - \mu \\ X_2 - \mu \end{pmatrix}$$

[5]Der Kehrwert der Varianz $\frac{1}{\sigma}$ heißt Präzision.

Bezogen auf unser ursprüngliches Problem der Analyse der Grauwerte g_A und g_T an Aufpunkt A und Testpunkt T kann die Realisierung des Eigenvektors Y_1 als **Summengrauwert** g_S bezeichnet werden, da gilt:

$$g_S = \frac{1}{\sqrt{2}}(g_A + g_T)$$

Die Realisierung des Eigenvektors Y_2 kann als **Differenzgrauwert** g_D bezeichnet werden, da gilt:

$$g_D = \frac{1}{\sqrt{2}}(g_A - g_T)$$

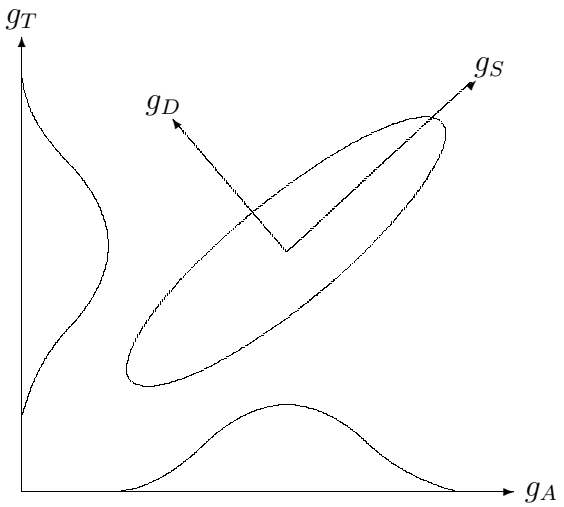


Abbildung 3.20: Hauptachsentransformation der Cooccurrence-Matrix und ihre marginalen Häufigkeiten.

Die Varianzen dieser neuen Merkmale sind nun natürlich nicht mehr gleich. Sie entsprechen den Eigenwerten:

$$\sigma_S^2 = \sigma^2(1+\rho)$$
$$\sigma_D^2 = \sigma^2(1-\rho)$$
$$\text{mit } \rho_{12} = \rho_{21} = \rho$$

Die **diagonalisierte Kovarianzmatrix** im neuen Koordinatensystem (g_S, g_D) lautet nun:

$$\Sigma_{KLT} = \begin{pmatrix} \sigma_S^2 & 0 \\ 0 & \sigma_D^2 \end{pmatrix}$$

Wie die Diagonalstruktur ausdrückt, sind die Merkmale g_S und g_D **dekorreliert**.

Außerdem sind die Varianzen (σ_S^2, σ_D^2) der Größe nach geordnet, wobei die größte Varianz σ_S^2 ist.

Approximation mit minimalem mittleren, quadratischem Abstand:
Dieses Beispiel wird nun auf den Fall einer n-dimensionalen stochastischen Variablen $X = (X_1, ..., X_n)^T$ verallgemeinert. Dabei wird aber wieder lediglich die Statistik 2.Ordnung berechnet. Es sei o.B.d.A. $E\{X\} = \mu = 0$ angenommen, so dass für die Kovarianzmatrix gilt $\Sigma_X = E\left\{XX^T\right\}$.
Im obigen Beispiel wurde eine Rotation (lineare, orthogonale Transformation) auf der Variablen X ausgeführt. Dies soll auch hier geschehen:

$$Y = A^T X = \sum_{i=1}^{n} a_i^T X_i$$

mit A ist eine orthogonale $n \times n$-Matrix (nicht-singulär und reell) und a_i^T sei der i-te Spaltenvektor von A^T. Wegen der Linearität der Abbildung gilt auch für die neue stochastische Vektorvariable $E\{Y\} = 0$. Außerdem folgt aus der Orthogonalität der Transformation $A^T = A^{-1}$. Demzufolge ist die obige Gleichung einfach invertierbar:

$$X = AY = \sum_{i=1}^{n} a_i Y_i$$

Die **Diagonalisierung der Kovarianzmatrix** Σ_X läßt sich nun formulieren als:

$$\begin{aligned} \Sigma_Y &= E\left\{YY^T\right\} = E\left\{A^T X X^T A\right\} \\ &= A^T E\left\{XX^T\right\} A = A^T \Sigma_X A \end{aligned}$$

Die Spalten der Matrix A sind die Eigenvektoren von Σ_X. Es gilt die Eigenwertgleichung:

$$\Sigma_X a_i = \lambda_i a_i, \quad i = 1, ..., n$$

Da Σ_X symmetrisch ist, wird A durch orthogonale Einheitseigenvektoren aufgespannt. Die Matrix Σ_Y ist eine Diagonalmatrix mit den Varianzen $\sigma_{ii}^2 > 0$ als Eigenwerte, wobei gilt $\sigma_{ii}^2 > \sigma_{(i+1)(i+1)}^2$.
Damit führte die multidimensionale Rotation von X mit der Matrix A zu einem Merkmalsvektor Y, dessen Varianzen in den einzelnen Komponenten maximal unterschiedlich sind.

Nachdem der erste Schritt, die **Dekorrelation der Merkmale**, durchgeführt wurde, wird nun der zweiten Schritt, das **Abschneiden der kleineren Eigenwerte** und damit die Reduktion der Dimension des Merkmalsvektors Y auf $k < n$ Komponenten behandelt.

Sei $Y = \left(Y_{(k)} : Y_{(n-k)}\right)^T = (Y_1, ..., Y_k, Y_{k+1}, ..., Y_n)^T$ eine Partitionierung von Y in zwei Teilvektoren der Länge k (d.h. $Y_{(k)}$) bzw. der Länge $(n-k)$ (d.h. $Y_{(n-k)}$). Sei außerdem die Matrix A in zwei Teilmatrizen $A_{(k)}$ bzw. $A_{(n-k)}$ so zerlegt, dass

$$A = \left(A_{(k)} : A_{(n-k)}\right)$$

die Aneinanderkopplung dieser aus k bzw. $(n-k)$ Spalten bestehenden Teilmatrizen bedeutet. Wegen $X = AY$ folgt:

$$X = \left(A_{(k)} : A_{(n-k)}\right) \begin{pmatrix} Y_{(k)} \\ Y_{(n-k)} \end{pmatrix}.$$

Werden nun die letzten Merkmalsvektoren $Y_{(n-k)}$ abgeschnitten, so repräsentiert

$$X_{(k)} = A_{(k)} Y_{(k)}$$

die ursprünglichen Merkmale im n-dimensionalen Merkmalsraum X als Approximation durch k dekorrelierte Merkmale. Der dabei entstehende Fehler ist

$$\epsilon_{(k)} = X - X_{(k)} = \sum_{i=k+1}^{n} a_i Y_i \quad \text{mit } \epsilon_{(k)}^2 = \sum_{i=k+1}^{n} Y_i^2$$

wegen der Orthonormalität der Eigenvektoren.

Für den Erwartungswert des quadratischen Fehlers gilt:

$$E\left\{\epsilon_{(k)}^2\right\} = \sum_{i=k+1}^{n} \left(E\left\{Y_i\right\}\right)^2 = \sum_{i=k+1}^{n} a_i^T \Sigma_X a_i = \sum_{i=k+1}^{n} \sigma_{ii}^2$$

Die σ_{ii}^2 sind die Varianzen von Y_i.
Die KLT liefert in den Merkmalen $Y_i, i = 1, ..., k$, eine **Dimensionsreduktion** der Repräsentation mit einem minimalen mittleren quadratischen Fehler für jedes $k < n$.

3.10.2 Fisher's lineare Diskriminanzanalyse

Das auf Fisher (1936) zurückgehende Verfahren liefert einen 1D-Merkmalsraum durch eine nach gewissen Gesichtspunkten optimale lineare Projektion eines n-dimensionalen Merkmalsraumes.
Seien $x \in \mathbb{R}^n$ ein Merkmalsvektor und $w \in \mathbb{R}^n$ ein Vektor justierbarer Gewichte.
Dann liefert das Skalarprodukt

$$y = w^T x$$

die Projektion von x auf w mit $y \in \mathbb{R}$. Der damit verbundene Verlust an Information wird durch **Optimierung unter Zwang** minimiert.

Annahme: 2-Klassen-Problem
Der Merkmalsraum X sei also partitioniert in $X = X_1 \cup X_2$. Aus den Stichprobenelementen $x^t \in X_i$ erhält man die klassenbedingten Populationsmittelwerte

$$\mu_1 = \frac{1}{T_1} \sum_{x^t \in X_1} x^t \text{ und } \mu_2 = \frac{1}{T_2} \sum_{x^t \in X_2} x^t$$

A) Optimierung: **Maximale Klassentrennung**
Gesucht ist die Lösung

$$|m_2 - m_1| = |w^T(\mu_2 - \mu_1)| \rightarrow \max \quad \text{mit } m_i = w^T\mu_i, \quad m_i \in \mathbb{R},\ \mu_i \in \mathbb{R}^n$$

unter dem **Zwang**:

$$\|w\| = \sum_{i=1}^{n} w_i^2 = 1$$

Die Zwangsbedingung des normierten Gewichtsvektors ist erforderlich, weil das Maximum auch durch wachsendes w erreicht würde, was keinen Sinn ergäbe.
Die optimale Lösung wird mit der **Methode der Lagrangeschen Multiplikatoren** erreicht.
Eine bessere Strategie stellt die folgende Variante dar.

B) **Optimierung: Gleichzeitig Maximierung der Interklassentrennung und Minimierung der Intraklassenstreuung**
Dies wird erreicht durch Maximierung einer Funktion, welche die Differenz der projizierten Klassenmittelwerte μ_1 und μ_2 repräsentiert und auf die Intraklassenstreuungen S_1 und S_2 entlang des Gewichtsvektors normiert ist.
Intraklassenstreuung:

$$\begin{aligned} S_i^2 &= \sum_{x^t \in X_i} (x^t - \mu_i)^2 \\ S^2 &= S_1^2 + S_2^2 \end{aligned}$$

Fisher's Kriterium:

$$J(w) = \frac{(m_2 - m_1)^2}{S^2}$$

Einsetzen von $|m_2 - m_1| = |w^T(\mu_2 - \mu_1)|$ und von S^2 liefert:

$$J(w) = \frac{w^T\Sigma_{\text{Inter}}w}{w^T\Sigma_{\text{Intra}}w} \quad \text{mit}$$

$$\begin{aligned} \Sigma_{\text{Inter}} &= (\mu_2 - \mu_1)(\mu_2 - \mu_1)^T \\ \Sigma_{\text{Intra}} &= \sum_{x^t \in X_1} (x^t - \mu_1)(x^t - \mu_1)^T + \sum_{x^t \in X_2} (x^t - \mu_2)(x^t - \mu_2)^T \end{aligned}$$

Die Richtung des Gewichtsvektors wird so gewählt, dass $J(w)$ maximiert wird.

notwendige Bedingung: $\frac{\partial J}{\partial w} = 0$
Hieraus folgt:

$$\left(w^T\Sigma_{\text{Inter}}w\right)\Sigma_{\text{Intra}}w = \left(w^T\Sigma_{\text{Intra}}w\right)\Sigma_{\text{Inter}}w$$

Und schließlich folgt **Fisher's lineare Diskriminanzfunktion**:

$$w \sim \Sigma_{\text{Intra}}^{-1}(\mu_2 - \mu_1)$$

$\Sigma_{\text{Intra}}^{-1}$ spielt hier die Rolle einer **Metrik**, ähnlich wie bei der Mahalanobis-Distanz. Sind die Datencluster isotrop, so liegt w parallel zu $\Delta\mu = \mu_2 - \mu_1$.
Ist schließlich der optimale Gewichtsvektor gefunden, liefert $y = w^T x$ den neuen Merkmalsraum. Das Problem besteht nun darin, eine Schwelle y_0 zu finden, so dass das 2-Klassen-Problem gelöst werden kann. Das heißt:

$$e : \begin{cases} j = 1 \quad (x \in X_1), & \text{wenn } y(x) \geq y_0. \\ j = 2 \quad (x \in X_2), & \text{sonst.} \end{cases}$$

Das neue Merkmal $y = w^T x$ kann als Summe von Zufallsvariablen interpretiert werden. Aus dem zentralen Grenzwertsatz der Wahrscheinlichkeitsrechnung folgt dann, dass die Dichtefunktionen $p(y|c_i)$ Normalverteilung annehmen. Mit der in Abschnitt 3.8 eingeführten Methode der ML-Schätzung der Dichtefunktionen kann auch auf die optimale Schwelle y_0 geschlossen werden.
Sind der geeignete Gewichtsvektor und die Schwelle gefunden, dann entspricht die Klassenzuordnung eines unbekannten Merkmalsvektors dem im Fall des Perzeptrons behandelten Entscheidungsproblem. Von größerer Bedeutung als diese Analogie ist aber die erreichte Dimensionsreduktion des Merkmalsraumes.

Verallgemeinerung auf das Mehrklassenproblem:
Sei $n > K$, d.h. die Dimension des Merkmalsraumes ist größer als die Anzahl der Klassen. Dann ist eine Dimensionsreduktion auf $1 < n' < n$ möglich, also auf eine gewisse Anzahl linearer Merkmale:

$$y_k = w_k^T x, \quad k = 1, ..., n'$$

Mit $y = (y_1, ..., y_{n'})^T$ folgt hieraus $y = Wx$.
Fisher's Kriterium, das zu maximieren ist, ist nun

$$J(W) = \text{tr}\left\{ \left(W\Sigma_{\text{Intra}}W^T\right)^{-1} \left(W\Sigma_{\text{Inter}}W^T\right) \right\}$$

und aus $\frac{\partial J}{\partial w_{ij}} = 0$ folgt das Gleichungssystem für die optimale Schätzung der Gewichtsmatrix W. Diese Lösung ist bestimmt durch die n' maximalen Eigenvektoren von $\Sigma_{\text{Intra}}^{-1}\Sigma_{\text{Inter}}$.

Im Fall des Mehrklassenproblems besteht also eine Analogie zur Karhunen-Loeve-Transformation aus Abschnitt 3.10.1.

Wie ist die Dimension zu wählen?
Da der Rang von Σ_{Inter} höchstens $K - 1$ ist, können höchstens $K - 1$ Eigenwerte ungleich Null sein. Die Projektion der Merkmalsvektoren auf einen $(K - 1)$-dimensionalen Unterraum, der von den Eigenvektoren von Σ_{Inter} aufgespannt wird, ändert nicht den Wert von $J(W)$. Also können auf diesem Weg höchstens $K - 1$ lineare Merkmale y_k gefunden werden.

4 Mehrschicht-Netze (Multi-Layer-Perzeptron)

In Kapitel 2 wurde die Architektur eines Mehrschicht-Netzes eingeführt, um nichtlineare Probleme zu lernen. Betrachtet wird dort das XOR-Problem, das mit einem recht einfachen zweischichtigen Netz, bestehend aus drei Perzeptrons, gelernt werden konnte.

Problemspezifisch können Mehrschicht-Netze unterschiedliche Topologien besitzen. Es gibt aber kein Verfahren, um systematisch a priori eine geeignete Topologie zu wählen, die ein gegebenes Problem optimal löst. Vielmehr ist diese durch **Versuch und Irrtum** (trial and error) herauszufinden. Dafür stehen verschiedene Techniken zur Verfügung, die das Hinzufügen und Herausschneiden einzelner Neuronen, ganzer Schichten oder gewichteter Verbindungen zu bewerten gestatten, siehe Abschnitt 4.5.

Nach den Ausführungen in Kapitel 2 ist bekannt, dass beim überwachten Training eines Perzeptrons in folgender Weise vorgegangen wird: Der Fehler ϵ zwischen tatsächlicher Ausgabe y und geforderter Ausgabe r, $\epsilon = r - y$, wird rückgekoppelt, um durch die Änderung der Gewichte den Ausgabefehler zu reduzieren.

Im Fall eines Mehrschicht-Netzes entsteht hierbei das Problem, dass nur die Ausgabe der letzten Schicht abgreifbar und mit der geforderten Antwort vergleichbar ist. Eigentlich wird zum sinnvollen Training der Neuronen in einer **verdeckten Schicht** auch deren Vergleich mit einer geforderten Ausgabe benötigt. Diese ist gewöhnlich unbekannt. Die Lösung dieses Problems ist das Training eines Mehrschichtnetzes nach dem Backpropagation-Verfahren. Es wurde in den 1970er Jahren vorgeschlagen (in Werbos (1974)). Aber erst das Buch „Parallel Distributed Processing: Explorations in the Microstructure of Cognition“ (siehe Rumelhart, Hinton und Williams (1986)) führte zum Durchbruch dieses Lernverfahrens und damit der Architektur der Mehrschicht-Netze. Meist werden in diesen Netzen modifizierte Perzeptrons als Neuronen verwendet. Deshalb ist im Englischen auch der Begriff „multilayer perceptron“ (MLP) gebräuchlich.

In diesem Kapitel werden folgende Themen behandelt:

- Backpropagation-Lernen
- Mehrschichtnetze und Regression
- Generalisierung in Mehrschichtnetzen und Bias-Varianz-Dilemma
- Bildkomprimierung mittels Mehrschichtnetzen.

4.1 Lernen in Mehrschichtnetzen, Backpropagation

Dem Backpropagation-Lernen eines Mehrschichtnetzes liegt ein **überwachtes Training** im **Batchmode-Verfahren** zu Grunde. Es wird das in Kapitel 2 eingeführte **Gradientenabstiegsverfahren** (steepest descent) auf der Fehlerkurve im Gewichtsraum angewendet. Das heißt, es wird ein **quadratisches Fehlermaß** verwendet.
Außerdem kommt eine sigmoide Aktivierungsfunktion, $f_\sigma(u)$, zur Anwendung:

$$f_\sigma(u) = (1 + \exp(-\beta u))^{-1} \quad \text{mit } \beta > 0, \text{ meist } \beta = 1$$

Würde nämlich wie im Fall des Perzeptrons eine harte Schwellenfunktion (z.B. Stufenfunktion oder Signumfunktion) verwendet werden, entstünde das **Credit Assignment Problem**: Es besteht keine Möglichkeit zu bestimmen, welcher der verdeckten Knoten (Neuronen) für den Fehler eines Ausgabeneurons verantwortlich gemacht werden kann und dessen Gewichte folglich geändert werden müßten.

Ist aber die Aktivierungsfunktion **differenzierbar**, dann läßt sich ein Zusammenhang zwischen dem Fehler und dem Gewicht über den Gradientenabstieg im Gewichtsraum herstellen. Dazu muß die Aktivierungsfunktion nach dem Argument (dem Bild der Propagierungsfunktion) abgeleitet werden. Für die Sigmoid-Funktion erhält man den einfachen Zusammenhang:

$$f'_\sigma(u) \equiv \frac{d}{du} f_\sigma(u) = f_\sigma(u)(1 - f_\sigma(u))$$

Also für $\beta = 1$:

$$f'_\sigma(u) = \frac{\exp(-u)}{(1 + \exp(-u))^2}$$

Neben der Sigmoid-Funktion kommt auch häufig der Tangens-Hyperbolikus, $f_{th}(u)$, zur Anwendung:

$$f_{th}(u) = \tanh(u) = \frac{\exp(u) - \exp(-u)}{\exp(u) + \exp(-u)}$$

Beide Funktionen lassen sich durch eine lineare Funktion in einander überführen.

4.1.1 Notation

Das Backpropagation-Verfahren wird für ein MLP als vorwärts gerichtetes Netz mit dem Aufbau aus Abbildung 4.1 hergeleitet.

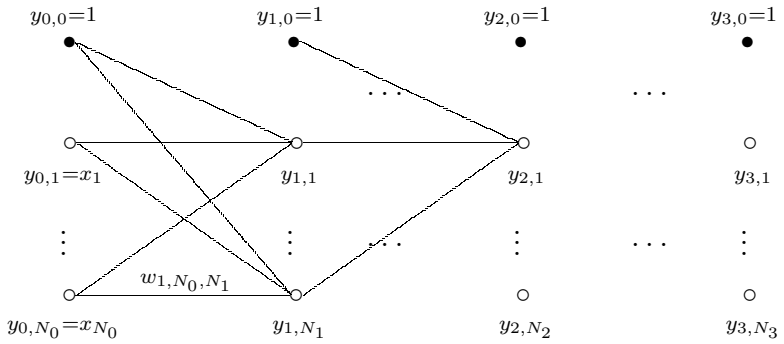


Abbildung 4.1: Architektur eines MLP.

Es wird folgende Notation verwendet:

N_ℓ Zahl der Knoten in Schicht ℓ

L Zahl der Schichten

P Zahl der Trainingsvektoren

Ω_P Stichprobenmenge

x^p p-ter Trainingsvektor

$y_{\ell,j}$ Ausgabe des j-ten Knotens in Schicht ℓ, $0 \leq \ell \leq L$, $0 \leq j \leq N_\ell$

$y_{0,j}$ j-te Komponente des Eingabevektors ($y_{0,j} = x_j$)

$w_{\ell,i,j}$ Gewicht, welches den i-ten Knoten in Schicht $\ell-1$ mit dem j-ten Knoten in Schicht ℓ verbindet

$r_j(x^p)$ Geforderte Antwort des j-ten Ausgabeknotens für den p-ten Trainingsvektor

Die Architektur entspricht dem erweiterten Perzeptronmodell mit der konstanten Eingabe $y_{\ell,0} = 1$ und dem (negativen) Schwellwert $w_{\ell,i,0}$.

Phase 1:	Vorwärts-Propagierung
Phase 2:	Bestimmung des Fehlers
Phase 3:	Rückwärts-Propagierung

Tabelle 4.1: Backpropagation Trainingsphasen.

4.1.2 Herleitung des Backpropagation-Verfahrens

Das Training eines MLP durchläuft iterativ drei Phasen (siehe Tabelle 4.1) für jeweils die komplette Stichprobe oder für einen einzelnen Stichprobenvektor:

1. Vorwärts-Propagierung
Die Ausgabe der Vorgängerschicht bildet die Eingabe der Nachfolgeschicht. Die Eingabedaten des Trainingsvektors x^p mit $x_j^p = y_{0,j}^p$ durchlaufen die L Schichten, bis die Ausgabe $y_{L,j}^p$ am Knoten j der Ausgabeschicht anliegt.

Die Ausgabe des Knotens j in der Schicht ℓ ist gegeben durch:

$$y_{\ell,j} = f_\sigma(u_{\ell,j}) = f_\sigma\left(\sum_{i=0}^{N_{\ell-1}} w_{\ell,i,j} y_{\ell-1,i}\right)$$

2. Bestimmung des Fehlers
Für jede Eingabe x^p des Stichprobenelementes (x^p, r^p) wird die Summe e^p der quadratischen Fehler über alle Ausgabeknoten $y_{L,j}$ durch Vergleich mit den geforderten Antworten $r_j^p(x^p)$ berechnet:

$$e^p(w) = \frac{1}{2}\sum_{j=1}^{N_L}(y_{L,j}(x^p; w) - r_j^p(x^p))^2$$

Der Ausdruck $y(x; w)$ drückt aus, dass die berechnete Ausgabe nicht nur von den Eingabedaten x, sondern auch von der Abbildungsfunktion (der neuronalen Architektur und dem Einfluß seiner Komponenten), repräsentiert durch den Gewichtsvektor w abhängt. Bezogen auf das gesamte kNN muß w durch die Gewichtsmatrix W ersetzt werden. Es wird aber in der Folge meist nur $y(x)$ notiert. Die Zielfunktion r hängt hingegen nur von den Eingabedaten x ab.

