JOHANN WOLFGANG GOETHE UNIVERSITÄT FRANKFURT AM MAIN

www.uni-frankfurt.de

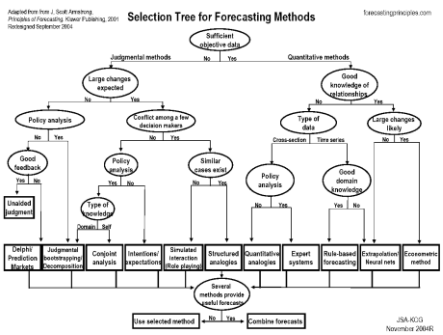
Adaptive Modellierung und Simulation

Kapitel 2: Black-Box-Modellierung

Rüdiger Brause

Vorhersagemodelle

JOHANN WOLFGANG GOETHE UNIVERSITÄT FRANKFURT AM MAIN



Input-Output-Tabellen

JOHANN WOLFGANG GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Beispiel

- Inländische Produktion und Importe -
Mrd. EUR

Lfd. Nr.	Verwendung / Aufkommen	Input der Produktionsbereiche*									
		Erzg. v. Prod. der Land- u. Forstwirtschaft, Fischerei	Gew. v. Bergbau-erz., Steinen u. Erden, Erzg. v. Energie und Gew. v. Wasser	H.v. Mineralölerz., chemischen Erz., Glas, Verarb. v. Steinen u. Erden	Erzg. und Beruh. von Metallen	H.v. Maschinen, Fahrzeugen, DV-Geräten, e-techn. Geräten	H.v. Textilien, Bekleidung, Leder, Holz, Papier, Sekundärrohstoffen u.ä.	H.v. Nahrungsmitteln, Getränken und Tabakwaren	Bauarbeiten	Dl. des Handels u. Verkehrs, Dl. der Nachrichtenübermittlung, Berkehr- u. Gastständen-Dl.	Dl. der Kredit. Inst. u. Vers., Dl. des Grundst.- u. Wohnungswesens u. unternehmensbezogene Dl.
		1	2	3	4	5	6	7	8	9	10
	Output nach Gütergruppen (Zeile 1 bis Zeile 12)										
1	Erzeugn. der Land- u. Forstwirtschaft, Fischerei	1,4	0,0	0,5	-	0,0	2,1	34,0	0,0	1,0	0,0
2	Bergbauerzeugnisse, Steine und Erden, Energie und Wasser	1,4	8,7	31,7	7,5	4,0	3,3	2,5	2,6	7,6	2,6
3	Mineralölerzeugnisse, chemische Erzeugnisse, Glas, Keramik, bearbeitete Steine und Erden	4,0	1,7	64,1	7,9	30,0	11,5	3,5	35,0	12,5	2,7
4	Metalle	0,4	1,0	4,7	47,1	53,0	2,7	1,0	12,1	3,3	0,5
5	Maschinen, Fahrzeuge, DV-Geräte, e-techn. Geräte	1,1	4,3	4,1	5,3	176,3	2,1	1,0	12,9	14,1	3,4
6	Textilien, Bekleidung, Leder und Lederwaren, Erz. des Holz-, Papiergewerbes, Sekundärrohstoffe u.ä.	0,3	0,4	4,7	2,4	10,0	47,7	3,4	8,7	12,1	7,5
7	Nahrungs- und Futtermittel, Getränke, Tabakerzeugnisse	4,0	0,0	2,7	0,0	0,1	0,0	19,7	0,0	13,9	0,0
8	Bauarbeiten	0,4	2,1	1,5	1,0	1,5	0,7	0,6	4,7	5,3	21,8
9	Handelsleistungen, Verkehrs- und Nachrichtenübermittlungs-Dl., Gastständen-Dl.	3,8	3,1	13,6	11,4	37,5	12,9	12,6	13,0	113,0	13,5
10	Dl. der Kreditinst. u. Versicherungen, Dl. des Wohnungw. und sonstige unternehmensbezogene Dl.	4,5	10,0	29,7	10,2	51,2	10,0	14,6	33,2	89,0	257,2
11	Dl. des Gesundheits-, Veterinär- u. Sozialwesens, Erziehungs- u. Unterrichts-Dl., Erholungsind.	1,5	0,4	1,6	0,7	0,7	0,9	0,0	0,5	4,6	3,0
12	Dl. d. öffentl. Verwaltung, Verteid., Sozialvers., Dl. von Körchen, Kultur-Dl. u.ä., Dl. priv. Haushalte	0,3	4,1	1,5	0,7	1,7	3,0	1,0	1,1	0,1	16,2
13	**Vorleistungen der Produktionsbereiche (lfp. 1 bis lfp. 12) bzw. letzte Verwendung von Gütern (lfp. 14 bis lfp. 19)**	**23,3**	**36,7**	**160,3**	**94,3**	**366,0**	**106,5**	**96,2**	**124,5**	**264,5**	**336,0**

Black-Box- Modellierung

JOHANN WOLFGANG GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Lineare Modellierung

Nichtlin. Modellierung: MLP

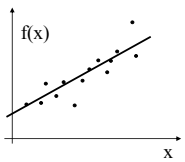
Nichtlin. Modellierung: RBF

Eigenschaften von NN

Lineare Approximation

JOHANN WOLFGANG GOETHE UNIVERSITÄT FRANKFURT AM MAIN

- ***m* Messungen**



- **Modellierung als Gerade**

  $$y = f(x) = y_0 + ax$$

- **Beispiel: Ökonomie**

  Konsum y = f(Einkommen x)

  = Konsumsockel + a ·Einkommen

Lineare Approximation - rauschfrei

JOHANN WOLFGANG GOETHE UNIVERSITÄT FRANKFURT AM MAIN

- **Parameterbestimmung**

  2 Messwerte $y_1(x)$, $y_2(x)$ reichen aus für a, y_0

  RECHNUNG

- **x,y,y**$_0$ sind n-dim Vektoren, **a** eine n × n-Matrix

  m = n^2+n Parameter ⇒ *m* Datenpunkte bei *n* Variablen nötig

Lineare Approximation - verrauscht

- **Modellierung als verrauschte Gerade**

  $\mathbf{y_t = y_0 + a x_t + u_t}$

- **Parameterbestimmung** **Rechnung**

  aus Varianz und Kovarianz

- **Parameterbestimmung** **Rechnung**

  mittels Gauß-Methode

$$R(a) = \langle (y_t - f(a))^2 \rangle_t \overset{!}{=} \min_a$$

Lineare Approximation - verrauscht

- **Mehrere Variable** *Multiple Regression*

$y(t) = f(x_1,x_2,\ldots,x_k) = a_0 + a_1x_1 + a_2x_2 + \ldots + a_kx_k + u$

Zusammenfassung zu

$y_t = (1, x_1(t), \ldots, x_k(t))\,(a_0, a_1, \ldots, a_k)^T + u_t = \mathbf{x}\mathbf{a}^T + u_t.$

