Script zur Vorlesung :

Neuronale Netze

Vorlesung SS 02

HD Dr. Jens Timmer

July 7, 2003

Contents

1 Einleitung

The brain, my second favorite organ.
Woody Allen

Org-Krams:

- Übung ganz wichtig! Wer kann nicht hacken ?
- Accounts besorgen
- Numerik ? Recipes
- Script auf homepage. Ist eine Baustelle.
- Fragt bei Unklarheiten !

Literatur:

- J. Hertz, A. Krogh, R.G. Palmer: Introduction to the theory of neural computation [22]
- P. Dayan, L.F. Abbott: Theoretical neuroscience : computational and mathematical modeling of neural systems [12]
- R. Rojas: Neural Networks: A systematic introduction [56]
- R. Brause: Neuronale Netze [4]
- B.D. Ripley: Pattern Recognition and Neural Networks [53]
- Speziell Back-Propagation: [58]
- Speziell Kohonen [35, 54]
- Braun, Heinrich Praktikum neuronale Netze [3]
- Patterson [48]

Zwei Richtungen in NN:

- Verstehen des Gehirns

- Neue Rechnerarchitekturen bauen, "neural computation"

Ein paar Eigenschaften neuronaler Informationsverarbeitung :

- Robust und fehlertolerant. Nervenzellen sterben jeden Tag, System bricht nicht zusammen
- Flexibel. "Lernt" in neuer Umgebung, keine explizite Umprogrammierung
- Kann unscharfe Information verarbeiten
- Hochparallel, geringe Taktrate, $O(ms)$
- geringer Energieverbrauch
- Nur für Numerik ist Rechner besser

Kontroverse :

- "Neuronale Netze bilden ihre Begriffe selber"
- "Neural nets are an inefficient implementation of well-known algorithms" (NAG)

Begriffe:

- Netze:
  - Backpropagation-Netz, Multi-Layer-Feedforward
  - Recurrent networks
  - Hopfield-Netz
  - Kohonen-Netz
  - Support Vektor Maschinen
- "Lernregeln"
  - Supervised
  - Unsupervised
  - Competetive

Sonstiges:

- Self-organisation
- "Auswendiglernen"
- "Verallgemeinern"
- ...

Vom Algorithmus zum Verständnis:

- Häufig:
  "Wie macht man es "
- Ziel:
  "Was macht man eigentlich ?"
- parallel zu "geratene Bewegungsgleichung" vs. "Hamiltonian"

2 Ein bißchen Neurobiologie

Neurone

- FOLIE

  Zellkörper (Soma)
- Dendritenbaum (Input)
- Axon (Output)
- Ruhepotential -70 bis $-80mV$
- Unterschreitung von kritischem Potential (Depolarisierung): Neuron "feuert"

  Beschreibung durch Hodgkin-Huxley-Gleichungen [25, 26]
- Kopplung über Synapsen: One neuron, one neurotransmitter: i.e. entweder exitatorisch, depolarisiernd (senkt -Potential) oder inhibitorisch, hyperpolarisiernd (erhöht -Potential)

  Synapsen Sinn:

  - Entkopplung der Neurone (á la Opto-Koppler)
  - Möglichkeit auf viele Synapsen simultan einzuwirken (Alkohol)
- Elektrische Kopplung über Gap-Junctions

Neuronentypen:

- Pyramidenzellen, excitatorisch, auch langreichweitige Axone
- Glatte Sternzellen, inmhibitorisch, kurzreichweitig
- Dornige Sternzellen, excitatorisch, kurzreichweitig

Menschlicher Cortex:

- Dicke des Cortex: 2-5 mm
- 10^{10} Neurone
- 10^4 Kopplungen pro Neuron
- Neurone pro mm^3: 10^5
- Dendriten pro mm^3: 400 m
- Axone pro mm^3: 3000 m
- Nervenleitgeschwindigkeit 3-20 m/s
- Funktionell stark strukturiert (WWI)
  - hinten (occipetal) Sehzentrum (Sterne sehen)
  - seitlich (temporal) Hörzentrum
  - oben (central) Motorik
  - frontal : Höheres Denken
  - Ganz tief unten: Emotionales

Anmathematisierung
Leaky Integrate-and-fire Mechanismus (modulo Vorzeichen):

$$Pot(t) = \int_{t_0}^{t} dt'\, e^{-(t-t')/\tau} input(t-t'), \quad \tau: \text{ Leak-Parameter}$$

Fire, if

$$Pot(t) > S, \text{ danach } Pot(t) = 0. \quad t_0 = t$$

"Leaky" erklären.

Gilt für die Feuerwahrscheinlichkeit:

$$Prob(\text{Feuern in } (t, t + \Delta t)) = \rho \, \Delta t$$

so nennt man den Prozeß Poisson-Prozess mit Intensität ρ.

ZEICHUNGEN Integrate and fire, τ groß, τ klein

Zwei Fälle:

- τ groß, Ratenkodierung, zeitliche Summation, Anatomische Verbindung
- τ klein, Raumzeitliche Kodierung, räumliche Summation, Koinzidenzdetektor, Funktionelle Verbindung

3 The early days

um 1890 Pavlov
Assoziativen Lernen: Klingelt man immer mit der Glocke, wenn es Essen gibt, produziert irgendwann alleiniges Klingeln Speichelfluß

um 1900 Cayal
Färbetechniken für Gehirnschnitte

FOLIEN

McCulloch & Pitts, 1943 [42, 43]
"A logical calculus of the ideas immanent in nervous activity"
"How we know universals: The Perception of Auditory and visual forms"
Schön diskutiert in [9]

Betrachte simples Neuron:

$$y = \Theta(ax_1 + bx_2 - S), \quad y, x_1, x_2 \in \{0, 1\} \tag{1}$$

S wie Schwellwert, ergibt mit

- $a = b = 1$, $S = 1.5$: logisches UND
- $a = b = 1$, $S = 0.5$: logisches OR

Idee McCulloch & Pitts: Unser logisches Denken ist 1:1 auf neuronale Aktivität abuzbilden.
"Starke Vereinfachung" (grottenfalsch), aber (vom Guten des Schlechten :-) triggerte Entwicklung.
Bleibender Verdienst: Formalisierung der Neurone á la Gl. (1)

Hebb, 1949 [46]
"The organization of behavior: A neuropsychological theory"
Erste Lernregel Idee
Hebb'sches Lernen:
Ist prä- und postsynaptisches Neuron gleichzeitig aktiv, wird Synapse verstärkt.

Rosenblatt, 1959 [57] "Principles of Neurodynamics: Perceptrons and the theory of brain mechanism"
Erste systematische (auch numerische) Untersuchung

- Größte Liebe zum Detail
- Maximales Unverständnis in der Sache :-)

Ab hier ein Bogen bis Backpropagation, am Ende trivial, Verwirrung soll transportiert werden

Drei Schichten Perceptron ("einfaches Perceptron")

- S wie sensory, denke an Retina
- A wie association, Feature Detection
- R wie response, recognition

Neurone:
Einfache lineare Schwellwertüberschreitungsfeuerer á la McCulloch & Pitts

$$R_i \text{ feuert, wenn } u_i = a_{ij} S_j > \Theta$$

Mittlere A-Schicht

- sieht Ausschnitt der S-Schicht

  FOLIE 3 layer Perceptron

- Hat festverdrahtete Kopplungen von S-Schicht
- feuert bei bestimmtem Eingangsmuster
- Sinn: Spezifität erhöhen

  Zeichnung 0 vs. 8

  FOLIEN aus Rosenblatt

- Ist biologisch gefunden wurden [30], Nobelpreis

Arbeitspferd: Aschenputtel-Problem: Menge von Mustern richtig in 2 Klassen zu teilen, im einfachsten Falle:
ZEICHNUNG 2 Flatschen
Start: Zufälliges a_{ij}
Supervised learning:

- Externer "Lehrer" sagt, ob R-Neuron richtig reagiert hat
- Lernsystem: Wie werden Kopplungen geändert
  - positive Verstärkung: Kopplung $a_{ij} \rightarrow a_{ij} + \delta$ mit $\text{sign}(\delta) = \text{sign}(u_i)$
  - negative Verstärkung: pos. Verst. mit $\text{sign}(\delta) = -\text{sign}(u_i)$
  - α Regel: $a_{ij} \rightarrow a_{ij} + \delta$, wenn R_i aktiv α-Perceptrons
  - γ Regel: α Regel mit Renormierung, so daß Summe über alle $a_{i.}$ konstant bleiben γ-Perceptrons
- Perceptron Trainingsprozedur: Abhängig wovon werden Kopplungen geändert

- R-System: Nur abhängig vom response ("spontanes Lernsystem", unsupervised), Response bewirkt positives δ
- S-System: Nur abhängig vom Stimulus und einer Vorklassifikation: sign(δ) = f(Klasse)
- E-System: Error correcting.
  * Klassifikation richtig: $\delta = 0$
  * Klassifikation falsch: $\text{sign}(\delta) = \text{sign}(R_{i,soll} - R_{i,ist})$

Damit nicht genug: Andere Perceptrons:

- Vierschichtige mit zwei A Schichten

  Nur Kopplungen von 2. A-Schicht nach R variable, da unklar, wie A-A Kopplungen zu lernen sind: "Credit Assignment Problem", Lösung siehe Backpropagation, Kap. 4. Gilt natürlich auch für die einzelne A-Schicht

- Rückgekoppelte Perceptrons: Kopplungen zwischen A-Neuronen

Vielfalt der Kombinationen: Spielerischer Ansatz, Konzept unklar

Analytik und Simulationen:

- $\alpha-$, $\gamma-$Perceptron, einfach, vierschichtig, rückgekoppelt, mit S- oder E-System trainiert: Funktionieren
- R-System instabil, siehe Kohonen-Netz Kap. 5 für erfolgreiches unsupervised learning
- Theorem I: Für jedes binäre Klassifikationsproblem gibt es ein einfaches Perceptron, das den Job tut.

  Beweis: Trivial.

- Theorem II: Gegeben ein α Perceptron für eine binäre Klassifikationsaufgabe: Wenn eine Lösung existiert, wird sie mit der E-System Prozedur in endlicher Zeit gefunden.

  Beweis:

  Nicht inspirierend, inspirirender Beweis siehe unten

Problem einfaches Perceptron:
Es kann nicht generalisieren: Verschieb sich das Muster, wird es nicht mehr erkannt.
Geht mit vierschichtigem Perceptron, aber Komplezität explodiert: Ein A_1 Neuron für jede Position.

Wahrnehmungstheorie hinter dem Perceptron

- Atomistisch: Komplexe Muster setzen sich additiv zusammen
- Großmutter-Zelle: Am Ende pro komplexem Stimulus ein R-Neuron

Widrow & Hoff, 1960 [73]
ADALINE (ADAptive LINear Element)
Erster Schritt Richtung Back-Propagation

Minsky & Papert, 1969 [45]
"Perceptrons" Killerbuch, beendete Perceptron-Forschung
Offene Probleme Perceptron:

- Systematische Ableitung von Lernregeln
- Komplexität: Können die A-Neurone begrenztes "Sichtfeld" haben und trotzdem jedes Problem lösen ?
- Wie verhalten sich die Koeffizienten bei Skalierung des Problems ?

Konvergenzbeweis α-Perceptron
Motivation:

- Seien die zu trennenden Muster F^+ und F^- mit Elementen X^+ und X^-
- Das Problem sei lösbar.
- Ziel: $R(X^+) = 1$, $R(X^-) = 0$
- Wähle beliebige Kopplungen a_i
- Ist die Zuordnung richtig, ist man fertig

- Idee: Gilt für ein X^+: $R(X^+) = 0$, so können die A_i mit $A_i(X^+) = 0$ daran nicht schuld sein, sondern Kopplungen der A_i mit $A_i(X^+) = 1$ waren zu kein $\Longrightarrow$ Addiere $A_i(X^+)$ zu a_i.
- Für X^- aus F^- entsprechend $a_i \to a_i - A_i(X^-)$
- Für Klarheit:
  - $A(X^-) \to -A(X^-)$
- Damit: Aufgabe: Suche $\vec{a}$, so daß $\forall X$ gilt: $\vec{a}\vec{A}^T > 0$

Theorem:
Wähle jeweils $|\vec{A}(X_i)| = 1$
Gibt es einen Einheitsvektor $\vec{a}^*$ (eine Lösung) mit

$$\vec{a}^*\vec{A}^T(X_i) > \delta, \quad \forall X_i$$

dann wird `ADDIERE` des folgenden Algorithmus nur endlich häufig durchlaufen.