Für die gesamte Stichprobe erhält man den Fehler E: $E(w) = \sum_{p=1}^{P} e^p(w)$
Überschreitet der Fehler eine Güteschwelle, müssen entsprechend Phase 3 die Gewichte modifiziert werden. Ansonsten erfolgt Abbruch des Trainings und eventuell Übergang in die Testphase für das MLP.

3. Rückwärts-Propagierung
Der am Ausgang festgestellte Fehler wird iterativ durch Änderung der Gewichte $w_{\ell,i,j}$ reduziert. Hierzu wird ein Gradientenabstiegsverfahren verwendet. Dies wird

für den Fall gezeigt, dass der Fehler e^p für jedes Stichprobenelement x^p zu einer Gewichtskorrektur verwendet wird. Aber auch die Akkumulation der Fehler zu dem der gesamten Stichprobe, E, und die nachfolgende Korrektur der Gewichte sind möglich:

$$w_{\ell,i,j}(k+1) := w_{\ell,i,j}(k) - \mu \left.\frac{\partial e^p(w)}{\partial w_{\ell,i,j}}\right|_{w(k)} \tag{4.1}$$

Hierbei ist μ die Lernkonstante.

Diese Korrekturen der Gewichte werden in umgekehrter Richtung zum Datenfluß durchgeführt. Beginnend mit den Korrekturen der Gewichte der Schicht L (Ausgabeschicht), folgt Schicht $L-1$ u.s.w. Diese schichtweise Abhängigkeit der Korrekturschritte bildet den Kern des Backpropagation-Lernens.
Während die Ausführung der Korrekturen in Richtung absteigender Schichtnummern erfolgt, wird die Ableitung dieser Abhängigkeiten analytisch in Richtung aufsteigender Schichtnummern geschehen (Vorwärts-Propagierung). Dies soll nun dargestellt werden.

Die in Gleichung (4.1) auftretende partielle Ableitung des Fehlers e^p nach dem Gewicht $w_{\ell,i,j}$ erfolgt nach der Kettenregel:

$$\frac{\partial e^p(w)}{\partial w_{\ell,i,j}} = \frac{\partial e^p(w)}{\partial y_{\ell,j}} \cdot \frac{\partial y_{\ell,j}}{\partial w_{\ell,i,j}}$$

Zunächst wird die innere Ableitung $\frac{\partial y_{\ell,j}}{\partial w_{\ell,i,j}}$ betrachtet. Mit $y = f_\sigma(u)$ folgt für $\frac{\partial y}{\partial w}$:

$$\frac{\partial y}{\partial w} = \frac{\partial f_\sigma(u)}{\partial u} \cdot \frac{\partial u}{\partial w} = f'_\sigma(u)\frac{\partial u}{\partial w}$$

Mit $y_{\ell,j} = f_\sigma(u_{\ell,j})$ folgt:

$$\begin{aligned}
\frac{\partial y_{\ell,j}}{\partial w_{\ell,i,j}} &= \frac{\partial}{\partial w_{\ell,i,j}}\left[f_\sigma\left(\sum_{n=0}^{N_{\ell-1}} w_{\ell,n,j}\cdot y_{\ell-1,n}\right)\right] \\
&= f'_\sigma\left(\sum\cdots\right)\cdot\frac{\partial}{\partial w_{\ell,i,j}}\left(\sum\cdots\right) \\
&= f'_\sigma(u_{\ell,j})\cdot y_{\ell-1,i} \\
&= y_{\ell,j}(1-y_{\ell,j})y_{\ell-1,i}
\end{aligned}$$

Hierbei wird die angenehme Ableitungsregel der Sigmoid-Funktion genutzt.
Einsetzen in den ursprünglichen Ausdruck liefert:

$$\frac{\partial e^p(w)}{\partial w_{\ell,i,j}} = \frac{\partial e^p(w)}{\partial y_{\ell,j}} \cdot y_{\ell,j}(1-y_{\ell,j})y_{\ell-1,i}, \tag{4.2}$$

wobei die äußere Ableitung $\frac{\partial e^p(w)}{\partial y_{\ell,j}}$ die **Sensitivität** von $e^p(w)$ bzgl. der Ausgabe $y_{\ell,j}$ darstellt.

Der Knoten (ℓ, j) reicht seinen Einfluß auf e^p durch alle Knoten der nachfolgenden Schichten weiter. Das nennt man Vorwärts-Propagierung (forward propagation).

Aber $\frac{\partial e^p(w)}{\partial y_{\ell,j}}$ kann auch als Funktionen der Sensitivität der nachfolgenden Schichten ausgedrückt werden:

$$\begin{aligned}
\frac{\partial e^p(w)}{\partial y_{\ell,j}} &= \sum_{n=1}^{N_{\ell+1}} \frac{\partial e^p(w)}{\partial y_{\ell+1,n}} \cdot \frac{\partial y_{\ell+1,n}}{\partial y_{\ell,j}} \quad \text{Kettenregel} \qquad (4.3)\\
&= \sum_{n=1}^{N_{\ell+1}} \frac{\partial e^p(w)}{\partial y_{\ell+1,n}} \cdot \frac{\partial}{\partial y_{\ell,j}} \left[f_\sigma \left(\sum_{q=0}^{N_\ell} w_{\ell+1,q,n} \cdot y_{\ell,q} \right) \right] \\
&= \sum_{n=1}^{N_{\ell+1}} \frac{\partial e^p(w)}{\partial y_{\ell+1,n}} f'_\sigma(u_{\ell+1,n}) \frac{\partial}{\partial y_{\ell,j}} u_{\ell+1,n} \\
&= \sum_{n=1}^{N_{\ell+1}} \frac{\partial e^p(w)}{\partial y_{\ell+1,n}} y_{\ell+1,n}(1 - y_{\ell+1,n}) \cdot w_{\ell+1,j,n} \qquad (4.4)
\end{aligned}$$

An der Ausgabeschicht haben wir:

$$\frac{\partial e^p(w)}{\partial y_{L,j}} = y_{L,j}(x^p) - r_j^p(x^p) \qquad (4.5)$$

Beginnend mit der Ausgabeschicht kann die Sensitivität von $y_{L,j}$ bis zu $y_{\ell,j}$ zurückverfolgt werden. Diese rückwärts (von der Ausgabe zur Eingabe) gerichtete Berechnung des verdeckten Fehlers heißt Backpropagation-Algorithmus. Das Prinzip des BP-Algorithmus ist nicht auf das MLP beschränkt.

Der Backpropagation-Algorithmus benutzt beim Gradientenabstieg nur **lokale Information**. Das heißt, Eingaben, die ein Neuron während der Vorwärtspropagierung an einer speziellen Eingabeverbindung erhält, werden beim Backpropagation durch die gleiche Verbindung weitergeleitet. Nur die Summenbildung über den Komponenten des Skalarproduktes in der Propagierungsfunktion besitzt die Eigenschaft der Lokalität.

Seien a und b die Eingaben eines Neurons an zwei Kanälen und liege an der Ausgabe die Summe $a + b$ an. Dann liefern die partiellen Ableitungen:

$$\frac{\partial(a+b)}{\partial a} = 1 \quad \text{und} \quad \frac{\partial(a+b)}{\partial b} = 1$$

Also wird der a-Anteil bzw. der b-Anteil aus der Summe in die Richtung des richtigen Eingabekanals rückwärts propagiert. Das Produkt ab als Ausgabe besitzt

diese Eigenschaft nicht, z.B. $\frac{\partial(ab)}{\partial a} = b$.

Allgemeiner, seien $x_1, \ldots, x_n$ die Eingaben eines Neurons über n Leitungen. Die Propagationsfunktion sei u und die Aktivierungsfunktion sei die Identität. Dann ist die Funktion

$$g_i(x_i) = \frac{\partial u}{\partial x_i}, \quad i = 1, \ldots, n$$

die Funktion, welche beim Backpropagation weitergeleitet wird. Die partiellen Integrationen von $g_i(x_i)$ liefern folgende Einzelbeiträge zur Propagierungsfunktion:

$$\begin{aligned} u(x_1, \ldots, x_n) &= G_1(x_1) + R_1(x_2, x_3, \ldots, x_n) \\ u(x_1, \ldots, x_n) &= G_2(x_2) + R_2(x_1, x_3, \ldots, x_n) \\ &\vdots \\ u(x_1, \ldots, x_n) &= G_n(x_n) + R_n(x_1, \ldots, x_{n-1}) \end{aligned}$$

Die R_i sind unabhängig von x_i und Konstante. Hieraus folgt mit R als Konstante:

$$u(x_1, \ldots, x_n) = G_1(x_1) + G_2(x_2) + \ldots + G_n(x_n) + R$$

Diese Bildungsregel für die Propagierungsfunktion garantiert die für den Backpropagation-Algorithmus erforderliche Lokalität in dem Sinn, dass an Leitung i keine Information über die Eingabe x_j gespeichert werden muß.

4.1.3 Implementierung des Backpropagation-Algorithmus

Die Implementierung des Gradientenabstiegsverfahrens erfolgt durch Einsetzen der Gleichungen (4.2) und (4.3) mit den Lösungen (4.4) und (4.5) in Gleichung (4.1). Im Fall des Lernens aus dem Fehler der Stichprobenmenge wird dabei in (4.1) anstelle des Fehlers $e^p(w)$ des einzelnen Stichprobenelementes x^p der Fehler $E(w)$ über der gesamten Stichprobe benutzt.

Das folgende Programm nutzt die Prozeduren feed-forward, compute-gradient und update-weights entsprechend der Herleitung des BP-Verfahrens.

Algorithmus Backpropagation:

```
proc back_prop
{ initialize weights
  { repeat
    choose next training sample (x,r)
    y[0,i] = x[i]
    feed_forward
    compute_gradient
    update_weights
  } until (termination_condition reached)
}
```

```
proc feed_forward
{ for l = 1 to L
    for j = 1 to N[l]
      y[l,j] = f_sigma(summe(i=0,N[l-1]
                                ,w[l,i,j] * y[l-1,i]))
}

proc compute_gradient
{ for l = L to 1
  { for j = 1 to N[l]
    { if (l == L) then
        e[L,j] = y[L,j]-r[j]
      else
        e[l,j] = summe(n=1,N[l+1],e[l+1,n]
                                    * y[l+1,n]
                                    * (1-y[l+1,n])
                                    * w[l+1,j,n])

      for i = 1 to N[l-1]
        g[l,i,j] = e[l,j] * y[l,j] * (1-y[l,j]) * y[l-1,i]
    }
  }
}

proc update_weights
{ for l = L to 1
    for j = 1 to N[l]
      for i = 1 to N[l-1]
        w[l,i,j](k+1) = w[l,i,j](k) - mu * g[l,i,j]
}
```

Bemerkung: `e[l,j]`$= \frac{\partial e^p(w)}{\partial y_{\ell,j}}$ und `g[l,i,j]`$= \frac{\partial e^p(w)}{\partial w_{\ell,i,j}}$

Termination des Algorithmus:
Der Algorithmus terminiert nicht zwangsläufig. Probleme mit dem BP-Algorithmus und Möglichkeiten ihrer Überwindung werden im folgenden Abschnitt behandelt. Folgende Möglichkeiten der Termination bestehen:

- Abbruch nach fester Anzahl von Iterationsschritten
- Abbruch, wenn $|e^p(w)| <$ Schwelle
- Abbruch, wenn $|\frac{\partial e^p(u)}{\partial w_{\ell,i,j}}| <$ Schwelle.

Wird der richtige Moment für einen Abbruch verpaßt, kann es zu einem erneuten Anstieg des Fehlers führen. Man nennt diese Situation **Overfit**. Das Netz geht dabei in den Zustand des **„Auswendiglernens“** der Stichprobenmenge über. Dies hat einen totalen Verlust an Generalisierung zur Folge.

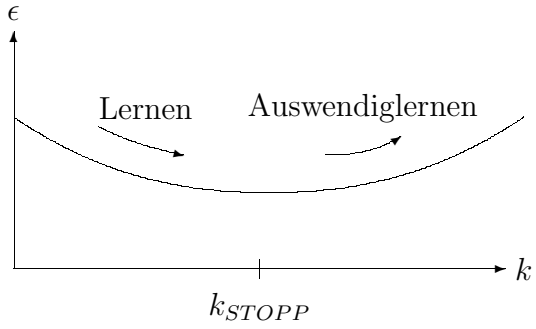


Abbildung 4.2: Fit und Overfit beim Lernen.

4.1.4 Probleme des BP-Lernens

Im Unterschied zu Fehleroberfläche eines einzelnen Adalines ist die Oberfläche eines MLP nicht mehr eine einfache quadratische Funktion. Sie besitzt diverse **lokale Minima**, in die sich der BP-Algorithmus verirren kann. Gesucht ist das **globale Minimum**. Das Gradientenabstiegsverfahren hat aber nur eine lokale Sicht auf die Fehleroberfläche. Es bewertet nur den Anstieg der Tangentialhyperebene an die Fehlerfunktion. Deshalb können unterschiedlichste Probleme auftreten. Im Fall einer mehrdimensionalen Fehleroberfäche muß auch beachtet werden, dass der Gradient des Fehlers nicht in die Richtung des Minimums zeigt (außer im Fall der Rotationssymmetrie des Fehlerfunktionals). Insbesondere zeigt Gleichung (4.1), dass das Produkt aus lokalem Gradienten und Lernrate μ (das ist die gewählte Sprungweite auf der Fehleroberfläche) in die Gewichtskorrektur eingehen. Die Form der Fehleroberfläche hängt vom Problem ab und seiner Beschreibung mittels einer Menge von Merkmalsvektoren. Sowohl die Dimension des Merkmalsraums als auch der Stichprobenumfang sind Einflußgrößen der Form der Fehleroberfläche. Deshalb hat der Nutzer problemspezifisch mit unterschiedlichen Lernraten zu experimentieren. Aber auch die Art der Initialisierung des Netzes hat Einfluß auf dessen Konvergenzverhalten.

Es treten folgende Probleme auf:
1. Lokale Minima
Hängenbleiben in einem lokalen Minimum bedeutet, dass nur eine suboptimale Lösung gefunden wird. Da die Lage des globalen Minimums prinzipiell unbekannt ist, besteht die Zielstellung im Finden einer hinreichend guten Lösung.
Mit wachsender Dimension des Merkmalsraumes und des Netzes (wachsende Anzahl von Verbindungen) wird die Fehleroberfläche zunehmend zerklüftet. Damit wächst die Wahrscheinlichkeit, in einem lokalen Minimum zu landen.
Lösungswege:

- Schrittweite μ nicht zu groß wählen und mehrere Initialisierungen der Gewichte ausprobieren.
  Große Schrittweite bedeutet große Sprünge auf der Fehleroberfläche und trägt

damit das Risiko, ein globales Minimum zu überspringen. Sinnvoll ist es, zunächst mit größerem μ zu beginnen und dieses schrittweise zu reduzieren.

- Die Initialisierung hat Konsequenzen für die Trajektorie des Systems auf der Fehleroberfläche. Niemals mit gleich großen Gewichten initialisieren, da infolge der Lernregel hieraus folgen würde, dass alle Gewichte zwischen zwei Schichten stets den gleichen Wert erhalten. Vielmehr sollten zufällige, nicht zu große Werte gewählt werden.

- "Stochastische Neuronen" verwenden:
  Die berechnete Ausgabe eines stochastischen Neurons wird nur mit einer gewissen Wahrscheinlichkeit übernommen. Diese Wahrscheinlichkeit kann in Abhängigkeit von den zurückgelegten Iterationsschritten variiert werden. Das erlaubt eine Exploration der Fehleroberfläche und u.U. das Herausspringen aus lokalen Minima.

2. Flache Gradienten
Da die Größe der Gewichtsänderung vom Betrag des Gradienten abhängt, stagniert der Gradientenabstieg an flachen Plateaus und kann auch das Vorliegen eines Minimums vortäuschen.

3. Steile Gradienten
Ist der Gradient sehr steil und die Eindellung der Fehleroberfläche eng, springt das System auf die gegenüber liegende Seite der Fehlerfunktion. Ist dort der Gradient gleich groß, kommt es zu Oszillationen.

Betrachtet wird ein einzelner linearer Assoziator, der den n-dimensionalen Eingabevektor x überwacht verarbeitet. Aus Abschnitt 2.4 ist bekannt, dass die (integrale) Fehlerfunktion ein Paraboloid

$$E(w) = E_{\min} + \frac{1}{2}(w - w^*)^T R(w - w^*)$$

um das Minimum $E_{\min}$ am Ort w^* bildet. R ist die Autokorrelationsmatrix der Eingabevektoren. Die Eigenvektoren von R bilden die Hauptachsen der Ortskurven (Kurven gleichen Fehlers) von $E(w)$. Dann ist der Gradientenabstieg am meisten effektiv, wenn die Hauptachsen alle von gleicher Ausdehnung sind. In diesem Fall weist der Gradientenvektor $\frac{dE}{dw}$ direkt in Richtung des Minimums der Fehlerfunktion. Sind die Achsen des Paraboloids aber von unterschiedlicher Größe, so kann es in Richtung mancher Hauptachsen zu Oszillationen kommen und in anderen Richtungen konvergiert der Algorithmus langsam. Dies kann dadurch vermieden werden, dass jedes Gewicht w_i seine eigene Lernrate μ_i erhält.

Beispiel: Zweidimensionale Fehlerfunktion
Sei $E(w) = aw_1^2 + bw_2^2$. Der Gradientenabstieg würde liefern:

$$\begin{aligned} \Delta w_1(k) &= -2a\mu_1 w_1(k) \\ \Delta w_2(k) &= -2b\mu_2 w_2(k) \end{aligned}$$

Die optimalen Lernraten $\mu_1 = \frac{1}{2a}$ und $\mu_2 = \frac{1}{2b}$ würden in einem Schritt in Richtung w_1 oder w_2 das Neuron nach $E_{\min} = 0$ überführen. Ein Kompromiss für eine Lernrate μ könnte zwischen μ_1 und μ_2 gewählt werden. Gilt z.B. $\mu_1 < \mu < \mu_2$, dann ist μ zu groß für die Korrektur von w_1. Das führt in dieser Richtung eventuell zu Oszillationen. Da μ aber zu klein für die optimale Korrektur von w_2 ist, reduziert sich in dieser Richtung die Konvergenz.

Im Mehrschichtnetz sind die Verhältnisse aber weitaus komplexer, weil die resultierende Gesamt-Fehlerfunktion komplexer gestaltet ist.

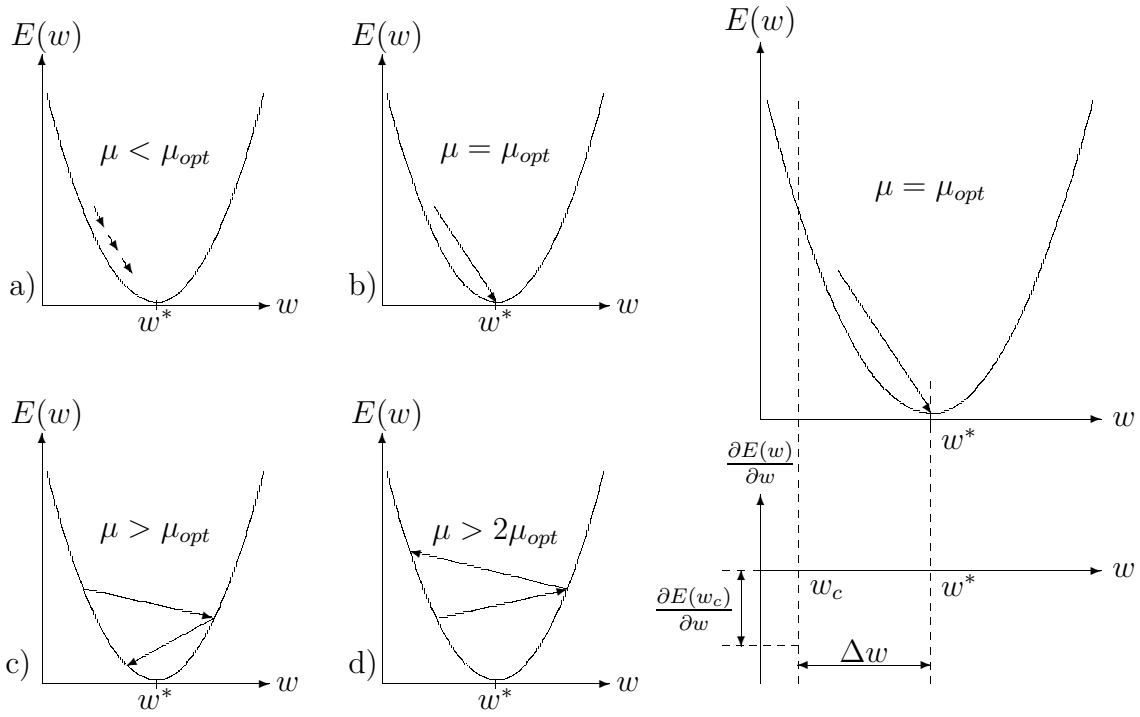


Abbildung 4.3: Gradientenabstieg für unterschiedliche Lernraten.

4.1.5 Varianten des BP-Lernens

Es sind viele Modifikationen des BP-Verfahrens entwickelt worden, um besseres Konvergenzverhaltens zu erzielen. Dies kann aber nur für Spezialfälle erreicht werden, die heuristisch in das Verfahren einfließen. Werden die heuristischen Annahmen nicht erfüllt, sind die Varianten oft schlechter als der Basisalgorithmus.

4.1.5.1 Momentum-Verfahren

Das Verfahren (siehe (Rumelhart, Hinton und Williams, 1986, S. 318-362)) heißt auch konjugierter Gradientenabstieg. Ziel ist die Vergrößerung der Schrittweite μ in

flachen Regionen und deren Reduzierung in steilen Regionen. Damit werden auch Oszillationen vermieden. Um die Eigenschaften der Fehlerregionen zu erkennen, wird die in der Vergangenheit durchgeführte Gewichtsänderung berücksichtigt. Die Heuristik lautet: In flachen Gebieten bleibt das Vorzeichen der Gewichtsänderung aufeinander folgender Schritte erhalten. In Senken erfolgt ein Vorzeichenwechsel.
Gewichtskorrektur:

$$w_i(k+1) = w_i(k) + \Delta w_i(k) \quad \text{mit}$$

$$\begin{aligned} \Delta w_i(k) &= -(1-\alpha)\mu \frac{\partial e^p(k)}{\partial w_i(k)} + \alpha \Delta w_i(k-1) \\ &= -(1-\alpha)\mu \sum_{m=0}^{k} \alpha^m \frac{\partial e^p(k-m)}{\partial w_i(k-m)} \end{aligned}$$

Hierbei ist $\alpha \in [0,1[$ der Momentum-Term. Durch Addition von $\alpha \Delta w$ wird das Gradientenabstiegsverfahren träger. Je älter ein Korrekturschritt ist, umso weniger trägt er zur aktuellen Korrektur bei. Bei gleichem Vorzeichen aufeinander folgender Korrekturen wächst die Summe und damit $\Delta w(k)$ schneller als im Fall alternierender Vorzeichen.