$\mathbf{y} = (y_1,\ldots,y_T)^T = \mathbf{X}\mathbf{a}^T + \mathbf{u}$ *T* Zeilen für *T* Messungen, rang(**X**) = k+1

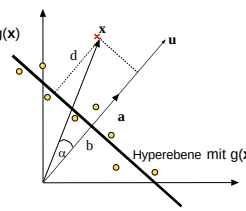
Parameterbestimmung **Rechnung**

Lineare Approximation - verrauscht

- **Minimaler mittlerer Abstand zur Gerade (Hyperebene)**

  im Datenraum $\mathbf{x} = (y,x_i)$

$d = x^T\mathbf{a} - b = g(\mathbf{x})$



Hyperebene mit $g(\mathbf{x}^*) = 0 = \mathbf{x}^{*T}\mathbf{u}/|\mathbf{u}| - b = \mathbf{x}^{*T}\mathbf{a} - b$

Hessesche Normalform

TLMSE = $R(\mathbf{a},b) = \langle d^2 \rangle$

Rechnung: Minimum des TLMSE (Kap.2.2)

TLMSE - Parameterschätzung

- **Vorhanden:** Messungen $\mathbf{x} = (y, \mathbf{x}^-)$
- **Gesucht:** Eigenvektor von $\mathbf{C}_{xx}$ mit min. λ

- **1.Lösung**: **Fixpunktalgorithmus für EV**

  $\mathbf{a}(t+1) = \mathbf{C}\mathbf{a}(t)$, $|\mathbf{a}| = 1$ →EV mit max. EW
  Neuer Eingaberaum $\mathbf{x}' = \mathbf{x} - \mathbf{a}\mathbf{a}^T\mathbf{x}$, $C_{x'x'}$ bilden, nächsten EV lernen.

- **2. Lösung**: **Anti-Hebb-Lernalgorithmus**

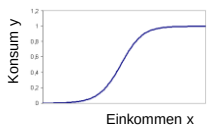
  $\mathbf{a}(t) = \mathbf{a}(t-1) - \mathbf{x}(\mathbf{a}^T\mathbf{x})$, $|\mathbf{a}| = 1$ *Anti-Hebb Lernregel*

Lineare Approximation - Grenzen

- **Grenzen des linearen Modells**
  - linearer Zusammenhang der Daten ? Linearisieren!
  - Parameter sind konstant (oder zeitabhängig?)
  - Einflüsse von x_1, x_2, ... sind nur eingebildet ?

Lineare Approximation - Grenzen

- **Nichtlineare Modellierung: Beispiel Ökonomie**
  Grossverdiener: Wer alles hat, kauft nichts mehr.
  Sigmoidaler Zusammenhang ist wahrscheinlicher:



Linearisierung

- **Linearisieren von nichtlin. Messungen**
  - $y = y_0 + ax^{1/2} + u$

    $z = x^{1/2} \rightarrow y(z) = y_0 + az + u$
  - $y = ax_1^{\alpha} \cdot x_2^{\beta} \cdot e^u$ („Cobb-Douglas-Produktion“)

    $z = \ln(y) \rightarrow z(x) = \ln a + \alpha \ln x_1 + \beta \ln x_2 + u$

    $= a_0 + a_1 x'_1 + a_2 x'_2 + u$
  - nicht-lin. Kontext, z.B. „männlich“ bei $y = y_0 + ax_1 + u$

    $x_2 = 1$ bei „männlich“, sonst null

    $\rightarrow y = y_0 + ax_1 + bx_2 + \ldots + u$

Lineare Approximation

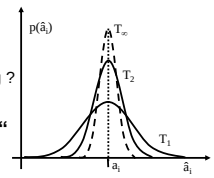
- **Parameter konstant ?**
  - Schätzung $\hat{a}_i$ liegt vor.
  - Parameter $a_i \neq 0$? Oder nur Störung ?
- **Kriterium „erwartungswerttreu“** (unbiased)

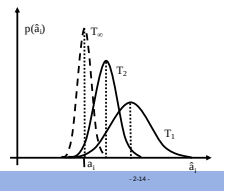
$$\langle \hat{a}_i(t) \rangle_t = a_i$$

- **Kriterium „konsistent“**

$$\forall \varepsilon > 0$$

$$\lim_{t \to \infty} \mathrm{Prob}\left(\left| \hat{a}_i(t) - a_i \right| < \varepsilon \right) = 1$$





Lineare Approximation - Störterm

- **Störterm** $\langle u_t \rangle = u_0 \neq 0$

  Was tun ? Mittelwert in Konstante verschieben:

$$y_t = (a_0 + u_0) + a_1 x_1(t) + a_2 x_2(t) + \ldots + a_k x_k(t) + (u_t - u_0)$$

$$= \quad \acute{a}_0 \quad + a_1 x_1(t) + a_2 x_2(t) + \ldots + a_k x_k(t) + \quad \acute{u}_0$$

- **Forderung**: Keine Korrelation des Störterms mit den Variablen x !

$$\mathrm{cov}\,(u_t, u_{t'} | \mathbf{x}_t) = 0 \qquad \forall t,t' = 1,\ldots,T \quad t \neq t'$$

Lineare Approximation - Multikollinearität

- **Test der Abhängigkeit der Variablen** unter einander
  - Bilde Korrelationskoeffizienten

$$r_{ij} = \frac{\mathrm{cov}(x_i, x_j)}{\sqrt{\mathrm{var}(x_i)}\sqrt{\mathrm{var}(x_j)}} = \frac{\mathrm{cov}(x_i, x_j)}{\sigma_i \sigma_j}$$

  - Bilde Bestimmtheitsmaß

$$R^2 = \frac{\left\langle (\hat{y}_t - \bar{y})^2 \right\rangle}{\left\langle (y_t - \bar{y})^2 \right\rangle} \le \frac{\left\langle \hat{u}_t^{\,2} \right\rangle}{\left\langle (y_t - \bar{y})^2 \right\rangle} + \frac{\left\langle (\hat{y}_t - \bar{y})^2 \right\rangle}{\left\langle (y_t - \bar{y})^2 \right\rangle} = 1$$

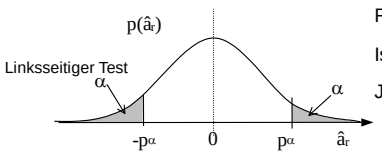
  - Teste: Gilt $r_{ij}^2 \ge R^2$?

    JA: **Multikollinearität liegt vor!**

Lineare Approximation

- **Test der Einflüsse: Ist Variable x_r nötig?**
  Ist der Parameter $a_r = 0$?

Nullhypothese H_0 : $a_r = 0$



Verteilung der beobachteten Parameterwerte

Rechtsseitiger Test:

Ist Prob($\hat{a}_r > p_\alpha$) $\le \alpha$?

JA: H_0 ok.

Hypothesentest auch für $a_r = s$ brauchbar!