- `START`: Setze $\vec{a}_0 = \vec{A}^T(X_i)$ für beliebiges X_i
- `TEST` : Wähle zufällig X_j
  - Falls $\vec{a}_t\vec{A}^T(X_j) > 0$ go to `TEST`
  - Sonst: go to `ADDIERE`
- `ADDIERE`: Ersetze $\vec{a}_t \to \vec{a}_{t+1} + \vec{A}(X_j)$

Beweis:

- Sei

$$G(a_t) = \frac{\vec{a}^*\vec{a}_t}{|\vec{a}_t|}$$

  Da $|\vec{a}^*| = 1$ ist $G(a_t) = \cos(\vec{a}^*, \vec{a}_t) \leq 1$
- Betrachte $G(a_t)$ bei wiederholtem Durchlauf von `ADDIERE`

- Zähler:
$$\vec{a}^* \vec{a}_{t+1} = \vec{a}^* (\vec{a}_t + A(X_j)) = \vec{a}^* \vec{a}_t + \vec{a}^* A(X_j)$$

$$\vec{a}^* \vec{a}_{t+1} > \vec{a}^* \vec{a}_t + \delta$$

  N mal `ADDIERE`: $\vec{a}^* \vec{a}_{t+1}$ wächst i.w. linear:

$$\vec{a}^* \vec{a}_{t+1} > N\delta$$

- Nenner:
  $a_t A(X_j) < 0$, sonst nicht in `ADDIERE`

$$\begin{aligned} |a_{t+1}|^2 &= (a_t + A(X_j))(a_t + A(X_j))^T && (2) \\ &= |a_t|^2 + 2a_t A + |A|^2 && (3) \\ &< |a_t|^2 + 1 && (4) \end{aligned}$$

  N mal `ADDIERE`: $|a_{t+1}|^2$ wächst i.w. langsamer als linear

$$|a_{t+1}| < \sqrt{N}$$

- Folgt:
$$1 \geq G(a_N) = \frac{\vec{a}^* \vec{a}_t}{|\vec{a}_t|} > \frac{N\delta}{\sqrt{N}}$$

  und
$$N < 1/\delta^2$$

  `ADDIERE` wird nur endlich häufig durchlaufen.

Comments:

- Lernregel ebenso handgestirckt wie Rosenblatt's Lernregel
- Aber: Optimierungsstrategie:
  - Maximiere $G(a)$

– Man weiß, wann ein Update: $a\,A < 0$

– Man weiß, wie: $a \to a + A$

Komplexitätsbetrachtungen:
Ordnung eines Perceptrons: maximale Anzahl von S-Neuronen, zu denen A-Neurone Verbindung haben

Betrachte Paritäts-Problem:
Besteht Muster aus grader oder ungrader Anzahl von "1"en ?

Einfachste Version: XOR-Problem:

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

Versuche Lösung mit Perceptron 1. Ordnung

- Mehr als 2 A-Neurone machen keine Sinn
- Wg. Skalierbarkeit $A_i = S_i$
- Gibt es Kopplungen a_i ?

$$R(0,0) = 0 \implies 0 < \Theta \tag{5}$$
$$R(0,1) = 1 \implies a_2 > \Theta \tag{6}$$
$$R(1,0) = 1 \implies a_1 > \Theta \tag{7}$$
$$R(0,0) = 0 \implies a_1 + a_2 < \Theta \tag{8}$$

Folgt $a_1 + a_2 > 2\Theta$, $\Theta < 2\Theta$, $\Theta < 0$, i.W. zu oben.

Ergo: XOR hat Ordnung 2.

ZEICHNUNG: Graphische Veranschaulichung für OR and XOR mit 1. Ordnungs Perceptrons

Erläutere: lineare Separabilität (auf Basis der A-Schicht)

Theorem:
Paritätsproblem hat immer maximale Ordnung

ZEICHNUNG Lösung für XOR-NN
ZEICHNUNG: Graphische Veranschaulichung für XOR mit 2. Ordnungs Perceptrons (auf Basis der A-Schicht) lineare Separabilität
ZEICHNUNG : Lösung des Paritätsproblems-NN

Analoge Argument für Zusammenhangs-Problem: maximale Ordnung

Komplexitätsverhalten
Theorem:
Ist ein Problem mit einem Perceptron endlicher Ordnung lösbar, so wächst das Verhältnis des größten zum kleinsten Koeffizienten stark an [45].

Zusammengefaßt:

- Maximale Ordnung des Pariatäts (XOR)-Problems
- Anwachsen der Koeffizienten(verhältnisse) bei Skalierung des Problems

... killte das Perceptron (schlagartig).

Folgejahre: **KI als Alternative**:

- Symbol-Verarbeitung
- Ableiten von if-then Beziehungen, Expertensysteme

Hielt auch nicht, was es versprach.

Werbos, 1974 [72]
Revival von NN, führt zu Backpropagation

4 Back-Propagation

4.1 Neuro-Sicht

Rumelhart & McLelland, 1986 [58]

- Hauptproblem der bisherigen Versuche in Sachen Lernregel: Unstetigkeit des Neuronenverhaltens
- Wähle stattdessen sigmoide Funktionen:

$$\begin{aligned} \lim_{x \to -\infty} &= 0 \text{ oder } a \\ \frac{d}{dx} f(.) &> 0 \\ \lim_{x \to \infty} &= 1 \text{ oder } b \end{aligned}$$

ZEICHNUNG sigmoide Funktionen

z.B.:

$$f(x) = \arctan(x), \; f(x) = \frac{1}{1+e^{-x}} \text{ oder } f(x) = \tanh(x)$$

Ergibt:

$$y_k(x) = f\left(\sum_{i=1}^{L} w_{ki}^{(2)} f\left(\sum_{j=1}^{M} w_{ij}^{(1)} x_j\right)\right)$$

ZEICHNUNG NN mit Eingabe- hidden und Ausgabe Schicht

Sei $L_k(x)$ die "Lehrervorgabe" für die Zuordnungen der Muster x, so misst

$$E = \frac{1}{2} \sum_x \sum_k (y_k(x) - L_k(x))^2$$

Abweichungen.

Lernregel soll E minimieren:

$$w_{ij} \to w_{ij} - \gamma \sum_x \frac{\partial E}{\partial w_{ij}}, \quad \gamma: \text{ Learning rate}$$

Betrachte erstmal einzelne Muster x

Notation:

$$z_k = \sum_i w_{ki}^{(2)} x_i^{(2)}$$

$$\frac{\partial y_k}{\partial z_k} = \frac{\partial f(z_k)}{\partial z_k} = f'(z_k)$$

$$\delta_k := -\frac{\partial E_x}{\partial z_k} = -\frac{\partial E_x}{\partial y_k}\frac{\partial y_k}{\partial z_k} = -\frac{\partial E_x}{\partial y_k} f'(z_k)$$

Zweite Schicht:
Somit folgt für $w_{ki}^{(2)}$ mit

$$x_i^{(2)} = f\left(\sum_{j=1}^{M} w_{ij}^{(1)} x_j\right)$$

und

$$\frac{\partial z_k}{\partial w_{ki}^{(2)}} = x_i^{(2)}$$

finally:

$$\Delta w_{ki}^{(2)} = \gamma \delta_k^{(2)} x_i^{(2)} \qquad \text{"delta - Regel"}$$

Da:

$$\frac{\partial E_x}{\partial y_k} = y_k(x) - L_k(x), \qquad \text{gilt: } \delta_k^{(2)} = -(y_k(x) - L_k(x)) f'(z_k)$$

Erste Schicht:
Sei

$$y_i^{(1)} = f\left(\sum_{j=1}^{M} w_{ij}^{(1)} x_j\right) (= x_i^{(2)})$$

dann

$$\frac{\partial E_x}{\partial y_i^{(1)}} = \sum_k \frac{\partial E_x}{\partial z_k}\frac{\partial z_k}{\partial y_i^{(1)}} = -\sum_i \delta_k^{(2)} \frac{\partial z_k}{\partial y_i^{(1)}}$$

und

$$\frac{\partial z_k}{\partial y_i^{(1)}} = w_{ki}^{(2)}$$

Insgesamt:

$$\delta_i^{(1)} = \frac{\partial E_x}{\partial y_i^{(1)}} \frac{\partial y_i^{(1)}}{\partial x_i^{(1)}} = -\left(\sum_k \delta_k^{(2)} w_{ki}^{(2)} \right) f'(x_j)$$

δ wird backpropagated

FOLIE zum backpropagated

und

$$\Delta w_{ij}^{(1)} = \gamma \delta_i^{(1)} x_j^{(1)}$$

Bei beliebig vielen Schichten (Erinnere: Mehrschichtige Perceptrons, Lernregel für A_1- Schicht: "Credit Assignment Problem"):

$$\delta_i^{(n-1)} = \left(\sum_k \delta_k^{(n)} w_{ij}^{(n)} \right) f'(z_j^{(n-1)})$$

und

$$\Delta w_{ij}^{(n-1)} = \gamma \delta_i^{(n-1)} x_j^{(n-1)}$$

Aufschrei ??
Backpropagation = Kettenregel
Kein anderer Begriff aus Analysis I hat solch eine Karriere gemacht !

Betrachte im folgenden:

$$f(x) = \frac{1}{1 + e^{-x}}$$

Wie man leicht sieht:

$$f'(x) = (1 - f(x)) f(x)$$

Dann:

$$\Delta w_{ki}^{(2)} = \gamma(y_k - L_k)(1 - f(z_k))f(z_k)x_i^{(2)} \quad (9)$$

$$\Delta w_{ij}^{(1)} = \gamma\left(\sum_k \delta_k^{(2)} w_{ki}^{(2)}\right)(1 - f(z_k^{(1)}))f(z_k^{(1)})x_j^{(1)} \quad (10)$$

Am Ende, nehme

$$y(x)\begin{cases} > 0.5 \text{ setze auf } 1 \\ < 0.5 \text{ setze auf } 0 \end{cases}$$

Zusammenfassung:

- Back-Propagation formuliert Klassifikationsproblem als Optimierungsproblem
- Gradientenabstieg im Zielfunktional definiert Lernregel

Zu klären:

- Wieviele Schichten braucht man ?
- Wieviele hidden units braucht man ?

1. Versuch:
13. Hilbert Problem:

- Sei I^n der n-dimensionale Einheitskubus
- Abbildung $\phi : I^n \to R^m$
- Kann ϕ durch Überlagerung von endlichen Summen von Funktionen der einzelnen Variablen dargestellt werden ?

Antwort: Ja [36, 65, 20], siehe auch A. Heinz
Und zwar mit:

$$z_k = \sum_{j=1}^{n} \lambda^k f(x_j + \epsilon k) + k$$

mit $\lambda \in R$, $f(.)$ sigmoid, ϵ rational
und

$$y_i = \sum_{k=1}^{2n+1} g_i(z_k)$$

I.e.:

- 2 nichtlineare Schichten
- $2n + 1$ hidden units

ABER:
g_i hängt von unbekannterweise ϕ ab.