4.1.5.2 Gewichtsabnahme (weight decay)

Das Verfahren geht auf Werbos (1974) zurück. Es bestraft zu große Gewichte, da diese die Fehleroberfläche steiler und zerklüfteter machen. In der Fehlerfunktion

$$e^p_{wd} = e^p + \frac{d}{2} \sum_i w_i^2$$

bestraft der zusätzliche Term zu große Gewichte. Die Bildung der partiellen Ableitung entsprechend

$$\frac{\partial e^p_{wd}}{\partial w_i} = \frac{\partial e^p}{\partial w_i} + d w_i$$

ergibt eine Modifikationsregel der Gewichte, die diese gleichzeitig minimiert (weight decay):

$$\Delta w_i(k) = -\mu \frac{\partial e^p(k)}{\partial w_i(k)} - \mu d w_i(k-1)$$

Dabei sind Werte $0.005 \leq d \leq 0.03$ gebräuchlich.

4.1.5.3 Quickprop-Verfahren

Das Verfahren von Fahlman (1988) beruht auf der Annahme, dass das Tal einer Fehlerfunktion näherungsweise durch eine Parabel beschreibbar ist. Es wird der Scheitelpunkt der Parabel aus der aktuellen und der letzten Analyse der Fehlerfunktion geschätzt, um in einem Schritt in das geschätzte (lokale) Minimum zu springen. Daß dies nur annährend möglich ist, stört wegen des iterativen

Verfahrens nicht.

Sei die Steigung der Fehlerfunktion in Richtung w_i zum Zeitpunkt k gegeben durch:

$$s(k) = \frac{\partial e^p}{\partial w_i(k)}$$

Aus dem Strahlensatz ergibt sich:

$$\frac{\Delta w_i(k)}{\Delta w_i(k-1)} = \frac{s(k) - s(k+1)}{s(k-1) - s(k)}$$

Wegen $s(k+1) = 0$ ist die erforderliche Gewichtsänderung zum Erreichen eines lokalen Minimums nur aus der lokalen Information eines Neurons (wie auch beim BP-Verfahren) ableitbar:

$$\Delta w_i(k) = \frac{s(k)}{s(k-1) - s(k)} \Delta w_i(k-1)$$

Das Quickprop-Verfahren entspricht der Verwendung eines veränderlichen Momentum-Terms. Es läßt sich auch mit Gewichtsabnahme verknüpfen. Als iteratives Verfahren zweiter Ordnung benötigt es für die einzelnen Schritte aber einen viel höheren Rechenaufwand als das klassische BP-Verfahren.

4.2 Mehrschichtnetze und Regression

In diesem Abschnitt werden Mehrschichtnetze als nichtlineare Funktionen, die eine (mehrdimensionale) Eingabe auf eine (mehrdimensionale) Ausgabe abbilden, interpretiert.
Diese Funktion ist nicht a priori bekannt, sondern soll mit Hilfe eines Verfahrens zur Fehlerminimierung durch überwachtes Training gelernt werden. Dieses Lernverfahren ist das Backpropagation-Verfahren, das den Gradientenabstieg auf der Fehleroberfläche im Gewichtsraum realisiert.

Fehlerminimierung bedeutet nicht, den Fehler auf Null reduzieren zu können. Vielmehr wird die Funktion gesucht, die den mittleren quadratischen Fehler über der Stichprobe Ω_P minimiert. Dieses Problem ist uns aus der Behandlung des Adalines (siehe Abschnitt 2.4) bekannt. Dort war eine lineare Funktion zwischen Eingabe und Ausgabe gesucht. Das angewendete Ausgleichsverfahren heißt **lineare Regression** oder LMS-Algorithmus (least mean square). Andere Bezeichnungen sind Gaußsches Fehlerminimierungsverfahren oder MMSE-Verfahren (minimum mean square error). Alle Begriffe bezeichnen das Gleiche. Die unterschiedlichen Namen kommen lediglich in verschiedenen Problemen vorzugsweise zur Anwendung.

Der Backpropagation-Algorithmus ist geeignet, eine **nichtlineare Regression** zu realisieren. Er ist ebenfalls ein MMSE-Verfahren. In diesem Abschnitt wird

zunächst die lineare Regression mittels eines einzelnen linearen Assoziators wiederholt, um schrittweise das Berechnungsmodell zu vervollständigen, bis Aussagen über die Regression eines MLP gemacht werden können. Dabei werden auch Zusammenhänge zwischen der Qualität der Funktionsapproximation und den freien Parametern eines MLP hergestellt.

An die Güte der gesuchten Approximationsfunktion sind im Wesentlichen drei Forderungen zu stellen:

1. **Topologieerhaltung**: Die Abbildung der Eingabedaten auf die Ausgabedaten soll topologieerhaltend sein:
   Nachbarschaften (Ähnlichkeiten) der Trainingseingaben werden auf Nachbarschaften der Trainingsausgaben abgebildet. Es wird eine stetige Funktion gesucht. Unterbrechungen und Sprünge sind nicht zugelassen.

2. **Plastizität**: Die Abbildung soll möglichst genau die Trainingsdaten widerspiegeln.

3. **Steifigkeit**: Die Abbildung soll von Störungen, die den Trainingsdaten überlagert sind, möglichst gut abstrahieren. Sie soll möglichst glatt sein. Diese Eigenschaft heißt in der Neuroinformatik **Generalisierungsfähigkeit**.

Offensichtlich sind die Forderungen nach Plastizität und Steifigkeit widersprechend. Wie das daraus resultierende **Plastizitäts-Stabilitäts-Dilemma** behandelt werden kann, wird in Abschnitt 4.4 ausgeführt.

Die Freiheitsgrade eines MLP und die Wahl der Stichprobe müssen gezielt eingesetzt werden, um die Lösung des Regressionsproblems in die eine oder andere Richtung zu drängen. Erinnert sei hierbei an die bekannte Problemstellung der Anpassung von Polynomen mit verschiedener Ordnung an eine Punktmenge.

Einschub: Wiederholung aus der linearen Algebra
An dieser Stelle werden einige Grundlagen aus der linearen Algebra wiederholt, die für die Lösung eines linearen Gleichungssystems wesentlich sind und im Folgenden für die Diskussion der linearen und nichtlinearen Regression benötigt werden.

Eine **quadratische Matrix** $A \in \mathbb{R}^{n \times n}$ heißt **regulär**, wenn es zu A eine quadratische Matrix $B \in \mathbb{R}^{n \times n}$ gibt, so dass gilt:

$$AB = BA = I$$

Hierbei ist I die **Identitätsmatrix**, ebenfalls eine quadratische Matrix, deren Diagonalelemente die Einträge 1 besitzen und deren Nicht-Diagonalelemente alle den Wert 0 tragen. Die Identitätsmatrix spielt die Rolle eines neutralen Elements:

$$IA = AI = A$$

Sei A eine reguläre Matrix. Die durch A eindeutig bestimmte Matrix $B := A^{-1}$ heißt die zu A **inverse Matrix**. Offensichtlich ist auch B wieder eine reguläre Matrix.

Das Produkt zweier regulärer Matrizen ergibt wieder eine reguläre Matrix, und es gelten die beiden Gleichungen $(AB)^{-1} = B^{-1}A^{-1}$ und $(A^{-1})^{-1} = A$.

Jeder quadratischen Matrix A läßt sich die **Determinante**, $\det A \equiv |A|$, als reelle Zahl zuweisen. Es gilt der Produktsatz:

$$|AB| = |A| \cdot |B|$$

Da $|I| = 1$ gilt, folgt nach dem Produktsatz $|A| \cdot |A^{-1}| = 1$. Es gilt offenbar, dass die Determinante einer regulären Matrix von Null verschieden ist. Umgekehrt gilt auch: Ist die Determinante einer Matrix A ungleich Null, so ist A regulär. Gilt für eine Matrix $\det A = 0$, so heißt die Matrix A **singulär**.

Hieraus ergibt sich die Möglichkeit, die inverse Matrix A^{-1} einer regulären Matrix A mittels ihrer Unterdeterminanten zu berechnen. Es wird hier aber eine andere Methode im Zusammenhang mit der Lösung eines Gleichungssystems benutzt. In diesem Fall beschreibt die Matrix A die lineare Abbildung eines Spaltenvektors $x \in \mathbb{R}^n$ auf einen Spaltenvektor $y \in \mathbb{R}^m$ durch die Gleichung:

$$Ax = y \tag{4.6}$$

Offensichtlich gilt hier $A \in \mathbb{R}^{m \times n}$. Der Gleichung (4.6) entspricht das **lineare Gleichungssystem**:

$$\begin{array}{ccccccc} a_{11}x_1 & + & \ldots & + & a_{1n}x_n & = & y_1 \\ \vdots & & & & & \vdots & \vdots \\ a_{m1}x_1 & + & \ldots & + & a_{mn}x_n & = & y_m \end{array} \tag{4.7}$$

Jeder Matrix $A \in \mathbb{R}^{m \times n}$ kann eine Matrix $A^T \in \mathbb{R}^{n \times m}$, die **transponierte Matrix** A^T, zugeordnet werden. Die transponierte Matrix entsteht aus A durch Vertauschen von Spalten und Zeilen. Ist A eine quadratische Matrix, so erhält man A^T durch Spiegelung an der Hauptdiagonalen. Ist A regulär, so ist auch A^T regulär und es gilt $(A^T)^{-1} = (A^{-1})^T$. Weiterhin gilt allgemein $(AB)^T = B^T A^T$ und $(A^T)^T = A$.

Nun wird die Invertierung des Gleichungssystems (4.6) bzw. (4.7) betrachtet. Das heißt, es wird die Lösung $x = A^+ y$ gesucht, wobei A^+ aus A durch Invertierung hervorgeht. Das heißt aber nicht zwingend, dass $A^+ \equiv A^{-1}$.
Vielmehr hängt die Lösbarkeit des Systems (4.6) vom **Rang** der Matrix A ab.
Der Rang der Matrix $A \in \mathbb{R}^{m \times n}$ gibt die Maximalzahl der linear unabhängigen Zeilen bzw. die Maximalzahl der linear unabhängigen Spalten an. Beide Zahlen stimmen überein. Im Allgemeinen sind folgende drei Fälle bei der Lösung eines linearen Gleichungssystems zu unterscheiden:

1. Rang$(A) = n = m$
   Nur in dem Fall, dass A regulär und der Rang maximal ist, existiert **eine eindeutige Lösung**
   $$x^* = A^{-1}y$$
   mit Fehler $E(x^*) = 0$.

2. Rang$(A) = n < m$
   Man spricht von einem **überbestimmten Gleichungssystem**, wenn z.B. n die Anzahl der Freiheitsgrade eines (linearen) Systems ist und m die Anzahl der Beobachtungen des Systems. Es existiert keine eindeutige Lösung, aber eine **MMSE-Lösung**, die formuliert werden kann als:
   $$\widehat{x} = (A^T A)^{-1} A^T y = A^+ y$$
   Man bezeichnet die $(n \times m)$-Matrix A^+ als (Moore-Penrose) **Pseudoinverse**. Wie diese Lösung zustande kommt, zeigt der folgende Abschnitt. Die Lösung hat einen minimalen mittleren quadratischen Fehler bezogen auf alle m Beobachtungen.
   Sei $e_i(x)$ der Einzelfehler der i-ten Beobachtung des Systems und sei $E(x)$ die Summe der quadrierten Einzelfehler:
   $$\begin{aligned} E(x) &= \frac{1}{2}\sum_{i=1}^{m} e_i^2 \\ &= \frac{1}{2}|Ax - y|^2 = \frac{1}{2}(Ax - y)^T(Ax - y) \end{aligned}$$
   Dann ist der MMSE gegeben durch:
   $$E_{\min}(x) = E(\widehat{x}) = y^T(I - AA^+)y \geq 0$$
   Hier ist I die $(m \times m)$-Identitätsmatrix.

3. Rang$(A) = m < n$
   Man spricht von einem **unterbestimmten Gleichungssystem** zur Beschreibung eines linearen Systems mit n Freiheitsgraden aus m Beobachtungen. Hierfür existieren unendlich viele Lösungen. Aber die Lösung
   $$\widehat{x} = A^T(AA^T)^{-1}y = A^+ y$$
   hat minimale Norm (minimalen Abstand zum Optimum). Auch für den Fehler dieser Lösung gilt $E(\widehat{x}) \geq 0$.

Allgemein besitzt die Pseudoinverse die folgenden drei Eigenschaften:

$$\begin{array}{lll} AA^+A & = & A \\ A^+AA^+ & = & A^+ \\ A^+A & \text{und} & AA^+ \text{ sind symmetrisch} \end{array}$$

Im Fall, dass A regulär ist, gilt $A^+A = I$ und $AA^+ = I$.
Ist A^TA oder AA^T nicht singulär, so gilt:

$$A^+ = (A^TA)^{-1}A^T = A^T(AA^T)^{-1}$$

Damit werden die obigen Fälle 2 und 3 erfaßt. Aber auch im Fall der Singularität von A^TA oder AA^T existiert die Pseudoinverse als Grenzwert:

$$A^+ = \lim_{\nu \to 0}(A^TA + \nu^2 I)^{-1}A^T = \lim_{\nu \to 0} A^T \left(AA^T + \nu^2 I\right)^{-1}$$

Im Normalfall verrauschter Daten sind Singularitäten der Matrizen A^TA oder AA^T nicht auszuschließen. Die Methode der **Singulärwertzerlegung** (SVD) einer Matrix leistet in diesem Fall gute Dienste. Siehe hierzu auch als Standardempfehlung: „Numerical Recipes in C“ von Press, Flannery, Teukolsky und Vetterling (1992).

4.2.1 Linearer Assoziator und lineare Regression

Ein linearer Assoziator entspricht dem Adaline. Er zeichnet sich dadurch aus, dass die Aktivierungsfunktion die Identität ist und demzufolge die Ausgabe der Propagierungsfunktion u entspricht.

Annahme: Überwachtes Training eines einzelnen linearen Assoziators im Batchmode
Sei $P \in \mathbb{N}$ und sei $p \in \mathbb{N}_{\leq P}$. Gegeben sei eine Stichprobenmenge $\Omega_P = \{X, r\}$ mit Daten $X = \left(x^1, \ldots, x^P\right)^T$ und Zuordnung $r = \left(r^1, \ldots, r^P\right)^T$. Der Eingaberaum des Assoziators sei n-dimensional, also $x^p = (x_1^p, \ldots, x_n^p)^T \in \mathbb{R}^n$, und die Ausgabe sei skalar, also $u^p \in \mathbb{R}$. Es wird das erweiterte Modell mit $x_0^p = 1$ und $w_0 = -\theta$ verwendet. Also ist X eine $P \times (n+1)$-Matrix:

$$X = \begin{bmatrix} 1 & x_1^1 & \ldots & x_n^1 \\ 1 & x_1^2 & \ldots & x_n^2 \\ \vdots & \vdots & & \\ 1 & x_1^P & \ldots & x_n^P \end{bmatrix}$$

Der Zusammenhang zwischen den Vektoren x^p und der gesuchten Antwort u^p des linearen Assoziators sei gegeben durch das allen Daten gemainsame lineare Modell im $(n+1)$-dimensionalen Eingaberaum

$$u^p = b^T x^p \quad \text{für } p = 1, \ldots, P$$

mit den stichprobenunabhängigen Koeffizienten der Hyperebene b.

Diese Gleichung beschreibt für jeden Punkt in dem durch den Koeffizientenvektor $b = (b_0, b_1, \ldots, b_n)^T$ aufgespannten Lösungsraum eine n-dimensionale Hyperfläche. Gesucht ist der Schnittpunkt dieser Hyperflächen für alle x^p als unbekannter

Koeffizientenvektor. Offensichtlich liegen alle Punkte (x^p, u^p) auf einer Hyperebene im n-dimensionalen Eingaberaum, die den Abstand b_0 vom Ursprung hat.

Die gegebenen Datenpaare (x^p, r^p) während des Trainings des linearen Assoziators folgen aber nicht exakt diesem eindeutigen Zusammenhang, weil sie gestört sind. Folglich gilt für jedes $p \in \{1, \ldots, P\}$:

$$u^p = (b^p)^T x^p$$

Wegen der Stichprobenabhängigkeit von b schneiden sich die Hyperebenen im Lösungsraum nicht in einem Punkt. Sie besitzen allerdings weiterhin den Abstand b_0 zum Ursprung im Eingaberaum.

Es wird eine gemeinsame analytische Lösung für alle P linearen Modelle des unbekannten Systems gesucht. Dazu wird das lineare Gleichungssystem

$$r = Xb$$

mit Zielvektor $r = (r^1, \ldots, r^P)^T$ und Lösungsvektor $b = (b_0, b_1, \ldots, b_n)^T$ betrachtet.

Im Fall $P = n + 1$ muß diese Gleichung eine eindeutige Lösung haben, die durch

$$b^* = X^{-1} r$$

gegeben ist, vorausgesetzt, dass X regulär und der Rang maximal ist. Die Lösung ist die Schnittmenge der P Hyperebenen. Im Fall verrauschter Daten erhält man aber nur die triviale Lösung $b^* = (b_0, 0, \ldots, 0)$. Das heißt, es kann lediglich eine Schwelle $\theta = -b_0$ bestimmt werden, die im Eingaberaum den gesuchten Abstand der aus P Punkten gebildeten Hyperebene zum Koordinatenursprung liefert, aber nicht deren Lage im Raum.
Im Fall $P > n + 1$ kann aber analytisch ein linearer Ausgleich zwischen allen P Hyperebenen im Lösungsraum berechnen werden. Im Eingaberaum wird die Hyperebene gesucht, welche die Summe der quadratischen Abstände zu den Stichprobenelementen minimiert.

Im Lösungsraum entspricht dies dem Punkt $\widehat{b}$, der minimalen quadratischen Abstand zu allen Hyperebenen hat. Dazu wird für den Punkt (x^p, r^p) der Ansatz

$$r^p = b^T x^p + \epsilon^p$$

mit dem für alle P Punkte gemeinsamen, durch b aufgespannten, Lösungsraum und der punktweisen Störung ϵ^p gemacht.

Für die gesamte Stichprobe gilt in Vektor-Matrix-Schreibweise:

$$r = Xb + \epsilon$$

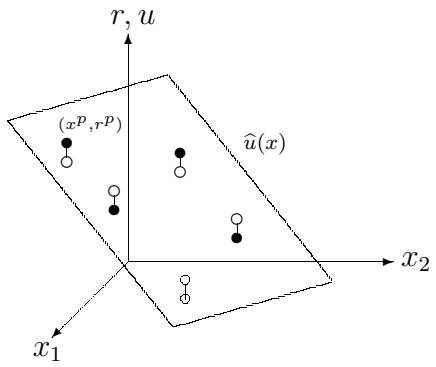


Abbildung 4.4: Eingaberaum eines 2-dim. linearen Ausgleichproblems.

mit $r = (r^1, \ldots, r^P)^T$, $b = (b_0, \ldots, b_n)^T$, $\epsilon = (\epsilon^1, \ldots, \epsilon^P)^T$ und $X \in \mathbb{R}^{P \times (n+1)}$.

Gesucht wird die Lösung $\widehat{b}$, die den mittleren quadratischen Fehler über der Stichprobenmenge Ω_P minimiert[1], das heißt, $E_{\min}(b) := \min\{E(b)\}$ mit:

$$E(b) := \frac{1}{2}\|\epsilon\|_2^2 = \frac{1}{2}\sum_{p=1}^{P}(\epsilon^p)^2 = \frac{1}{2}(r - Xb)^T(r - Xb)$$

Die Ableitung $\nabla = \left(\frac{\partial}{\partial b_0}, \frac{\partial}{\partial b_1}, \ldots, \frac{\partial}{\partial b_n}\right)$ nach dem Parametervektor b liefert:

$$\nabla E(b) = -X^T r + X^T X\widehat{b} \stackrel{!}{=} \mathbf{0}$$

Dies sind die aus Abschnitt 2.4.2.1 bekannten Normalengleichungen: $X^T X\widehat{b} = X^T r$
Falls $(X^T X)^{-1}$ existiert, folgt die Lösung:

$$\widehat{b} = (X^T X)^{-1} X^T r$$

In Abschnitt 2.4.2.1 wurde dies als die Wiener-Lösung diskutiert. In der Wiederholung zur linearen Algebra aus Abschnitt 4.2 wurde $X^+ = (X^T X)^{-1} X^T$ als die Pseudoinverse eingeführt. Die lineare Regression liefert also für eine im Vergleich zur Dimension des Eingaberaums hinreichend große Stichprobenmenge die pseudoinverse Lösung, siehe auch (Stoer, 2005, Kapitel 4.8).

Diese durch die Gleichung $\nabla E(b) = \mathbf{0}$ gefundene analytische Lösung ist identisch mit derjenigen, die als Folge des Trainings eines linearen Assoziators als neuronales Berechnungselement mittels Gradientenabstieg erhalten wird. Nur wird in diesem Fall das Optimum nicht in einem einzigen Berechnungsschritt gefunden, sondern iterativ.

[1] E bezeichnet hier nicht den Erwartungswertoperator, sondern eine Fehlerfunktion.

Im Fall des Gradientenabstiegsverfahrens wird der Lösungsraum durch den Gewichtsvektor $w = (w_0, \ldots, w_n)^T$ aufgespannt (also $b \equiv w$). Jedes Stichprobenelement weist den Fehler

$$\epsilon^p = r^p - u^p = r^p - \sum_{i=0}^{n} w_i x_i^p$$

auf. Der lineare Assoziator berechnet für die gesamte Stichprobe Ω_P den Ausgabevektor $\widehat{u} = (\widehat{u}^1, \ldots, \widehat{u}^P)^T$, der zum Zielvektor r den mittleren quadratischen Abstand $E(w)$ besitzt. Die iterative Bestimmung der Gewichtskoeffizienten durch die Korrekturschritte

$$\Delta w_i = -\mu \frac{\partial E(w)}{\partial w_i} \quad , i = 0, \ldots, n$$

kann als mehrfache lineare Regression betrachtet werden. Ist das Minimum des Fehlerfunktionals $E_{\min}$ gefunden worden, wird der optimale Ausgabevektor $\widehat{u}$ für die Stichprobe berechnet. Im Gegensatz zur analytischen (pseudoinversen) Lösung, kann für das Optimum $\widehat{w}$ bzw. $\widehat{u}$ kein geschlossener Ausdruck angegeben werden.

Wie ist die gefundene Lösung $\widehat{w}$ bzw. $\widehat{b}$ geometrisch zu interpretieren? Hierfür ist abermals das Bild der Dualität eines Punktes im Eingaberaumes und einer Hyperebene im Koeffizienten-(Gewichts-)Raum bzw. umgekehrt, einer Hyperebene im Eingaberaum und eines Punktes im Koeffizienten-(Gewichts-)Raum heranzuziehen. Die Lösung spannt im Eingaberaum eine Hyperebene auf, die einen Ausgleich zwischen allen Stichprobenelementen liefert. Dem entspricht im Gewichtsraum der durch $\widehat{w}$ gegebene Punkt. Dieser Punkt hat die Eigenschaft, zu allen Hyperebenen der Stichprobenelemente (x^p, r^p) die Summe der quadratischen Abstände zu minimieren und ist somit der Ort des Minimums des Fehlerfunktionals $E(w)$.

Beispiel: $n = 2, P = 3$
Die drei Hyperebenen im Gewichtsraum sind durch Geraden repräsentiert. Die Lösung $\widehat{w}$ ist ein Punkt, der die Summe der quadratischen Abstände zu den Geraden minimiert. Abbildung 4.5 zeigt veranschaulicht dieses Beispiel.