Black-Box- Modellierung

Lineare Modellierung

Nichtlin. Modellierung: MLP

Nichtlin. Modellierung: RBF

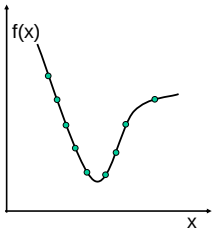
Eigenschaften von NN

NichtLineare Approximation

- **Polynomansatz (Taylorentwicklung)**

$f(x) = f(x_0)+(x-x_0)f^{(1)}(x_0)+(x-x_0)^2 f^{(2)}(x_0)/2 + (x-x_0)^3 f^{(3)}(x_0)/6+...$

$= A + B(x-x_0) + C(x-x_0)^2 + D(x-x_0)^3 + ...$

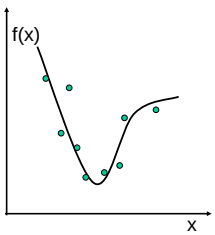


Direkte Bestimmung der Parameter aus den Messwerten

NichtLineare Approximation

- **Polynomansatz mit Messfehlern**

$f(x) = A + B(x-x_0) + C(x-x_0)^2 + D(x-x_0)^3 + ...$

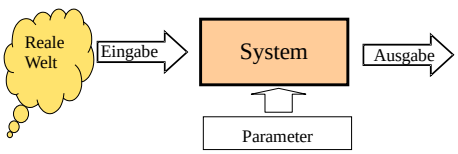


Indirekte Bestimmung der Parameter aus den Messwerten:

Regression *n*-ter Ordnung,

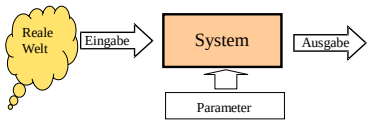
z.B. mit kleinstem quadr. Fehler

Nichtlineare Approximation



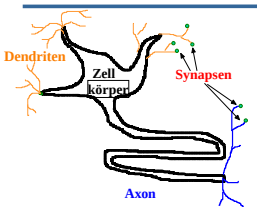
- **Einsatzarten**
  - Adaptive Schätzung von Prozeßparametern
    Nicht-lin. Reaktionen, Produktionsoptimierung,...
  - Adaptive Kontrolle und Regelung
    Landekontrollsysteme, Roboterkontrolle,..
  - Adaptive Klassifikation
    Qualitätskontrolle, med. Diagnose, Bonitätsprüfung,..

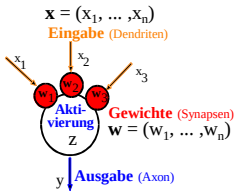
Nichtlineare Approximation



- **Einsatzgebiete**
  - Echtzeitreaktionen
    z.B. Stahlwalzstraßen, Flugzeugsteuerung,..
  - Analytisch unbekannte Abhängigkeiten
    z.B. Polymerchemie, DNA-Schätzungen, ..
  - Analytisch nicht zugängige Abhängigkeiten
    z.B. psychische Faktoren, ergebnisverändernde Messungen,..
  - Analytisch nur unter großem Aufwand bearbeitbare, hochdimensionale Gesetzmäßigkeiten
    z.B. Wechselkursabhängigkeiten,..
  - Statistische Analysen durch untrainierte Benutzer (?!)

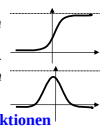
Was sind Neuronale Netze ?





Quetschfunkion

Radiale Basis-funktion



$y = S(z)$

$$z = \sum_{i=1}^{n} w_i x_i = \mathbf{w}^T\mathbf{x}$$

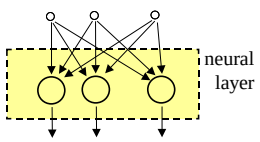
Aktivierung

Ausgabefunktionen

Was sind Neuronale Netze ?

- **DEF** ***Schicht***

$\mathbf{x} = (x_1 \quad x_2 \quad \cdots \quad x_n)$



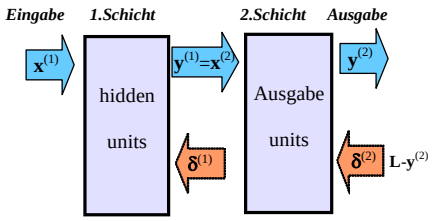
$\mathbf{y} = (y_1 \quad y_2 \quad \cdots \quad y_m)$

Lineare Schicht

$$\mathbf{y} = \left(\sum_i w_{1i} x_i, \ldots, \sum_i w_{mi} x_i\right)^T = \mathbf{W}\cdot\mathbf{x}$$ *Matrixmultiplikation*

Backpropagation-Grundidee

Netzarchitektur und Lernen



Schichtweise Verbesserung durch Rückführung des Fehlers

Anwendung - NetTalk

Sejnowsky-Rosenberg 1986

- Automatisches System Text -> Sprache
- Vorläufer: DECtalk, 20 Mann-Jahre, 95% Genauigkeit
- NetTalk: 15 CPU-Stunden, 98% Genauigkeit
- Eingabe: Texte

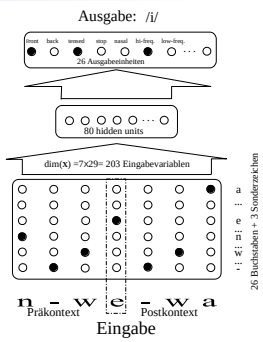
Anwendung - NetTalk

- **Architektur**

Eingabe: 29 Zeichen, binär (26 Buchstaben+ 3Sonderzeichen: Pkt., Wortgrenze), 7 Buchstaben als Kontext.

Hidden units: 80 Stück

Ausgabe: 26 Merkmale, binär (23 Laut- und drei Artikulations-merkmale: Continuation, Wortgrenze, Stop)



Anwendung - NetTalk

- **Training**
  - protokollierte, von Kindern gesprochenen Sätze,
  - zufallsmäßig eingegebene Worte eines Wörterbuchs aus 20.000 Einträgen

Eingabe x: Buchstabe eines Worts im Kontext
Lehrervorgabe L(x): Phonologische Transkription
Einfügen eines Sonderzeichens „Continuation“, wenn Buchstabe nicht gesprochen wird (!)

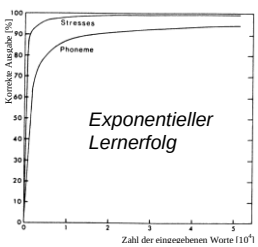
Anwendung - NetTalk

- **Ergebnisse: 3 Phasen des Sprachlernens**
  - Zuerst wurden die Konsonanten und Vokale als Klassen getrennt. Innerhalb der Klassen blieben die Phoneme aber noch gemischt, so dass sich die Laute wie "Babbeln" anhörten.
  - Dann wurden die Wortgrenzen als Merkmale entwickelt, so dass "Pseudoworte" erkennbar wurden.
  - Zuletzt, nach ca. 10 Durchgängen pro Wort, entstand eine verständliche Sprache, die sich mit fortlaufender Erfahrung weiter verbesserte

NetTalk: Training

Training
- transkribiertes Wörterbuch 20.000 Einträge
- Protokollierte Kindersätze

Ergebnis
- Trennung der Konsonanten von Vokalen („Babbeln“)
- Entwicklung der Wortgrenzen („Pseudoworte“)
- Verständliche Sprache (10xTraining pro Wort)



Backpropagation-Grundidee

- Zielfunktion minimieren

Ziel = minimaler quadrat. Fehler

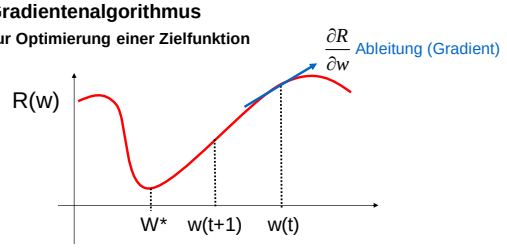
$$R(w) = \langle \, (L(x) - y^{(2)}(x))^2 \, \rangle_x = \min_w$$

Wie erreichen?