2. Versuch:
Approximationstheorie [29, 11]:

- Theorem: Sei $f(.)$ sigmoid, dann kann

$$y_k(x) = \sum_{i=1}^{L} w_{ki}^{(2)} f\left(\sum_{j=1}^{M} w_{ij}^{(1)} x_j + \theta_i\right) + \phi_k$$

  jede kontinuierliche Funktion Abbildung $\phi : I^n \to R^m$ beliebig genau approximieren

- Beweis:

  Rückführung auf Approximationseigenschaften der trigonometrischen Funktionen

- Ab jetzt: Binäres Klassifikations back-Propagationsnetz: 1 nichtlineare hidden + lineare Ausgabe dafür mit offsets (true units)

$$y(x) = \sum_{i=1}^{L} w_i f\left(\sum_{j=1}^{M} w_{ij} x_j + \theta_i\right) + \phi$$

Probleme:

- Wieviele hidden units ?
  - Zuwenig: Klassifikation klappt nicht
  - Zuviele: NN "lernt auswendig", kann nicht "verallgemeinern"
  - Richtig viele: NN lernt und verallgemeinert
  - Merke: Es gibt nicht "DAS neuronale Netz"

- Lokale Optima im Optimierungsprozeß
- Beobachtung: Langsame Konvergenz
  - Momentum term Erweitere Lernregel

$$\Delta w_{ij}(t) = \gamma \delta_k x_i + \alpha \Delta w_{ij}(t-1)$$

  - Erinnere: $f'(x) = (1 - f(x))f(x)$ Wird klein, wenn f bei 0 oder 1. Addiere 0.1 zu dem Faktor
  - ...

Lernmodi:

- Batch-update

  Berechne einen Update nachdem alle Muster präsentiert wurden sind
- Serieller Update

  Berechne einen Update nach jeder Musterpräsentation

  Erfahrung: kann aus lokalen Minima herausführen, Beweis, siehe: [51]

  Erfahrung: Man muß Lernrate mit der Zeit kleiner machen.

Da man nicht weiss, wieviele hidden units man braucht:
Beginne mit vielen und verwende:

- Weight decay :

  Nach jedem Update zusätzlich

$$w_{ij}^{new}(t) = (1 - \epsilon)\, w_{ij}^{old}(t)$$

  so daß nur immer wieder aufgewertete Gewichte im Geschäft bleiben

  Entspricht Änderung des Zielfunktionals

$$E_{neu} = E_{alt} + \gamma \frac{1}{2} \sum_{ij} w_{ij}^2$$

  mit steepest descent

$$\Delta w_{ij} = -\eta \partial E_{neu} / \partial w_{ij}$$

mit $\epsilon = \eta\gamma$

Bestraft zu große Gewichte

- pruning : Setze Gewichte, die im Verlauf der Optimierung stark schwanken auf Null (scheinen ja nicht genau bestimmt zu sein)
- early stopping :
  - Teile Datensatz in zwei Teile
  - Beginne mit kleinen Anfangswerten für Gewichte
  - Optimiere Gewichte an Teil I und betrachte Fehler
  - Evaluiere Fehler gleichzeitig an Teil II
  - Beende Optimierung wenn beide Fehler gleich
- Siehe auch "Optimal Brain Damage" [39]

Nach der Klassifikation :
Verwendung von Backpropagationnetzen zur allgemeinen Funktionsdarstellung :

$$y = f(x) + \epsilon$$

4.2 Statistik-Numerik-Sicht

-1950
Funktionsdarstellungssicht
Parametrische Modelle
Notwendige Statistikkenntnisse:

- Zufallsvariable: Etwas, das eine Verteilung hat.
- Gaußverteilung N(0,1)
- χ^2 Verteilung: Summe quadrierter unabhängiger N(0,1)

Lineare Regression:
Betrachte Modell:

$$y_i = ax_i + \epsilon_i, \quad \epsilon_i \in N(0, \sigma_i^2)$$

$$p(y_i, x_i, a) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\left(\frac{y_i - a\,x_i}{\sigma_i}\right)^2}$$

Gemessene Daten x_i, y_i. Schätzer für a ?
Betrachte Likelihood:

$$L(y_1, \ldots, y_N, x_1, \ldots, x_N | a) \propto \prod_{i=1}^{N} p(y_i, x_i | a)$$

Idee: Wähle Parameter so, dass die Daten maximal wahrscheinlich werden.

$$\frac{d}{da} L(a) = 0$$

Im Falle Gaußverteilter Fehler:

$$L(y_1, \ldots, y_N | a) \propto \prod_{i=1}^{N} \left\{ \exp\left[-\frac{1}{2} \left(\frac{y_i - ax_i}{\sigma_i} \right)^2 \right] \Delta y \right\} \quad (\dagger)$$

Maximierung von (†) entspricht Minimierung des negativen Logarithmus von (†):

$$\mathcal{L} = \left[\frac{1}{2} \sum_{i=1}^{N} \frac{[y_i - ax_i]^2}{2\sigma_i^2} \right] - N \log \Delta y$$

Relevanter Teil :

$$\chi^2 = \frac{1}{2} \sum_{i=1}^{N} \frac{[y_i - ax_i]^2}{\sigma_i^2}$$

Alles geht analytisch:

$$0 = \frac{\partial \chi^2}{\partial a} = \sum_{i=1}^{N} \frac{x_i (y_i - \hat{a} x_i)}{\sigma_i^2} \quad (\dagger)$$

Mit

$$S_{xx} = \sum_{i=1}^{N} \frac{x_i^2}{\sigma_i^2}, \quad S_{xy} = \sum_{i=1}^{N} \frac{x_i y_i}{\sigma_i^2}$$

folgt aus (†)

$$\hat{a} S_{xx} = S_{xy} \tag{11}$$

$$\hat{a} = \frac{S_{xy}}{S_{xx}} \tag{12}$$

Fehlerfortpflanzung:

$$\sigma_a^2 = \sum_{i=1}^{N} \left(\frac{\partial a}{\partial y_i} \right)^2 \sigma_i^2$$

$$\frac{\partial a}{\partial y_i} = \frac{\frac{x_i}{\sigma_i^2}}{S_{xx}}$$

Eingesetzt ergibt:

$$\sigma_a^2 = 1/S_{xx}$$

$\hat{a}$ ist Zufallsvariable, Kurz statistischen Test erläutern.

Ermöglicht Test auf $a = 0$: Ist 0 in $[\hat{a} - 1.96\sigma_a, \hat{a} + 1.96\sigma_a]$?

Nichtlineare Regression:
Einfachste Fortsetzung von oben:

$$y(x) = a_1 + a_2 x + a_3 x^2 + a_4 x^3 + \ldots + a_M x^{M-1}$$

oder genereller:

$$y(x) = \sum_{k=1}^{M} a_k X_k(x)$$

$X_k(x)$ Basisfunktionen, z.B. $\sin(\omega_k x)$ (ω_k kein freier Parameter)
Modell ist

- linear in den Parametern,
- hat aber nichtlineare Basisfunktionen.

Nun:

$$\chi^2 = \sum_{i=1}^{N} \left[\frac{y_i - \sum_{k=1}^{M} a_k X_k(x_i)}{\sigma_i} \right]^2$$

Definiere:

$$A_{ij} = \frac{X_j(x_i)}{\sigma_i}, \quad b_i = \frac{y_i}{\sigma_i}$$

A heißt Design-Matrix.

Minimumsbedingung für χ^2 :

$$0 = \sum_{i=1}^{N} \frac{1}{\sigma_i^2} \left[y_i - \sum_{j=1}^{M} a_j X_j(x_i) \right] X_k(x_i) \quad (\dagger)$$

Mit

$$\alpha_{kj} = \sum_{i=1}^{N} \frac{X_j(x_i) X_k(x_i)}{\sigma_i^2}, \text{ oder } \alpha = A^T A$$

α ist $(M \times M)$ Matrix
und

$$\beta_k = \sum_{i=1}^{N} \frac{y_i X_k(x_i)}{\sigma_i^2} \text{ oder } \beta = A^T b$$

und vertauschen der Summen folgt:

$$\sum_{j=1}^{M} \alpha_{kj} a_j = \beta_k \quad (\ddagger)$$

Die Gleichungen (†), resp. (‡) heissen Normalen-Gleichungen
Schätzer:

$$\hat{\vec{a}} = A^{-1} \vec{\beta}$$

Fehler für die Parameter:

Definiere:

$$C = \alpha^{-1}$$

Betrachte:

$$a_j = \sum_{k=1}^{M} \alpha_{jk}^{-1} \beta_k = \sum_{k=1}^{M} C_{jk} \left[\sum_{i=1}^{N} \frac{y_i X_k(x_i)}{\sigma_i^2} \right]$$

Erinnere

$$\sigma^2(a_j) = \sum_{i=1}^{N} \sigma_i^2 \left(\frac{\partial a_j}{\partial y_i} \right)^2$$

α, und damit C, ist unabhängig von y_i:

$$\frac{\partial a_j}{\partial y_i} = \sum_{k=1}^{M} C_{jk} X_k(x_i) / \sigma_i^2$$

Also:

$$\sigma^2(a_j) = \sum_{k=1}^{M} \sum_{l=1}^{M} C_{jk} C_{jl} \left(\sum_{i=1}^{N} \frac{X_k(x_i) X_l(x_i)}{\sigma_i^2} \right)$$

Der Term (.) ist aber das grade $\alpha = C^{-1}$, somit:

$$\sigma^2(a_j) = C_{jj}$$

Wählt man $X_k(.)$ und x_i geschickt, ist C_{ij} diagonal und man kann alle Koeffizienten separat testen.

Auswahl notwendiger Basisfunktionen: Modellselektion

Eventuell $F-$Test erwähnen

Logistische Regression
Oft (speziell in Medizinstatistik)

- binäre Zielvariable (Heilung, Tod) (0,1)
- binäre und/ oder kontinuierliche Einflussgrößen $X_1, ...$

Sei p Wahrscheinlichkeit für Ereignis

- $p/(1-p)$ ist relative Chance $\in (0, \infty)$
- $\log(p(1-p))$ ist $\in (-\infty, \infty)$
- Das logistische Regressionsmodell:

$$log(p(1-p)) = \sum_i a_i X_i$$

oder

$$p = \frac{e^{\sum_i a_i X_i}}{1 + e^{\sum_i a_i X_i}}$$

ZEICHNUNG logistische Funktion

Nichtlineare Modelle
Ist Modell nichtlinear in den Parametern, i.e.

$$f(x, a) \neq \sum_i a_i g(x)$$

$\Longrightarrow$ i.a. keine analytische Lösung, iterative Verfahren (erinnere Gradientenabstieg in BP)

- Gradientenabstieg (lineares, 1. Ordnung-Verfahren):

$$\Delta a = -\gamma \frac{\partial f}{\partial a}$$

Probleme

  - Skala für γ unklar
  - Konvergiert i.w. nie, da $\frac{\partial f}{\partial a} \to 0$ für $a \to a_{true}$

- Quadratische, 2. Ordnung Verfahren

  Betrachte Taylorentwicklung nahe bei Minimum für allgemeines Funktional $F(a)$:

  $$F(a) = F(a_i) + (a - a_i)\nabla F(a_i) + \frac{1}{2}(a - a_i)A(a - a_i)$$

  mit A Hesse Matrix an a_i

  Soll der nächste Schritt ins Minimum gehen $\Longrightarrow \nabla F(a_{i+1}) = 0$

  also

  $$\nabla F(a_{i+1}) = \nabla F(a_i) + A(a_i)(a_{i+1} - a_i) = 0$$

  oder Newton-Schritt

  $$\Delta a = a_{i+1} - a_i = -A^{-1}\,\nabla F(a_i)$$

  MERKE (für die Freunde der ART): Hesse-Matrix legt Metrik fest.

Optimierungsstrategien

- Batch-update

  Spezielle Form des zu minimierenden Funktionals:

  $$E(a) = \frac{1}{2}\sum_i (L(x_i) - y(x_i, a))^2$$

  Levenberg-Marquardt-Verfahren [49]

  - Braucht nur 1. Ableitungen
  - nahe dem Optimum so gut wie 2. Ordnungsverfahren
  - Weit weg vom Optimum, "gutes" Gradientenverfahren

Erinnerung:
Nah am Optimum gilt in guter Näherung die quadratische Näherung, und der Newton-Schritt

$$a_{i+1} = a_i - A^{-1}\nabla E(a_i) \quad (\dagger)$$

führt direkt zum Ziel.

