



# Musterklassifikation und Parameterbestimmung mit Funktionen

- analytische Verfahren
- regelbasierte Verfahren
- lernbasierte Verfahren

Prof. Dr.-Ing. Andreas Meisel

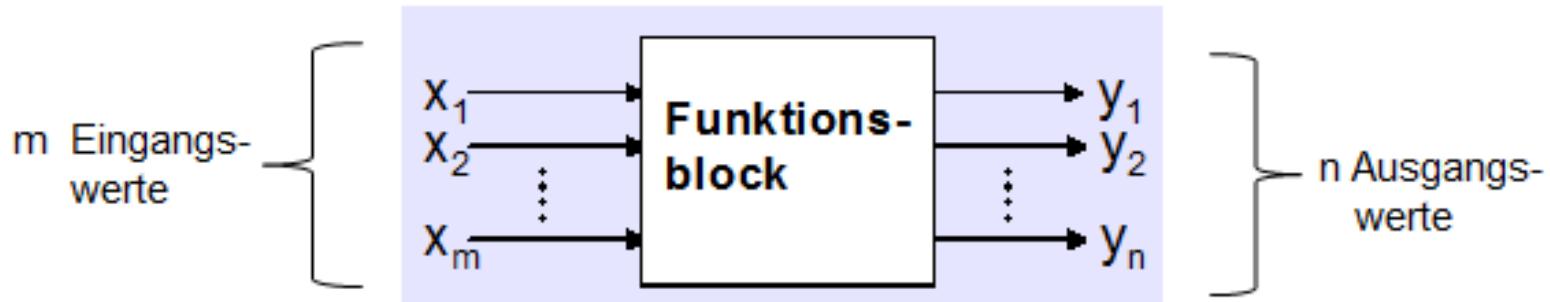


## 5. Klassifikation/Parameterbestimmung mit Funktionen

### 5.1 „Funktionsblöcke“ = Funktionsvektoren

#### 5.1.1 Einführung

Die  $n$  Ausgangswerte des Funktionsblocks ( $y_1, y_2, \dots, y_n$ ) hängen ab von den  $m$  Eingangswerten ( $x_1, x_2, \dots, x_m$ ).



Jeder Ausgang ( $y_1, y_2, \dots, y_n$ ) wird durch eine Funktion  $f_i(x_1, x_2, \dots, x_m)$  mit  $i = 1 \dots n$  realisiert.

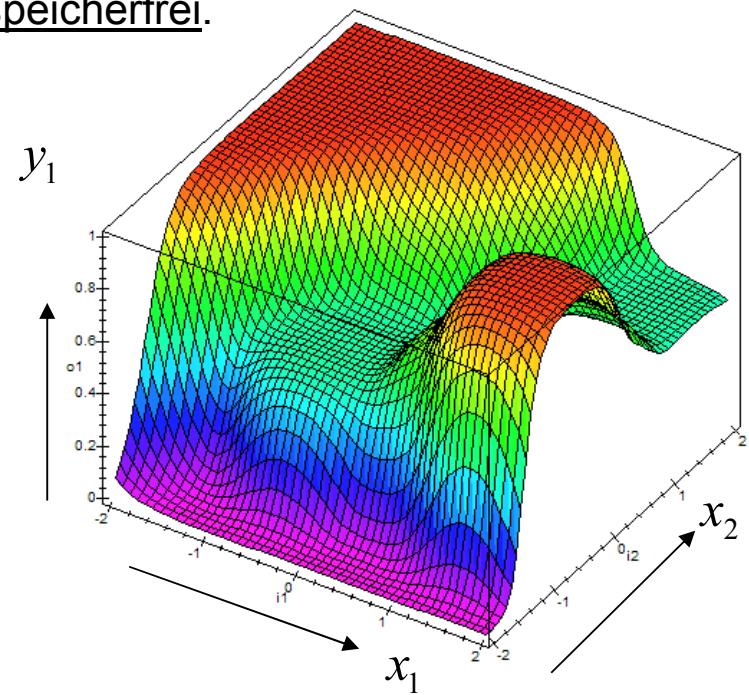
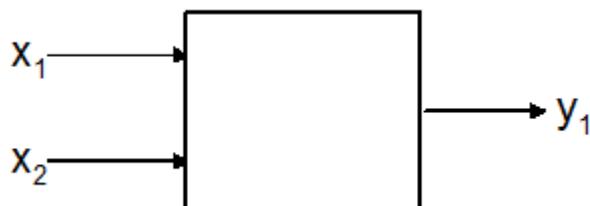


## 5.1.2 Zustandsfreiheit

- Eingangswerte (Eingangsvektor):  $\vec{x} = (x_1, x_2, x_3, \dots, x_m)$   $x_i \in \mathbb{R}$
- Ausgangswerte (Ausgangsvektor):  $\vec{y} = (y_1, y_2, y_3, \dots, y_n)$   $y_i \in \mathbb{R}$
- Zustandsfrei heißt, die Ausgangswerte sind nur von den gerade anliegenden Eingangswerten abhängig :  $\vec{y} = \vec{f}(\vec{x})$

D.h. der Funktionsblock berücksichtigt nicht die Eingangswerte, die in der Vergangenheit angelegen haben. Er ist also speicherfrei.