4.2.2 Logistische Regression und Backpropagation

Wird der in Abschnitt 4.2.1 behandelte lineare Assoziator um eine sigmoide Aktivierungsfunktion erweitert, entsteht eine neuronale Berechnungseinheit, die eine **nichtlineare Regression** durchführt. Derartige Neuronen sind die Knoten eines MLP. Die Nichtlinearität der Sigmoidfunktion $f_\sigma(u)$ ist wesentlich für die Verwendbarkeit des Neurons als nichtlineare Approximationsfunktion und die Differenzierbarkeit der Sigmoidfunktion ist wesentlich für die MMSE-Schätzung dieser Ausgleichsfunktion.

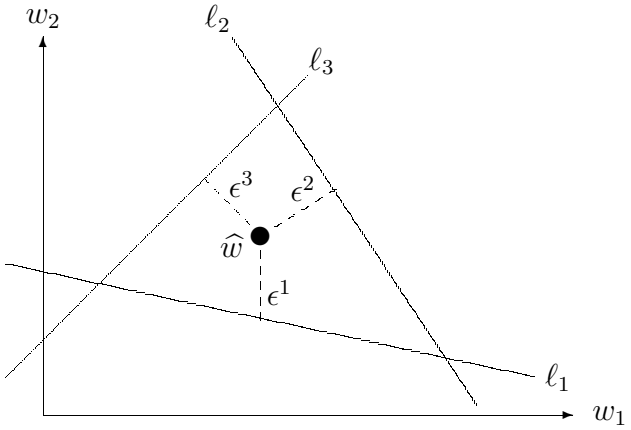


Abbildung 4.5: Beispiel einer 1-elementigen Lösungsmenge für $\widehat{w}$.

Die Folge der Modifikation der Ausgabe eines linearen Assoziators durch eine Sigmoidfunktion ist das Verbiegen der durch lineare Regression schätzbaren Hyperebene entsprechend der Nichtlinearität der Aktivierungsfunktion. Ein neuronales Netz mit h verdeckten Neuronen erzeugt eine h-fach gefaltete Regressionsfunktion. Im Abschnitt 2.6.1 wurde ein einfaches MLP zur Lösung des XOR-Problems vorgestellt, das aus zwei verdeckten Neuronen, welche die Booleschen Funktionen f_2 und f_4 realisieren, besteht. Die folgende Abbildung 4.6 gibt die Ausgabe der zweifach gefalteten Funktion qualitativ wieder. In der Nähe der Punkte $(1, 0)$ und $(0, 1)$ nimmt die Funktion näherungsweise den Wert 1 an und in der Nähe der Punkte $(0, 0)$ und $(1, 1)$ den Wert 0. Damit bildet diese nichtlineare Funktion die **Diskriminanzfunktion** (siehe Abschnitt 3.4) des XOR-Problems.

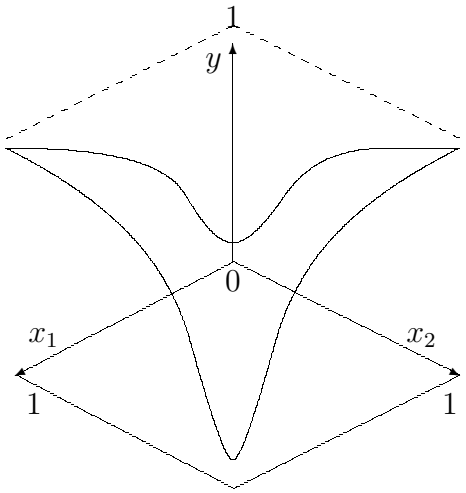


Abbildung 4.6: Diskriminanzfunktion des XOR-Problems.

Die sigmoide Aktivierungsfunktion

$$f_\sigma(u) = (1 + \exp(-u))^{-1}$$

ist also nicht nur hinsichtlich ihrer angenehmenen Differenzierbarkeit von Interesse. Es werden hier nun verschiedene Aspekte beleuchtet, die mit ihren Symmetrieeigenschaften in Verbindung stehen. Abbildung 4.7 zeigt, das einer sigmoiden Aktivierungsfunktion entsprechende nichtlineare Regressionsproblem.

Die Sigmoidfunktion bildet das Intervall $(-\infty, \infty)$ des Argumentbereiches auf das Intervall $(0, 1)$ ab. Ist $|u|$ klein, so kann die Sigmoidfunktion als lineare Funktion betrachtet werden. Darüber hinaus kann ein durch die Sigmoidfunktion repräsentiertes nichtlineares Regressionsproblem, man nennt dies **logistische Regression**, einfach in ein lineares Regressionsproblem transformiert werden. Hierfür wird die Sigmoidfunktion seit langer Zeit in den Biowissenschaften genutzt.

Sei das ideale Modell des Zusammenhanges zwischen den Eingabevektoren x^p und den geforderten Ausgaben r^p durch folgende Beziehung gegeben:

$$r^p = \left(1 + \exp\left(-\sum_{i=1}^{n} w_i x_i^p\right)\right)^{-1}$$

Dann folgt durch Umstellung

$$\exp\left(-\sum_{i=1}^{n} w_i x_i^p\right) = \frac{1}{r^p} - 1$$

für alle $p = 1, \ldots, P$. Logarithmierung liefert:

$$\sum_{i=1}^{n} w_i x_i^p = -\log\left(\frac{1 - r^p}{r^p}\right) = \log\left(\frac{r^p}{1 - r^p}\right) \equiv \varrho^p$$

Den Ausdruck $\varrho^p = \log(\frac{r^p}{1-r^p})$ bezeichnet man als **Logit-Transformation** von r^p. Tatsächlich gilt $\varrho = f_\sigma^{-1}$ und ϱ ist linear in u.

Für eine gegebene Stichprobenmenge $\Omega_P = \{X, r\}$ mit der Eingabematrix X und dem gestörten Ausgabevektor $r = (r^1, \ldots, r^P)^T$ wäre das durch nichtlineare Regression zu minimierende Fehlerfunktional:

$$E(w) = \frac{1}{2} \sum_{p=1}^{P} \left(r^p - f_\sigma(w^T x^p)\right)^2$$

An dieser Stelle muß bemerkt werden, dass diese Fehlerfunktion keine quadratische Funktion mit eindeutigem Minimum ist. Vielmehr hat diese Funktion neben dem absoluten (globalen) Minimum noch lokale (Neben-)Minima. Der Unterschied zum Fehlerfunktional des linearen Assoziators hat folgenden Grund. Beim linearen Assoziator beschreibt der Ausdruck $u^p = w^T x^p$ einen linearen Zusammenhang zwischen Eingabe und Ausgabe. Deshalb ist $(\epsilon^p)^2 = (r^p - w^T x^p)^2$ eine reine

quadratische Funktion und die Summe über alle P Stichproben ändert daran nichts. Hier ist aber $y^p = f_\sigma(w^T x^p)$ nichtlinear, also auch $\left(\epsilon^p\right)^2 = \left(r^p - f_\sigma(w^T x^p)\right)^2$ keine reine quadratische Funktion und folglich auch nicht $E(w)$.
Die Logit-Transformation korrigiert den nichtlinearen Zusammenhang im Originalraum in einen linearen im Bildraum.

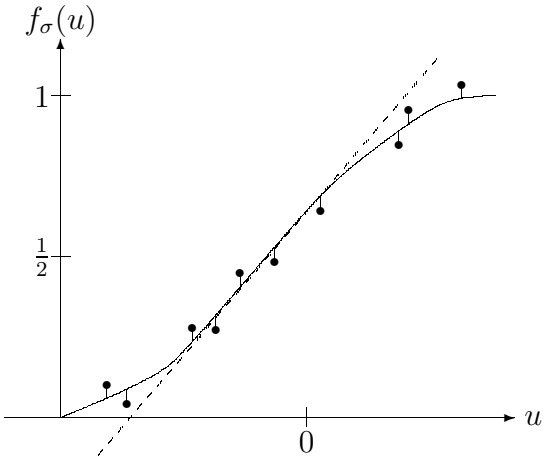


Abbildung 4.7: Nichtlineares Regressionsproblem.

Nach Logit-Transformation lautet die lineare Regressionsaufgabe:
Berechne $E'_{\min}$ mit

$$E'(w) = \frac{1}{2} \sum_{p=1}^{P} \left(f_\sigma^{-1}(r^p) - w^T x^p\right)^2 .$$

In dem durch r^p und $(1 - r^p)$ aufgespannten Raum ist die Ausgleichsgerade über der Stichprobenmenge Ω_P zu berechnen. Dabei entstehen größere Fehler in den Bereichen mit $|u| >> 0$, siehe Abbildung 4.8. Im Gegensatz hierzu ist der mittels Backpropagation in dem durch $f_\sigma(u)$ und u aufgespannten Originalraum erreichbare Fehler ϵ^p gleichmäßig über alle u verteilt, siehe Abbildung 4.7.

4.2.3 Regression im Mehrschichtnetz

In diesem Abschnitt wird der Fall untersucht, dass nichtlineare Neuronen in einem Mehrschichtnetz (MLP) angeordnet werden, um eine Abbildung der Eingaben x^p auf die Ausgaben y^p zu berechnen. Es wird zunächst eine vorborgene Schicht angenommen, deren Ausgaben bzgl. x^p bzw. Eingaben bzgl. y^p mit z^p bezeichnet werden. Es gelte $x^p \in \mathbb{R}^n$, $y^p \in \mathbb{R}^m$ und $z^p \in \mathbb{R}^l$. Die Ausgabeschicht habe also m Neuronen und die verdeckte Schicht habe l Neuronen. Für Ω_P existieren demzufolge die Matrizen X als $(P \times n)$-Matrix, Y als $(P \times m)$-Matrix und Z als $(P \times l)$-Matrix. Die Zeilen dieser Matrizen werden durch die Vektoren x^p, y^p und z^p (als Spaltenvektoren dargestellt) gebildet.

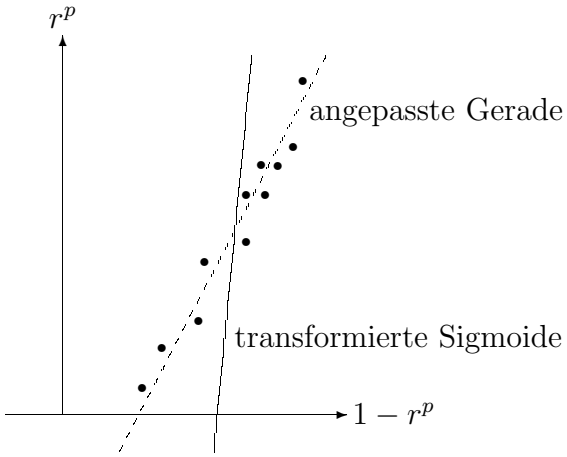


Abbildung 4.8: Lineare Regression nach Logit-Transformation.

Sei W_1 die $(n \times l)$-Matrix der Gewichte zwischen Eingabeschicht und verdeckter Schicht und sei W_2 die $(l \times m)$-Matrix der Gewichte zwischen verdeckter Schicht und Ausgabeschicht. Es wird zunächst angenommen, dass alle Neuronen des Mehrschichtnetzes aus linearen Assoziatoren bestünden. Dann beschreiben

$$z^p = W_1^T x^p \quad \text{und} \quad y^p = W_2^T z^p$$

die beiden linearen Teilschritte der Abbildung. Offensichtlich können beide Abbildungen zu einer einzigen

$$y^p = W x^p$$

mit $W = W_2^T W_1^T$ vereinigt werden. Die verdeckte Schicht wäre also verzichtbar. Beliebige Netze von linearen Assoziatoren sind als Einschichtnetze behandelbar. Die verdeckte Schicht macht also nur Sinn, wenn sie eine Nichtlinearität in die Abbildung einbringt.

Deshalb wird Annahme getätigt, dass die verdeckte Schicht aus Neuronen mit sigmoider Aktivierungsfunktion besteht. Die Ausgabeschicht darf durchaus, wie häufig in der Praxis zu finden, aus linearen Assoziatoren bestehen. Wenn die Ausgaben z^p der verdeckten Neuronen, welche die Zeilen in der Matrix Z bilden, linear unabhängige Vektoren sind, läßt sich die Ausgabe des Netzes für die gesamte Stichprobe Ω_P schreiben:

$$Y = ZW_2$$

In diesem Fall existiert die Pseudoinverse Z^+ mit $ZZ^+ = I$, welche die durch Backpropagation ermittelte Lösung der Gewichtsmatrix liefert:

$$W_2 = Z^+ Y$$

Dieser Zusammenhang kann in folgender Weise interpretiert werden. Die verdeckte Schicht repräsentiert einen Merkmalsraum, in dem alle Stichprobenelemente x^p

des Eingaberaums auf Vektoren z^p mit **linear unabhängigen Komponenten** abgebildet werden. Dann wird bei der Abbildung aller z^p der Trainingsmenge ein lineares Regressionsproblem gelöst, das den mittleren quadratischen Fehler für Ω_P minimiert.

Sind die Vektoren z^p nicht linear unabhängig, kann zwar die obige Gleichung nicht die analytische Lösung beschreiben. Dennoch findet das Backpropagation-Verfahren ein Minimum der Fehlerfunktion. Um die Eindeutigkeit dieses Minimums sicherzustellen, ist es u.U. nötig, mittels Gewichtsabnahme (Abschnitt 4.1.5) die Norm der Gewichtsmatrix zu begrenzen, um die Pseudoinverse berechnen zu können.

4.2.4 Levenberg-Marquardt-Algorithmus

In Abschnitt 4.1.4 wurden einige Probleme beim Umgang mit dem Backpropagation-Algorithmus im Fall global nicht-konvexer Fehlerfunktionen angesprochen und in Abschnitt 4.1.5 einige Verfahren vorgestellt, um diesen Problemen zu begegnen. Hier wird nun ein weiteres Verfahren vorgestellt, das sich in der Praxis gut bewährt hat. Seine Grundidee besteht darin, lokal einen linearen Lösungsansatz zur Minimierung der Summe des quadratischen Fehlers zu nutzen. Dazu ist die lokale Konstruktion einer konvexen Fehlerfunktion erforderlich. Ein ähnlicher Ansatz wurde beim Quickprop-Verfahren (siehe Abschnitt 4.1.5.3) betrachtet.

Da aber die Abweichung dieser Ersatz-Fehlerfunktion von der wahren Fehlerfunktion mit der Schrittweite der Gewichtskorrekturen wächst, muß eine geeignete Nebenbedingung zur Begrenzung der Schrittweite erfüllt werden ($\rightarrow$ line search). Ein sehr bekanntes Verfahren hierzu, das Abstiegsrichtung und Schrittweite simultan bestimmt, ist der Levenberg-Marquardt-Algorithmus, siehe hierzu auch (Schwarz, 1997, Kapitel 7.4).

Sei

$$E(w) := \frac{1}{2}\|\epsilon(w)\|_2^2 = \frac{1}{2}\sum_{p=1}^{P}\left(\epsilon^p(w)\right)^2$$

das Fehlerfunktional eines Mehrschichtnetzes für die Stichprobe Ω_P. Es wird die Gewichtskorrektur

$$w(k+1) = w(k) + \Delta w(k)$$

für den Gewichtsvektor w betrachtet. Ist $\Delta w(k)$ klein, so soll der Fehlervektor $\epsilon(w)$ in eine Taylorreihe bis zur 1.Ordnung entwickelt werden:

$$\epsilon(w(k+1)) = \epsilon(w(k)) + V(w(k+1) - w(k))$$

wobei V die Matrix ist, deren Elemente v_{pi} gegeben sind durch:

$$v_{pi} := \frac{\partial \epsilon^p}{\partial w_i(k)}$$

Hierbei ist $w_i(k)$ die i-te Komponente des Gewichtsvektors $w(k)$.

Diese lineare Näherung gestattet, folgendes konvexe Fehlerfunktional zu bilden:

$$E'(w) := \frac{1}{2}\|\epsilon(w(k+1)) + V(w(k+1) - w(k))\|_2^2$$

Die Minimierung bezüglich $w(k+1)$ liefert:

$$\begin{aligned}\frac{\partial E'(w)}{\partial w(k+1)} &= w(k) - (V^T V)^{-1} V^T \epsilon(w(k)) - w(k+1) \stackrel{!}{=} 0 \\ &\Longrightarrow w(k+1) = w(k) - (V^T V)^{-1} V^T \epsilon(w(k)) \end{aligned} \tag{4.8}$$

Gleichung (4.8) ist die pseudoinverse Lösung mit $V^+ = (V^T V)^{-1} V^T$, da die Summe der quadratischen Fehler des linearen Modells minimiert wird.

Um die Güte der lokalen linearen Näherung zu bewerten, die durch V beschrieben wird, müssen Terme zweiter Ordnung der Taylorreihen-Entwicklung betrachtet werden. Diese repräsentieren die Änderung des Anstieges des Fehlers im Gewichtsraum, also die lokale Krümmung (siehe auch Abschnitt 3.7.3 des Skripts zur Vorlesung „Computer Vision“) des Fehlerfunktionals. Die **Hesse-Matrix** H hat die Elemente

$$\begin{aligned} h_{ik} &:= \frac{\partial^2 E'(w)}{\partial w_i \partial w_k} \\ &= \sum_{p=1}^{P} \left\{ \frac{\partial \epsilon^p}{\partial w_i} \frac{\partial \epsilon^p}{\partial w_k} + \epsilon^p \frac{\partial^2 \epsilon^p}{\partial w_i \partial w_k} \right\} \end{aligned}$$

Bei Vernachlässigung des zweiten Terms ergibt sich die Hessematrix zu:

$$H = V^T V$$

Für ein Netz linearer Elemente (ohne sigmoide Aktivierungsfunktion) ist diese Beziehung exakt. Für ein Netz nichtlinearer Elemente ist diese Beziehung nur exakt im Grenzfall einer unendlich großen Trainingsmenge, $P \to \infty$. In der Realität wird der zweite Term jedoch in der Nähe eines Minimums klein, wenn die iterative Approximation der Lösung bereits zu kleinen ϵ^p geführt hat. Dann läßt sich die Pseudoinverse formulieren als:

$$V^+ = H^{-1} V^T$$

Ist bei der iterativen Berechnung die Schrittweite nicht klein genug, gilt die Näherung durch das konvexe Fehlerfunktional $E'(w)$ nicht mehr gut. Um dem zu begegnen, wird im Levenberg-Marquardt-Algorithmus das folgende Fehlerfunktional mit Schrittweitenparameter λ vorgeschlagen:

$$E''(w) = \frac{1}{2}\|\epsilon(w(k+1)) + V(w(k+1) - w(k))\|_2^2 + \lambda \|w(k+1) - w(k)\|_2^2$$

So wird eine Minimierung unter dem Zwang realisiert, dass die Schrittweite proportional λ^{-1} begrenzt ist.

Nullsetzen der ersten Ableitung von $E''(w)$ liefert die Gewichtskorrektur:

$$w(k+1) = w(k) - (V^T V + \lambda I)^{-1} V^T \epsilon(w(k))$$

Dies ist wieder die Singulärwertlösung der Pseudoinversen V^+ (s.o.).

Folgende Fälle dieser Lösung sind zu unterscheiden:

1. Für sehr kleines λ geht die Lösung in die **Newtonsche Formel** für den Gradientenabstieg über:
$$\widehat{w} = w - H^{-1} \nabla E$$
Das Newton-Raphson-Verfahren (NR) wird hier aber nicht weiter behandelt, denn es erfordert die aufwendige Berechnung bzw. Approximation der inversen Hesse-Matrix ($\rightarrow$ Methode der Finiten Differenzen).

2. Für große λ beschreibt die Lösung den **Standard-Gradientenabstieg** mit Schrittweite λ^{-1}. Hieraus folgt, dass die für die Gewichtskorrektur vorgenommen Korrekturschritte sehr klein sind.

In der Praxis wird mit einem willkürlichen Wert für λ gestartet (z.B. $\lambda = 0.1$) und die Änderung des Fehlers beobachtet. Sinkt der Fehler ab, wird λ z.B. um einen Faktor 2 verkleinert und der Iterationsprozeß wiederholt. Vergrößert sich aber der Fehler, wird der alte Zustand wieder hergestellt und λ z.B. um einen Faktor 2 vergrößert. Dies wird wiederholt, bis eine Abnahme im Fehler erreicht wird.

4.3 Statistische Interpretation von Mehrschichtnetzen

In diesem Abschnitt wird das Ergebnis des Backpropagation-Trainings eines Mehrschichtnetzes mit den Grundlagen der statistischen Entscheidungstheorie (siehe Kapitel 3) verknüpft. In Abschnitt 4.3.1 wird die Rolle der sigmoiden Entscheidungsfunktion für die Interpretation der Ausgabe eines Neurons als Wahrscheinlichkeit betrachtet. In Abschnitt 4.3.2 wird die Ausgabe eines MLP im Zusammenhang mit der Zielfunktion, in die Eingaben durch überwachtes Training eingehen, dargestellt.

4.3.1 Ausgabe als a posteriori Wahrscheinlichkeit

Die Sigmoidfunktion der Neuronen eines MLP stellt eine natürliche Verbindung der in einem MLP mittels Backpropagation iterativ berechneten nichtlinearen Regression zu den Grundlagen der statistischen Entscheidungstheorie her. Dies wird am

Beispiel des **2-Klassen-Problems** (siehe Abschnitt 3.3) im Fall normalverteilter klassenbedingter Dichtefunktionen $p(x|c_k) = N(\mu_k, \Sigma_k)$ mit $\Sigma_1 = \Sigma_2 = \Sigma$ gezeigt, siehe Fall 2 in Abschnitt 3.7. Es wird nun zunächst ein SLN (single layer network) angenommen, das aus zwei Neuronen besteht. Diese Neuronen sollen die klassenbedingte a posteriori Wahrscheinlichkeiten $P(c_k|x)$ berechnen. In Abschnitt 3.7 wurde gezeigt, dass die **linearen Diskriminanzfunktionen**

$$d_k(x) = w_k^T x + w_{k0}$$

die beiden Klassen c_1 und c_2 zu beschreiben gestatten und dass die durch $d_1(x) = d_2(x)$ bestimmte Hyperebene

$$w^T(x - x_0) = 0$$

eine lineare Trennung der Klassen erlaubt.

Die Funktion

$$y = f_\sigma(w^T x + w_0)$$

kann aber ebenfalls als lineare Diskriminanzfunktion betrachtet werden. Wie in Abschnitt 3.4 erläutert wurde, ändern monotone Transformationen einer Diskriminanzfunktion nichts am Charakter ihrer Klassentrennung. Die Entscheidungsgrenze bleibt auch in diesem Fall linear.

Nimmt man an, dass die a posteriori Wahrscheinlichkeiten durch die Ausgabe des k-ten Neurons beschrieben werden,

$$P(c_k|x) = f_\sigma(u(c_k)),$$

so folgt umgekehrt die wichtige Aussage, dass die Ausgabe von Neuronen mit sigmoider Aktivierungsfunktion als a posteriori Wahrscheinlichkeiten betrachtet werden kann. Man kann diesen Zusammenhang zeigen, indem folgender Ansatz für die Propagierungsfunktion gewählt wird:

$$u(c_1) = \log \frac{p(x|c_1)P(c_1)}{p(x|c_2)P(c_2)} \quad \text{und} \quad u(c_2) = \log \frac{p(x|c_2)P(c_2)}{p(x|c_1)P(c_1)}$$

Für $u(c_1)$ folgt dann beispielsweise:

$$\begin{aligned} f_\sigma(u(c_1)) &= \left(1 + \exp\left(-\log \frac{p(x|c_1)P(c_1)}{p(x|c_2)P(c_2)}\right)\right)^{-1} \\ &= \left(1 + \exp\left(\log \frac{p(x|c_2)P(c_2)}{p(x|c_1)P(c_1)}\right)\right)^{-1} \\ &= \left(1 + \frac{p(x|c_2)P(c_2)}{p(x|c_1)P(c_1)}\right)^{-1} \\ &= \frac{p(x|c_1)P(c_1)}{p(x|c_1)P(c_1) + p(x|c_2)P(c_2)} \end{aligned}$$

Dies ist aber die a posteriori Wahrscheinlichkeit der Klasse c_1.