Gradient:

$$\frac{\partial E(a)}{\partial a_k} = -\sum_{i=1}^{N}(L(x_i) - y(x_i, a))\frac{\partial y(x_i, a)}{\partial a_k}, \quad k = 1, 2, \ldots, M$$

Hesse-Matrix:

$$\frac{\partial^2 E(a)}{\partial a_k \partial a_l} = \sum_{i=1}^{N}\left[\frac{\partial y(x_i, a)}{\partial a_k}\frac{\partial y(x_i, a)}{\partial a_l} - (L(x_i) - y(x_i, a))\frac{\partial^2 y(x_i, a)}{\partial a_k \partial a_l}\right]$$

Konvention:

$$\beta_k = -\frac{\partial E(a)}{\partial a_k}, \quad \alpha_{kl} = \frac{\partial^2 E(a)}{\partial a_k \partial a_l}$$

Damit und mit $\delta_l = (a_{i+1} - a_i)_l$ wird $(\dagger)$ zu

$$\sum_{i=1}^{M}\alpha_{kl}\delta a_l = \beta_k \quad (*)$$

Beachte : Steepest Descent lautet:

$$\delta a_l = \text{constant}\beta_l \quad (\ddagger)$$

Idee :

- Wenn der Fit gut ist, gilt für 2.Term der Hesse-Matrix

$$\sum_{i=1}^{N}\frac{1}{\sigma_i^2}(L(x_i) - y(x_i, a))\frac{\partial^2 y(x_i, a)}{\partial a_k \partial a_l} \approx 0,$$

da die Fehler $\epsilon_i = (L(x_i) - y(x_i, a))$ unkorreliert sind.
Also, definiere:

$$\alpha_{kl} = \sum_{i=1}^{N} \frac{1}{\sigma_i^2} \left[\frac{\partial y(x_i, a)}{\partial a_k} \frac{\partial y(x_i, a)}{\partial a_l} \right]$$

- * Weit weg von Minimum mag Newton-Schritt ($*$) schlecht sein.
  * Mache Gradienten-Schritt ($\ddagger$). Wie "constant" wählen ?
  * Dimension $[\beta_l]$ = Dimension $[1/a_l] \Longrightarrow$
    $1/\alpha_{ll}$ ist Skalierungskanditat.
  * Zur Sicherheit (damit's nicht zu groß wird), wähle $\lambda \gg 1$ und setze:

$$\delta a_l = \frac{1}{\lambda \alpha_{ll}} \beta_l$$

- Kombiniere diese Gleichung und den Newton-Schritt Gl. ($*$) durch

$$\alpha'_{jj} = \alpha_{jj}(1 + \lambda) \quad (13)$$
$$\alpha'_{jk} = \alpha_{jk}, \quad (j \neq k) \quad (14)$$

in Gleichung

$$\sum_{i=1}^{M} \alpha'_{kl} \delta a_l = \beta_k \quad (**)$$

Ist λ groß $\Longrightarrow$ α'_{kl} diagonal-dominant $\Longrightarrow$ kleiner Gradienten-Schritt
Geht $\lambda \to 0$, Hesse-Schritt

Procedere:

- Wähle Startschätzung für a, Berechne $\chi^2(a)$
- Wähle ein kleines λ: $\lambda = 0.001$
- $(+)$ Löse $(**)$ und berechne $\chi^2(a + \delta a)$
- Wenn $\chi^2(a + \delta a) \geq \chi^2(a)$, verwerfe δa, wähle $\lambda = 10\lambda$, go to $(+)$

- Wenn $\chi^2(a+\delta a) < \chi^2(a)$, akzeptiere δa, wähle $\lambda = 0.1\lambda$, go to (+).
- Interpretation:
  Wenn Newton-Schritt
  * gut, dann mehr davon,
  * schlecht, dann mit Gradienten-Schritt auf Sicht fahren.

Nach Konvergenz:

$$\alpha^{-1} = \left\{ \sum_{i=1}^{N} \frac{1}{\sigma_i^2} \left[\frac{\partial y(x_i, a)}{\partial a_k} \frac{\partial y(x_i, a)}{\partial a_l} \right] \right\}^{-1}$$

ist Kovarianz-Matrix der Fehler in den geschätzten Parametern.

I.d.R. nie diagonal.

- Single update

  Erfahrung :

  - Single update mit fester Lernrate konvergiert nicht.
  - Lernrate muß runtergefahren werden

  Stochastische Approximation (1951) [55, 34]

$$\Delta w = -\gamma \frac{\partial E(x)}{\partial w}$$

für ein einzelnes Muster kann als Realisierung von Zufallsvariable angesehen werden.

Gesucht:

$$w^* \text{ so daß } < \frac{\partial E(x)}{\partial w} >_x = 0$$

Theorem:

Gilt

$$\sum_{t=1}^{\infty} \gamma(t) = \infty \tag{15}$$

$$\sum_{t=1}^{\infty} \gamma(t)^2 < \infty \tag{16}$$

so konvergiert

$$w(t) = w(t-1) - \gamma(t) \frac{\partial E(x)}{\partial w}$$

gegen w^*.

Mögliche Wahl für $\gamma(t)$: $\gamma(t) = \frac{1}{t}$:

Nichtparametrik, Kernschätzer

Hat man Zusammenhang

$$y_i = f(x_i) + \epsilon_i \,,$$

so kann man f(x) nichtparametrisch schätzen durch Kernschätzer :

$$\hat{f}_h(x) = \sum_i K((x - x_i)/h)\, y_i$$

mit $K(z)$ z.B.:

$$K(z) = \begin{cases} 1/2 & \text{if } |z| < 1/2 \\ 0 & \text{sonst} \end{cases}$$

Gewichtetes Mittel, h bestimmt Einzugsbereich.

ZEICHNUNG Kerschätzer

Man kann zeigen:
Jedem Kern $K(z)$ entspricht ein lokales Polynom mit Träger [x-h/2,x+h/2]

ZEICHNUNG Kerschätzer und lokales Polynom

- sehr flexibel
- Nichtparametrisch = Sehr viele Parameter

k-nearest neighbor
Tong/Chan Buch p.136 ff
Irgendwo: Early stopping nutzt nur die Hälfte der Daten richtig zur Modellbildung

4.3 Anwendungen

- XOR

  FOLIE XOR

- Aussprache von englischem Text NETtalk [64]:
  Geschriebener Text → Phoneme

  FOLIE NETtalk

  - 80 hidden units
  - 26 Output units
  - 95 % korrekt auf Trainingsdaten
  - 78 % korrekt auf Testdaten
  - klingt o.k.

  Vergleich zu DEC-talk mit festverdrahteten linguistischen Regeln

  - Schlechtere Performance
  - Einfachere Implementation

- Spracherkennung, for review see [41]
- Worttrennung
  prop-a-ga-tion, pro-pa-gan-da, pro-pane
- U-Boot Erkennung [16, 17] Unterscheidung von

  - Felsen
  - U-Booten

  auf Grund spektraler Sonar Information

  - 100 % korrekt auf Training
  - 90 % korrekt auf Testdaten

  Vorverarbeitung (Zeit- nach Frequenzraum) sehr wichtig, das gilt allgemein

- Sekundäre Protein Struktur [50, 2]
  - DNA: 20 Buchstaben Aminosäuren, kodieren für Proteine
  - Sekundärstrukturen
    * α- helix
    * β- sheets

  Zusammenhang :

  - Input: 13 Aminosäuren
  - Output:
    * α- helix
    * β- sheets
    * Anderes

  Ergebnis: 62 % richtig, best so far: 53 %

- Backgammon [66] Schlug alle Computerprogramme, aber nicht den Menschen. Das wurde erst durch Reinforcement-Learning geschafft.
- Handgeschriebene Postleitzahlen [15]

  FOLIE ZIP codes

- Machbarkeitsstudien, keine Überlegenheitsstudien
- Aber: Probleme behandelt, an die sich Statistiker nicht ranwagten

4.4 Synopsis

- Backpropagation ist "potenzierte" logistische Regression
- "Optimal Brain Damage", early stopping, pruning:
  - Mit sehr vielen (lokalen) Basisfunktionen (=Parametern) beginnen = nichtparametrisch, flexibel
  - Mit wenigen Basisfunktionen enden = parametrisch
- Nichtlineare Regression, nichtlinear in den Parametern
  - Keine analytische Lösung des Schätzproblems
  - man muss numerisch optimieren
  - Euphemismus: Lernen

Neurosprache	konventionelle Sprache
Back Propagation	nichtlineare Regression mit sigmoidalen Basiskunktionen
Back Propagation	Kettenregel
Back Propagation	Steepest descent
lernen	schätzen
verallgemeinern	glätten
auswendig lernen	interpolieren
Neurone	Basisfunktionen
Gewichte	Parameter
pruning	Variablenselektion
# versteckte Neurons	Modellselektion
learning rate	Schrittweite

Nachtrag: Statt sigmoidaler Basisfunktionen jüngst auch Radiale Basisfunktionen in Mode

$$y = \sum_{i=1}^{L} \alpha_i \Phi\left(|\vec{x} - \vec{c}_i)|\right)$$

mit z.B.:

$$\Phi(z) = \frac{1}{1 + z^2/r^2} \text{ oder } \Phi(z) = e^{-z^2/r^2}$$

ZEICHNUNG

- Können auch jede vernünftige Funktion approximieren
- Wenn $\vec{c}_i$, r frei: Nichtlinear in Parametern
- Wenn $\vec{c}_i$, r fest: Linear in Parametern

Bemerkung zu Schwarzer Studie [62]: In Praxis werden alle Regeln der Kunst misachtet.

5 Kohonen Map

Bisher:

- Präsentation von Muster
- Vergleich: Output mit Lehrervorgabe
- Korrektur der Gewichte aus Mismatch
- Supervised Learning
- Erinnere:
  Rosenblatt in Kap. 3: Error-correcting learning

Jetzt:

- Präsentation von Muster
- Ergibt Output
- Änderung der Gewichte nur in Abhängigkeit des Outputs
- Unsupervised Learning Selbstorganisation
- Erinnere:
  Rosenblatt in Kap. 3: Response learning, war instabil

5.1 Competitive Learning

5.1.1 Neuro-Sicht

Betrachte 2 Schichten Netz mit linearer Kopplung :

$$y_i = \sum_j w_{ij} x_j$$

Nur der Gewinner

$$i^* = \text{ index max } (y_i) \text{ mit } \vec{w}_{i^*}\vec{x} \geq \vec{w}_i\vec{x} \quad \forall i$$

feuert :

$$O_i^* = 1, \quad O_i = 0, \forall i \neq i^*$$

Sind die Gewichte und Muster auf $|\vec{w}_i| = 1$ normiert, entsprechend:

$$|\vec{w}_{i^*} - \vec{x}| \leq |\vec{w}_i - \vec{x}| \ \forall i$$

Biologische Implementation durch laterale Inhibition mit gutem Feintuning

ZEICHNUNG laterale Inhibition und self excitation

Lernregel: *The winner takes it all*

- Bestimme Sieger i^* unter Ignoration aller Normierung aus

  $$|\vec{w}_{i^*} - \vec{x}| \leq |\vec{w}_i - \vec{x}| \ \forall i$$

- Update der Sieger-Gewichte
  - Version 1:

    $$\Delta w_{i^* j} = \gamma x_j$$

    Führt zu $w_{ij} \to \infty$
  - Version 2:

    $$\Delta w_{i^* j} = \gamma \left(\frac{x_j}{\sum_j x_j} - w_{i^* j} \right)$$

    sorgt für $\sum_j w_{ij} = 1$

– Version 2: Standard Competitive learning rule

$$\Delta w_{i^* j} = \gamma(x_j - w_{i^* j})$$

Zieht $\vec{w}_{i^*}$ in Richtung $\vec{x}$.
Oder auch

$$\Delta w_{ij} = \gamma O_i(x_j - w_{i^* j})$$

– Alles Spielarten des Hebb'schen Lernens

FOLIE Competitive Learning

Ergebnis:
Clusterzuordnung

Allgemeines zu Clustering : [33]
Zum Vergleich zweier Clustereien [52] und adjusted Rand-Index: [31, 44]