**Beispiel:** Zustandsfreie Funktion mit 2 Eing./1 Ausg.

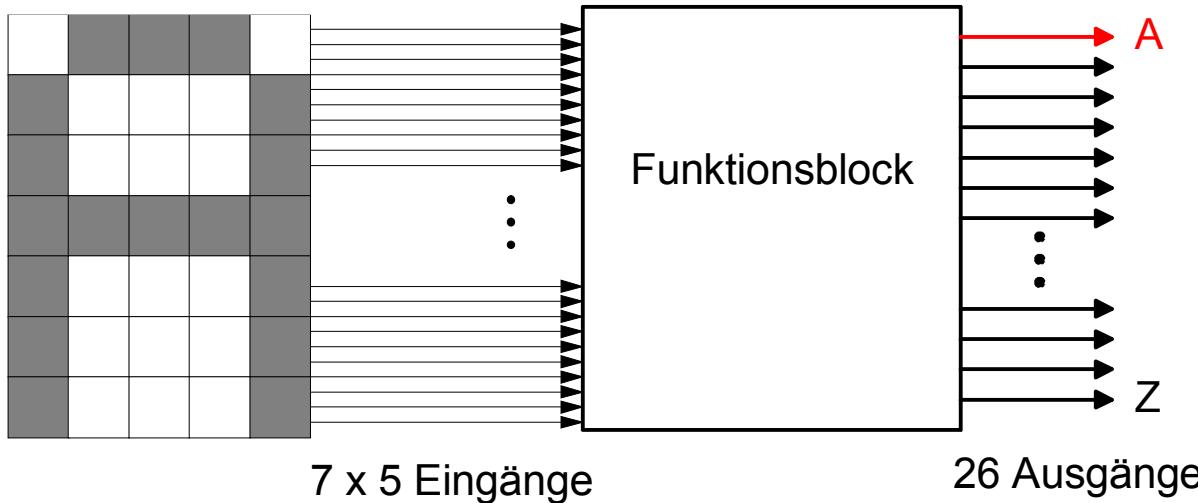




## 5.1.3 Anwendungsbeispiele von zustandsfreien Funktionsblöcken

### 5.1.3.1 Zeichenklassifikator

7 x 5 Sensorfeld



**Eingangswertebereich:** 0...1

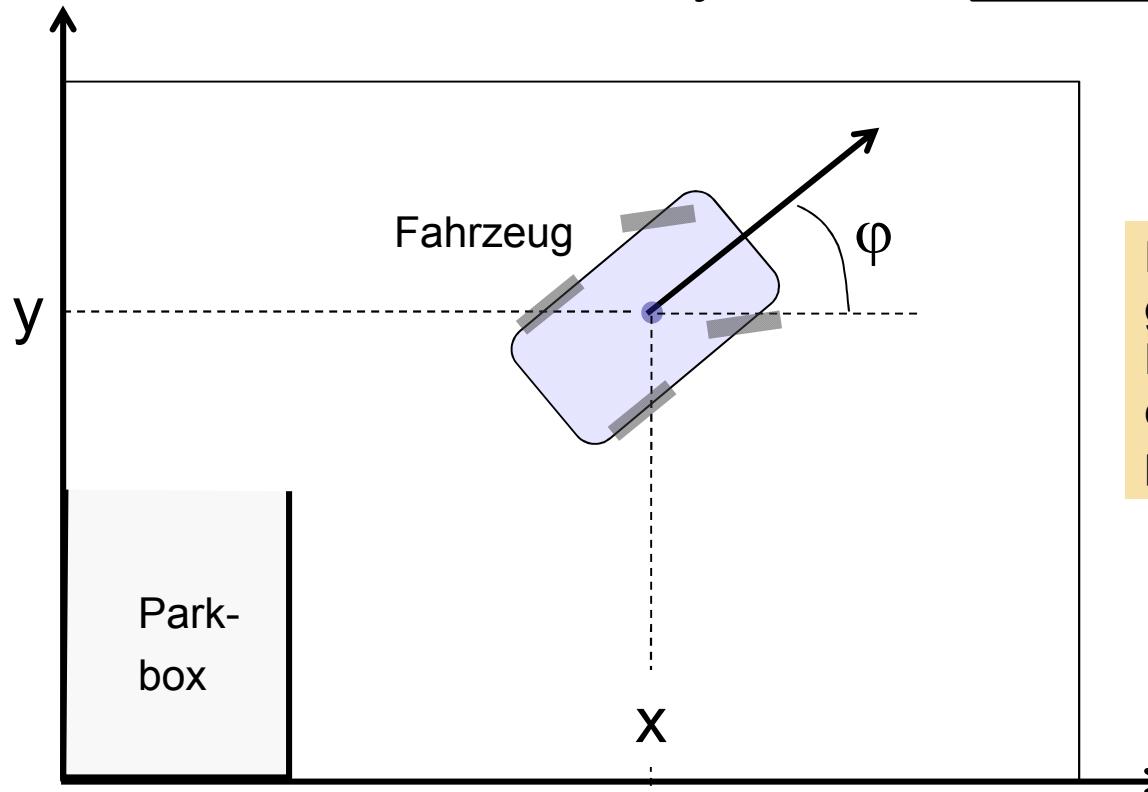
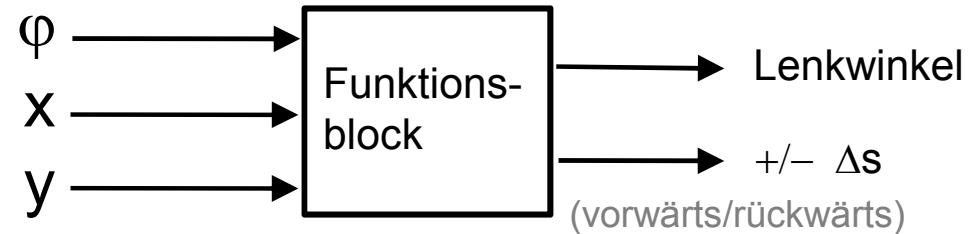
**Ausgangswertebereich:** 0...1

Typisch für Klassifikatoren ist, daß der Ausgang mit dem höchsten Ausgangswert als „Gewinner“ gedeutet wird. Der Gewinner legt die Klasse fest.

Der Ausgangswertebereich ist typ. begrenzt, z.B. [-1, 1] oder [0, 1].