Das für die Normalverteilung erhaltenen Ergebnis läßt sich verallgemeinern auf die Familie der Verteilungen, die eine Exponentialfunktion enthalten, wie Exponentialverteilung und Poissonverteilung, aber auch auf solche, die eine Exponentialfunktion approximieren, wie Binomialverteilung und Bernoulliverteilung (siehe Skript zur Vorlesung „Computer Vision“, Kapitel 4).

Auch die Ausgaben verdeckter Neuronen in einem MLP können als Wahrscheinlichkeiten interpretiert werden, wenn sie mit einer sigmoiden Aktivierungsfunktion ausgestattet sind.

4.3.2 Ausgabe als bedingter Mittelwert der Zielfunktion

Ein Mehrschichtnetz ist eine neuronale Architektur zur Berechnung einer nichtlinearen (vektorwertigen) Funktion über einem (mehrdimensionalen) Eingaberaum X. Diese Funktion kann nur die gesuchte Zielfunktion approximieren. In einem überwachten Trainingsschema werden gestörte Stichproben (x^p, r^p) aus einem Stichprobenvorrat Ω_P entnommen. $r(x)$ wird als **Zielfunktion** bezeichnet. Sie repräsentiert für jede Eingabe $x \in X$ die geforderte Ausgabe r. Die MMSE-Lösung $\widehat{y}(x)$ des Mehrschichtnetzes ist aber nur eine Approximation der für $P \to \infty$ erreichbaren idealen Lösung $y^*(x)$. Da $r^p(x^p)$ Störungen enthält, wird sowohl $y^*(x)$ als auch $\widehat{y}(x)$ eine glattere Funktion sein. Die Störungen müssen durch das Mehrschichtnetz gewissermaßen herausgemittelt werden.
In diesem Abschnitt wird die MMSE-Lösung als Approximation des bedingten Erwartungswertes der Zielfunktion $\mathrm{E}\{r|x\}$ diskutiert. Das Ergebnis hat grundlegende Bedeutung für das bessere Verständnis neuronaler Berechnungen. Insbesondere wird in Abschnitt 4.4 daraus eine statistische Interpretation des Plastizitäts-Stabilitäts-Dilemmas abgeleitet.

Für eine unendlich große Stichprobenmenge Ω, d.h. $P \to \infty$, läßt sich das Fehlerfunktional $E(w)$ in folgender Weise umformen:

$$\begin{aligned} E(w) &= \lim_{P\to\infty} \frac{1}{2P} \sum_{p=1}^{P} \sum_{j=1}^{m} \left(y_j(x^p; w) - r_j^p\right)^2 \\ &= \frac{1}{2} \sum_{j=1}^{m} \iint \left(y_j(x; w) - r_j\right)^2 p(r_j, x) dr_j dx \\ &= \frac{1}{2} \sum_{j=1}^{m} \iint \left(y_j(x; w) - r_j\right)^2 p(r_j|x) p(x) dr_j dx \end{aligned} \tag{4.9}$$

Um das weitere Verfahren zu motivieren, wird an dieser Stelle noch einmal auf die Bedeutung der Dichtefunktionen eingegangen. Die Verbunddichte $p(r, x)$ repräsentiert das gesuchte Modell der Daten des Stichprobenraumes. Es assoziiert

die Eingaben x mit den Ausgaben r. In Abschnitt 2.4.2.1 wurde gezeigt, dass die **Wienerlösung** für das Modell den Zusammenhang zwischen der Struktur der Eingabedaten (dort repräsentiert durch die Autokorrelationsmatrix) und dem Modell der Abbildung der Eingaben auf die Ausgaben (dort repräsentiert durch den Kreuzkorrelationsvektor) beschreibt. Im statistischen Modell muss auch eine derartige Trennung der Einflußgrößen auf die Verbunddichtefunktion vorgenommen werden. Deshalb wird der Ansatz

$$p(r, x) = p(r|x)p(x)$$

verwendet, wobei $p(r|x)$ die Wahrscheinlichkeitsdichte für eine Ausgabe r beschreibt, gegeben eine Eingabe x und eine unbedingte Wahrscheinlichkeitsdichte der Eingaben $p(x) = \int p(r, x)dr$.

Dieses Modell ist insofern simpler als das Wiener-Modell, weil letzteres mit der Autokorrelationsmatrix eine Statistik 2.Ordnung beschreibt, während $p(x)$ lediglich eine Statistik 1.Ordnung repräsentiert. Die bedingte Dichte $p(r|x)$ ist gerade der Anteil des gesuchten Abbildungsmodelles, der die gesuchte Prädiktion von Ausgaben y für unbekannte Eingaben x erlaubt. Ein vorwärtsgerichtetes neuronales Netz kann als Modell der bedingten Dichtefunktion $p(r|x)$ betrachtet werden. Deshalb wird ihr Anteil am Fehlerfunktional eines Mehrschichtnetzes explizit ausgewiesen.

Mittels der bedingten Dichtefunktion $p(r_j|x)$ werden nun die ersten und zweiten gewöhnlichen Momente über der bedingten Zielfunktion gebildet, um damit den Differenzterm im obigen Fehlerfunktional umzuformulieren.

$$\begin{aligned} \mathrm{E}\{r_j|x\} &= \int r_j p(r_j|x)dr_j \\ \mathrm{E}\left\{r_j^2|x\right\} &= \int r_j^2 p(r_j|x)dr_j \end{aligned}$$

Der quadratische Differenzterm $(y_j - r_j)^2$ im Gleichung (4.9) wird nun durch das erste Moment erweitert und ausformuliert.

$$\begin{aligned} (y_j - r_j)^2 &= (y_j - \mathrm{E}\{r_j|x\} + \mathrm{E}\{r_j|x\} - r_j)^2 \\ &= (y_j - \mathrm{E}\{r_j|x\})^2 + 2(y_j - \mathrm{E}\{r_j|x\})(\mathrm{E}\{r_j|x\} - r_j) + (\mathrm{E}\{r_j|x\} - r_j)^2 \end{aligned}$$

Substitution dieses Ausdruckes im obigen Fehlerfunktional führt unter Berücksichtigung der Definition der Erwartungswerte zu folgender Formulierung des MSE:

$$E(w) = E_1(w) + E_2$$

mit:

$$E_1(w) := \frac{1}{2}\sum_{j=1}^{m}\int (y_j(x; w) - \mathrm{E}\{r_j|x\})^2 p(x)dx \tag{4.10}$$

$$E_2 := \frac{1}{2}\sum_{j=1}^{m}\int \left(\mathrm{E}\left\{r_j^2|x\right\}) - (\mathrm{E}\{r_j|x\})^2\right) p(x)dx \tag{4.11}$$

Zunächst ist bemerkenswert, dass der Fehleranteil E_2 nicht von der gegebenen Abbildungsfunktion $y_j(x;w)$ des Mehrschichtsnetzes abhängt und folglich auch unabhängig vom Gewichtsvektor ist. Für die Suche des optimalen Gewichtsvektors w^* nach dem MMSE-Verfahren trägt also der Anteil E_2 des Fehlerfunktionals nichts bei. Er drückt aber den Anteil der Auswahl der Stichprobenmenge auf das Fehlerfunktional aus, wenn auf eine endliche Stichprobe Bezug genommen wird und w^* in $\widehat{w}$ übergeht.

Offensichtlich wird das Minimum des Fehlerfunktionals erreicht, wenn $E_1(w) = 0$ gilt, d.h.:

$$y_j^*(x;w^*) = \mathrm{E}\{r_j|x\}$$

Im praktisch interessanten Fall einer endlichen Stichprobengröße gehen die Erwartungswerte in die bedingten Mittelwerte $< r_j|x >$ bzw. $< r_j^2|x >$ über. Dann hat Gleichung

$$\widehat{y}_j(x;\widehat{w}) = < r_j|x > \tag{4.12}$$

die Bedeutung, dass die optimale Abbildungsfunktion $\widehat{y}$ des Netzes durch den bedingten Mittelwert der Zielfunktion $< r_j|x >$ repräsentiert wird.

In Abbildung 4.9 ist die nichtlineare Regression einer eindimensionalen Funktion von gestörten Daten $(x,r) \in \Omega_P$ dargestellt. Für irgendein x_0 liefert das Netz $\widehat{y}(x_0)$ als Mittelwert über alle r in Bezug auf die bedingte Dichte $p(r|x_0)$.

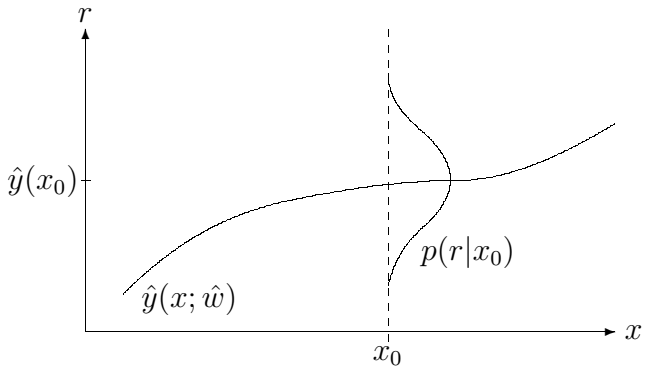


Abbildung 4.9: Nichtlineare Regression gestörter Daten.

Der in Gleichung (4.12) dargestellte Zusammenhang zwischen dem Ergebnis der nichtlinearen (iterativen) Regression und dem bedingten Mittelwert der Zielfunktion beruht auf drei Annahmen:

1. Die Trainingsmenge muß hinreichend groß sein, um den Erwartungswert gut zu approximieren.

2. Die Abbildungsfunktion $y_j(x; w)$ des Netzes muß hinreichend allgemein sein, um den Term $E_1(w)$, Gleichung (4.10), hinreichend klein zu machen. Dazu muß die Funktion hinreichend viele Parameter haben, um die geforderte Anpassung zu ermöglichen.

3. Die Optimierung der Netzparameter muß so erfolgen, dass auch das Minimum der Fehlerfunktion gefunden werden kann.

Die Gestaltung der Trainingsmenge (1) und der Abbildungsfunktion (2) können nur als verkoppelte Randbedingung behandelt werden. Diese Problematik wird im nächsten Abschnitt näher betrachtet. Im Zusammenhang mit Annahme (3) sei auf die unterschiedlichsten Techniken der Parameteroptimierung verwiesen, wie konjugierter Gradientenabstieg oder Levenberg-Marquardt-Algorithmus.

Obwohl das Ergebnis nicht auf neuronale Netze beschränkt ist, zeichnen sich diese besonders aus, weil sie nichtlineare Abbildungen mit hinreichender Qualität zu modellieren gestatten.

Für viele Regressionsprobleme ist der in Gleichung (4.12) dargestellte Zusammenhang optimal. Sei zum Beispiel eine Zielfunktion aus einer Menge deterministischer Funktionen $h_j(x)$ gebildet, die von mittelwertfreiem Rauschen mit $< \epsilon^p >= 0$ gestört sind:

$$r_j^p = h_j(x^p) + \epsilon_j^p$$

Dann rekonstruiert die Ausgabe des Netzes die deterministischen Funktionen:

$$\widehat{y}_j(x) =< r_j|x >=< h_j(x) + \epsilon_j|x >= h_j(x)$$

Aber nicht alle Regressionsprobleme funktionieren so simpel.

Frage: Welche Rolle spielt die Dichtefunktion der Eingaben $p(x)$ in Gleichung (4.10)?
Antwort: Offensichtlich wichtet dieser Faktor die Abweichung der Netzausgabe $\widehat{y}_j(x, \widehat{w})$ vom bedingten Mittelwert der Zielfunktion. Kleine Dichten $p(x)$ begrenzen die Auswirkung dieser Abweichung und große Dichten verstärken diese.

Schließlich wird noch der konstante Term E_2, Gleichung (4.11), diskutiert. Die Differenz der Erwartungswerte unter dem Integral entspricht der Varianz der Zielfunktion als Funktionen der Eingabedaten und ist gegeben durch:

$$\begin{aligned}
\sigma_j^2(x) &= < r_j^2|x > - < r_j|x >^2 \\
&= < (r_j - < r_j|x >)^2|x > \\
&= \int (r_j - < r_j|x >)^2 p(r_j|x) dr_j
\end{aligned}$$

Ist das neuronale Netz im optimierten Zustand, Gleichung (4.12) ist also erfüllt, so ist $E_1(w)$ minimal. Im Fall einer unendlich großen Stichprobe wäre $E_1(w) = 0$ und $E(w) = E_2$. Der Fehleranteil

$$E_2 = \frac{1}{2} \sum_{j=1}^{m} \int \sigma_j^2(x) p(x) dx$$

beschreibt den Restfehler $E_{\min}$, welcher ein Maß für die mittlere Varianz der Zieldaten ist.

4.4 Generalisierung und Lernen

In diesem Abschnitt werden Generalisierung und Lernen als zwei untrennbare Aspekte der Aufgabe angesehen, aus einer endlichen Stichprobenmenge Ω_P **Wissen abzuleiten**. Diese Aufgabe ist nicht an neuronale Netze gebunden, sondern von allgemeinem Charakter. Während ein Datenbanksystem nur einen komfortablen Zugriff auf Ω_P erlaubt, gilt das Interesse hier Systemen, die gewisse **Regeln** aus den Daten ableiten. In neuronalen Netzen sind diese Regeln als nichtlineare Funktionen zu verstehen. Die Regelhaftigkeit zu erkennen wird dadurch erschwert, dass die Stichproben (x^p, r^p) gestörte Daten darstellen. Außerdem sind die unregelmäßigen Abtastpositionen x^p unter Umständen ungünstig verteilt, um die Regelhaftigkeit der Daten zu erkennen. Da Ω_P immer nur einen Ausschnitt des für $P \to \infty$ zutreffenden Stichprobenraumes Ω ist, wird erwartet, dass die gelernte Regelhaftigkeit erlaubt, für Abtastpositionen $x \notin \Omega_P$ die Zielfunktion $r(x)$ durch **Interpolation** und/oder **Extrapolation** berechnen zu können.

Neuronale Netze stellen Berechnungsarchitekturen dar, die hierzu besonders geeignet sind. Das Lernen einer glatten Funktion $r(x)$ aus diskreten verrauschten Daten (x^p, r^p) ist völlig verschieden zum Auswendiglernen der Daten. Vielmehr ist **Abstraktion** (Generalisierung) erforderlich, um die Störungen in den Daten $r^p(x^p)$ herauszurechnen und für $x \in \Omega$, $x \notin \Omega_P$ die unbekannte Zielfunktion $r(x)$ zu berechnen. Hierzu ist es erforderlich, ein **statistisches Modell** von Ω zu lernen.

In Abschnitt 4.4.1 wird das dabei entstehende Plastizitäts-Stabilitäts-Dilemma informal diskutiert. In Abschnitt 4.4.2 wird dieses Dilemma statistisch formalisiert und als Bias-Varianz-Problem behandelt. In Abschnitt 4.5 werden einige Methoden zur Herstellung einer Balance zwischen Bias und Varianz in der geschätzten Zielfunktion $\widehat{y}(x)$ vorgestellt.

4.4.1 Plastizitäts-Stabilitäts-Dilemma

Generalisierung beschreibt die Fähigkeit eines (neuronalen) Systems, die Regelhaftigkeit in einer Menge von Daten in der Weise zu erlernen, dass das Modell der

Daten auch Gültigkeit für bisher nicht gesehene Daten hat.

Die Modell- bzw. Regelextraktion aus Daten wird in folgender Weise veranschaulicht. Sei Ω_U die Menge aller möglichen Ein-Ausgabe-Paare (x, r). Das System hat nicht den Anspruch, das Universum zu verstehen, sondern begrenzt seinen Horizont auf die Menge $\Omega_R \subset \Omega_U$. Ω_R repräsentiert die Aspekte des Universums, für das ein Modell aus der Regelhaftigkeit der Paare (x, r) gelernt werden soll. Dazu muß aber der Aufwand in der Lernphase auf die Trainingsmenge $\Omega_T \subset \Omega_R$ begrenzt werden. Es besteht die Hoffnung, dass die Regeln bzw. das Modell, welches aus den $(x, r) \in \Omega_T$ gelernt wird, auch für Ω_R gilt. Dies wird mit Hilfe der Testmenge $\Omega_E \subset \Omega_R$, $\Omega_T \cap \Omega_E = \emptyset$, überprüft. Wird diese Hoffnung bezüglich Ω_E erfüllt, wird die Hypothese aufgestellt, dass das gelernte Modell auch für Ω_R gilt.

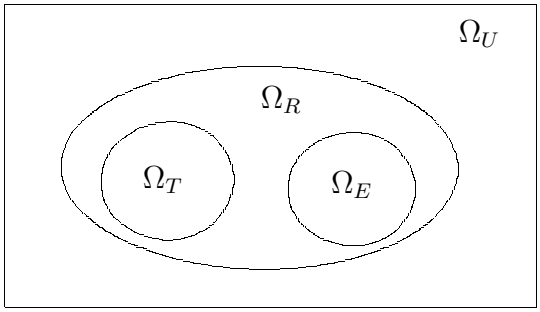


Abbildung 4.10: Zusammenhang von Test- und Trainingsmenge.

Im engeren Sinn wird als **Gedächtnis** der Erfolg des Systems bezüglich Ω_T und als **Generalisierung** der Erfolg bezüglich Ω_E verstanden. Im weiteren Sinn bezieht sich die gewünschte Generalisierung auf den Erfolg bezüglich Ω_R. Gedächtnis und Generalisierung sind zwei entgegengesetzte Optimierungsziele eines Systems. Der Erfolg des Systems bezüglich $\Omega_R \backslash Omega_T$ heißt **Prädiktion**.

Dieses Schema ist auf den Fall des Offline-Lernens angepaßt. Beim Online-Lernen werden die Unterscheidungen zwischen Ω_T und Ω_E nicht gemacht. Dann bezieht sich der Begriff Generalisierung allein auf die Befähigung, eine glatte Funktion aus gestörten Daten zu lernen. Die Steifigkeit (Stabilität) dieser Funktion erlaubt auch eine **Prädiktion** in die Menge Ω_R.

Das Generalisierungsproblem eines mittels Ω_T antrainierten Systems besteht darin, dass die begrenzte Sicht auf den durch Ω_R gegebenen Problemraum ohne Weiteres nicht die gewünschte Generalisierung liefert.
Anstatt eine Generalisierung bezüglich Ω_R zu erreichen, kann sich ebenso eine Generalisierung bezüglich Ω'_R einstellen. Hieraus folgt, dass geeignete **Zwänge** erforderlich sind, um die Systemparameter in geeigneter Weise zu optimieren.

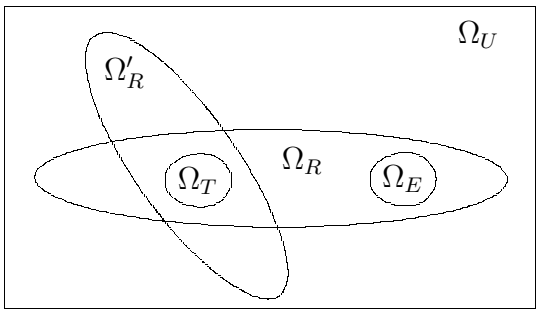


Abbildung 4.11: Problematik bei der Wahl von Test- und Trainingmenge.

Ein solcher Zwang ist z.B. die **Auswahl der Trainingsstichprobe**. Sie sollte so gewählt werden, dass die für die Modellierung von Ω_R erforderlichen Parameter auch aus Ω_T gelernt werden können. Das erfordert aber insbesondere, nicht nur hinreichend viele Trainingsdaten zu verwenden, sondern auch die richtigen. Damit reduziert sich die Unsicherheit der Generalisierung.

Beispiel: Lernen eines Entscheidungsproblems mit dem Perzeptron (Abbildung 4.12)
Zu wenig Trainingsdaten der Mengen Ω_- und Ω_+ erlauben viele mögliche Lösungen (linke Abbildung). Wird die Anzahl der Trainingsdaten in geeigneter Weise erhöht, reduziert sich die Varianz des Anstiegs der linearen Trennfunktionen (rechte Abbildung). „In geeigneter Weise“ bedeutet, z.B. allein die Anzahl der (eingekreisten) **Supportvektoren** zu erhöhen.

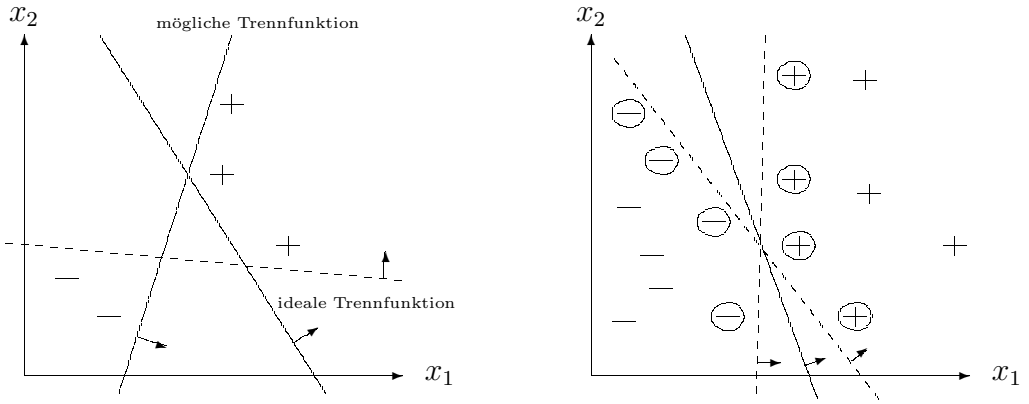


Abbildung 4.12: Vielfalt an Trennfunktionen.

Dieses Beispiel enthält aber gleichzeitig einen weiteren Zwang: Die Trennfunktion ist ein Polynom 1.Ordnung. Bestünde dieser Zwang nicht bereits im gewählten Neuronenmodell, könnte ein neuronales Netz versuchen, die Klassentrennung mit Hilfe eines Polynoms höherer Ordnung zu realisieren. Ist die Ordnung des Lösungspolynoms nicht vorgegeben, trägt die Erhöhung der Trainingsbeispiele

nichts zur Verbesserung der Lösung bei.

Das **Plastizitäts-Stabilitäts-Dilemma** besteht darin, dass das zu lernende Modell der Daten diese gleichzeitig gut widerspiegeln und hinreichend gut von ihnen abstrahieren soll. Es muß eine für das anliegende Problem geeignete Balance zwischen Plastizität und Stabilität der Systemparameter eingestellt werden. Hierzu hat der Designer verschiedene Möglichkeiten:

- Das Netz muß hinreichend viele **Freiheitsgrade** besitzen, um sich an die erforderliche Komplexität der Lösung anpassen zu können. Solche Freiheitsgrade sind zum Beispiel die Anzahl der verdeckten und Ausgabeneuronen und die Gesamtzahl der Gewichte.
- Diese Freiheitsgrade müssen durch **Zwänge** geführt angepaßt werden.