5.1.2 Mathematisch betrachtet

Betrachte:

$$E(w_{ij}) = \frac{1}{2}\sum_{ijk} M_i^k (x_j^k - w_{ij})^2 = \frac{1}{2}\sum_k |\vec{x}^k - \vec{w}_{i^*}|^2$$

mit Membership-Matrix

$$M_i^k = \begin{cases} 1 & \text{wenn } i = i^*(x^k) \\ 0 & \text{sonst} \end{cases}$$

Beachte: M_i^k ist iterationsabhängig.
Gradienten-Abstieg:

$$\Delta w_{ij} = -\gamma \frac{\partial E(w_{ij})}{\partial w_{ij}} = \gamma \sum_k M_i^k (x_j^k - w_{ij})$$

Entspricht

- Batch-Mode Version der Standard Competitive Learning Rule.
- Klassischem k-means Clustering [37]

Bemerkungen:

- Zyklische Repräsentation der Muster kann Zyklen in M_i^k und keiner Konvergenz führen
- Addition von Momentum-Term

  $$\Delta w_{ij}(t) \rightarrow \Delta w_{ij}(t) + \eta \Delta w_{ij}(t-1)$$

  a la Backpropagation beschleunigt Konvergenz.
- **Stochastische Approximation**

  Endgültige Konvergenz nur bei

  $$\lim_{t \rightarrow \infty} \gamma(t) = 0$$

- Eindeutige Lösung nur bei klar getrennten Clustern
- Vom zu minimierenden Funktional zur Lernregel

  Coulomb Energy network [63]

  Betrachte Fehlerfunktional:

  $$E_C(w_{ij}) = -\frac{1}{p} \sum_{ijk} Q_i^k |x_j^k - w_i j|^{-p}$$

  mit

  $$Q_i^k = 2M_i^k - 1$$

  und Lernregel

  $$\Delta w_{ij} = -\gamma \frac{\partial E_C(w_{ij})}{\partial w_i j}$$

  Ergibt

  – Überdämpfte Bewegung elektrostatischer Punktladungen

- Attraktiv auf Gewinner
- Repulsiv von allen Verlierern

Steinbruch:
Fischer'sche Diskriminanzanalyse [14, 18]
lineare Dis-Analyse (DLD)
CART hat tendency zum overfitting, überoptimistik, Korrektur a la [38].
Allgemein: [33]
Zum Vergleich zweier Clustereien [52] und adjusted Rand-Index: [31, 44]

Siehe auch Schuchard in CHAOS

Erst PC, dann Clustering [8] [74]

ROC Kurven
Sensitive und spezifische Tests

Zusammenfassung:
Unsupervised, self-organized learning :

- Lehrer stellt Fehlerfunktional auf.
- Lehrer leitet per Gradienten-Abstieg Lernregel ab
- Minimierung des Fehlerfunktionals benötigt keinen Lehrer

5.2 Kohonen Map

Auch als Self organizing map (SOM) oder Self organizing feature map (SOFM) bezeichnet

Motivation:

- Nahe beieinander liegende sensorische Inputs sind kortikal nahe beieinander repräsentiert (Topologie erhaltend).
- Sehr sensible Areale sind großflächiger repräsentiert (verzerrend)

  FOLIE Homunkulus

- Zuordnung kann nicht genetisch vorgegeben sein.
  Aber Biologie sagt, es geht mehr per Botenstoffen als mit Kohonen
- Abbildung ist flexibel

  FOLIE AFFENFINGER

  Bei Geige lernenden Menschen bestätigt

5.2.1 Neuro-Sicht

- Competitive Learning:
  Keine Nachbarschaft-Beziehung zwischen Ausgabe-Neuronen
- Kohonen-Netz [35] :
  Räumliche Organisation der Output-Neurone
  Aufgeweichte Winner-takes-it-all Lernregel
  Abstrahierte "Mexican hat" Kopplung

  ZEICHNUNG "Mexican hat"

Kohonen - Netz:
Struktur:

- Analog zu competitive learning
- Output-Neurone i.d.R. (geordnete) 1D oder 2D-Felder ($\vec{r}_i$)

Vorgehen:

- Initialisiere Gewichte
- Präsentiere Stimulus
- Bestimme Sieger i^* unter Ignoration aller Normierung aus

$$|\vec{w}_{i^*} - \vec{x}| \leq |\vec{w}_i - \vec{x}| \ \forall i$$

- Update:

$$\Delta w_{i,j} = \gamma(t)\,\Lambda(i, i^*)\,(x_j - w_{ij})$$

mit z.B.

$$\Lambda(i, i^*) = e^{-\frac{|r_i - r_{i^*}|^2}{\sigma(t)^2}}$$

oder

$$\Lambda(i, i^*) = \begin{cases} 1 & \text{wenn } |r_i - r_{i^*}|^2 < \sigma(t)^2 \\ 0 & \text{sonst} \end{cases}$$

mit

$$\sigma(t) \to_t 0 \quad (\gamma(t) \to_t 0, \text{ Stochastische Approximation})$$

oder

$$\sigma(t) \to_t \sigma_{final} \text{wenn es flexibel bleiben soll}$$

  - Anfangs: Globale Ordnung richtig machen
  - Final: Lokales Fine-tuning

5.2.2 Anwendungen

Abbildungen: Output-Gewichte in Input-space

- Simulationen
  FOLIEN
- Somatotopie

  FOLIE aus Martinetz
- Ballistik movement, erweitern
- Travelling Sales Man Problem
  - Task: finde kürzesten Reiseweg durch N Städte

- Problem: Es gibt $(N-1)!$ Möglichkeiten, e.g. für $N = 120$: $6\,10^{196}$, mehr als Teilchen im Universum
- Finde subopimale, aber sehr gute Lösung (schnell)
- Wähle Output-Struktur S^1, *somit* : Abbildung $R^2 \to S^1$

  FOLIE

- Analog: Kürzeste Verbindung auf Leiterplatten
- Dimensionsreduktion:

  Phonem-mapping

  21 Finnische Phoneme zerlegt in 15 Frequenz-Kanäle

  Abbildung $R^{15} \to R^2$

  Hier nicht Output-Gewichte in Input-space, sondern Orte der Output-Neurone bei maximaler Antwort auf Input

  FOLIE KOHONEN-BUCH

- Output-Neurone als solche nicht notwendig
- Nur Gewichte dahin
- Nur für Nachbarschaftsverhaeltnisse

5.2.3 Mathematisch betrachtet

Kohonen-Netz ordnet kontinuierlichem Input, diskreten Output zu:
Vektorquantisierung
im einfachsten (gleichverteilten) Falle:

- 1D $\to$ 1D: Analog/Digital-Wandler

  ZEICHNUNG AD Wandler

- 2D $\to$ 2D: Voronoi-Diagramme

  ZEICHNUNG Voronoi-Diagramme

Zu optimierendes Funktional

- Basierend auf Mutual Information

  Entropie H einer Dichte $p(x)$:

  $$H = -\int dx\, p(x) \log p(x)$$

  Maß für die Überraschendheit einer Verteilung (MEM erwähnen)

  Für zwei Zufallsvariablen betrachte: Mutual Information

  $$MI = -\int dx\, dy\, p(x,y) \log \frac{p(x,y)}{p(x)\, p(y)}$$

  - $p(x), p(y)$ unabhängig: $MI = 0$, gegeben x ist y maximal überraschend
  - Abhängigkeiten verringern MI

  Sei

  - $p(x)$ die Dichte der Reize in der Eingabe-Schicht
  - $p(y)$ die Dichte der Gewichte zur Ausgabe-Schicht

  Handwaving Arguments [40]:

  Kohonen i.w. Gradientenabstieg der Mutual Information[1]

  Macht Sinn

- Analog zu Competitive Learning:

  $$E(w_{ij}) = \frac{1}{2} \sum_{ijkl} M_k^l \Lambda(i,k)(x_j^l - w_{ij})^2 = \frac{1}{2} \sum_{ijl} \Lambda(i,i^*)(x_j^l - w_{ij})^2$$

  mit Membershipmatrix M_k^l von oben. Erinnere M_k^l, resp. i^* ist iterationsabhängig.

  Gradientenabstieg liefert Batch-Version des Kohonen-Algorithmus.

[1] In [40] andere Vorzeichenkonvention: Gradientenaufstieg

Asympotik in 1D [54]
Bei Konvergenz gilt:

$$\Delta w_{ij} = -\gamma \frac{\partial E(w_{ij})}{\partial w_{ij}} = 0$$

also:

$$0 = \sum_l \Lambda(i, i^*)(x_j^l - w_{ij}) \quad \forall i, j$$

Kontinuums Approximation:

- $\sum_l \to \int_x dx p(x)$
- $\Lambda(i, i^*) \to \Lambda(r - r^*)$
- $w_{ij} \to w(r)$
- $\Lambda(r - r^*)$ sei scharf gepeakt

$$0 = \int dx p(x)\, \Lambda(r - r^*)\, (x - w(r)) \quad (\dagger)$$

- Entwickle nach
  $$\epsilon = r^*(x) - r$$
  und vernachlässige höhere Terme als ϵ^2 :
- $\Lambda(.)$ symmetrisch:
  $$\Lambda(r - r^*) = \Lambda(-\epsilon) = \Lambda(\epsilon)$$
- Bei Konvergenz : $w(r^*) = w(r + \epsilon) = x$.
  Damit:
  $$x - w(r) = \epsilon w'(r) + \frac{1}{2}\epsilon^2 w''(r)$$
- Bei Konvergenz: $p(x) = p(w(r^*)) = p(w(r + \epsilon))$ und
  $$p(r + \epsilon) = p(w(r)) + \epsilon p'(w(r)) w'(r)$$

- $$dx = dw(r+\epsilon) = w'(r+\epsilon)d\epsilon = (w'(x) + \epsilon w''(x))d\epsilon$$

Alles zusammen in (†) und bis 2. Ordnung :

$$\begin{aligned} 0 &= \int \Lambda(\epsilon)(\epsilon w' + \frac{1}{2}\epsilon^2 w'')(p(w) + \epsilon p'(w)w')(w' + \epsilon w'')d\epsilon & (17) \\ &= \int \Lambda(\epsilon)(\epsilon w'^2 p(w) + \epsilon^2 w'[\frac{3}{2}w''p(w) + w'^2 p'(w)])d\epsilon & (18) \\ &= w'[\frac{3}{2}w''p(w) + w'^2 p'(w)] \int \Lambda(\epsilon)\epsilon^2 d\epsilon & (19) \end{aligned}$$

da $\int \Lambda(\epsilon)\epsilon$ Term wg. Symmetry wegfällt.
Damit für $w' \neq 0$:

$$\frac{3}{2}w''p(w) + w'^2 p'(w) = 0$$

oder:

$$\frac{d}{dr}\log|w'| = \frac{w''}{w'} = -\frac{2}{3}\frac{p'(w)\,w'}{p(w)} = -\frac{2}{3}\frac{d}{dr}\log p(w)$$

also:

$$|w'| \propto p(w)^{-2/3}$$

Da Dichte der Output-Neurone $M(x)$ im Input-Raum $|dr/dw| = 1/w'$ ist, folgt das Endresultat:

$$M(x) \propto p(x)^{2/3}$$

- Idealerweise hätte man sich $M(x) = p(x)$ gewünscht
- Kohonen-Result :
  - untersamplet Regionen hoher Wahrscheinlichkeit
  - übersamplet Regionen niedriger Wahrscheinlichkeit

Konvergenz

2 FOLIEN zu Kinks

Lernen kann sehr langsam gehen, da Kinks immer nur um eine Position wandern. De facto meist fix.