Lernen mit Zielfunktionen

Gradientenalgorithmus

zur Optimierung einer Zielfunktion



$$w(t+1) = w(t) - \gamma \frac{\partial R}{\partial w}$$

Lernrate

Lernregel

Stochastisches Lernen

Lernen mit Zielfunktion $R(w) = \langle R_x(w,x) \rangle_x$

$$\mathbf{w}(t) = \mathbf{w}(t-1) - \gamma(t) \, \nabla_w \, R_x(\mathbf{w}(t-1))$$

wird ersetzt durch

Lernen mit **stochast. Zielfunktion** $R_x(w,x)$

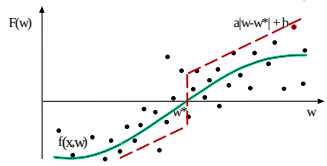
$$\mathbf{w}(t) = \mathbf{w}(t-1) - \gamma(t) \, \nabla_w \, R_x(\mathbf{w}(t-1), \mathbf{x}(t))$$

stochastisches Lernen

Wieso darf man das?

Stochastische Approximation

Gesucht: Nullstelle einer stochast. Funktion f(x,w) $= R_x'(x,w)$



Methode 1: Alle Ereignisse x abwarten und dann F(w) = $\langle f(x,w)\rangle_x$ bilden

$$\mathbf{w}(t) = \mathbf{w}(t\text{-}1) - \gamma(t)\, F(\mathbf{w}(t\text{-}1))$$

Methode 2: Einfach f(x,w) verwenden *Robbins, Monro 1951*

$$\mathbf{w}(t) = \mathbf{w}(t\text{-}1) - \gamma(t)\, f(\mathbf{w}(t\text{-}1),\mathbf{x}(t))$$ ***stochastische Approximation***

Stochastische Approximation

Voraussetzungen *das klein Gedruckte...*

- die Funktion F(w) := $\langle f(x,w)\rangle_x$ ist *zentriert*, d.h. F(w*) = 0
- F(w) ist *ansteigend*, d.h. F(w<w*) < 0, F(w>w*) > 0
- F(w) ist *beschränkt* mit $|F(w)| \leq a|w-w^*|+b < \infty$ a,b > 0
- f(x,w) hat *endliche Varianz*, d.h. $\sigma^2(w) = \langle (F(w) - f(x,w))^2\rangle_x < \infty$
- $\gamma(t)$ ***verschwindet***, $\gamma(t) \to 0$
- $\gamma(t)$ wird ***nicht zu*** schnell ***klein*** $\sum_{t=1}^{\infty} \gamma(t) = \infty$
- $\gamma(t)$ wird ***nicht zu groß*** $\sum_{t=1}^{\infty} \gamma(t)^2 < \infty$

Dann ex.

$$\lim_{t\to\infty} \langle (w(t) - w^*)^2\rangle = 0$$ mittl. quadr. Konvergenz *Robbins-Monro*

$$P(\lim_{t\to\infty} w(t) = w^*) = 1$$ *Blum*

Backpropagation-Lernregel letzte Schicht

Lernziel: $R(w^*) = \min \langle (y(\mathbf{x},\mathbf{w}) - L(\mathbf{x}))^2 \rangle_{\mathbf{x}}$ *min.mittl. quadr. Fehler*

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) - \gamma \frac{\partial R}{\partial w_i}$$ *Gradienten-Lernregel*

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) - \gamma\, (y(\mathbf{w}_i) - L(\mathbf{x}))\, \frac{\partial S(z)}{\partial w_i}$$ *stoch. Approximation*

Rechnung: Ableitung der Zielfunktion

Backpropagation-Lernregel letzte Schicht

Lernziel: $R(w^*) = \min E(y(w) - L(x))^2$ *min.mittl. quadr. Fehler*

$w_i(t+1) = w_i(t) - \gamma \frac{\partial R}{\partial w_i}$ *Gradienten-Lernregel*

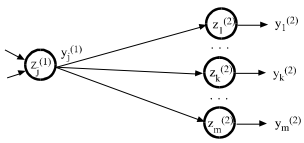
$w_i(t+1) = w_i(t) - \gamma\,(y(w_i) - L(x))\,\frac{\partial S(z)}{\partial w_i}$ *stoch. Approximation*

mit $\frac{\partial S(z)}{\partial w_i} = \frac{S'(z)\partial z}{\partial w_i} = \frac{S'(z)\partial}{\partial w_i}\sum_j w_j x_j = S'(z)x_i$

$\delta_i := -(y(w_i) - L(x))\,S'(z)$

$\Delta w_{ij}(x) = \gamma\,\delta_i\,x_j$ *Delta-Regel*

Fehler-Backpropagation



Beeinflussung voriger Schichten $z_i^{(1)} \rightarrow R$

Delta-Regel für Schicht 1

$$\delta_i^{(1)} = \frac{-\partial R_x}{\partial y_i^{(1)}}\frac{\partial y_i^{(1)}}{\partial z_i^{(1)}} = \left(\sum_{k=1}^{m} \delta_k^{(2)} w_{ki}^{(2)}\right) S'\left(z_i^{(1)}\right)$$

Online vs Offline-Lernen

- **ONLINE-Learning:**

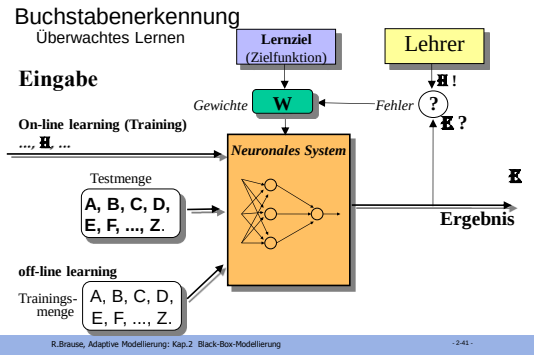
```
WHILE NOT Abbruchbedingung erfüllt:
    Delta := 0
    FORALL Trainingsmuster x
        berechne Delta(W(x))
        W(t) := W(t-1) + Delta  // Lernen mit jedem Muster
    END FOR
END WHILE
```