### 5.1.3.2 Einparkassistent



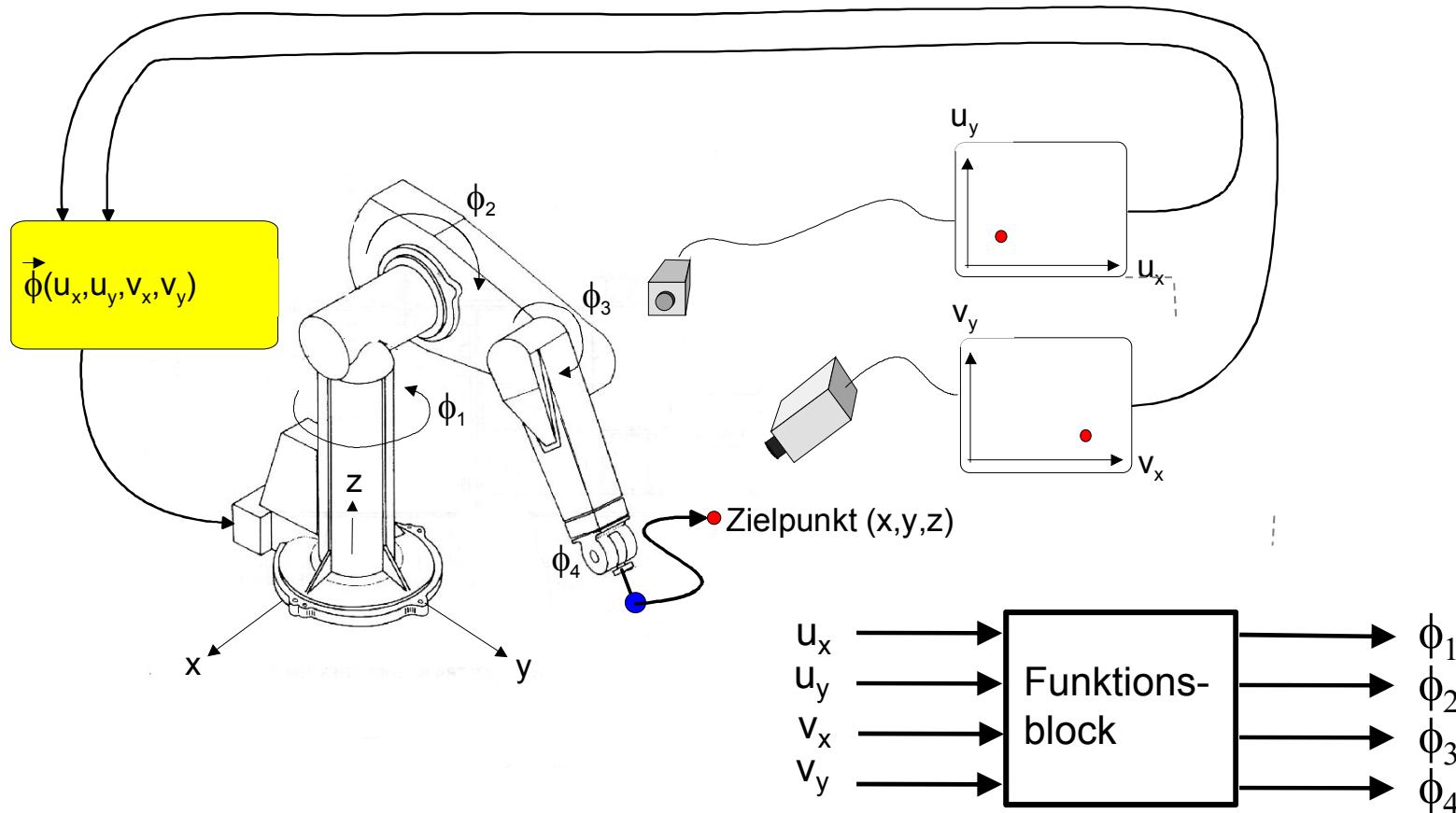
Hier werden die ausgegebenen Werte als Parameter verwendet, mit deren Hilfe der Einparkprozess gesteuert wird.

**Eingangswertebereich:** durch die Anwendung festgelegt

**Ausgangswertebereich:** durch die Anwendung festgelegt



### 5.1.3.3 Visumotorische Koordination eines Roboters

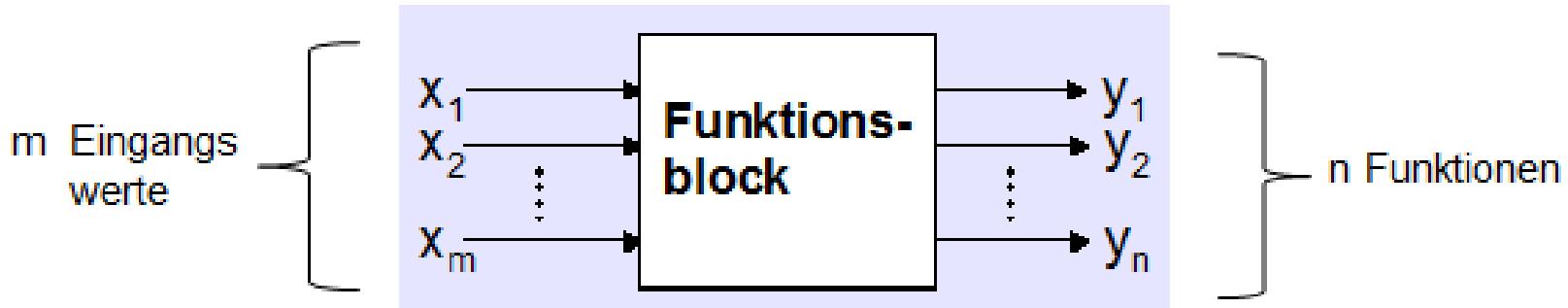


**Eingangswertebereich:** durch die Anwendung festgelegt

**Ausgangswertebereich:** durch die Anwendung festgelegt



## 5.1.4 Wie erhält ein Funktionsblock sein Verhalten ?



Das Verhalten des Funktionsblocks kann auf unterschiedliche Arten festgelegt werden :

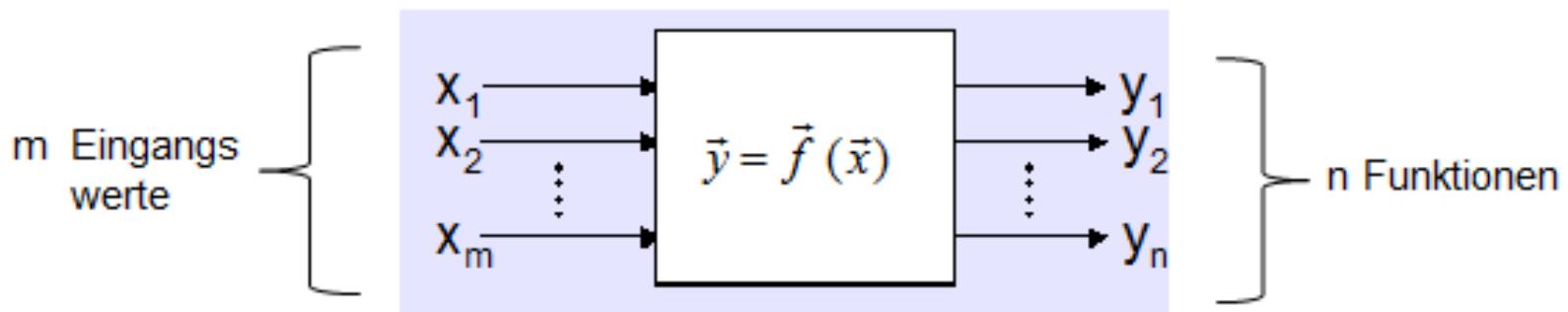
- **analytisch** : durch gegebene Funktionen oder Gleichungen → Kap. 5.2
- **regelbasiert** : durch scharfe oder unscharfe (d.h. fuzzy) Regeln → Kap. 5.3
- **durch Lernen** : anhand von Beispielen (mit oder ohne Modellwissen) → Kap. 5.4