5.2.4 Dimensionsreduktion

Häufiges Setting:
Projeziere Struktur im Hochdimensionalen ins Niedrig (2-3) dimensionale, a la Finnische Phoneme oben

FOLIE DIM-RED Aufgabe

Standard-Verfahren:
Hauptkomponenten-Analyse (auch als Karhunen-Loeve Trafo, Principal component Analyse (PCA), Singulärwert-Zerlegung bekannt)

- Muster $\vec{x}^k$, Mittelwertsbereinigt
- Berechne Kovarianz-Matrix

$$C_{ij} = \frac{1}{N} \sum_k x_i^k x_j^k$$

- Bestimme (geordnete) Eigenwerte λ_i und Eigenvektoren ξ_i

  ZEICHNUNG zur Varianzzerlegung

- Projeziere Ausgangsmuster (linear) auf d-dimensionalen Raum der d größten Eigenwerte.
- Gab es im Hochdimensionalen einer (lineare) d-dimensionale Struktur, wird sie im d-dimensionalen sichtbar.

Je nach Setting, auch interessant:

- Faktorenanalyse
- Partial least squares
- Principal regression analysis
- Independent component analysis
- Sliced inverse regression

Oja und Sanger Regeln

Vorstufe:

- Betrachte:
$$y = \sum_i w_i x_i$$
- Hebb'sches Lernen:
$$\Delta w_i = \gamma\, y\, x_i$$
- (Bekanntes) Problem:
Gewichte wachsen unbeschränkt
- Nehme an, es sei stabil
$$0 =< \Delta w_i >=< y\, x_i >=< \sum w_j\, x_j x_i >= \sum C_{ij} w_j = C\vec{w}$$
Fixpunkt: 0-Eigenvector
- Kann nicht stabil sein, da jede Fluktuation (mit positivem Eigenwert) exponentiel wächst.
- $\vec{w}$ läuft tendenziell zu Richtung von λ_{max}-Eigenvektor.

Oja-Regel [47]

Neuro-Sicht:

- Verhindere Divergenz der w_i
- Betrachte dazu:
$$\Delta w_i = \gamma\, y\, (x_i - y w_i)$$
Diskussion y und $-y^2$-Terme.
- Oja in action
FOLIE 8.2a

- Sieht aus wie erster Eigenvektor
- Hausaufgabe:

  Ermittle das Funktional, aus dem sich durch Gradientenabstieg die (Batch-Variante) von Ojas Regel ergibt.

Mathematisch betrachtet:
Zu zeigen:

1. Bei Konvergenz: $|w| = 1$
2. w liegt in Richtung des maximalen Eigenvektors von C

Ad 1.:

$$\begin{aligned} 0 =< \Delta w_i > &= < y\, x_i - y^2 w_i > \\ &= < \sum_j w_j\, x_j x_i - \sum_{jk} w_j\, x_j\, w_k x_k w_i > \\ &= \sum_j C_{ij} w_j - \left[\sum_{jk} w_j C_{jk} w_k \right] w_i \end{aligned}$$

oder

$$0 =< \Delta w >= Cw - [w^t C w] w$$

Damit

$$Cw = \lambda w \quad (\ddagger)$$

mit

$$\lambda = w^t C w = w^t \lambda w = \lambda |w|^2$$

q.e.d

Ad 2.:

- Gl.($\ddagger$): Ergebnis-w ist Eigenvektor
- z.z.: Es ist maximaler

- Betrachte Störung eines Eigenvektors c^α

$$w = c^\alpha + \epsilon$$

und wiederhole Rechnung von oben in $0(\epsilon)$:

$$\begin{aligned} <\Delta\epsilon> &= <\Delta w> = C(c^\alpha+\epsilon) - \left[(c^\alpha+\epsilon)^t C(c^\alpha+\epsilon)\right](c^\alpha+\epsilon) \\ &= \lambda^\alpha c^\alpha + C\epsilon - (c^{\alpha^t} C c^\alpha)c^\alpha - (\epsilon^t C c^\alpha)c^\alpha - (c^{\alpha^t} C\epsilon)c^\alpha - (c^{\alpha^t} C c^\alpha)\epsilon + 0(\epsilon^2) \\ &= C\epsilon - 2\lambda^\alpha(\epsilon^t c^\alpha)c^\alpha - \lambda^\alpha\epsilon + 0(\epsilon^2) \end{aligned}$$

- Betrachte Projektion auf anderen Eigenvektor c^β

$$\begin{aligned} c^{\beta^t} <\Delta\epsilon> &= \lambda^\beta c^{\beta^t}\epsilon - 2\lambda^\alpha(\epsilon^t c^\alpha)\delta_{\alpha\,\beta} - \lambda^\alpha c^{\beta^t}\epsilon \\ &= [\lambda^\beta - \lambda^\alpha - 2\lambda^\alpha\delta_{\alpha\,\beta}]c^{\beta^t}\epsilon \end{aligned}$$

- Also: w instabil, wenn $\lambda^\beta > \lambda^\alpha$
  Nur stabil, wenn $\lambda^\alpha = \lambda_{max}$
  q.e.d
- Konvergenz nur bei Stochastischer Approximation, i.e. $\gamma(t) \to_t 0$
- Erweiterungen:
  Betrachte:

$$y_i = \sum_j w_{ij}x_j = \vec{w}_i^t\vec{x}, \quad i = 1,\ldots,M$$

mit Oja-Rule:

$$\Delta w_{ij} = \gamma\, y_i \left(x_i - \sum_{k=1}^{M} y_k w_{kj}\right)$$

oder Sanger-Rule [60]:

$$\Delta w_{ij} = \gamma\, y_i \left(x_i - \sum_{k=1}^{i} y_k w_{kj}\right)$$

Analyse:

- Analog zu oben:
  $\vec{w}_i$ konvergieren gegen die M größten Eigenvektoren
- Bei Sanger in absteigender Reihenfolge
- Bei Oja wird nur der Raum aufgespannt
- Oja und Sanger sind nicht-lokal, i.e. brauchen Information von "ferne".

MERKE:
Oja und Sanger: Online-PCA
Kohonen: "Nichtlineare PCA"

Eine (klassische) Alternative für nichtlineare Dimensionsreduktion:
Multidimensional scaling [59]
Betrachte N Daten $\vec{x}^*$ im hochdimensionalen und ihre Abstände

$$d^*(i,j) = \sum_k (x_k^{*\,i} - x_k^{*\,j})^2$$

Suche Punkte $\vec{x}$ im z.B. 2-dimensionalen durch Minimierung von

$$E(\vec{x}) = \sum_{n=1}^{N} \frac{(d(i,j) - d^*(i,j))^2}{d^{*p}(i,j)}$$

mit

$$d(i,j) = \sum_{k=1}^{2} (x_k^i - x_k^j)^2$$

FOLIEN aus Sammon

Eventuell ergänzen: Bestimmung der richtigen Zieldimension [1]
Eventuell: die beiden Methoden (Isomap, Tenenbaum, LLE, Roweis) aus Science, Dez. 2000, Jan 2002

5.3 Synopsis

Somatotopie bei rezeptiven Feldern kommt nicht a la Kohonen sondern wohl über Gradienten von Botenstoffen.

supervised learning = Lehrer fragt jede Vokabel ab
Selbstorganisation, unsupervised learning = Lehrer sagt: Lernt Vokabeln.

6 Hopfield-Netze

Im weiteren alle Muster binär.

Motivation:
Das Assoziative-Gedächnis-Problem :

- Speichere p Muster x^k, so daß bei Präsentation eines neuen Musters x das Netz das dazu ähnlichste ausgibt.
- "Inhalts-addressierter" Speicher

Mögliche Lösungen:

- Triviale Lösung:

  Hamming-Abstand: Anzahl verschiedener Bits $\in \{0, 1\}$ zweier Muster:

  $$HA(x^1, x^2) = \sum_i x_i^1(1 - x_i^2) + (1 - x_i^1)x_i^2$$

  Berechne Hamming-Abstände $HA(x, x^k)$ und gebe das mit kleinstem Hamming Abstand aus.

- Raffinierter: Lösung eines dynamischen Models

  ZEICHUNG: Fixpunktdynamik

  Genau das tut das Hopfield - Netz [27, 28]

Ab jetzt statt $x_i \in \{0, 1\}$: x_i (zu lernende Muster), S_i (Systemzustand) $\in \{-1, 1\}$

Betrachte Netz mit Dynamik:

$$S_i(t + 1) = sgn\left(\sum_j w_{ij} S_j(t)\right) \quad (\dagger)$$

Dies ist:

- Rückgekoppelte Dynamik ohne eigentlichen Input.

- Input ist Anfangszustand

Update:

- Synchron
- Asynchron

Wie die Gewichte wählen ?

- Betrachte ein Muster $\vec{x}$
- Bedingung für Fixpunkt unter der Dynamik Gl.(†):

$$sgn\left(\sum_j w_{ij}x_j\right) = x_i \quad \forall\, i$$

- Wird erfüllt durch die Wahl

$$w_{ij} \propto x_i x_j, \text{ da } x_j^2 = 1$$

- Konvention:

$$w_{ij} = \frac{1}{N} x_i x_j, \text{ mit } N: \text{ Anzahl der Neurone}$$

- Hebb'sches Lernen, aber mit positivem Gewicht auch für simultan nicht feuernde Neurone $x_i = x_j = -1$.
- Stabilität ?
  - $sgn\left(\sum_j w_{ij}S_j\right)$ ändert sich nicht, wenn einige Komponenten $S_j \neq x_j$.
  - $\vec{x}$ ist ein Attraktor der Dynamik des Netzes.
- Kommentar:

  Symmetrische Kopplungen: Unbiologisch, aber methodisch stimulierend

- Mit $\vec{x}$ ist auch $-\vec{x}$ eine stabile Lösung

Soweit nur ein Muster.

Viele Muster :

- Wahl der Kopplungen:

$$w_{ij} = \frac{1}{N} \sum_{k=1}^{p} x_i^k x_j^k, \text{ mit } N: \text{ Anzahl der Neurone}$$

- Bemerkung:

  Fügt man neue Muster hinzu, kann w_{ij}, Vorzeichen wechseln, unbiologisch. Aber methodisch stimulierend.

- Stabilität

  Sei:

$$y_i = \sum_j w_{ij} x_j$$

  Fixpunkt, wenn

$$sgn(y_i^k) = x_i^k, \quad \forall i \quad (\dagger)$$

$$y_i^k = \sum_j w_{ij} x_j^k = \frac{1}{N} \sum_j \sum_l x_i^l \, x_j^l x_j^k$$

  Ziehe $k = l$ heraus:

$$y_i^k = \sum_j w_{ij} x_j^k = x_i^k + \frac{1}{N} \sum_j \sum_{l \neq k} x_i^l \, x_j^l x_j^k$$

  Ist der 2. Term ("Crosstalk")

  - $= 0$, dann Muster x^k stabil nach Gl. ($\dagger$)

- < 1 immer noch stabil

Muster sind Attraktoren: "Error correcting"

- Uneindeutigkeiten :
  - Mit $\vec{x}$ ist auch $-\vec{x}$ eine Lösung
  - Linearkombinationen

$$x_i^{mix} = sgn(\pm x_i^{k_j} \pm x_i^{k_l} \pm x_i^{k_m})$$

Zu minimierendes Funktional
Betrachte Energiefunktional:

$$H = -\frac{1}{2}\sum_{ij} w_{ij} S_i S_j \qquad (\ddagger)$$

Die Dynamik kann H nur reduzieren
Sei

$$S_i' = sgn(\sum_j w_{ij} S_j) \qquad (\dagger)$$

Betrachte asynchronen Update für i

- Wenn $S_i' = S_i$ ist die Energie unverändert
- Wenn $S_i' = -S_i$

$$\begin{aligned} H' - H &= -\sum_{j\neq i} w_{ij} S_i' S_j + \sum_{j\neq i} w_{ij} S_i S_j \\ &= 2S_i \sum_{j\neq i} w_{ij} S_j \\ &= 2S_i \sum_j w_{ij} S_j - 2w_{ii} \end{aligned}$$

  - 1. Term negativ wg. $(\dagger)$ und Voraussetzung
  - 2. Term negativ wg. $w_{ii} = p/N$

$i = j$ Term in H ist unabhängig von $S_i \rightarrow$ setze i.d.R. $w_{ii} = 0$.
Stabilisiert den Algorithmus.