Online vs Offline-Lernen

- **OFFLINE-Learning:**

WHILE NOT Abbruchbedingung erfüllt:
  GesamtDelta := 0
  FORALL Trainingsmuster x
    berechne Delta(W(x))
    GesamtDelta := GesamtDelta + Delta(W(x))
  END FOR
  W(t) := W(t-1) + GesamtDelta // Lernen am Schluss!
END WHILE

Arten des Lernens: Beispiel



Frage: Offline- oder Online-Lernen?
Backpropagation- Code

```
REPEAT (* jeweils einen Trainingszyklus *)
 dw1:=0.0; dw2:=0.0; γ:=0.1;
 REPEAT      (* Für alle Trainingsmuster im PatternFile *)
  Read( PatternFile,x1,L);          (*Einlesen der Eingabe, Ausgabe *)
   (* Ausgabe errechnen *)
  FOR i:=1 TO p DO    x2[i]:= S(z(w1[i],x1)) END (* Für hidden units *)
  FOR i:=1 TO m DO      (* Für alle Ausgabeneuronen*)
     y2[i]:= S(z(w2[i],x2))
     d2[i]:= -(y2[i]-L[i])*(1-y2[i])*y2[i]          (*  (y-L)(1-S)S  *)
  END
  FOR i:=1 TO m DO        (* Gewichtsveränderungen in 2. Schicht *)
     FOR j:=1 TO p DO
       dw2[i,j] := dw2[i,j] + γ*d2[i]*x2[j]
     END;
  END
  FOR i:=1 TO p DO        (* Gewichtsveränderungen in 1. Schicht *)
     FOR j:=1 TO n DO        (* Für alle Eingabemusterkomp.*)
       dw1[i,j] := dw1[i,j]+γ*SumProd(i,m,d2,w2)*(1-x2[i])*x2[i]*x1[j]
     END;
  END
 UNTIL    ( EOF( PatternFile))
 w1:=w1+dw1;  w2:=w2+dw2;                (* Korrektur der Gewichte *)
UNTIL    Fehler_klein_genug
```

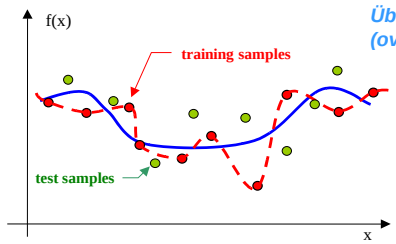
Lernen - Probleme

- Die Konvergenz ist relativ *langsam*, besonders bei mehreren Schichten
- Das System kann in einem *lokalen Optimum* "stecken" bleiben
- Trotz guter Trainingsleistung zeigt der Test *schlechte Ergebnisse*

Verbesserungen des BP-Algorithmus

Problem

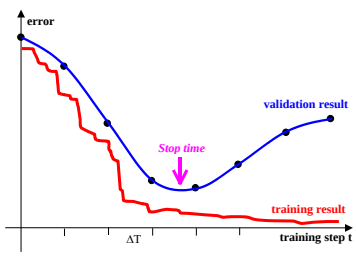
- Trotz guter Trainingsleistung zeigt der Test ***schlechte Ergebnisse***



Überanpassung (overfitting) !

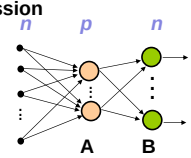
Verbesserungen des BP-Algorithmus

Lösung: ***Stopped Training***



Initialisierung der Neuronengewichte

- **Beispiel Informationskompression**



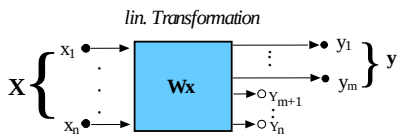
- **Lin. Approximation**
(1. Glied Taylorentwicklung)
Beispiel: n-p-n Netz Kodierer
$\mathbf{y} = \mathbf{B}_{pxn}\, \mathbf{A}_{nxp}\, \mathbf{x}$

- **Min. quadr. Fehler** bei globalem Minimum

$\Rightarrow$ **A = ?**

Transformation mit minimalem MSE

Allgemeine Situation



$$\min_{\mathbf{w}} R(\mathbf{W}) = \min \langle (\mathbf{x}-\hat{\mathbf{x}})^2 \rangle \quad \textit{least mean squared error} \ \text{(LMSE)}$$

Wann minimal ?

Transformation mit minimalem MSE

Minimaler Rekonstruktionsfehler

$$\min_{\mathbf{w}} R(\mathbf{W}) = \min \langle (\mathbf{x}-\hat{\mathbf{x}})^2 \rangle \quad \textit{least mean squared error} \ \text{(LMSE)}$$

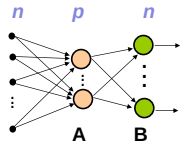
$$\mathbf{x} = \sum_{i=1}^{m} y_i \mathbf{w}_i + \sum_{i=m+1}^{n} y_i \mathbf{w}_i \qquad \hat{\mathbf{x}} = \sum_{i=1}^{m} y_i \mathbf{w}_i + \sum_{i=m+1}^{n} c_i \mathbf{w}_i \qquad y_i = \mathbf{x}^T \mathbf{w}_i$$

- **Was ist die beste Schätzung für die Konstanten c_i?**
  min $R(c_i)$ = ? **Rechnung!**

- **Bei welchen Basisvektoren w_i ist der Fehler minimal ?**
  min $R(w_i)$ = ? **Rechnung!**

Analyse der Neuronengewichte

Hauptkomponentenanalyse

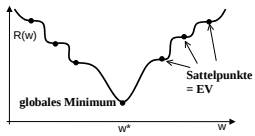


Lin. Approximation
(1. Glied Taylorentwicklung)

- **A** besteht aus **Eigenvektoren** der **Kovarianzmatrix**

$C_{xx} = \langle (\mathbf{x}-\langle\mathbf{x}\rangle)(\mathbf{x}-\langle\mathbf{x}\rangle)^T \rangle$

$$(C_{ij}) = \left(\frac{1}{N} \sum_{k=1}^{N} \left(x_i^k - \langle x_i^k \rangle \right) \left(x_j^k - \langle x_j^k \rangle \right) \right)$$



Normierung der Eingangsvariablen

- **Problem:** unterschiedliche Skalierungen

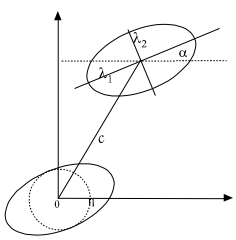
z.B. x_1 : Weg in [m]
x_2 : Grösse in [cm]
x_3 : Gewicht in [kg]
x_4 : Farbwert in [RGB-Zahl]

Normierung aller numerischen Werte auf gleiche Skala !