## 5.2 Festlegung des Funktionsblock-Verhaltens durch Funktionen oder Gleichungen

### 5.2.1 Gegebene Funktionsvektoren $\vec{y} = \vec{f}(\vec{x})$

In diesem Fall sind die Ausgangsgrößen bereits als Funktion der Eingangsgrößen angegeben. → trivial





## Beispiel: Affine Transformation (Target-to-source)

$$\vec{x}_z = \begin{pmatrix} x_z \\ y_z \end{pmatrix} \xrightarrow{\quad} \boxed{\quad} \xrightarrow{\quad} \vec{x}_q = \begin{pmatrix} x_q \\ y_q \end{pmatrix}$$

$$\begin{pmatrix} x_q \\ y_q \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \cdot \begin{pmatrix} x_z \\ y_z \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}$$



Ausgangsgrößen



## 5.2.2 Gleichungen der Form $\vec{f}(\vec{x}, \vec{y}) = 0$

### 5.2.2.1 Lineare Gleichungen:

Der Zusammenhang zwischen Ein- und Ausgang liegt als lineares Gleichungssystem vor. Die Ausgangsgrößen sind die Unbekannten des Gleichungssystems.

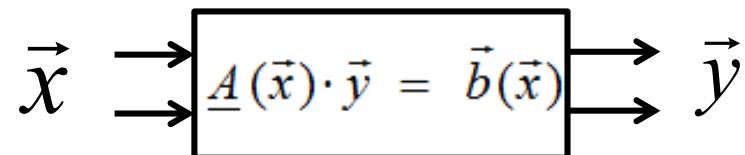
$$\underline{A}(\vec{x}) \cdot \vec{y} = \vec{b}(\vec{x}) \quad \xrightarrow{\hspace{1cm}} \quad \underline{A}(\vec{x}) \cdot \vec{y} - \vec{b}(\vec{x}) = 0$$

\underbrace{\hspace{100pt}}

$\vec{f}(\vec{x}, \vec{y})$

auch als Nullstellensuche formulierbar

mit      Eingangsvektor  $\vec{x}$   
           Ausgangsvektor  $\vec{y}$



Die Ausgangsgrößen erhält man durch Lösen des Gleichungssystems, im überbestimmten Fall durch Bestimmung der Ausgleichslösung.

Anm.: .... oder durch eine Nullstellensuche.



## Beispiel: Inverse affine Transformation

Die Vorwärtstransformation (Source-to-Target) sei gegeben.  
Die Rückwärtstransformation (Target-to-Source) ist gesucht.

$$\vec{x}_z = \begin{pmatrix} x_z \\ y_z \end{pmatrix} \xrightarrow{\quad} \boxed{\begin{pmatrix} x_z \\ y_z \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \cdot \begin{pmatrix} x_q \\ y_q \end{pmatrix} + \begin{pmatrix} b_1 \\ b_2 \end{pmatrix}} \xrightarrow{\quad} \vec{x}_q = \begin{pmatrix} x_q \\ y_q \end{pmatrix}$$



$$\begin{pmatrix} x_z - b_1 \\ y_z - b_2 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \cdot \begin{pmatrix} x_q \\ y_q \end{pmatrix}$$



Ausgangsgrößen



**vorbereitende Übung:** Welche Funktionsparameter (a,b,c...) lassen sich linear aus Messwerten ( $x_i$ ,  $y_i$ ) bestimmen ?

a)  $y = a \cdot x^4 + b \cdot \sin(\frac{\pi}{2} \cdot x^3) + c$

b)  $y = a \cdot x^b + 2 \cdot \sin(cx) + 1$

c)  $y = \frac{ax^2 + bx + c}{dx^2 + ex + 1}$

d)  $y = \frac{ax + b}{cx + d} + f$



### 5.2.2.2 (a) Nichtlineare Gleichung: $f(x, y) = 0$

Der Zusammenhang zwischen Ein- und Ausgang liegt als Gleichung vor und eine Umstellung der Gleichung nach der Ausgangsgröße  $y$  ist nicht möglich.

#### Beispiel: einfache Polynomtransformation

$$x_z = a_0 + a_1 x_q + a_2 x_q^4$$



**Lösung:** durch iterative Nullstellensuche (z.B. Newton-Verfahren)

**Anmerkungen:** Es kann auch keine oder mehrere Lösungen geben  
(evtl. Lösungsintervall angeben)

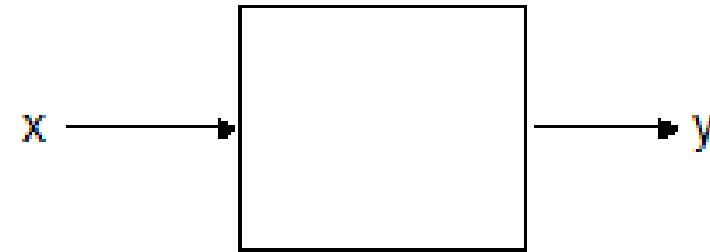
#### Beispielproblemstellung:

- nichtlin. geom. Bildtransformationen  
(z.B. wenn Vorwärtstransformation gegeben – Rückwärtstransformation gesucht)



## ÜBUNG: Iterative Nullstellensuche 1

$$x^2 + 2xy^2 - e^{\frac{y}{x}} - 7 = 0$$



Welchen Wert hat y für x=1 ?  
Startwert sei  $y_0=3$ .



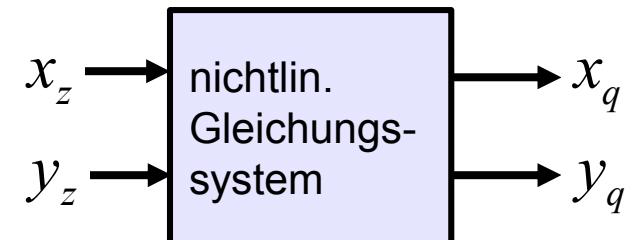
### 5.2.2.2 (b) Nichtlineares Gleichungssystem: $\vec{f}(\vec{x}, \vec{y})=0$

Der Zusammenhang zwischen Ein- und Ausgangsvektor liegt als Gleichungssystem vor und eine Umstellung der Gleichung nach den Ausgangsgrößen ist nicht möglich.