Durch Versuch und Irrtum (trial and error) hat der Designer diese Zwänge gezielt einzusetzen, um den Lösungsraum der Systemarchitektur, d.h. $\widehat{y}(x)$, an Ω_R anzupassen. Mögliche Zwänge sind:

- Umfang, Auswahl und Repräsentation der Lernstichprobe
- Änderung der Struktur (Knoten, Gewichte) des Netzes
- Aufgabenteilung im Netz (Mixture of Experts)
- algebraische Methoden (Erhöhung der Dimension des Eingaberaumes, Wahl einer „Problem-Algebra“).

4.4.2 Bias und Varianz der Abbildungsfunktion

Das Ziel des Netztrainings ist nicht, eine exakte Repräsentation der Daten zu lernen (Auswendiglernen), sondern ein statistisches Modell des Prozesses, der die Daten erzeugt, geeignet zu parametrisieren. Dieses Modell sollte auch für unbekannte Daten Gültigkeit haben.

Um die beste Generalisierung zu erreichen, ist die Optimierung der Komplexität der Modellfunktion erforderlich, ohne auf diese selbst zugreifen zu können. Das Konzept des Balancierens von Bias und Varianz liefert hierfür Einsicht, wenngleich auch keine konkrete Lösung. Es wurde von Geman, Bienenstock und Doursat (1992) in „Neural networks and the bias/variance dilemma“ vorgeschlagen.

Die Kernidee beruht darauf, einen Ausdruck für den Generalisierungsfehler herzuleiten, der sich als Summe aus dem Quadrat des Bias und der Varianz darstellt.

- Ein zu starres Modell (hohe Stabilität) bewirkt großen **Bias** (systematische Abweichung).

- Ein zu flexibles Modell (hohe Plastizität) bewirkt große **Varianz** (zufällige Abweichung).

Die Aufgabe der Optimierung der Komplexität der Modellfunktion wird damit überführt in das

Ersatzproblem: Minimierung des Generalisierungsfehlers
Diese Minimierung wird unter der Kontrolle der effektiven Komplexität des Systemmodelles durchgeführt. Dieser Aspekt wird im nächsten Abschnitt behandelt.

Es folgt zunächt die Ableitung des Generalisierungsfehlers und die Interpretation des Bias- und Varianzterms dieses Fehlerfunktionals. Dies knüpft an Abschnitt 4.3.2 an, in dem das Fehlerfunktional $E(w)$ in den systemabhängigen Anteil $E_1(w)$ (siehe Gleichung (4.10)) und den systemunabhängigen Anteil E_2 (siehe Gleichung (4.11)) aufgespaltet wurde. Im Fall einer unendlich großen Stichprobe und für den Fall einer einzigen Ausgabe des Netzes ($m = 1$) gilt:

$$E_1(w) = \frac{1}{2} \int (y(x;w) - \mathrm{E}\{r|x\})^2 p(x) dx \quad \text{und}$$

$$E_2 = \frac{1}{2} \int \left(\mathrm{E}\left\{r^2|x\right\} - (\mathrm{E}\{r|x\})^2\right) p(x) dx$$

Es wird daran erinnert, dass $p(x)$ die Dichtefunktion der Eingabedaten ist und $\mathrm{E}\{r|x\}$ bzw. $\mathrm{E}\{r^2|x\}$ erstes und zweites gewöhnliches Moment der bedingten Zielfunktion $r|x$ sind.

Während die Paare $(x, r) \in \Omega$ die Struktur und Dynamik von Ω vollständig implizit repräsentieren, stellt die Lösung für $E_1(w) = 0$, d.h.

$$y^*(x, w^*) = \mathrm{E}\{r|x\}$$

die gelernte Regel dar, die für jede Eingabe x, ohne Fehler zu begehen, die entsprechende Ausgabe y^* berechnet. Hier repräsentiert w^* die nach dem MMSE-Prinzip optimierte Parametermenge des Systems.

Dabei kann der Systemfehler $E(w)$ nicht kleiner werden als der Anteil E_2, der die Varianz der Eingabedaten in Ω repräsentiert. Selbstverständlich hat dieser Restfehler des Systems keine Auswirkung auf die gefundene Ausgabe y^*.

In der Praxis existieren keine unendlich großen Stichproben Ω. Vielmehr werden die Stichprobenelemente (x^p, r^p), $1 \leq p \leq P$, der Menge $\Omega_T \subset \Omega_R$ entnommen. Sie gibt Anlaß zu der approximierenden Schätzung $\widehat{y}(x; \widehat{w}) =< r|x >$, siehe auch Gleichung (4.12). Dabei wird $< r|x >$ **Regressionsfunktion** genannt. Ferner wird die gewählte Stichprobenmenge mit Ω_P bezeichnet, weil sie P Elemente aus Ω_R enthält. Sei Ω_T beschrieben durch die Verbunddichtefunktion $p(x, r)$. Dann darf

erwartet werden, dass auch Ω_P diese Eigenschaft widerspiegelt. Besser ist es jedoch, Ω_R hinreichend oft Stichprobenmengen Ω_{P_i} zu entnehmen, mit denen das Modell $y(x)$ antrainiert wird. Wird ein Ensemble aus I Stichprobenmengen Ω_{P_i} verwendet, dann beschreibt der Mittelwert über all diese $I \cdot P$ Stichproben

$$D(x) =< (y(x) - < r|x >)^2 > \tag{4.13}$$

den mittleren quadratischen Abstand der jeweiligen Abbildungsfunktion $y(x)$ zur Regressionsfunktion $< r|x >$, siehe Gleichung (4.12), für alle I Stichprobenmengen.

Wäre die jeweilige Abbildungsfunktion stets in der Lage, die Regressionsfunktion zu realisieren, würde dieser Integrand der Fehlerfunktion $E_1(w)$ verschwinden und damit $E_1 = 0$ liefern. Tatsächlich muß aber fehlerfreies Lernen nicht eintreten. Dies hat zwei verschiedene Ursachen:

1. **Bias-Komponente** des Lernfehlers: Im Mittel ist die Abbildungsfunktion des Netzes verschieden von der Regressionsfunktion.

2. **Varianzkomponente** des Lernfehlers: Die Abbildungsfunktion des Netzes ist empfindlich abhängig von der Wahl der $\Omega_{P_i} \subset \Omega_R$.

Um diese Anteile an dem Integranden von $D(x)$ explizit darzustellen, wird folgende Umformung vorgenommen:

$$\begin{aligned}
(y(x) - < r|x >)^2 &= (y(x) - < y(x) > + < y(x) > - < r|x >)^2 \\
&= (y(x) - < y(x) >)^2 + (< y(x) > - < r|x >)^2 \\
&\quad +2\,(y(x) - < y(x) >)\,(< y(x) > - < r|x >)
\end{aligned}$$

Hier bedeutet $< y(x) >$ den Mittelwert über alle Abbildungen $y(x)$ der $I \cdot P$ Stichprobenelemente. Um den Mittelwert $D(x)$ der Gleichung (4.13) zu berechnen, wird in der obigen Gleichung beiderseits der Mittelwertoperator $< \cdot >$ über $I \cdot P$ Stichprobenelementen angewendet. In diesem Fall verschwindet der dritte Summand auf der rechten Seite, wegen der Linearität und Idempotenz des Mittelwertoperators. Es folgt:

$$D(x) = (< y(x) > - < r|x >)^2 + < (y(x) - < y(x) >)^2 >$$

Dies wird geschrieben als

$$D(x) = D_B^2(x) + D_V(x)$$

mit

$$\begin{aligned}
D_B^2(x) &= (< y(x) > - < r|x >)^2 \\
D_V(x) &= < (y(x) - < y(x) >)^2 >
\end{aligned}$$

- $D_B^2(x)$ ist der **quadrierte Bias-Anteil** am Integranden des Fehlerfunktionals E_1. Er ist ein Maß der Abweichung der über alle Datensätze Ω_{P_i} gemittelten Abweichungen der Ausgabefunktion $y(x)$ des Netzes von der geforderten Regressionsfunktion $< r|x >$. Damit ist beschreibt $D_B^2(x)$ den Einfluß der Architektur.

- $D_V(x)$ ist der **Varianz-Anteil** am Integranden des Fehlerfunktionals E_1. Er ist ein Maß für die Sensitivität der Abbildungsfunktion $y(x)$ des Netzes bezüglich der speziellen Wahl der Daten Ω_{P_i}. Damit ist beschreibt $D_V(x)$ den Einfluß der Stichprobenauswahl.

Die Abhängigkeit von x kann durch Integration beseitigt werden. Die Terme B^2 und V werden im Zusammenhang mit dem Generalisierungsproblem auch als **Bias** und **Varianz** bezeichnet:

$$\begin{aligned} B^2 &= \frac{1}{2}\int D_B^2(x)p(x)dx \\ V &= \frac{1}{2}\int D_V(x)p(x)dx \end{aligned}$$

Beispiel: Die Bedeutung der beiden Terme kann mit der Wahl zweier extrem unterschiedlicher Abbildungsfunktionen $y(x)$ illustriert werden. Die Zieldaten r^p

$$r^p := h(x^p) + \epsilon^p$$

seien durch eine stochastisch gestörte glatte Funktion $h(x^p)$ erzeugt und es gelte $< \epsilon^p >= 0$. Zielstellung ist die Rekonstruktion der optimalen Abbildungsfunktion $< r|x >= h(x)$. Es werden folgende Fälle unterschieden:

1. Wahl einer festen Funktion $g(x)$ als Modell der Abbildungsfunktion des Netzes $y(x)$, siehe Abbildung 4.13. Dabei soll $g(x)$ völlig unabhängig von Ω_T sein.

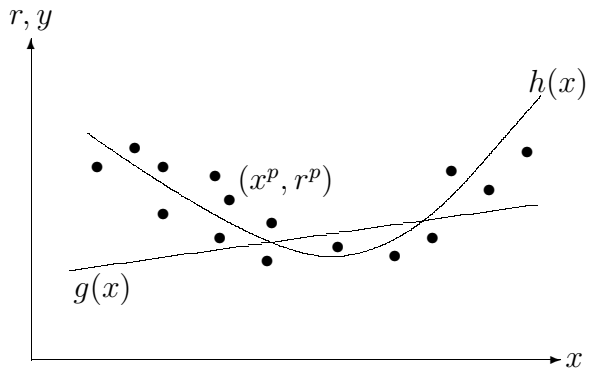


Abbildung 4.13: Falsche Modellwahl.

Die Abbildung verdeutlicht die schlechte Wahl des Modelles. Die Lösung hat folgende Eigenschaften:

- Die Varianz verschwindet, da $< y(x) >= g(x)$.
- Der Bias ist hoch, da $g(x)$ unabhängig von Ω_T.

2. Wahl einer flexiblen Funktion $g(x)$, die sich perfekt an die Daten aus Ω_T anpaßt, siehe Abbildung 4.14.

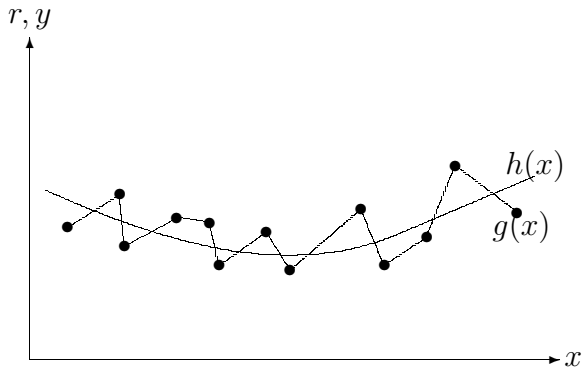


Abbildung 4.14: Falsche Plastizität.

Die Abbildung verdeutlicht die schlechte Wahl des Modelles, das die Daten auswendig lernt.

- Der Bias verschwindet an den Datenpunkten, da

$$< y(x) >=< h(x) + \epsilon >= h(x) =< r|x >$$

und ist klein in der Nachbarschaft der Datenpunkte.

- Die Varianz ist hoch, da

$$< (y(x) - < y(x) >)^2 >=< (y(x) - h(x))^2 >=< \epsilon^2 >$$

und entspricht der intrinsischen Varianz des Rauschens der Daten.

Bemerkung: Wenn $< \epsilon >= 0$, muß $< \epsilon^2 >= 0$ nicht gelten.
Hieraus schlussfolgern wir, dass Bias und Varianz ausbalanciert werden müssen durch die Suche eines gemeinsamen Minimums.

4.5 Methoden zur Verbesserung der Generalisierung

Aus dem letzten Beispiel des Abschnittes 4.4.2 ist erkennbar, welche Strategien zur Verbesserung des Lernerfolges mit gleichzeitiger Minimierung von Bias und Varianz führen.

1. Das **Einbringen von Vorwissen** bezüglich der unbekannten Funktion $h(x)$ liefert einen Zwang für die Modellfunktion $y(x)$, der vor allem den Bias reduziert.

2. Der **Zugriff auf mehr Daten** zum Training erhöht die Anzahl der Zwänge, welche die Varianz (Unsicherheit) des Modelles reduziert.

Die Verfügbarkeit von mehr Daten reduziert einerseits die Varianz. Andererseits wird aber auch die Verwendung komplexerer Modelle (mit mehr Freiheitsgraden) ermöglicht, was dazu beiträgt, den Bias zu reduzieren. Wächst die Zahl der Daten schneller als die verwendete Modellkomplexität, wird das Training eine Folge von Modellen liefern, die sich durch eine gleichzeitige Abnahme von Bias und Varianz auszeichnen.

Das Verhalten eines vorwärtsgerichteten neuronalen Netzes (FFNN) wie des MLPs im Grenzfall unendlich vieler Daten benennt man in der Statistik und in der Analysis auf unterschiedliche Weise, obwohl die Begriffe das Gleiche aussagen:

- Ein FFNN ist ein **konsistenter Schätzer** (siehe CV2-Skript) der Regressionsfunktion.
- Ein FFNN ist ein **universeller Approximator**.

Beide Aussagen drücken aus, dass eine beliebige Funktion mittels unbegrenzter Ressourcen (unendlich großer Stichprobe und unbegrenzter Modellkomplexität) beliebig gut approximiert werden kann. Diese theoretisch bedeutsame Aussage geht zurück auf ein berühmtes Theorem der Approximationstheorie von A.N. Kolmogorow (1957). G. Cybenko hat 1989 folgendes Theorem aufgestellt, welches die Rolle der von uns betrachteten sigmoiden Aktivierungsfunktion für die **universelle Approximation** unterstreicht:

Satz 4.1 *Für irgendeine kontinuierliche Funktion g, für die gilt $\lim_{x\to-\infty} g(x) = 0$ und $\lim_{x\to\infty} g(x) = 1$, sind Summen der Form*

$$S_\infty = \sum_{j=1}^{m} \lambda_j g(y_j^T x + \theta_j)$$

dicht in $\mathbb{C}([0,1]^n)$ in Bezug auf die L_∞-Norm.

Zur Erinnerung: Die L_∞-Norm ist die Maximum-Norm. Ein ähnlicher Satz kann für diskontinuierliche Funktionen wie die Stufenfunktion in Bezug auf die L_1-Norm aufgestellt werden.

Aber auch wenn Bias und Varianz verschwinden, wird der Fehler $E(w)$ bezüglich neuer Daten nicht verschwinden, weil der Fehleranteil E_2 (siehe Abschnitt 4.3.2) sensibel für das intrinsische Rauschen der Daten ist.

Im Folgenden werden drei Methoden angesprochen, um Lernen mit hinreichender Generalisierung zu ermöglichen. Dies sind die Regularisierung der Abbildungsfunktion, Aufgabenteilung durch Mixture of Experts und die Veränderung der Anzahl der Gewichte eines Netzes.

4.5.1 Regularisierung der Abbildungsfunktion

Ein komplexes nichtlineares Modell birgt die Gefahr in sich, die Adaption der Abbildungsfunktion des Netzes an die Daten zu übertreiben. Die Technik der Regularisierung erzeugt glattere Abbildungsfunktionen dadurch, dass ein **Bestrafungsterm** (penalty term) B zum Fehlerfunktional E hinzugefügt wird:

$$E_B(w) = E(w) + \alpha B(w)$$

Der Parameter α kontrolliert den Einfluß des Bestrafungsterms B auf die Lösung. Die Rolle des Bestrafungsterms ist darin zu sehen, die Funktion $y(x)$ daran zu hindern, sich zu gut an die Daten anzupassen. Damit wird die Steifigkeit von $y(x)$ erhöht. Die Schätzung $\widehat{y}_B(x)$ wird also einen Kompromiss darstellen zwischen der Anpassung an die Daten durch Minimierung von $E(w)$ und der Minimierung von B.

Die Methode der Regularisierung findet Eingang in viele bedeutende Schätzprobleme, z.B. in „Computer Vision“. Vielleicht am wichtigsten ist die Lösung eines inversen Problems. Die theoretischen Grundlagen der Regularisierung gehen auf Tikhonov und Arsenin (1977) zurück. Die Klasse der **Tikhonov-Regularisierer**, B_T, bringt den Bestrafungsterm B mit Ableitungen der zu regularisierenden Funktion in Verbindung. Ist $y(x)$ eine eindimensionale Funktion, dann gilt

$$B_T = \frac{1}{2} \sum_{r=0}^{R} \int_a^b h_r(x) \left(\frac{d^r y}{dx^r} \right)^2 dx,$$

wobei für die Funktionen $h_r(x)$ gilt: $h_r(x) \geq 0$ für $r = 0, \ldots, R-1$ und $h_R > 0$.

Bei der Methode der **Gewichtsabnahme** (weight decay) in Abschnitt 4.1.5.2 bestraft der Zusatzterm

$$B_{wd} = \frac{1}{2} \sum_i w_i^2$$

zu große Gewichte in der Fehlerfunktion des Backpropagation-Algorithmus. Ein Regularisierer dieser Art kann die Generalisierung wesentlich verbessern.

Betrachtet man den Gesamtfehler des MLP, dann gilt:

$$\nabla E_B = \nabla E + \alpha w$$

Ist μ die Lernrate, dann verändert sich der Gewichtsvektor $w(t)$ als Folge des Gradientenabstiegs, betrachtet in der kontinuierlichen Zeit t, entsprechend:

$$\frac{dw}{dt} = -\mu\alpha w$$

Diese Differentialgleichung hat die Lösung

$$w(t) = w(0) \exp(-\mu\alpha t),$$

worauf der Name des Verfahrens beruht.

Es ist leicht einzusehen, dass die Realisierung einer stark gekrümmten Abbildungsfunktion eines Mehrschichtnetzes hohe Werte der Gewichte erfordert. Andererseits führen kleine Gewichte zu kleinen Werten der Propagierungsfunktion. Aber für u klein ist $y = f_\sigma(u)$ nahezu linear. Also führt Gewichtsabnahme zu Generalisierung (mit Bias).

Regularisierung verschiebt das Optimum der Lösung von $\widehat{y}(x)$ nach $\widehat{y}_B(x)$. Dies entspricht natürlich einer Verschiebung der Systemparameter, d.h. der Gewichte. Im Fall einer quadratischen Fehlerfunktion $E(w)$ sei $\widehat{w}$ der Ort des Minimums von $E(w)$ ohne Regularisierung. Mit Regularisierung liegt das Minimum der Funktion $E_B(w)$ bei $\widehat{w}_B$ mit:

$$\widehat{w}_{jB} = \frac{\lambda_j}{\lambda_j + \alpha}\widehat{w}_j$$

Hier sind die λ_j die Eigenwerte der diagonalisierten Hessematrix der Fehlerfunktion.

Interessanterweise läßt sich auch die Anwendung bewußt verrauschter Daten im Sinn der Regularisierung interpretieren. Die Verwendung verrauschter Daten führt tatsächlich zu Generalisierung.

4.5.2 Kontrollierte Veränderung der Systemfreiheitsgrade

Die Architektur eines neuronalen Netzes hat bedeutenden Einfluß auf seine Leistungsfähigkeit. Unter Architektur soll die Zahl der verdeckten Schichten und der Neuronen sowie der Topologie der Verbindungen verstanden werden. Im Ergebnis der bisherigen Diskussion dieser Freiheitsgrade wird folgendes erwartet:
Die Speicherkapazität (Maß für Gedächtnis) bezüglich der Stichproben und die Fähigkeit zu Generalisierung aus den Stichproben verhalten sich invers zueinander.

- **Hohe Speicherkapazität** bedeutet hohe Plastizität der Abbildungsfunktion. Dies wird durch möglichst viele Freiheitsgrade erreicht.
- **Gute Generalisierung** bedeutet hohe Stabilität der Abbildungsfunktion. Dies wird durch Begrenzung der Freiheitsgrade des Systems erreicht.

Die Architektur hat in die Optimierung Zwänge einzubringen. Andererseits erfordert gute Generalisierung eine hinreichend große Stichprobe. Die durch die Daten eingebrachten **Zwänge** müssen ebenfalls groß genug sein.

Sei N_Ω die Anzahl der Lernstichproben, sei N_w die Anahl der Gewichte, sei N_y die Anzahl der Ausgabeknoten und sei N_z die Anzahl der verdeckten Knoten. Dann kann man die erforderlichen Randbedingungen für gute Generalisierung etwa in folgenden beiden Ungleichungen zusammenfassen:

$$N_z << N_\Omega \quad \text{und} \quad N_y N_\Omega >> N_w$$

Da solche Parameter wie N_z und N_w nicht a priori ermittelt werden können, geht man den alternativen Weg, durch Beobachtung der Leistung eines Netzes beim Training mit verschiedenen Parameterwerten für N_z und N_w das optimale Verhalten zu suchen. Dies bedeutet in der Praxis, dass der Basis-Backpropagation-Algorithmus nur ein Element in einer **zyklischen Trainingsprozedur** darstellt, die zudem noch folgendes ermöglichen muß:

- Systematische Exploration möglicher Architekturvarianten,
- Entscheidungen für eine systematische Veränderung der Architektur.

Eine erschöpfende Suche in einer begrenzten Menge möglicher Architekturen ist eine naive Variante, die erstens sehr rechenaufwendig und zweitens wegen der Begrenztheit der Netzmodelle auch nur unzureichend erfolgreich ist. Vielmehr bieten sich folgende systematischen Vorgehensweisen an:

1. **Erweiterung** der Zahl der Freiheitsgrade (**growing** algorithms): Ausgehend von einem kleinen Netz werden während des Trainings schrittweise Knoten und/oder Verbindungen hinzugefügt. Die Methode der Kaskadenkorrelation ist ein Beispiel hierfür im Fall von Mehrschichtnetzen. Im Fall von RBF-Netzen ist das DCS-Netz von Bruske und Sommer (1994) ein Beispiel für das Anwachsen des Netzes im Verlaufe des Trainings.

2. **Reduzierung** der Zahl der Freiheitsgrade (**pruning** algorithms): Ausgehend von einem relativ großem Netz werden während des Trainings schrittweise Knoten und/oder Verbindungen gestrichen.

3. **Aufgabenteilung** im Sinn der Konstruktion eines komplexeren Netzes aus einer Menge einfacher Netze stellt eine dritte Variante dar. Ein typischer Ansatz ist unter dem Namen **Mixture of Experts** bekannt. Diese Methode vermag auch, ein komplexeres Problem in eine Menge von einfacheren Teilproblemen automatisch zu zerlegen, wobei jedes Teilproblem durch ein separates Netz bearbeitet wird.