Bemerkungen:

- Muster (=Attraktoren der Dynamik): Lokale Minima der Energielandschaft Gl. (‡). Überdämpfte (Aristotelische) Dynamik.
- Es kann auch spurious state, andere lokale Minima geben.
- Energiefunktional (Lyapunovfunktion) existiert nur für $w_{ij} = w_{ji}$ "Clever step backwards from biological realism"

Ableitung der Hebb'schen Lernregel:

Energie soll minimal sein, wenn Überlapp zwischen Zustand S und Muster x maximal ist:
Wahl:

$$H = -\frac{1}{2N}(\sum_i S_i x_i)^2$$

oder für p Muster:

$$H = -\frac{1}{2N}\sum_k^p(\sum_i S_i x_i^k)^2$$

Ausmultiplikation:

$$H = -\frac{1}{2N}\sum_k^p(\sum_i S_i x_i)(\sum_j S_j x_j) = -\frac{1}{2}\sum_{ij}(\frac{1}{N}\sum_k^p x_i^k x_j^k)S_i S_j$$

ergibt Hebb'sche Lernregel.

Grundsätzliches Verfahren :

- Formuliere Energiefunktional, dessen Minimum das Probelm löst
- Multipliziere es aus
- Gewichte sind gegeben durch Term vor "$S_i S_j$"

Beispiel:
Travelling Salesman Problem revisited:

- d_{ij} Abstand zwischen Stadt i und j
- "Neuron" :

$$n_{ia} = \begin{cases} 1 & \text{wenn Stadt } i \text{ der } a\text{te Stop} \\ 0 & \text{sonst} \end{cases}$$

- Länge des Weges:

$$L = \sum_{ija} d_{ij} n_{ia} (n_{j\,a+1} + n_{j\,a-1})$$

Constraints:

$$\sum_i n_{ia} = 1 \quad \forall \text{ stops } a$$

$$\sum_i n_{ia} = 1 \quad \forall \text{ Städte } i$$

- Damit:

$$H = \sum_{ija} d_{ij} n_{ia} (n_{j\,a+1} + n_{j\,a-1}) + \lambda \left[\sum_a \left(1 - \sum_i n_{ia} \right)^2 + \sum_i \left(1 - \sum_a n_{ia} \right)^2 \right]$$

- Ausmultiplizieren als Hausaufgabe.
- Wahl von λ mitunter kritisch, Stichwort Regularisierung

Dynamische Muster durch [27]

$$w_{ij} = \frac{1}{N} \sum_{k=1}^{p} x_i^k x_j^k + \frac{\lambda}{N} \sum_{k=1}^{p} x_i^{k+1} x_j^k$$

mit $p + 1 = 1$.
Asymmetrische Kopplung, kein Energiefunktional
Ist System in Zustand $S_i = x_i^k$, folgt:

$$\begin{aligned} y_i^k &= \sum_j w_{ij} x_j^k = \frac{1}{N} \sum_j \sum_l x_i^l \, x_j^l x_j^k + \frac{\lambda}{N} \sum_j \sum_l x_i^{l+1} \, x_j^l x_j^k \\ &= x_i^k + \lambda x_i^{k+1} \frac{1}{N} \sum_j \sum_{l \neq k} x_i^l \, x_j^l x_j^k \end{aligned}$$

Wenn Crosstalk-Terme klein

- $\lambda < 1$: x_i^k stabil
- $\lambda > 1$: x_i^k wird zu x_i^{k+1} gezogen

Erfahrung: Dieses Schema ist instabil.
Für stabile Verfahren, siehe: [23, 24]
Wichtiges Beipiel für Idee, die nicht aus Spin-Systemen kommt.

Dynamik bei endlicher Temperatur

- Zufällige Einflüsse weichen deterministische Dynamik auf.
- Kann helfen, lokale Minima (z.B. Mixed states) zu umschiffen "Simulated Annealing"

In Analogie zum Ising Model, betrachte Glauber-Dynamik:

$$S_i = \begin{cases} +1 & \text{mit Wahrscheinlichkeit } g(y_i) \\ -1 & \text{mit Wahrscheinlichkeit } 1 - g(y_i) \end{cases}$$

mit

$$g(y_i) = \frac{1}{1 + \exp(-2\beta y_i)}$$

somit:

$$1 - g(y_i) = g(-y_i)$$

und, wie gehabt:

$$y_i = \sum_j w_{ij} S_j$$

Wählt man

$$\beta = \beta(t) = 1/T(t)$$

mit

$$T(t) = 1 - t/niter, \quad t = 1, \ldots, niter$$

oder theoretisch empfohlen

$$T(t) = \frac{\gamma}{\log(t)}$$

kann man lokale Minima umgehen, siehe Übung.

Analogie Statistische Mechanik von Magnetsystmen

- $w_{ij} > 0$: Ferromagnet
- $sgn(w_{ij})$ periodisch : Antiferromagnetisch
- w_{ij} zufällig: Spinglässer, im wesentlichen unser Fall

Speicherkapazität

Eventuell: **Replica Technik**
Grundlage: Ising-Modell
Betrachte finite temperature model von oben:
Durchschnittliche Magnetisierung:

$$\begin{aligned} < S_i > &= Prob(+1) \times (+1) + Prob(-1) \times (-1) \\ &= \frac{1}{1 + \exp(-2\beta y_i)} - \frac{1}{1 + \exp(2\beta y_i)} \\ &= \tanh(\beta y_i) \end{aligned}$$

Mean Field Ansatz :
Ersetze y_i durch $< y_i >$

$$< S_i >= \tanh(\beta < y_i >) = \tanh\left(\beta \sum_j w_{ij} < S_j >\right)$$

Fall $p \ll N$

Sei $p \ll N$ und Muster zufällig, i.e. Crosstalk vernachlässigbar.
Sei:

$$< S_i >= m x_i^k$$

Dann:

$$\begin{aligned} m x_i^k &= \tanh\left(\frac{\beta}{N}\sum_l \sum_j x_i^l x_j^l m x_j^k\right) \\ &= \tanh\left(\beta m x_i^k + \frac{\beta m}{N}\sum_j \sum_{l\neq k} x_i^l\, x_j^l x_j^k\right) \qquad (\dagger) \\ &= \tanh(\beta m x_i^k) \end{aligned}$$

Mit : $\tanh(-z) = -\tanh(z)$ folgt:

$$m = \tanh(\beta m)$$

ZEICHNUNGEN Lösung für m, m in Abhängigkeit von T

Mit

$$m =< S_i > / x_i^k = Prob(\text{ spin i correct}) - Prob(\text{ spin i incorrect})$$

ergibt sich

$$< N_{correct} >= \frac{1}{2} N(1+m)$$

ZEICHNUNG $N_{correct}(T)$

Phasenübergang 2. Ordnung

Fall p in der Ordnung von N

Gedankengang:

- Crosstalk nicht zu vernachlässigen
- Entwicklung von tanh(.) in Gl. (†)
- Zentraler Grenzwertsatz für Crosstalk: $N(0, \sigma^2)$
- Nähern und schlachten
- Fall 1: Limes $T \to 0$
  - Führt auf Selbstkonsistenzgleichung:

    $$y(\sqrt{2\alpha} + \frac{2}{\pi} e^{-y^2}) = \frac{2}{\sqrt{\pi}} \int_0^y dx\, e^{-x^2}$$

    mit $\alpha = \frac{p}{N}$

    FOLIE Selbstkonsistenzgleichung
  - $\alpha_{max} = \frac{p}{N} \approx 0.138$
- Fall 1: Endliches T
  - FOLIE PHASE DIAGRAMM A: Muster sind tiefste Minima, B: Spinglass tiefer als Muster, C:Spinglass-Zustand (viele Minima), D: $< S_i >= 0$
  - Für $\alpha \neq 0$ (siehe oben $p \ll N$):
    Phasenübergang 1. Ordnung:
    Speicherfähigkeit geht sprunghaft verloren
  - Potentialdarstellung FOLIE Fig. 2.18

7 Neuronale Netze und Zeitreihen

7.1 Gated Experts und HMMs

Backpropagation kann jede vernünfige Abbildung $y = f(x)$ darstellen
Auf Zeitreihen übertragen:

$$x(t+1) = f(x(t))$$

also Markov-Prozesse

Ein hidden Markov Modell hat nicht die Markov Eingenschaft
siehe: [70] [?] zitieren [71]
und erläutern.
Merke: Ein Hidden Markov Model ist kein Markov-Prozeß

7.2 Clearning und ZRM

Markov-Eigenschaft einführen
Clearning: [?]
Problemstellung von Gated Experts und Clearning ist nicht $y = f(x)$
Radiale Basis Funktionen [5]
Mit $\vec{x}(t-1) = (x(t-1), x(t-2), \ldots, x(t-m))$ und Zentren $\vec{c_i}$

$$x(t) = \sum_{i=1}^{L} \alpha_i \Phi\left(|(\vec{x}(t-1) - \vec{c_i})|\right)$$

mit z.B.:

$$\Phi(x) = \frac{1}{1 + x^2/r^2} \text{ oder } \Phi(x) = e^{-x^2/r^2}$$

Pflasterung des Phasenraums mit "Bumps".

$$y_i = \sum_k w_k e^{-\frac{(x_i - \mu_k)^2}{\sigma^2}}$$

- "Vorteil" gegen Polynome:
  Für Polynome gilt $|Pol(x)| \to \infty$ für $\to \pm\infty$. NN und RBF bleiben beschränkt. Kann einem aber auch was vorgauckeln.
- Nachteile:

  Nichtlinear in den Parametern: Muß numerisch, iterativ optimiert werden (Again: Euphemismus "Lernen")

  Interpretierbarkeit der Parameter ganz schwierig.

Anwendungen :

- NN und sunspots: [69]

  FOLIE aus [69]

- Laserdaten der Santa Fe competition. Sieger 100 Daten, mehr als 1000 Parameter, wenn man ?? Punkte vorhersagt. Glück gehabt. Wenn man ??? + ??? Punkte vorhersagt, gehts kaputt. [68]

8 Support Vektor Maschinen

- Linear nicht lösbare Klassifikationsprobleme können durch Einbettung ins höherdimensionale lösbar werden
- Vorsicht: Nicht lösbare Klassifikationsprobleme können durch Support Vektor Maschinen lösbar erscheinen.

[61, 13]
[10]

9 Reinforcement learning

10 Zusammenfassung

Gegenüberstellung: Neurosprache vs. konventionelle Sprache

Neurosprache	konventionelle Sprache
lernen, Selbstorganisation	schätzen, fitten
Back Propagation	nichtlineare Regression mit sigmoidalen Basiskunktionen
Back Propagation	Kettenregel
Back Propagation	Steepest descent
verallgemeinern	glätten
auswendig lernen	interpolieren
Neurone	Basisfunktionen
Gewichte	Parameter
pruning	Variablenselektion
erkennen	klassifizieren
# versteckte Neurons	Modellselektion
Muster	Beobachtungen
supervised learning	Regression / Diskriminanzanalyse
competitive learning	Clusteranalyse
unsupervised learning (Kohonen)	Optimierung der Mutual Information
higher order neurons	Wechselwirkungen
learning rate	Schrittweite

- Back-Propagation (98 % aller heutigen NN-Anwendungen):
  - Minimierung eines Fehlerfunktionals
  - Alle Probleme der nichtlineren Optimierung wieder entdeckt
  - Alle Probleme der nichtlineren Regression wieder entdeckt
- Kohonen (immer ausprobieren für Dimensionsreduktion):
  - Optimierung der Mutual Information, daher "unsupervised"
- Hopfield-Netz (historisch):
  - Analog zu Spin-System
  - Dynamik durch Energieminimierung

Frage nicht nach der Lernregel und der Struktur des Netzes

–

Frage nach dem zu optimierendes Funktional

Bewegungsgleichungen – Hamiltonian

Häufig: "Manchmal mögen neuronale Netze ja helfen."
Aber nur, weil eine Einlauf manchmal hilft, macht man ihn ja auch nicht bei jedem Problem.
Viele Machbarkeitsstudien, aber keine Überlegenheitsbeweise

"Neuronale Netze" suggeriert mehr als sind sind. Man könnte den Ansatz

$$y_i = ax_i^2 + \epsilon$$

"Den alleinseelig machenden Ansatz" nennen, und dann immer fragen, warum man nicht "Den alleinseelig machenden Ansatz" verwendet anstatt eines Neuronalen Netzes.