#### Beispiel: Polynomtransformation

$$x_z = a_0 + a_1 x_q + a_2 y_q + a_3 x_q y_q$$

$$y_z = b_0 + b_1 x_q + b_2 y_q + b_3 x_q y_q$$



**Lösung:** durch iterative Nullstellensuche für mehrere Unbekannte (Jacobimatrix, ....)

**Anmerkungen:** Es kann auch keine oder mehrere Lösungen geben

#### Beispielproblemstellungen:

- nichtlin. geom. Bildtransf. (z.B. wenn V.-transf. gegeben – R.-transf. gesucht)
- Inverse Roboterkinematik
- Abfanggerade



## ÜBUNG: Ausgleichskreis bestimmen

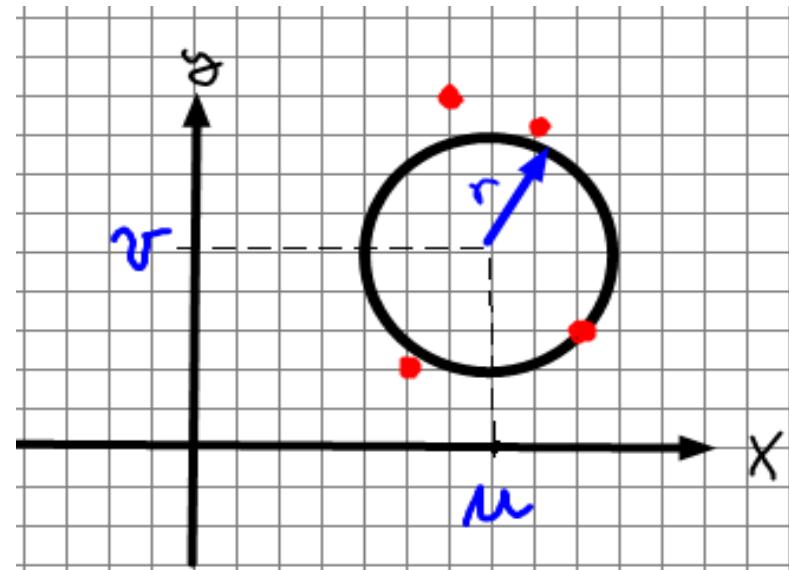
Gesucht sind die Parameter  $u$ ,  $v$  und  $r$  des Ausgleichskreises:

$$r^2 = (x - u)^2 + (y - v)^2$$

- a) Geben Sie die Iterationsformel für den 1. Iterationsschritt an.
- b) Geben Sie ein MATLAB-Programm zur Bestimmung der Kreisparameter an.

Als Kantenpunkte sind gegeben:

$$(x_a, y_a) = (5, 2), \quad (x_b, y_b) = (9, 3), \\ (x_c, y_c) = (6, 9), \quad (x_d, y_d) = (8, 8).$$



Verwenden Sie als Startwert  $u = v = r = 1$ .



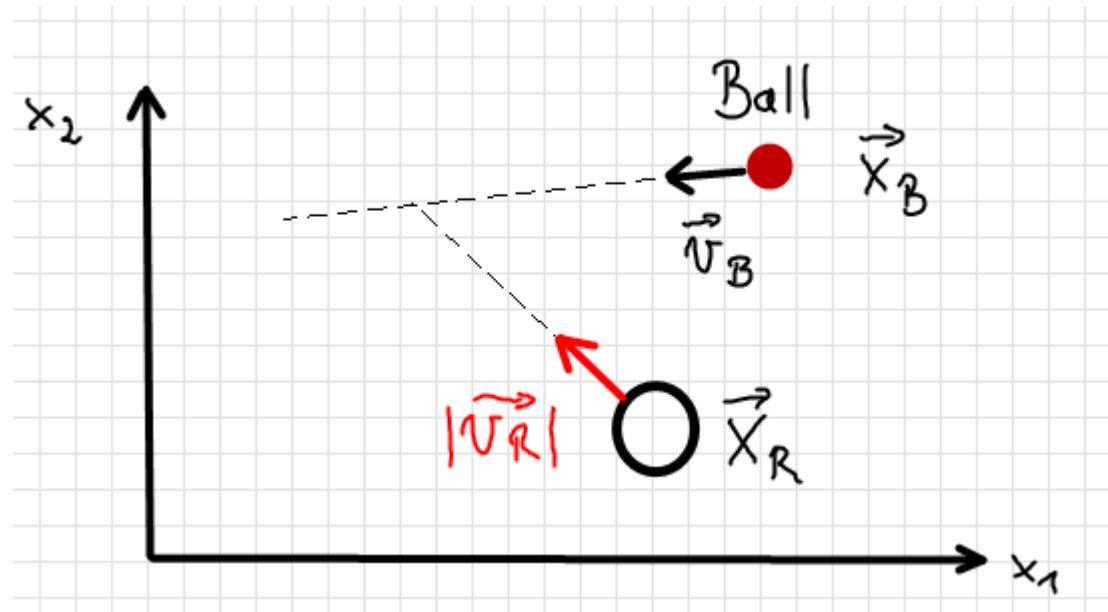
## ÜBUNG: Auffangen eines Balles

### Gegeben:

- Ort (zum Zeitpunkt  $t=0$ ), Geschwindigkeit und Richtung eines Balles
- Ort (zum Zeitpunkt  $t=0$ ) und Maximalgeschwindigkeit eines Fußballroboters

### Gesucht:

- Richtung des Roboters, um den Ball schnellstmöglich abzufangen





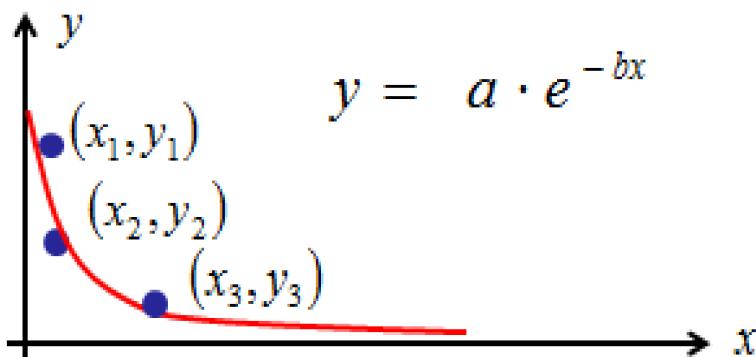
## ÜBUNG: Parameterbestimmung durch iterative Nullstellensuche