Kaskadenkorrelation Fahlman und Lebiere (1990)
Dieses Erweiterungsverfahren ist anwendbar für Mehrschichtnetze mit kontinuierlicher Ausgabefunktion. Die verdeckten Neuronen besitzen sigmoide Aktivierungsfunktionen.

- Start: als SLN, d.h. ohne verdeckte Schicht; voll verbunden mit justierbaren Verbindungen
- Erweiterung: mit einem verdeckten Neuron nach einigem Training, alternierend Training und Einfügen verdeckter Neuronen; Training stets separiert für die neu eingefügten Neuronen, d.h. die Gewichte der verdeckten Neuronen werden nach dem Antrainieren eingefroren.

Wegen der vollen Verbundenheit jedes verdeckten Neurons zu Eingabe- und Ausgabeschicht ergibt sich eine Kaskadenstruktur der Architektur, siehe Abbildung 4.15. Die Kaskadenkorrelation ist ein Spezialfall der allgemeinen Strategie, spezielle Module stets nur für eine Aufgabe anzutrainieren.

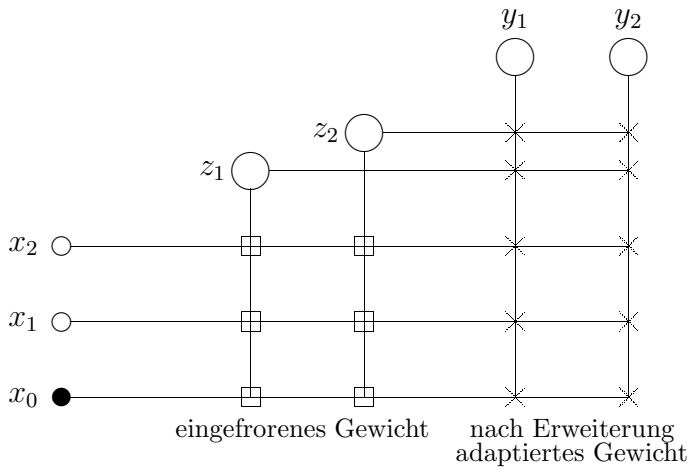


Abbildung 4.15: Kaskadenkorrelation.

Gewichtseliminierung
Es gibt verschiedene Algorithmen zum Abschneiden gewichteter Verbindungen. Diese können natürlich nur Verbindungen sein, deren Relevanz für die Abbildungsfunktion als niedrig eingeschätzt wird. Das allgemeine Schema umfaßt folgende Schritte als Zyklus organisiert:

- Training eines relativ großen, hochgradig verbundenen Netzes
- Ermittlung der relativen Bedeutung der Gewichte und Löschen der Verbindungen

Die Kernfrage, welche Verbindungen gelöscht werden sollen, wird entweder einfach über den **Betrag des Gewichtes** beantwortet oder über ein **Maß der Auffälligkeit** der Gewichte.

Im Fall normierter Eingaben und Ausgaben bedeutet $|w| \approx 0$, dass die Verbindung kaum Einfluß auf die Optimierung der Ausgabefunktion $y(x)$ hat.

Ähnlich zur Regularisierung der Abbildungsfunktion mittels Gewichtsabnahme

$$E_B(w) = E(w) + \frac{\alpha}{2} \sum_i w_i^2,$$

tendiert das Training eines solchen Netzes durch Minimierung des Funktionals $E_B(w)$ zur Verkleinerung jener Gewichte, die nicht wesentlich zur Reduktion des Fehlers $E(w)$ beitragen. Man könnte also jene Gewichte abschneiden, die unter eine

zu wählende Schwelle fallen. Dieses Verfahren unterstützt aber die Herausbildung vieler kleiner Gewichte anstelle weniger große Gewichte. Durch Modifikation des Gewichtsabnahmeterms kann diese Tendenz in Richtung weniger große Gewichte umgekehrt werden:

$$E_B(w) = E(w) + \alpha \sum_i \frac{w_i^2}{w_s^2 + w_i^2}$$

Hier ist w_s ein zu wählender Skalenparameter in der Größenordnung Eins.

Neben diesem Verfahren, das den Trainingsprozeß selbst modifiziert, gibt es solche, die zyklisch mit dem Training alternieren. Varianten eines solchen Verfahrens nutzen die Hesse-Matrix $H = (h_{ij})$, um ein Maß der Auffälligkeit für Gewichte zu berechnen. Dazu werden Änderungen des Fehlers beobachtet, die durch kleine Störungen δw_i der Gewichte w_i hervorgerufen werden:

$$\delta E = \sum_i \frac{\partial \epsilon}{\partial w_i} \delta w_i + \frac{1}{2} \sum_i \sum_j h_{ij} \delta w_i \delta w_j + O(\delta w^3) \quad \text{mit}$$

$$h_{ij} := \frac{\partial^2 E}{\partial w_i \partial w_j}$$

Im Zustand der Konvergenz des Netzes verschwindet der lineare Anteil des Fehlerfunktionals (Anstieg geht gegen Null), so dass

$$\delta E = \frac{1}{2} \sum_i h_{ii} \delta w_i^2$$

gilt, wenn die Hessematrix durch einen Diagonalmatrix approximiert wird. Der Wert von δE wird als Maß der Auffälligkeit bezeichnet. Dies führt zu einem zyklischen Verfahren des Gewichtsabschneidens mit folgenden Schritten.

1. Training eines relativ großen Netzes bis ein Abbruchkriterium erfüllt ist.
2. Berechnung der zweiten Ableitungen h_{ii} für jedes der Gewichte und damit der Auffälligkeit.
3. Sortierung der Auffälligkeiten und Löschen jener Gewichte mit niedriger Auffälligkeit.

Die Berücksichtigung der vollständigen Hessematrix

$$\delta E = \frac{1}{2} \delta w^T H \delta w$$

bringt eine weitere Verbesserung. Allerdings muß hier die inverse Hessematrix H^{-1} berechnet werden.

Ähnliche Verfahren, welche die Auffälligkeit eines Neurons bewerten, gestatten das Herausschneiden wenig signifikanter Neuronen.

4.5.3 Generalisierung im Ensemble und Spezialisierung

In diesem Abschnitt soll das Erreichen einer mittleren Generalisierungsleistung diskutiert werden, wenn beim Training ein Ensemble von neuronalen Netzen genutzt wird. Dabei sind zwei grundlegende Fälle zu unterscheiden:

1. Das Netz-Ensemble besteht aus identischen Individuen, die auf unterschiedlichen Datensätzen antrainiert werden. Das Ziel ist die optimale Parametrisierung eines Netzes.

2. Das Netz-Ensemble besteht aus unterschiedlichen Individuen, die auf dem gleichen Datensatz antrainiert werden. Das Ziel ist, die optimale Abbildungsfunktion eines Netzensembles zu finden.

Diese Herangehensweise ist nicht auf Mehrschichtnetze begrenzt und im Fall 2 können auch Netze ganz unterschiedlichen Typs verwendet werden.
Werden den einzelnen Netzen im Fall 2 unterschiedliche Datensätze zugeordnet, kann man das Ensemble als eine Menge von (lokalen) Experten ansehen. Diese Experten können als Sklaven unter der Regie eines Masters operieren, welcher die Aufgabenzuweisung übernimmt. Eine solche **Master-Slave-Architektur** ist auch unter dem Namen **Mixture of Experts** bekannt.

Zunächst wird Fall 1 betrachet. Mit der in Abschnitt 4.4.1 eingeführten Hierarchie von Datenmengen $(\Omega_T \cup \Omega_E) \subset \Omega_R \subset \Omega_U$, $\Omega_T \cap \Omega_E = \emptyset$, wird klar:

1. Die erreichte Generalisierung hängt kritisch von der Auswahl von Ω_T ab.

2. Die Bewertung der Generalisierung mit der Teststichprobe Ω_E trägt das gleiche Risiko. Dabei wird implizit die Annahme gemacht, dass die strukturellen und statistischen Eigenschaften von Ω_E denen von Ω_T entsprechen.

Es ist nicht das Ziel, Generalisierung bezüglich Ω_T zu erhalten, sondern bezüglich Ω_R. Es wird die beste Leistung des gefundenen Modells bezüglich neuer Daten gesucht. Der Generalisierungsfehler ist also eine Art Prädiktionsfehler. Es gibt verschiedener Methoden, um den Prädiktionsfehler zu reduzieren, indem aus einer gegebenen Stichprobenmenge Ω_P unterschiedliche Stichprobenmengen $\Omega_{P'}$, $P' < P$, ausgewählt werden, um das Training zu realisieren. Diese Methoden beruhen alle auf der faktischen Situation, dass der Zugriff auf neue Daten unverhältnismäßig hohe Kosten verursacht. Deshalb wird Generalisierung dadurch provoziert, dass über ein Ensemble von zufällig aus Ω_P ausgewählten Stichproben $\Omega_{P'}$ die mittlere Lösung ermittelt wird.

Jackknife (Leave-one-out-Methode):
Es gilt $P' = P - 1$, also werden P Berechnungen auf etwas unterschiedlichen Datensätzen durchgeführt. Dies kann parallel auf identischen Netzen oder sequentiell mit zufälliger Initialisierung geschehen. Eine Verallgemeinerung hiervon ist

Kreuzvalidierung:
Ω_P wird in K gleich große Mengen $\Omega_{P/K}$ zufällig unterteilt. Es werden K Berechnungen auf den Stichprobenmengen $\Omega_P \setminus \Omega_{P/K}$ durchgeführt und $\Omega_{P/K}$ wird zum Testen verwendet. Dies verhindert, dass die Teststichprobe einen Bias in das gemittelte Ergebnis der K Lösungsvorschläge einbringt.

Diese Lernstrategien sind von großer praktischer Bedeutung. Aber sie kosten Rechenleistung. Andere Methoden der Verbesserung der Generalisierung sind dem Fall 2 zuzuordnen. Die Methode Netz-Komitee erzeugt sehr gute Verbesserung des Prädiktionsfehlers bei begrenztem zusätzlichem Rechenaufwand. Die Leistung eines Netz-Komitees kann tatsächlich besser sein, als die seines besten Mitgliedes.

Netz-Komitee:
Seien $y_k(x)$, $k = 1, \ldots, K$ die Ausgabefunktionen $\mathbb{R}^n \to \mathbb{R}$ für K Netzmodelle. Diese Netzmodelle können neuronale Netze verschiedenen Typs sein, oder Mehrschichtnetze verschiedener Architektur oder gleichartige Netze, die bezüglich unterschiedlicher lokaler Minima der Fehlerfunktion antrainiert werden. Jede Abbildungsfunktion kann modelliert werden als gestörte ideale Funktion $h(x)$:

$$y_k(x) = h(x) + \epsilon_k(x)$$

Der Erwartungswert des MSE für das Modell $y_k(x)$ sei:

$$E_k = \mathrm{E}\left\{\epsilon_k^2\right\} = \mathrm{E}\left\{\left(y_k(x) - h(x)\right)^2\right\} \quad \text{bzw.}$$

$$\mathrm{E}\left\{\epsilon_k^2\right\} = \int \epsilon_k^2(x) p(x) dx$$

Der mittlere Fehler jedes Netzes bei isolierter Arbeitsweise ist:

$$E_I = \frac{1}{K} \sum_{k=1}^{K} E_k$$

Sei die Ausgabe des Komitees gegeben durch die mittlere Ausgabe aller K Mitglieder des Komitees $y_C(x) = \frac{1}{K} \sum_{k=1}^{K} y_k(x)$, dann ist der Fehler des Komitees:

$$E_C = \mathrm{E}\left\{\left(\frac{1}{K} \sum_{k=1}^{K} \epsilon_k^2\right)^2\right\}$$

Seien die Einzelfehler vom Mittelwert Null und unkorreliert:

$$\mathrm{E}\left\{\epsilon_k\right\} = 0 \quad \text{und} \quad \mathrm{E}\left\{\epsilon_k \epsilon_l\right\} = 0 \quad \text{für } k \neq l$$

Dann gilt für die Fehler bei isolierter Arbeitsweise und als Komitee die Relation:

$$E_C = \frac{1}{K^2} \sum_{k=1}^{K} \mathrm{E}\left\{\epsilon_k^2\right\} = \frac{1}{K} E_I$$

Das bedeutet, dass die Summe des quadratischen Fehlers um einen Faktor K reduziert werden kann, wenn die Ausgaben der K Netze gemittelt werden.

Tatsächlich ist der Gewinn an Performanz in der Praxis geringer, weil die über die Einzelfehler ϵ_k gemachten Annahmen gewöhnlich nicht erfüllt sind.

Da das Ergebnis so interpretiert werden kann, dass die Varianz der Lösungen $y_k(x)$ durch die Mittelung über mehrere Lösungen reduziert wird, erhält man folgende Empfehlung für die $y_k(x)$:
Die Einzellösungen sollten nicht einer optimalen Balance von Bias und Varianz entsprechen. Vielmehr sollten sie relativ kleinen Bias-Anteil am Fehler ϵ_k besitzen, da die Varianz durch die Mittelung über das Komitee reduziert wird.

Mixture of Experts:
Diese Methode verbindet gewissermaßen die Grundidee der Kreuzvalidierung mit dem Modell der Arbeitsteilung in einem Komitee. Die Architektur der Mixture of Experts bietet sich an, wenn die Abbildungsfunktionen $y_k(x)$ verschieden sind für unterschiedliche Regionen des Eingaberaums. Sei $\Omega_T = \cup_{t=1}^{T} \Omega_t$ mit $\Omega_t \cap \Omega_s = \emptyset$. Gesucht werden die Abbildungsfunktionen $y_t(x^t)$, $x^t \in \Omega_t$.
Ein zusätzliches Netz, das die Rolle des Masters übernimmt (gating net), hat Sicht auf Ω_T und entscheidet, welches der T Sklavennetze die Ausgabe berechnen soll.

Das Mixture-of-Experts-Modell von Jacobs, Jordan, Nowlan und Hinton (1991) findet außerdem die geeignete Partitionierung des Eingaberaumes und damit der Lernaufgabe. Hierzu wird auf die bedingte Wahrscheinlichkeitsdichte der Zielfunktion $p(r|x)$ zurückgegriffen (siehe Abschnitt 4.3.2) und diese als **Mischungsmodell** repräsentiert, das heißt als Linearkombination adaptiver Kernfunktionen (siehe Abschnitt 3.9.2):

$$p(r|x) = \sum_{t=1}^{T} \alpha_t(x) \varphi_t(r|x)$$

Die $\alpha_t(x)$ werden als Mischungskoeffizienten der Kernfunktionen $\varphi_t(r|x)$ bezeichnet. Als Kernfunktionen kommen Gaußfunktionen mit m als Dimension von r zur Anwendung:

$$\varphi_t(r|x) = \frac{1}{(2\pi)^{m/2}} \exp\left(-\frac{|r - \mu_t(x)|^2}{2}\right)$$

Es werden Einheits-Kovarianzmatrizen der Kernfunktionen angenommen.

Wird jedem Expertennetz eine solche Gaußfunktion zugeordnet, ist die Ausgabe des Experten t der entsprechende Mittelwert $\mu_t(x)$ der Funktion $\varphi_t(r|x)$. Wenn die Ausgaben des Masternetzes mit $\gamma_t(x)$ bezeichnet werden, dann sind die gesuchten Mischungskoeffizienten:

$$\alpha_t(x) = \frac{\exp(\gamma_t(x))}{\sum_{s=1}^{T} \exp(\gamma_s(x))}$$

Folglich berechnet das Masternetz einen Mischungskoeffizienten für jedes Sklavennetz, der dessen Anteil an der Ausgabe des Ensembles steuert.

Ein Mixture-of-Expert-Netz wird trainiert, indem die Fehlerfunktion

$$E = -\sum_{p=1}^{P} \log \left(\sum_{t=1}^{T} \alpha_t(x^p) \varphi_t(r^p | x^p) \right)$$

gleichzeitig für alle Gewichte der Expertennetze und des Masternetzes minimiert wird.

Das trainierte Netz wird so verwendet, dass die Eingabe dem Masternetz präsentiert und dessen maximale Ausgabe $y_{\max,\mathrm{t}}$ genutzt wird, um den geeigneten Experten zu selektieren. Die Eingabe wird dann diesem Experten präsentiert und dessen Ausgabe $\mu_t(x)$ stellt die Ausgabe des gesamten Systems dar, siehe Abbildung 4.16.

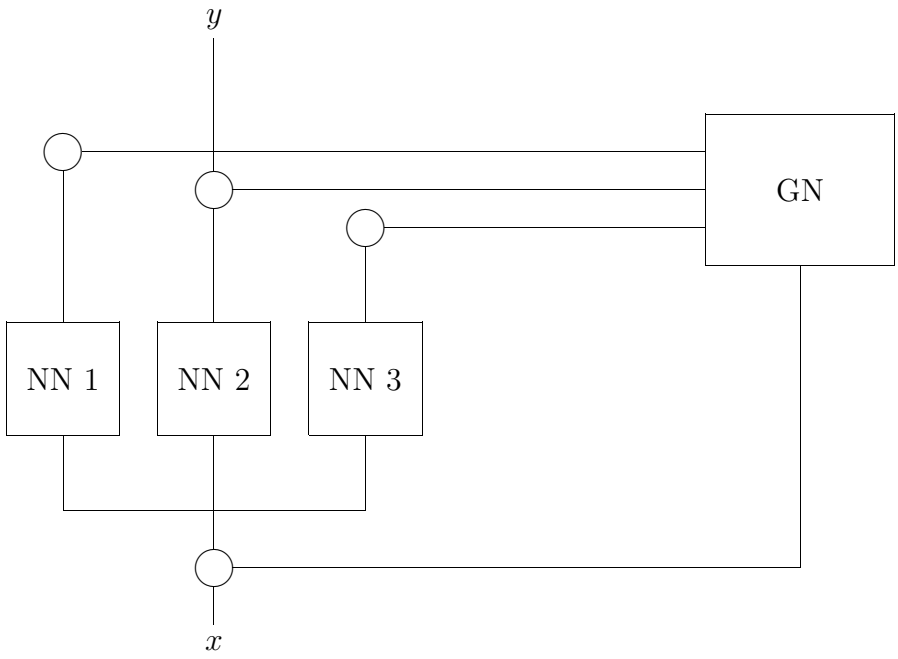


Abbildung 4.16: Ein Mixture-of-Expert-Netz.

4.6 Mehrschichtnetze und Komprimierung von Daten

4.6.1 Kodierung und Komprimierung von Daten

Wir haben bisher im Zusammenhang mit Mehrschichtnetzen sehr unterschiedliche Aufgabenklassen für diese neuronale Architektur angesprochen. Dies waren nichtlineare Regression, nichtlineare Entscheidungsgrenzen zur Klassifikation.

Ferner wurden im Zusammenhang mit der Generalisierung die Interpolation und Extrapolation (Prädiktion) mit Hilfe der Regressionsfunktion thematisiert.

Wir forderten für gute Generalisierung, dass in den verdeckten Schicht wenige Neuronen enthalten sein sollten, um hinreichende Abstraktion der Funktion $\widehat{y}(x)$ von den Stichproben (x^p, r^p) zu erhalten.

Diese Eigenschaft der verdichteten Repräsentation des funktionalen Zusammenhanges $y(x)$ in den verdeckten Schichten kann man als **Datenkomprimierung** interpretieren. Es ist möglich, ein Mehrschichtnetz speziell für den Zweck der Datenkomprimierung zu entwerfen und zu trainieren. Wir werden ein Beispiel zur **Bildkomprimierung** behandeln. Zuvor werden jedoch kurz in die Grundlagen eingeführt.

Die Komprimierung von Daten stellt eine spezielle Art von **Kodierung** dar. Der in der Nachrichtentechnik gebräuchliche Begriff Kodierung ist identisch mit dem in der Informatik verwendeten Begriff **Repräsentation**. Die Informatik beschäftigt sich zu einem wesentlichen Teil damit, für bestimmte Aufgaben geeignete Repräsentationen von Daten zu entwickeln. Ist der Kode für eine Datenmenge in einer speziellen Repräsentation kürzer als in einer anderen, spricht man von **Komprimierung**.

- **Verlustfreie Komprimierung**: Es existiert eine inverse Kodierungsvorschrift, so dass die Daten in der Ur-Repräsentation ohne Verlust aus der Bild-Repräsentation rekonstruiert werden können.
- **Verlustbehaftete Komprimierung**: Es existiert eine Vorschrift zur Dekodierung aus der Bild-Repräsentation in die Ur-Repräsentation, die aber nicht die inverse Kodierungsvorschrift ist. Demzufolge treten bei der Dekodierung Datenverluste auf.

In der linearen Algebra bedeutet Kodierung oft **Projektion** eines Vektors aus einem **Basissystem** auf ein anderes Basissystem. Die Karhunen-Loeve-Transformation (KLT), Fisher's Diskriminante, aber auch die Fourier-Transformation (FT) sind Beispiele hierfür. Sowohl bei der KLT als auch bei der FT sind beide Basissysteme gleich mächtig und lassen sich durch lineare Operationen ineinander überführen.
Werden jedoch Projektionskoeffizienten (Eigenwerte bzw. Frequenzanteile) weggeschnitten, so entsteht eine Approximation, die in gewisser Hinsicht auch als Komprimierung anzusehen ist. In der Nachrichtentechnik wird eine Modifikation des Kodes in der Bildrepräsentation, welche zur Komprimierung führt, als **Quantisierung** bezeichnet. Da bei der Kodierung keine relevante Information verloren gehen soll, muß sie im Fall der Komprimierung verlagert werden. Ist die Bild-Repräsentation (u.U. nach Quantisierung) also speichereffizienter als die Ur-Repräsentation, wird ein Teil der Information in die Kenntnis der Kodierungsvorschrift verlagert. Abbildung 4.17 faßt das Schema der Kodierung und

Dekodierung von Daten zusammen.

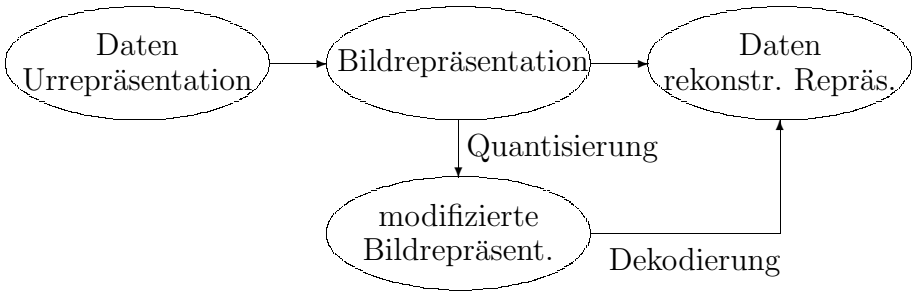


Abbildung 4.17: Kodierung und Dekodierung von Daten.

Komprimierung mittels Mehrschichtnetzen:
Aus dem obigen Schema folgt unmittelbar, wie ein Mehrschichtnetz (MLP) aussehen müßte, um Daten zu komprimieren. Dies zeigt Abbildung 4.18. Die Ausgabeschicht hat ebensoviele Knoten wie die Eingabeschicht. Ziel des Trainings ist die identische Abbildung der Eingabe auf die Ausgabe. Dies soll über das Nadelöhr der verdeckten Schicht mit weit weniger Knoten erfolgen.