- Gleichen Mittelwert, etwa $\langle x \rangle = 0$
- Gleiche Varianz $\sigma = 1$

Normierung der Eingangsvariablen

- **Abstände bilden nur mit normierten Variablen**



Mahalanobis-Abstand

$d^2 = (\mathbf{x} - \mathbf{c})^T \mathbf{C}^{-1} (\mathbf{x} - \mathbf{c})$

$= (\mathbf{x} - \mathbf{c})^T \mathbf{M}^T \mathbf{M} (\mathbf{x} - \mathbf{c})$

Entspricht einer Skalierung, Drehung, Verschiebung mit

$\mathbf{x} \rightarrow \mathbf{Mx} = \mathbf{SDVx}$ mit $\mathbf{x} \rightarrow (\mathbf{x}^T, 1)^T$

$\mathbf{S} = \begin{pmatrix} s_1 & 0 \\ 0 & s_2 \end{pmatrix}, \quad \mathbf{D} = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix},$

$\mathbf{V} = \begin{pmatrix} 1 & 0 & -c_1 \\ 0 & 1 & -c_2 \end{pmatrix}$

Verbesserungen des BP-Algorithmus

Problem $\Delta w_{ij}(x) = \gamma\, \delta_i\, x_j = \gamma\, (..)S'(z_i)\, x_j$

Die Konvergenz ist relativ ***langsam***, besonders bei mehreren Schichten

Problem Ausgabefunktion

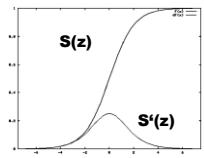
Bei Wahl von S = **Fermifunktion**

ist die Ableitung eine Glocken- Funktion mit $S'(-\infty) = 0 = S'(\infty)$

und damit bei sehr großem oder kleinem x

$\delta(x) = 0$

$\Rightarrow$ **Kein** Lernen mehr möglich!



$$S'(z) = \frac{\partial}{\partial z}\frac{1}{1+e^{-z}} = \frac{+1-1+e^{-z}}{\left(1+e^{-z}\right)^2} = \left(1-S(z)\right)S(z)$$

Problem Ausgabefunktion

Abhilfen:

- Andere Ausgabefunktion wählen (z.B. Sinus)
- Andere Zielfunktion wählen (Information statt quadr.Fehler)
- Andere Lernfunktion wählen, z.B. Ergänzung durch Backpropagation *Trägheitsmoment*

$\delta(t+1) = \alpha\delta(t) + (1-\alpha)\delta(t-1)$ z.B. $\alpha = 0.9$

Black-Box- Modellierung

Lineare Modellierung

Nichtlin. Modellierung: MLP

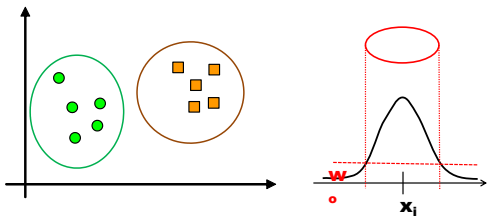
Nichtlin. Modellierung: RBF

Eigenschaften von NN

Klassifikation und RBF

Motivation: lokale Cluster-Klassenbildung

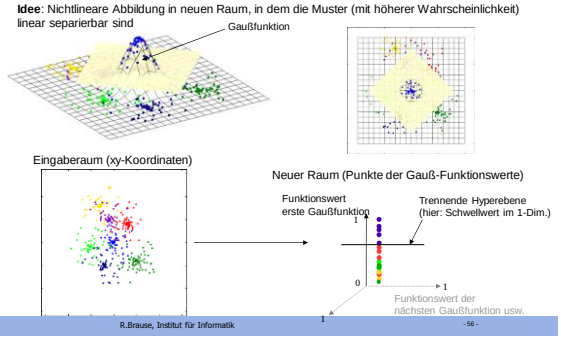
$$\Omega_i = \{ \mathbf{x} \mid S(|\mathbf{x}-\mathbf{x}_i|) > w_0 \}$$



Klassifikation und RBF

Bessere Klassifikation durch gekrümmte Entscheidungsgebiete

Idee: Nichtlineare Abbildung in neuen Raum, in dem die Muster (mit höherer Wahrscheinlichkeit) linear separierbar sind

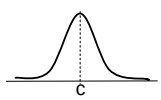


Radiale Basisfunktionen

Definition **Glockenfunktion**

Funktion S_G mit den Eigenschaften

- $S_G(z) \geq 0$, $S_G(-\infty) = S_G(\infty) = 0$,
- $0 < \int_{-\infty}^{\infty} S_G(x)dx < \infty$



- Es ex. ein $c>0$ mit $S_G(z)$ nicht anwachsend $\forall z \in [c,\infty)$, nicht abfallend $\forall z \in (-\infty,c)$

Also ist $S_G(c)$ globales Maximum.

RBF maximaler Information

Welche Basisfunktionen hat maximale Information ?

$H(p^*) = \max_p H(p(x))$ $\quad x \in \Re, \quad$ $p^*(x) = ?$

NB1: $\int p(x)\,dx = 1$ $\quad$ oder $g_1(x) := \int p(x)dx - 1 = 0$

NB2: $\sigma^2 = \langle x^2 \rangle = \int_{-\infty}^{+\infty} p(x)\, x^2\, dx$ $\quad$ oder $g_2(x) := \int_{-\infty}^{+\infty} p(x)x^2 dx - \sigma^2 = 0$

Ansatz ***Lagrange-Funktion***

$$L(p, \mu_1, \mu_2) := H(p) + \mu_1 g_1(p) + \mu_2 g_2(p)$$

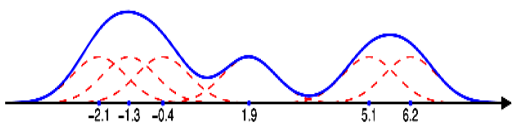
$$\frac{\partial L(p^*)}{\partial p} = 0, \quad \frac{\partial L(\mu_i)}{\partial \mu_i} = 0$$

(Rechnung Kap.5.2)

Ergebnis $p^*(x) = A \exp(-x^2/2\sigma^2)$ **Gauß'sche Glockenkurve**

Parzen Window - Methode

Approximation durch Überlagerung von Basisfunktionen



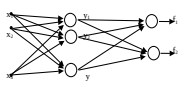
$$\rho(x) = \frac{1}{N}\sum_{i=1}^{N} W(x - x_i) \qquad W(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-x^2/2\sigma^2}.$$

Perfekte Approximation bei abnehmender Breite σ, wobei

$$\lim_{N\to\infty} \sigma(N) = 0, \qquad \lim_{N\to\infty} N\sigma^n(N) = \infty$$

RBF-Netze

Typisch: 2-Schichten Netzwerk

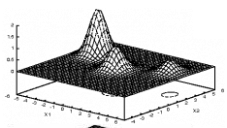


Aktivität

nicht normiert

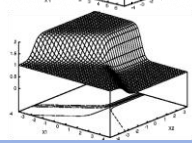
$$f_i(\mathbf{x}) = \sum_{k=1}^{m} w_k y_k = \sum_{k=1}^{m} w_k S_k(\mathbf{x})$$

mit $S_k(\mathbf{c}_k, \mathbf{x}) = e^{\frac{-(c_k - x)^2}{2s^2}}$



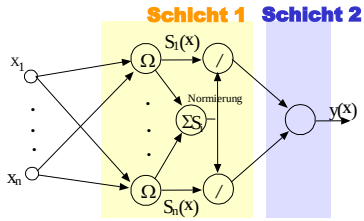
normiert

$$f_i(\mathbf{x}) = \sum_{k=1}^{m} w_k y_k = \frac{\sum_{k=1}^{m} w_k S_k(\mathbf{x})}{\sum_{j=1}^{m} S_j(\mathbf{x})}$$