Gegeben ist die Funktion  $y = a \cdot e^{-bx}$

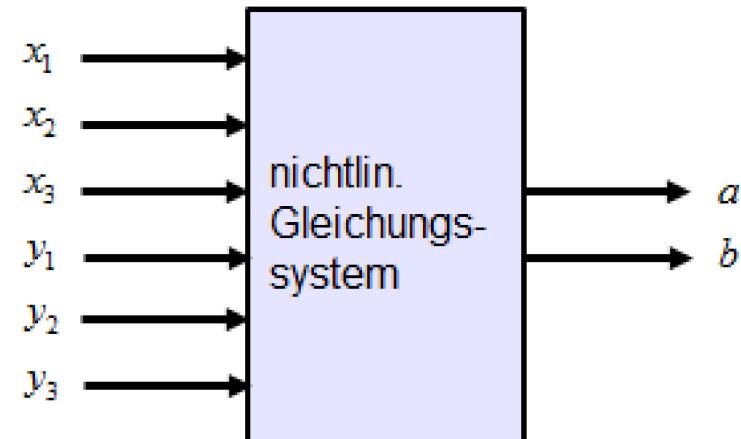
Weiter sind folgende Messwerte (x,y) gegeben: (0.5, 1.1), (1.0, 0.4), (2.0, 0.055)

Die initialen Schätzwerte der Parameter sind:  $a_0 = 4$ ,  $b_0 = 3$

Die Parameter a und b sind zu bestimmen.



$$y = a \cdot e^{-bx}$$





## 5.3 Festlegung des Funktionsblock-Verhaltens durch Regeln

### 5.3.1 Binäre Logik: am Beispiel einer Bremskraftsteuerung

Für ein autonomes Fahrzeug ist ein automatisches Bremsystem zu entwerfen, bei dem die Bremskraft  $F$  in Abhängigkeit von der Geschwindigkeit  $v$  und der Hindernisdistanz  $d$  berechnet wird:

$$F = f(v, d)$$

Es gibt 3 Geschwindigkeitsbereiche:

$$v_{\text{klein}} = 0 \dots 15 \text{ km/h}$$

$$v_{\text{mittel}} = 15 \dots 35 \text{ km/h}$$

$$v_{\text{groß}} = 35 \dots 70 \text{ km/h}$$

Es gibt 3 Entfernungsbereiche:

$$d_{\text{klein}} = 0 \dots 5 \text{ m}$$

$$d_{\text{mittel}} = 5 \dots 15 \text{ m}$$

$$d_{\text{groß}} = 15 \dots 30 \text{ m}$$

Ziel: Die Bremskraft  $F$  soll für die 9 Kombinationen von  $v$  und  $d$  wie im Bild angegeben ausgegeben werden.

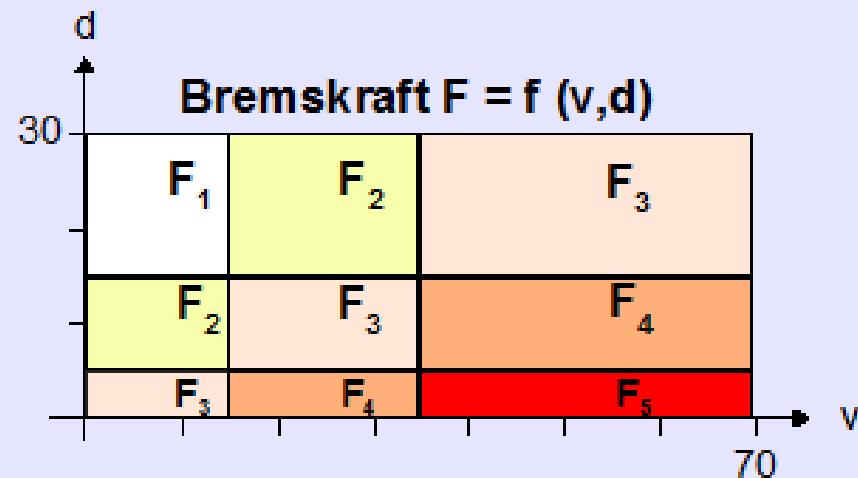
$$F_1 = 1 \text{ kN} \text{ (sehr klein)}$$

$$F_2 = 2 \text{ kN} \text{ (klein)}$$

$$F_3 = 3 \text{ kN} \text{ (mittel)}$$

$$F_4 = 4 \text{ kN} \text{ (groß)}$$

$$F_5 = 5 \text{ kN} \text{ (sehr groß)}$$





## Programmierte Regeln der Bremskraftsteuerung

```
enum Geschwindigkeit {VKLEIN=1, VMITTEL, VGROSS};  
enum Distanz {DKLEIN=1, DMITTEL, DGROSS };
```