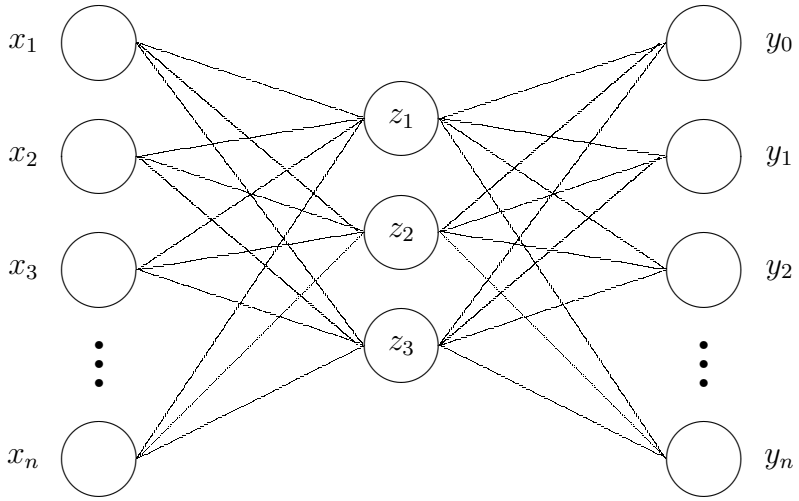


Abbildung 4.18: Mehrschichtnetz zur Datenkompression.

Ist das Netz trainiert, wird die Ausgabeschicht abgetrennt und die verdeckte Schicht wird zur Ausgabeschicht. Die verdeckte Schicht hat ein komprimiertes Modell der Daten gelernt.

In Abschnitt 2.6.1 wurde im Zusammenhang mit der Lösung des XOR-Problems die Umkodierung des Eingaberaumes in einem Regionenraum behandelt.

4.6.2 Bildkodierung mittels Gabortransformation

Diese Verfahren geht auf Daugman (1988) zurück.
Die Bildkodierung mittels Gabortransformation führt zu einer komprimierten Repräsentation. Die Gabortransformation spannt ein Basissystem auf, das diese sparsame Repräsentation erzeugt.

Kanonische Basis:
Die kanonische Basis eines Bildes f ist der **Signalraum**. Jeder Bildpunkt (m, n) im **Ortsraum** spannt eine Dimension des Signalraumes auf. Ein Bild der Ausdehnung 256×256 Bildpunkte wird demzufolge als ein Punkt (bzw. Vektor) im $256^2 = 65536$-dimensionalen Signalraum repräsentiert. Dabei ergibt sich das Volumen des Signalraumes aus dem Umfang maximaler Grauwerte, die jeder Bildpunkt tragen kann. Werden Grauwerte $g = f_{m,n}$ mit 8 Bit repräsentiert, variieren diese im Bereich $0 \leq g \leq 255$.
Fouriertransformation:
Die 2D-Fouriertransformation $\mathcal{F}$ ist eine lineare Integraltransformation $F = \mathcal{F}\{f\}$. Sie ist global, wirkt also über der gesamten Funktion f. Diese Transformation erzeugt eine vollständige and orthogonale Abbildung des Signalraumes auf sich selbst. Es kann auch gesagt werden, dass sie eine Abbildung des Ortsraumes mit den Koordinaten (m, n) auf den **Frequenzraum** mit den Koordinaten (u, v) erzeugt. Jeder Wert von $F_{u,v}$ ergibt sich in obigem Beispiel aus 65536 Werten von $f_{m,n}$. Umgekehrt entspricht jeder Wert $f_{m,n}$ der linearen Überlagerung von 65536 Werten von $F_{u,v}$. Man nennt dies die Fourier-Rücktransformation, $\mathcal{F}^{-1}$. Die Fouriertransformation ist also invertierbar.

Die Fouriertransformation selbst erzeugt keine Komprimierung der Daten eines Bildes, weil die Basisfunktionen jeder Repräsentation eine Deltafunktion ist (ein Bildpunkt (m, n) bzw. eine Frequenz (u, v)). Sie macht aber die spektralen Eigenschaften eines Signals explizit. Dies erfordert eine komplexwertige Repräsentation eines reellwertigen Signals. Die Fouriertransformation verdoppelt also den Speicheraufwand eines Signals. Eine verlustbehaftete geglättete Repräsentation eines Bildes erhält man durch Abschneiden hoher Frequenzen. Diese Operation sei mit Q bezeichnet. Dann ist $\widehat{f_Q} \approx f$ mit:

$$\widehat{f_Q} = \mathcal{F}^{-1}\{Q\{\mathcal{F}\{f\}\}\}$$

Gabortransformation:
Die Gabortransformation ist eine **lokale Integraltransformation** eines Signals in eine spektrale Repräsentation. Sie heißt auch **lokale Fouriertransformation**. Während das Basissystem der Fouriertransformation die komplexen harmonischen Funktionen sind, im 1D-Fall

$$\Phi_{u,x} = \exp(-j2\pi ux) = \cos(2\pi ux) - j\sin(2\pi ux),$$

werden diese harmonischen Funktionen bei der Gabortransformation lokal durch eine Gaußfunktion gewichtet bzw. ausgeblendet. Eine eindimensionale **Gaborfunktion** h_G,

$$h_G(m; m_0, u_0, \sigma) = \exp\left(-\pi \frac{(m-m_0)^2}{\sigma^2}\right) \exp\left(-j2\pi \frac{u_0(m-m_0)}{M}\right)$$

stellt eine Basisfunktion der (eindimensionalen) Gabortransformation eines Signals der Länge M dar. Hierbei sind m_0 der Aufpunkt der Gaußfunktion und u_0 die Modulationsfrequenz. Im 2D-Fall haben die Gaborfunktionen entsprechend mehr Freiheitsgrade: $h_G(m, n; m_0, n_0, u_0, v_0, \sigma, \theta)$
Hier gibt θ die Orientierung einer elliptischen Ortskurve der einhüllenden Gaußfunktion an.

Anstelle der Dirac-Basis der Fouriertransformation, bilden Gaußfunktionen das Basissystem der Gabor-Repräsentation eines Bildes. Es seien I (Indexmenge) Gaborfunktionen h_i mit unterschiedlichen Parametern (Frequenzen (u_{0i}, v_{0i}) ausgestattet, um eine Gaborrepräsentation $\widehat{f}_G$ am Aufpunkt (m_0, n_0) zu berechnen. Das Modell der Bildfunktion läßt sich dann für jeden Aufpunkt (m_0, n_0) formulieren als:

$$\widehat{f}_{m_0,n_0} = \sum_{i=0}^{I-1} \alpha_i h_i(m_0, n_0)$$

Die Repräsentation $\widehat{f}_G$ besitzt sehr viel Redundanz, weil sich die Basisfunktionen bei der Abtastung stark überlagern. Die **Gabor-Wavelet-Transformation** beseitigt diese Redundanz.

Im Unterschied zum Basissystem der Fouriertransformation ist das Basissystem der Gabortransformation nicht orthogonal. Es existiert also keine inverse Gabortransformation.

MMSE-Schätzung der Projektionskoeffizienten:
Die Projektionskoeffizienten α_i werden durch Minimierung des mittleren quadratischen Fehlers E zwischen dem Modell $\widehat{f}_G$ und der Bildfunktion f berechnet:

$$E = \frac{1}{2} \sum_{m_0=0}^{M-1} \sum_{n_0=0}^{N-1} \left(f_{m_0,n_0} - \widehat{f}_{m_0,n_0}\right)^2$$

Die partiellen Ableitungen des Fehlerfunktionals lauten:

$$\frac{\partial E}{\partial \alpha_i} = -\sum_{m_0,n_0} f_{m_0,n_0} h_i(m_0, n_0) + \sum_{m_0,n_0} \left(\sum_{k=0}^{I-1} \alpha_k h_k(m_0, n_0)\right) h_i(m_0, n_0)$$

Nullsetzen und Umstellen liefert die I Normalengleichungen $(0 \le i \le I-1)$:

$$\sum_{m_0,n_0} f_{m_0,n_0} h_i(m_0, n_0) = \sum_{m_0,n_0} \left(\sum_{k=0}^{I-1} \alpha_k h_k(m_0, n_0)\right) h_i(m_0, n_0)$$

Dies sind I Gleichungen mit I Unbekannten.

Wären die Basisfunktionen h_i orthogonal, würde die rechte Seite der Gleichung nur für $k = i$ nicht verschwinden. Jede der I Gleichungen hätte dann eine einzige Unbekannte und die Lösung für die Projektionskoeffizienten wäre

$$\alpha_i = \frac{\sum_{m_0,n_0} h_i(m_0,n_0) f_{m_0,n_0}}{\sum_{m_0,n_0} h_i^2(m_0,n_0)}$$

Da aber die h_i nicht orthogonal sind, würde eine geschlossene Lösung die Invertierung der Autokorrelationsmatrix

$$R = \left(\sum_{m_0,n_0} h_k(m_0,n_0) h_i(m_0,n_0) \right)$$

erfordern. Für ein 256×256-Bild hat R die Ausdehnung 65536×65536. Diese Matrix ist algebraisch nicht zu invertieren. Die Stirlingsche Näherung der Komplexität der Invertierung dieser Matrix liefert $2.5 \cdot 10^{287157}$ Gleitpunktoperationen.

Lösung mittels MLP:
Deshalb wird die Lösung des MMSE-Problems durch Trainieren eines MLP vorgeschlagen. In Abbildung 4.19 ist dargestellt, wie die Normalengleichungen auf die Struktur eines MLP mit drei verdeckten Schichten abgebildet werden. Dabei besitzt aber nur eine Schicht variable Gewichte, während die Basisfunktionen zwei Schichten mit festen Berechnungsvorschriften liefern. Die Basisfunktionen h_i stelle man sich am Aufpunkt (m_0, n_0) im Bild vor. Die Summation erfolgt über einen begrenzten Einzugsbereich dieser Funktionen, den Support. Das Netz ist also nicht voll verbunden. Die Optimierung erfolgt durch systematische Verschiebung dieses Supports über die gesamte Bildfunktion. Da die Fehlerfunktion ein globales Minimum hat, liefert Gradientenabstieg eine optimale Lösung α_i für ein einzelnes antrainiertes Bild.

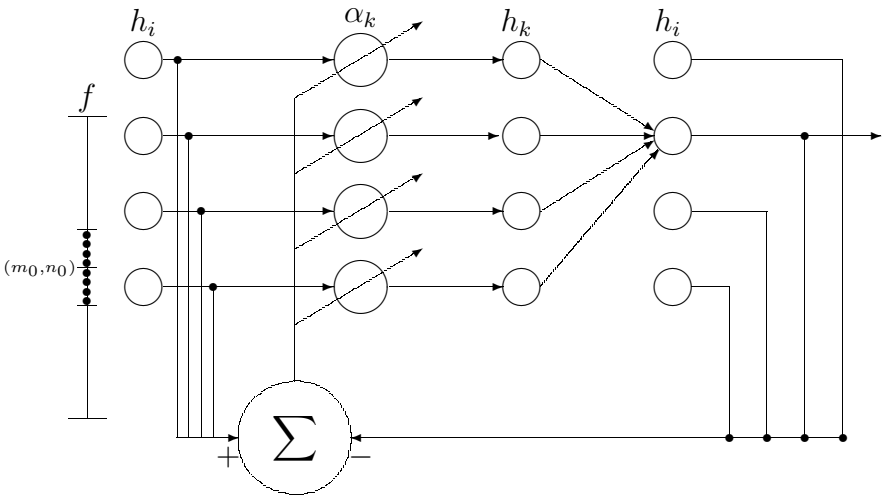
Kompression:
Die Projektionskoeffizienten $\widehat{\alpha}_i$ werden zur Berechnung des komprimierten Bildes $\widehat{f_G}$ benutzt.

Gewöhnlich wird die **Entropie** als Informationsmaß zur Bewertung einer Repräsentation herangezogen.

Die Entropie

$$S = -\sum_{g=0}^{G-1} \widehat{p}_g \log \widehat{p}_g$$

wertet die Häufigkeitsverteilung $\widehat{p}_g$ der Grauwerte g eines Bildes aus, also das Histogramm. Sie liefert die Anzahl der für die Repräsentation erforderlichen Bits.



$$\sum_{m_0,n_0} f_{m_0,n_0} h_i(m_0,n_0) = \sum_{m_0,n_0} \left(\sum_{k=0}^{I-1} \alpha_k h_k(m_0,n_0) \right) h_i(m_0,n_0)$$

Abbildung 4.19: Abbildung der Normalengleichungen auf ein MLP.

Im Fall $G = 256$ (8 Bit) liefert die Entropie für **weißes Rauschen** $S_R = 8$. Da in einem Bild, das nur weißes Rauschen darstellt, keinerlei Korrelation der Grauwerte zwischen den einzelnen Bildpunkten auftritt, ist die volle Speichertiefe von 8 Bit pro Bildpunkt zur Repräsentation nötig. Es gibt kein Verfahren für eine sparsamere Kodierung.

Für das Originalbild „Lena“[2] erhält man $S_L = 7.57$. Also ein Ergebnis, das der originalen Grauwertkodierung mit 8 Bit entspricht. Obwohl die Differenz $S_R - S_L = 0.43$ relativ klein ist, ist der Unterschied des Bildes „Lena“ zu einem Rauschbild sehr groß.

Für die Projektion $\widehat{f}_G$ des Bildes „Lena“ auf das Basissystem der Gabor-

[2]Die Geschichte des Bilds „Lena“ (`http://www.cs.cmu.edu/~chuck/lennapg/lenna.shtml`):

Alexander Sawchuk estimates that it was in June or July of 1973 when he, then an assistant professor of electrical engineering at the USC Signal and Image Processing Institute (SIPI), along with a graduate student and the SIPI lab manager, was hurriedly searching the lab for a good image to scan for a colleague's conference paper. They had tired of their stock of usual test images, dull stuff dating back to television standards work in the early 1960s. They wanted something glossy to ensure good output dynamic range, and they wanted a human face. Just then, somebody happened to walk in with a recent issue of Playboy.

The engineers tore away the top third of the centerfold so they could wrap it around the drum of their Muirhead wirephoto scanner, which they had outfitted with analog-to-digital converters (one each for the red, green, and blue channels) and a Hewlett Packard 2100 minicomputer. The Muirhead had a fixed resolution of 100 lines per inch and the engineers wanted a 512×512 image, so they limited the scan to the top 5.12 inches of the picture, effectively cropping it at the subject's shoulders.

Transformation erhält man die Entropie $S_G = 2.55$. Es werden also nur 3 Bits zur Kodierung dieses Bildes pro Bildpunkt benötigt. Der Komprimierungsfaktor $\beta = 2^{S_L}/2^{S_G} \approx 32$ macht deutlich, dass S_G weit mehr Korrelation enthält als S_L. Man sagt auch, S_L enthält mehr Details als S_G. Jedoch sind diese Details für die visuelle Wahrnehmung nicht zwingend relevant. Die Ursache liegt darin, dass eine Projektion auf die Gaborfunktion innerhalb des lokalen Supports die Korrelation der Bildpunkte in S_G stark erhöht. In den folgenden Abbildungen sind das Originalbild „Lena“, das Histogramm dieses Bildes und das Histogramm von $\widehat{f_G}$ zu sehen.

Abbildung 4.20: Lena

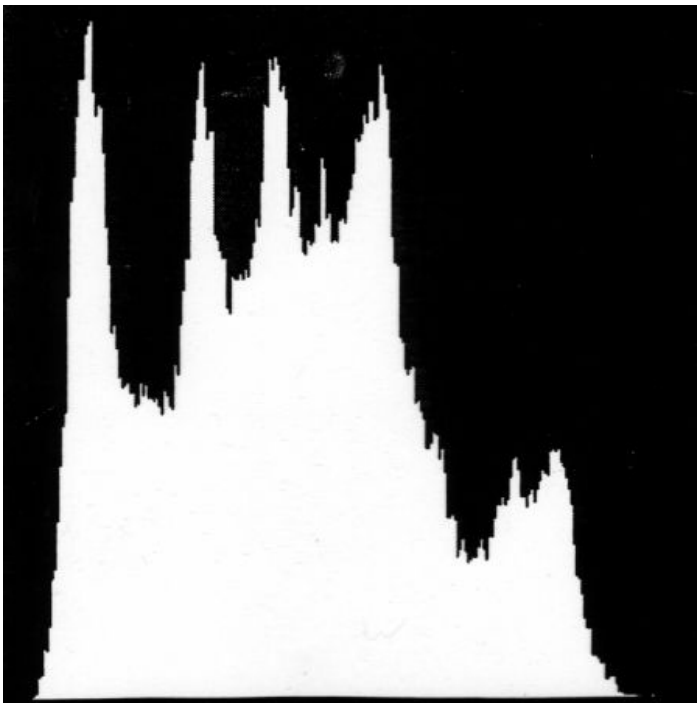


Abbildung 4.21: Histrogramm Lena, Entropie 7.57 Bit

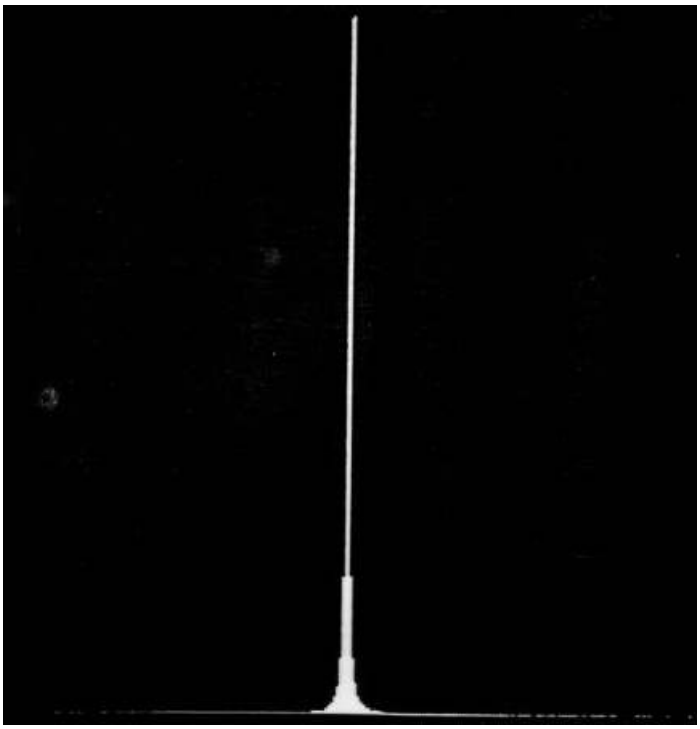


Abbildung 4.22: Histrogramm Lena, Entropie 2.55 Bit

Literaturverzeichnis

[Bishop 2004] Bishop, Christopher M.: Neuronal Networks for Pattern Recognition. Oxford University Press, 2004

[Bruske und Sommer 1994] Bruske, Jörg ; Sommer, Gerald: Dynamic Cell Structures. In: NIPS, 1994, S. 497–504

[Daugman 1988] Daugman, J.: Complete Discrete 2D-Gabor-Transforms by Neural Networks for Image Analysis and Compression. In: IEEE-Transactions on Acoustics, Speed and Signal Processing 36 (1988), S. 1169–1179

[Daugman 2000] Daugman, J.: Biometric decision landscapes / Computer Lab. 2000 (Technical Report UCAM-TR-482). – Forschungsbericht

[Duda u. a. 2004] Duda, Richard O. ; Hart, Peter E. ; Stork, David G.: Pattern Classification. 2. Wiley Singapore, 2004

[Fahlman 1988] Fahlman, S.E.: An empirical study of learning speed in back-propagation networks. In: Technical Report CMUCS. School of Computer Science, Carnegie Mellon University, 1988, S. 88–162

[Fahlman und Lebiere 1990] Fahlman, S.E. ; Lebiere, C.: The Cascade-Correlation Learning Architecture. In: Touretzky, D.S. (Hrsg.): Advances in Neural Information Processing Systems. Morgan-Kaufmann, 1990

[Geman u. a. 1992] Geman, S. ; Bienenstock, E. ; Doursat, R.: Neural Networks and the Bias/Variance Dilemma. In: Neural Computation 4 (1992), S. 1–58

[Giles u. a. 1988] Giles, C.L. ; Griffin, R.D. ; Maxwell, T.: Encoding Geometric Invariances in Higher-Order Neural Networks. Bd. 0. S. 301–309, American Institute of Physics, 1988

[Hotelling 1933] Hotelling, H.: Analysis of a complex of statistical variables into principal components. In: Journal of Educational Psychology 24 (1933), S. 417–441; 498–520

[Jacobs u. a. 1991] Jacobs, R.A. ; Jordan, M.I. ; Nowlan, S.J. ; Hinton, G.E.: Adaptive mixtures of local experts. In: Neural Computation Bd. 3. 1991, S. 79–89

[Minsky und Papert 1969] Minsky, M.L. ; Papert, S.A.: Perceptrons. MIT Press, 1969

[Neyman und Pearson 1933] Neyman, J. ; Pearson, E.S.: On the problem of the most efficient test of statistical hypothesis. In: Phil. Frans. Royal Soc. London, Ser. A 231 (1933), S. 289–337

[Pearson 1901] Pearson, K.: On Lines and Planes of Closest Fit to Points in Space. 2 (1901), S. 559–572

[Press u. a. 1992] Press, W.H. ; Flannery, B.P. ; Teukolsky, S.A. ; Vetterling, W.T.: Numerical Recipes in C - The Art of Scientific Computing. Bd. 2. Cambridge University Press, 1992

[Principe u. a. 2000] Principe, J.C. ; Euliano, N.R. ; Lefebvre, W.C.: Neural and Adaptive Systems. John Wiley & Sons, 2000

[Robbins und Monro 1951] Robbins, H. ; Monro, S.: A Stochastic Approximation Method. Bd. 22. S. 400–407, 1951

[Rojas 1996] Rojas, Raul: Theorie der neuronalen Netze, Eine systematische Einführung. Springer, Berlin, 1996

[Rosenblatt 1958] Rosenblatt, F.: The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. In: Psychological Review 65 (1958), S. 386–408

[Rumelhart u. a. 1986] Rumelhart, D.E. ; Hinton, G.E. ; Williams, R.J.: Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Bd. 1 & 2. MIT Press Cambridge, 1986

[Schwarz 1997] Schwarz, H.R.: Numerische Mathematik. B.G. Teubner Stuttgart, 1997

[Stoer 2005] Stoer, Josef: Numerische Mathematik 1. 9. Springer, 2005

[Tikhonov und Arsenin 1977] Tikhonov, A.N. ; Arsenin, V.Y.: Solutions of ill-posed Problems. Winston Wiley, 1977

[Werbos 1974] Werbos, P. J.: Beyond Regression, Harvard University, Dissertation, 1974

[Widrow 1962] Widrow, B.: Generalization and information Storage in Networks of Adaline Neurons. In: Yovits, M.C. (Hrsg.) ; Jacobi, G.T. (Hrsg.) ; Goldstein, G.D. (Hrsg.): Self-Organizing Systems. Spartan Books, 1962

[Widrow und Hoff 1960] Widrow, B. ; Hoff, M.E.: Adaptive Switching Circuits. In: IRE WESCON Convention Record (1960), S. 96–104

[Widrow und Stearns 1985] Widrow, B. ; Stearns, S.: Adaptive Signal Processing. Prentice Hall, 1985

[Widrow und Winter 1988] Widrow, B. ; Winter, R.: Neural Nets for Adaptive Filtering and Adaptive Pattern Recognition. In: IEEE Computer Magazine 21 (1988), S. 25–39

Schlagwortverzeichnis