RBF-Netze

Aktivität Normiertes RBF-Netzwerk



$$y(x) = f(\mathbf{x}) = \sum_i w_i \tilde{S}_i(\mathbf{x},\mathbf{c}_i) \quad \text{mit} \quad \tilde{S}_i(\mathbf{x},\mathbf{c}_i) = \frac{S_i(\mathbf{x},\mathbf{c}_i)}{\sum_k S_k(\mathbf{x},\mathbf{c}_k)}$$

Klassifikation mit RBF-Netzen

Beste Klassifizierung *Bayes-Klassifizierung*

Suche Klasse ω_l so, daß $p(\omega_l|\mathbf{x}) = \max_k p(\omega_k|\mathbf{x})$

Wir wissen: $p(\omega_l|\mathbf{x}) = \frac{p(\omega_l,\mathbf{x})}{p(\mathbf{x})} = \frac{p(\omega_l,\mathbf{x})}{\sum_k p(\omega_k,\mathbf{x})} = \frac{p(\mathbf{x}\,|\,\omega_l)p(\omega_l)}{\sum_k p(\mathbf{x}\,|\,\omega_k)p(\omega_k)}$

Situation:

Gaußvert. Abweichg. vom Klassenprototypen $\mathbf{c}_i$: $p(\mathbf{x}|\mathbf{c}_i) = Ae^{\frac{-(c_k-x)^2}{2\sigma^2}} = S(\mathbf{c}_i,\mathbf{x})$

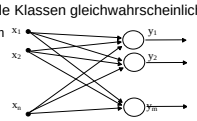
Bayes-Klassifizierung mit NN: Seien alle Klassen gleichwahrscheinlich

$p(\omega_l) = 1/m$

Suche Klasse *i* so, dass mit $y_i = \frac{S(\mathbf{c}_i,\mathbf{x})}{\sum_k S(\mathbf{c}_k,\mathbf{x})}$

$y_i = \max_k \; y_k$

winner take all



RBF *Suche Maximum*

Klassifikation mit winner-take-all

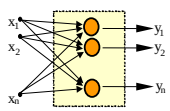
Suche Maximum der Aktivität

Ein-Schicht-Netzwerk

(1 Cluster = 1 Klasse)

Suche Klasse *i* so, dass mit $y_i = S(\mathbf{c}_i,\mathbf{x}) \,/\, \Sigma\, S(\mathbf{c}_k,\mathbf{x})$

$y_i = \max_k \; y_k$

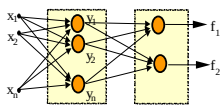


Zwei-Schichten-Netzwerk

(mehrere Cluster = 1 Klasse)

Suche Klasse *i* so, dass mit $f_i = \sum_i w_i y_i$

$f_i = \max_k \; f_k$



⇒ **Lernen nur** der Gewichte für y_i bzw. f_i

Frage

- **Was ist das Ziel der Bayes-Klassifikation?**

- **Antwort**
  1. Die minimale Fehlerwahrscheinlichkeit
  2. Die maximale bedingte Wahrscheinlichkeit für eine Entscheidung
  3. Die minimale Abweichung vom korrekten Wert
  4. Die maximale Korrelation mit dem korrekten Wert

RBF- Lernen

- **Lernmethode: Einzelne Anpassung der Schichten**
  - Anpassen der 1. Schicht: Lage $\mathbf{c}_i$ und Varianz $\mathbf{C}$
  - Anpassen der 2. Schicht: Gewichte $\mathbf{w}_j$

Pro: schneller, da weniger komplex
Contra: Suboptima möglich

- **Lernmethode: Gleichzeitige Anpassung beider Schichten**

  z.B. mit Backpropagation

Pro: Globales Optimum leichter erreichbar
Contra: langsam, mehr Daten für gegebenen max. Fehler nötig

RBF- Lernen: 1.Schicht

- **Erstellen der 1. Schicht**

Bekannte Trainingsdaten
- Festlegen der RBF- Zentren und Weiten durch Clusterung (k-mean, Kohonen-Netze, ...)
- Initiale Festlegung der **Anzahl** der Zentren, Iterative Verbesserung von Lage und Weite durch sequentielles Training

Unbekannte Daten
- Feste Unterteilung des Eingaberaumes, z.B. Netz potentieller Zentren bei uniformer, fester Weite für Datenbereich
- Inkrementeller Aufbau des Netzes: Sequentielle Regression

RBF-Lernen : 1.Schicht

- ***k-means*-Clusterung**

Wähle k zufällige Muster $\mathbf{x}_j \in A$ als Clusterzentren $\mathbf{c}_j$, bilde $C_j = \{\mathbf{c}_j\}$

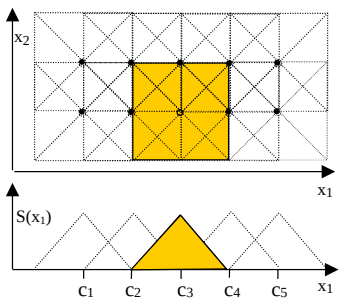
REPEAT

- Ordne alle $\mathbf{x}_i \in A$ zu den nächstgelegenen Clusterzentren zu: Suche für $\mathbf{x}_i$ das Cluster $\mathbf{c}_z$ so, dass $|\mathbf{x}_i - \mathbf{c}_z| = \min_k |\mathbf{x}_i - \mathbf{c}_k|$, und füge $\mathbf{x}_i$ zu C_z zu.
- Entferne alle Cluster i mit $|C_i| \leq 1$
- Bilde für jedes Cluster k ein neues Zentrum $\mathbf{c}_k = \langle \mathbf{x} \rangle$ als Mittelwert aller Muster in C_k

UNTIL Iterationszahl > Max

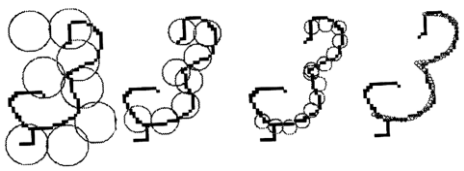
RBF- Lernen : 1.Schicht

- **Feste Unterteilung**



RBF- Lernen : 1.Schicht

- **Adaptive Unterteilung**



RBF-Lernen

Erstellen der 1. Schicht: Sequentielle Regression

- Start mit einem Neuron
- Füge ein neues Neuron hinzu für jedes Beispiel mit hohem Fehler (Abweichung vom gewünschten Netz-Ausgabewert)
- Verändere die Parameter bei den Nachbarn so, dass der Fehler verringert wird (Einpassen des neuen Neurons)

Das Netzwerk wächst solange, bis der Approximationsfehler auf das gewünschte Maß zurückgegangen ist.