### // Zuordnung des Eingangswertes

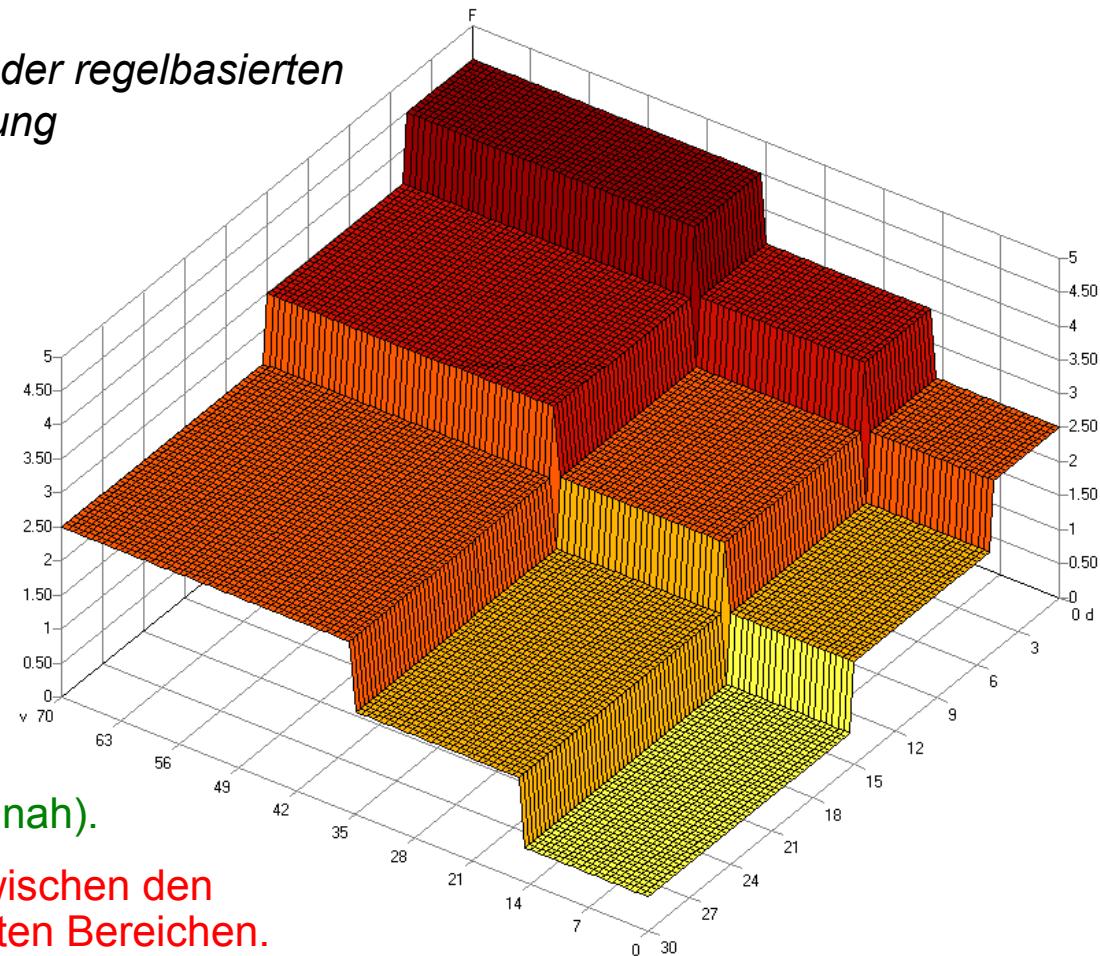
```
if(v_Mess> 0 && v_Mess< 15) then v=VKLEIN;  
if(v_Mess>=15 && v_Mess< 35) then v=VMITTEL;  
if(v_Mess>=35 && v_Mess<=70) then v=VGROSS;  
  
if(d_Mess> 0 && d_Mess< 5) then d=DKLEIN;  
if(d_Mess>= 5 && d_Mess< 15) then d=DMITTEL;  
if(d_Mess>=15 && d_Mess<=30) then d=DGROSS;
```

### // 9 Regeln

```
if(v==VKLEIN && d==DKLEIN) then F_Out=3; //kN  
if(v==VKLEIN && d==DMITTEL) then F_Out=2; //kN  
if(v==VKLEIN && d==DGROSS) then F_Out=1; //kN;  
.....
```



## Funktionsgebirge der regelbasierten Bremskraftsteuerung



### Diskussion:

- (+) Sehr einfach realisierbar.
- (+) Sehr anschaulich (sprachnah).
- (-) Sehr harte Übergänge zwischen den durch die Regeln definierten Bereichen.
- (-) Im Falle hochdimensionaler Eingangsvektoren kann das Aufstellen der Regeln sehr unübersichtlich werden.
- (-) Der Eingangsraum kann nur in (Hyper-)Quader unterteilt werden.



## 5.3.2 Fuzzy-Logic: am Beispiel einer Bremskraftsteuerung

### 5.3.2.1 Erstellungsphase: ... oder ... Wie beschreibt man das Fuzzy-System ?

**Schritt 1:** Bereichszugehörigkeiten und Zugehörigkeitsfunktionen festlegen

Der Eingangswertebereiche für  $v$  und  $d$  werden in 3 überlappende Bereiche unterteilt:

$$M_{v,klein} = \{v \mid v \in [0, 20]\}$$

$$M_{d,klein} = \{d \mid d \in [0, 7]\}$$

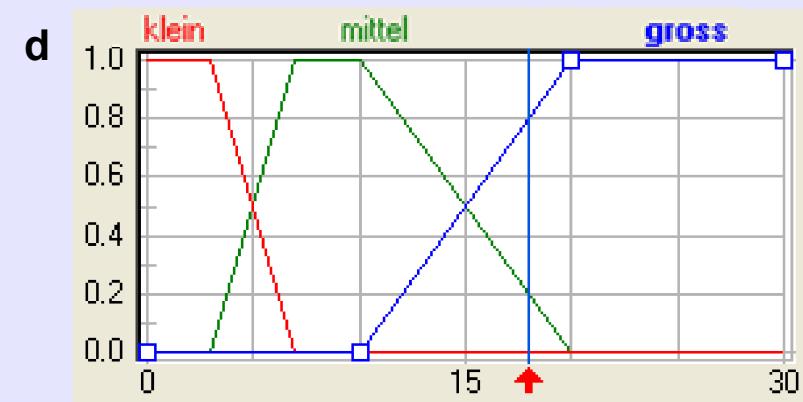
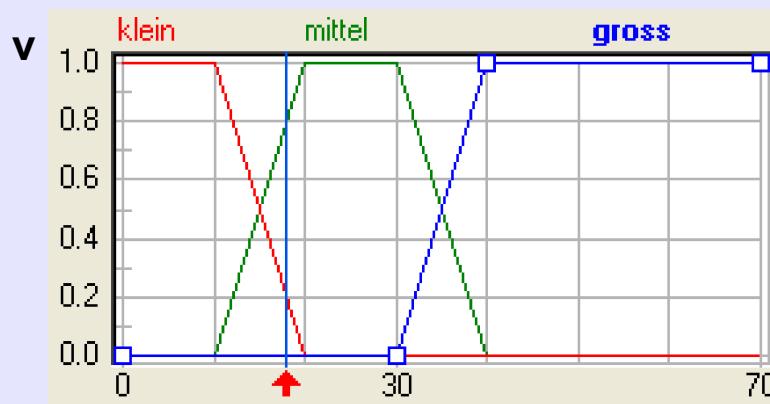
$$M_{v,mittel} = \{v \mid v \in [10, 40]\}$$

$$M_{d,mittel} = \{d \mid d \in [3, 20]\}$$

$$M_{v,groß} = \{v \mid v \in [30, 70]\}$$

$$M_{d,groß} = \{d \mid d \in [10, 30]\}$$

Die Bereichszugehörigkeiten werden durch Zugehörigkeitsfunktionen  $\mu_{i,k}$  festgelegt, z.B.  $\mu_{v,klein}$  oder  $\mu_{d,mittel}$ .