Herr Kästner, wo bleibt das Positive: Neuronale Netze haben viele Leute dazu gebracht, sich mit realen komplexen Systemen zu beschäftigen.

11 Was fehlt

- Adaptive resonance theory [6, 7]
- Counterpropagation [19, 21]
- Independent Component Analysis [32]
- Vapnik-Chervonenkis Dimension [67]
-

References

[1] H.U. Bauer and K.R. Pawelzik. Quantifying the neighborhood preservation of self-organizing feature maps. *IEEE Trans. Neural Networks*, 3:570–579, 1992.

[2] H. Bohr, J. Bohr, S. Brunak, R.M.J Cotterill, B. Lautrup, L. Norskov, O.H. Olsen, and S.B. Petersen. Protein secondary structure and homology by neural networks. *FEBS Letters*, 241:223–228, 1988.

[3] H. Braun. *Praktikum Neuronale Netze*. Springer, 1997, Berlin.

[4] R. Brause. *Neuronale Netze*. B.G. Teubner, Stuttgart, 1995.

[5] D.S. Broomhead and D. Lowe. Multivariate functional interpolation and adaptive networks. *Complex Systems*, 2:321–355, 1988.

[6] G.A. Carpenter and S. Grossberg. ART2: Self-organization of stable category recognition codes for analog input patterns. *Applied Optics*, 26:4919–4930, 1987.

[7] G.A. Carpenter and S. Grossberg. The ART of adaptive pattern recognition by a self-organization neural network. *Computer*, March:77–88, 1988.

[8] W.C. Chang. On using principal components before separating a mixture of two multivariate normal distributions. *Appl. Statist.*, 32:267–275, 1983.

[9] J.D. Cowan. McCulloch-Pitts and related neural nets from 1943 to 1989. *Bull. Math. Biol.*, 52:73–97, 1990.

[10] N. Cristianini and J. Shawe-Taylor. *An Introdution to Support Vector Machines*. Cambridge University Press, Cambridge, 2000.

[11] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control Signals Systems*, 2:303–314, 1989.

[12] P. Dayan and L.F. Abbott. *Theoretical neuroscience : computational and mathematical modeling of neural systems*. MIT Press, Cambridge, 2001.

[13] R. Dietrich, M. Opper, and H. Sompolinsky. Statistical mechanics of support vector networks. *Phys. Rev. Lett.*, 82:2975–2978, 1999.

[14] O. Duda and P.E. Hart. *Pattern classification and Scene Analysis*. Wiley, New York, 1973.

[15] Y. Le Cun et al. Backpropagation applied to handwritten zip code digit recognition. *Neural Computation*, 1:541–551, 1989.

[16] R.P. Gorman and T.J. Seynowski. Analysis of hidden units in a layered network trained to classify sonar targets. *Neural Networks*, 1:75–89, 1988.

[17] R.P. Gorman and T.J. Seynowski. Learned classification of sonar targets using a massively-parallel network. *IEEE Trans. Acoustics*, 36:1135–1140, 1988.

[18] D.J. Hand. *Discrimination and Classification.* Wiley, New York, 1992.

[19] R. Hecht-Nielsen. Counterpropagation networks. *Applied Optics*, 26:4979–4984, 1987.

[20] R. Hecht-Nielsen. Kolmogorov's mapping neural networks existence theorem. *Proc. IEEE 1. Int. Conf. Neural Networks*, III:11–14, 1987.

[21] R. Hecht-Nielsen. Application of counterpropagation networks. *Neural Networks*, 1:131–139, 1988.

[22] J. Hertz, A. Krogh, and R.G. Palmer. *Introduction to the theory of neural computation.* Addison-Wesley, Reading, 1991.

[23] A. Herz, B. Sulzer, R. Kühn, and J.L. van Hemmen. The Hebb rule: Storing static and dynamic objects in an associative neural network. *Europhy. Lett.*, 7:663–669, 1988.

[24] A. Herz, B. Sulzer, R. Kühn, and J.L. van Hemmen. Hebbian learning reconsiderd: representation of static and dynamic objects in associative neural nets. *Biol. Cybern.*, 60:457–467, 1989.

[25] A.L. Hodgkin. The local electric changes associated with repetitive action in a non-medullated axon. *J. Physiol.*, 107:165–181, 1948.

[26] A.L. Hodgkin and A.F. Huxley. A quantitatie description of ion currents and its application to conduction and excitation in nerve membranes. *J. Physiol.*, 117:500–544, 1952.

[27] J.J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Science*, 79:2554–2558, 1982.

[28] J.J. Hopfield and D.W. Tank. Computing with neural circuits: a model. *Science*, 233:625–633, 1986.

[29] K. Hornick, M. Stinchombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[30] D.H. Hubel and T.N. Wiesel. Receptive fields, binocular interaction, and the functional architecture in the cat's visual cortex. *J. Phys.*, 160:106–154, 1962.

[31] L. Hubert and P. Arabie. Comparing partitions. *J. Classifikation*, ??:193–218, 1985.

[32] A. Hyvaerinen, J. Karhunen, and E. Oja. *Independent Component Analysis.* John Wiley & Sons, New York, 2001.

[33] A.K. Jain and R.C. Dubes. *Algorithms for Clustering Data.* Prentice-Hall, Englewood Cliffs, NJ, 1988.

[34] J. Kiefer and J. Wolfowitz. Stochastic estimation of the maximum of a regression function. *??*, ??:462–466, 1952?

[35] T. Kohonen. *Self-organizing maps.* Springer, Berlin, 1995.

[36] A.N. Kolmogorov. On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition. *Dokl. Akad. Nauk. USSR*, 114:953–956, 1957.

[37] P.R. Krishnaiah and L.N. Kanal, editors. *Classification, Pattern Recognition and Reduction of Dimensionality*, volume 2 of *Handbook of Statistics.* North Holland, Amsterdam, 1982.

[38] B. Lausen and M. Schumacher. Maximally selected rank statistics. *Biometrics*, 48:73–85, 1992.

[39] Y. Le Cun, J.S. Denker, and S.A. Solla. Optimal brain damage. In D.S. Touretzky, editor, *Advances in Neural Information Processing Systems II*, pages 598–605, San Mateo, 1990. Morgan Kaufmann.

[40] R. Linsker. How to generate ordered maps by maximizing the mutual information between input and output signals. *Neural Computation*, 1:402–411, 1989.

[41] R.P. Lippmann. Review of neural networks for speech recognition. *Neural Computation*, 1:1–38, 1989.

[42] W. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biophys.*, 5:115–133, 1943.

[43] W. McCulloch and W. Pitts. How we know universals: The perception of auditory and visul forms. *Bull. Math. Biophys.*, 9:127–147, 1947.

[44] G.W. Milligan and M.C. Cooper. A study of the comparability of external criteria for hierarchial cluster analysis. *Multivariate Behav. Res.*, 21:441–458, 1986.

[45] M. Minsky and S. Papert. *Perceptrons.* MIT Press, Cambridge, 1969.

[46] D.O. Hebb: A neuropsychological theory. *The Organization of Behaviour.* John Wiley, New York, 1949.

[47] E. Oja. A simplified neuron model as a principal component analyzer. *J. Math. Biol*, 15:167–273, 1982.

[48] D.W. Patterson. *Künstliche neuronale Netze : Das Lehrbuch.* Prentice Hall, Englewood Cliffs, NJ, 1997.

[49] W.H. Press, B.P. Flannery, S.A. Saul, and W.T. Vetterling. *Numerical Recipes.* Cambridge University Press, Cambridge, 1992.

[50] N. Qian and T.J. Seynowski. Predicting the secondary structure of globular proteins using neural network models. *J. Molecular Biology*, 202:865–884, 1988.

[51] G. Radons. On stochastic dynamics of supervised learning. *J. Phys. A: Math. Gen.*, 26:3455–3461, 1993.

[52] W.M. Rand. Objective criteria for the evaluation of clustering methods. *J. Amer. Stat. Soc.*, 66:846–850, 1971.

[53] B.D. Ripley. *Pattern Recognition and Neural Networks.* Cambridge University Press, Cambridge, 1996.

[54] H. Ritter, T. Martinetz, and K. Schulten. *Neuronale Netze.* Addison-Wesley, New York, 1990.

[55] H. Robbins and S. Monro. A stochastic approximation method. *Annals Math. Stat.*, 22:400–407, 1952.

[56] R. Rojas. *Neural Networks: A Systematic Introduction.* Springer, Berlin, 1996.

[57] F. Rosenblatt. *Principles of Neurodynamics: Perceptrons and the theory of brain mechanis.* Spartan Books, New York, 1959.

[58] D. Rumelhart and J. McClelland. *Parallel Distributed Processing.* MIT Press, Cambridge, 1986.

[59] J.W. Sammon. A nonlinear mapping for data structure analysis. *IEEE Trans. Computers*, C-16:401–409, 1969.

[60] T.D. Sanger. Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural Networks*, 2:459–473, 1989.

[61] B. Schölkopf. *Support Vector Learning.* Oldenbourg, München, 1997.

[62] G. Schwarzer, W. Vach, and M. Schumacher. On the misuse of artificial neural networks for prognostic and diagnostic classification in oncology. *Statistics in Medicine*, 19:541–561, 2000.

[63] C.L. Scofield. Learning internal representations in the Coulomb energy network. In *IEEE Int. Conf. Neural Networks*, volume I, pages 271–276, New York, 1988. IEEE.

[64] T.J. Sejnowski and C.R. Rosenberg. Parallel networks that learn to pronounce english text. *Complex Systems*, 1:145–168, 1987.

[65] D.A. Sprecher. On the structure of continuous functions of several variables. *Trans. Amer. Math. Soc.*, 115:340–355, 1965.

[66] G. Tesauro. Neurogammon wins computer olympiad. *Neural Computation*, 1:321–323, 1990.

[67] V.N. Vapnik. *Estimation of Dependencies Based on Empirical Data.* Springer, Berlin, 1982.

[68] A.S. Weigend and M.A. Gerschenfeld, editors. *Time series prediction*, volume XV of *Studies in the Science of Complexity.* Addison-Wesley, Reading, 1994.

[69] A.S. Weigend, B.A. Huberman, and D.E. Rumelhart. Predicting the future: A connectionist approach. *Int. J. Neur. Comp.*, 1:193–209, 1990.

[70] A.S. Weigend, M. Mangeas, and A.N. Srivastava. Nonlinear gated experts for time series: Discovering regimes and avoiding overfitting. *Int. J. Neural Systems*, 6:373–399, 1995.

[71] A.S. Weigend and S. Shi. Predicting daily probability distributions of s&p500 returns. *J. Forecasting*, 19:375–392, 2000.

[72] P. Werbos. *Beyond regression: New tools for prediction and analysis in the behavorial sciences.* PhD thesis, Harvard University, 1974.

[73] B. Widrow and M. Hoff. Adaptive switching circuits. In *IRE WESCON*, pages 96–104. Convention Record, New York, 1960.

[74] K.Y. Yeung and W.L. Ruzzo. Prinipal component analysis for clustering gene expression data. *Bioinformatics*, 17:763–774, 2001.