RBF-Lernen

2. Schicht: Fehlerminimierung MSE der lin. Schicht

z.B. mit **Backpropagation-Lernen** 2. Schicht

$\Delta \mathbf{w}_i = -\gamma\,(y_i - L_i)\,\mathbf{x}$ *Gewichte von S_k zu Ausgabe i*

$\mathbf{x} := (S_1,..,S_n)$ und $\mathbf{w} := (w_1,..,w_n)$

oder: **TLMSE**

$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) - \gamma_1 \mathbf{x} y_i$ und $|\mathbf{w}_i(t+1)| = 1$ *Anti-Hebb Lernen*

$\mathbf{x} := (S_1,..,S_n,L_i)$ und $\mathbf{w} := (w_1,..,w_n,w_{n+1})$

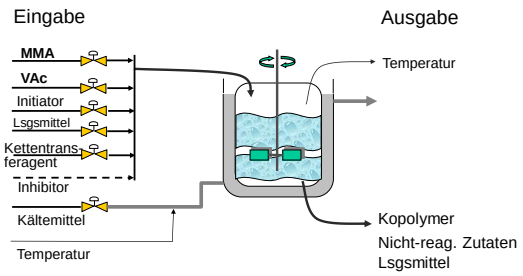
oder**: Konkurrenz-Lernen**

$y_c = \max_i y_i$

$\Delta \mathbf{w}_c = -\gamma\,(y_c - L_c)\,\mathbf{x}$ *Gewichte von S_k zu Ausgabe c*

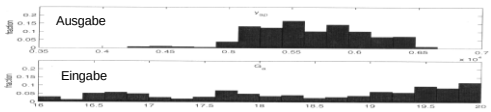
Anwendung: Industriereaktor

- **Chem. Synthese eines Polymers**

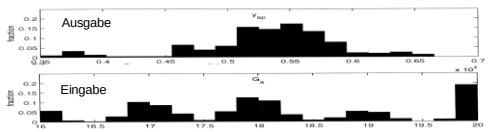


Inputgenerierung

- Strategie **RUS**: Random uniform sampling

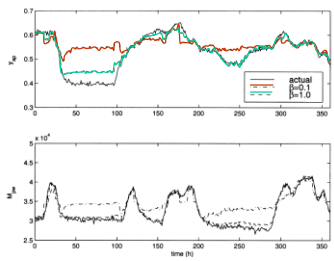


- Strategie **RDS**: Random distributed sampling



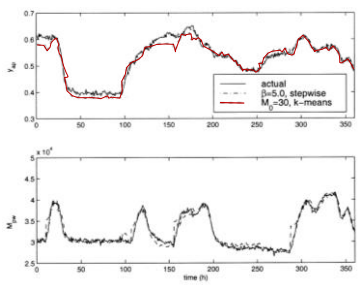
Ergebnisse

- Schmale vs. breite RBF



Ergebnisse

- K-means Clusterung vs. Sequ. Regression



Black-Box- Modellierung

Lineare Modellierung

Nichtlin. Modellierung: MLP

Nichtlin. Modellierung: RBF

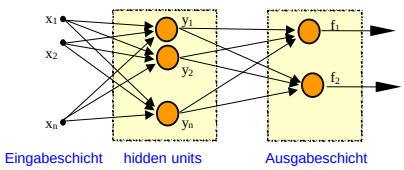
Eigenschaften von NN

Approximationsleistungen Neuronaler Netze

- **Allgemeine Eigenschaften von NN**
  - Kann man jede beliebige Funktion approximieren?
  - Wenn ja, mit welcher Architektur ?
  - Wie viele Schichten benötigt man ?
  - Wie viele Neuronen pro Schicht braucht man ?

Fähigkeiten: Nomenklatur

- **Mehrschichtennetze**



$$\hat{f}(\mathbf{x}) = \sum_{j=1}^{m} w_j^{(2)} S(\mathbf{w}_j, \mathbf{x})$$

Approximationsleistungen Neuronaler Netze

- **Voraussetzungen**
  - Sigma-Funktionen (2-Schicht. NN)

$$\Sigma^n(S) := \{ \hat{f} \mid \hat{f} : \Re^n \to \Re \quad \text{mit} \quad \hat{f}(\mathbf{x}) = \sum_{j=1}^{m} w_j^{(2)} S(z_j(\mathbf{x})) \}$$

wobei $w_j^{(2)}$ aus $\Re$, $\mathbf{x}$ aus $\Re^n$,

und $z_j \in z^n := \{ z \mid z(x) = w^{(1)T}x + b \}$ *affine Funktionen* $\Re^n \to \Re$

$$\lim_{x\to\infty} S(x) = 1, \quad \lim_{x\to\infty} S(x) = 0$$

Approximationsleistungen Neuronaler Netze

- **Aussage 1** (diskrete Punkte) *Hornik, Stinchkombe, White 1989*

Für die Funktionswerte *jeder beliebigen Funktion*

$f(x) : \Re^n \to \Re$ von *N* Mustern $x^1 .. x^N$

gibt es eine Sigma-Funktion f, so dass für alle Muster x^i

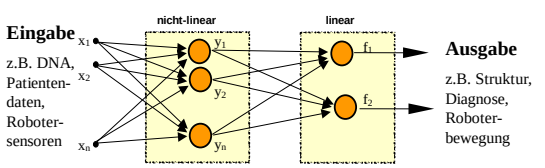
mit i = 1..N gilt

$\hat{f}(\mathbf{x}^i) = f(x^i)$ *punktweise Übereinstimmung*

Approximationsleistung Neuronaler Netze

- **Aussage 2** (stetige Funktionen)

Ein 2-Schichtennetzwerk mit nicht-linearer Ausgabefunktion S(z) kann JEDE beliebige Funktion beliebig dicht approximieren, wenn genügend Neuronen ex.



Fähigkeiten der Multilayer-Netzwerke

JOHANN WOLFGANG GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Satz

Jede beliebige, stetige Funktion f(x) in einem kompakten Intervall ("kompakte Teilmenge des $\Re^n$ ") kann beliebig dicht (*uniform dicht* im Sinne der L_s-Norm in der Menge C^n aller stetigen Funktionen und ρ_p-*dicht* in der Menge der Borel meßbaren Funktionen) durch eine Sigma-Funktion F(x) approximiert werden

Anmerkung: Gilt auch für S = stetig, begrenzt, nicht-konstant (RBF)

Fähigkeiten der Multilayer-Netzwerke

JOHANN WOLFGANG GOETHE UNIVERSITÄT FRANKFURT AM MAIN

Frage:

Wieviel Schichten muss ein Netzwerk mindestens haben, um eine beliebige Funktion beliebig gut zu approximieren?

? Antworten:

- eine
- zwei
- drei
- unendlich viele