## Schritt 2: Festlegung der Regeln

..... für alle Teilmengenkombinationen der Grundmengen ( $M_v$ ,  $M_d$ ).

WENN  $v \in M_{v,groß}$

UND  $d \in M_{d,klein}$

DANN  $F = F_{sehr\ groß}$

WENN  $v \in M_{v,groß}$

UND  $d \in M_{d,mittel}$

DANN  $F = F_{groß}$

WENN  $v \in M_{v,groß}$

UND  $d \in M_{d,klein}$

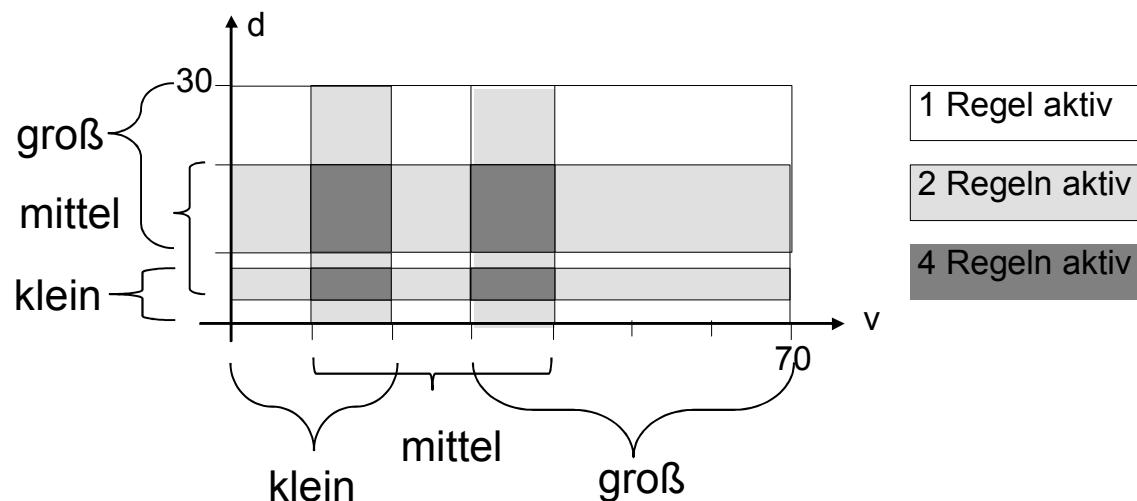
DANN  $F = F_{mittel}$

WENN  $v \in M_{v,klein}$

UND  $d \in M_{d,klein}$

DANN  $F = F_{mittel}$

..... usw.

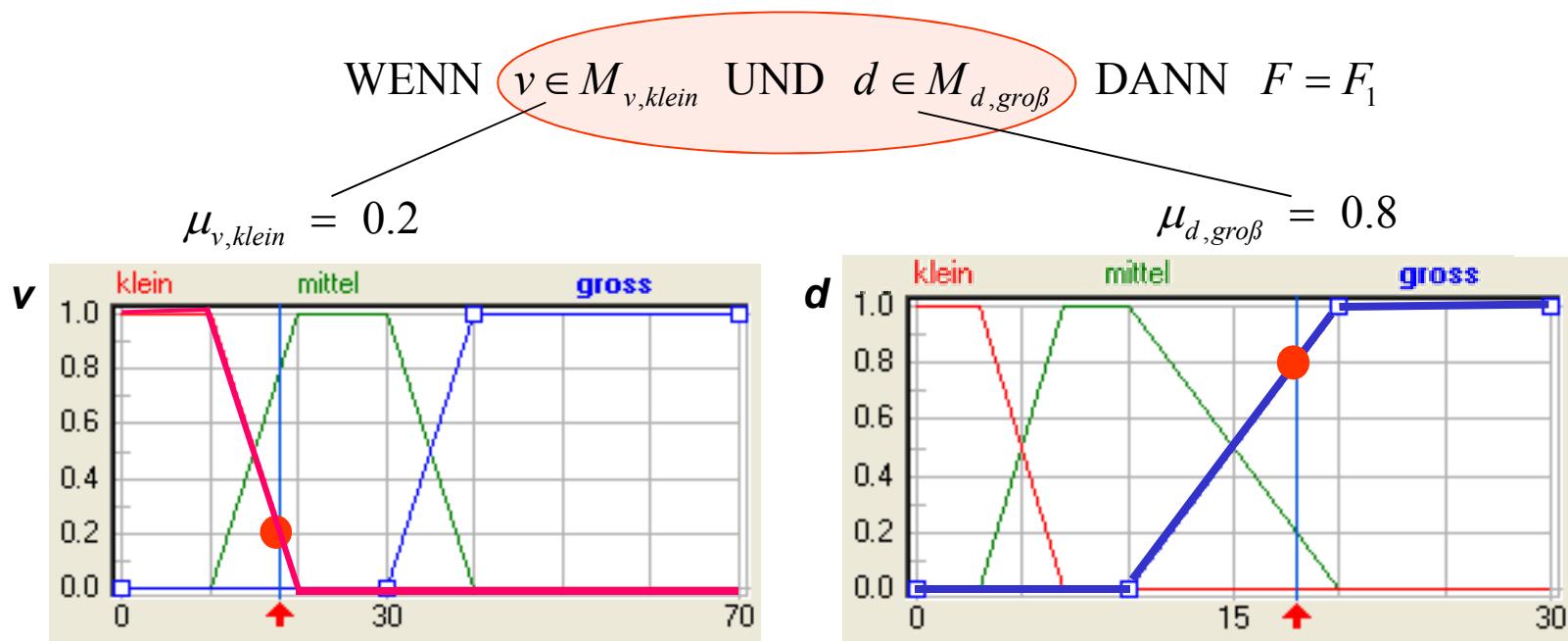


Je nach Eingangsvektor können auch **mehrere Regeln gültig sein !**



### 5.3.2.2 Arbeitsphase : ... oder ... Wie **arbeitet** das Fuzzy-System ?

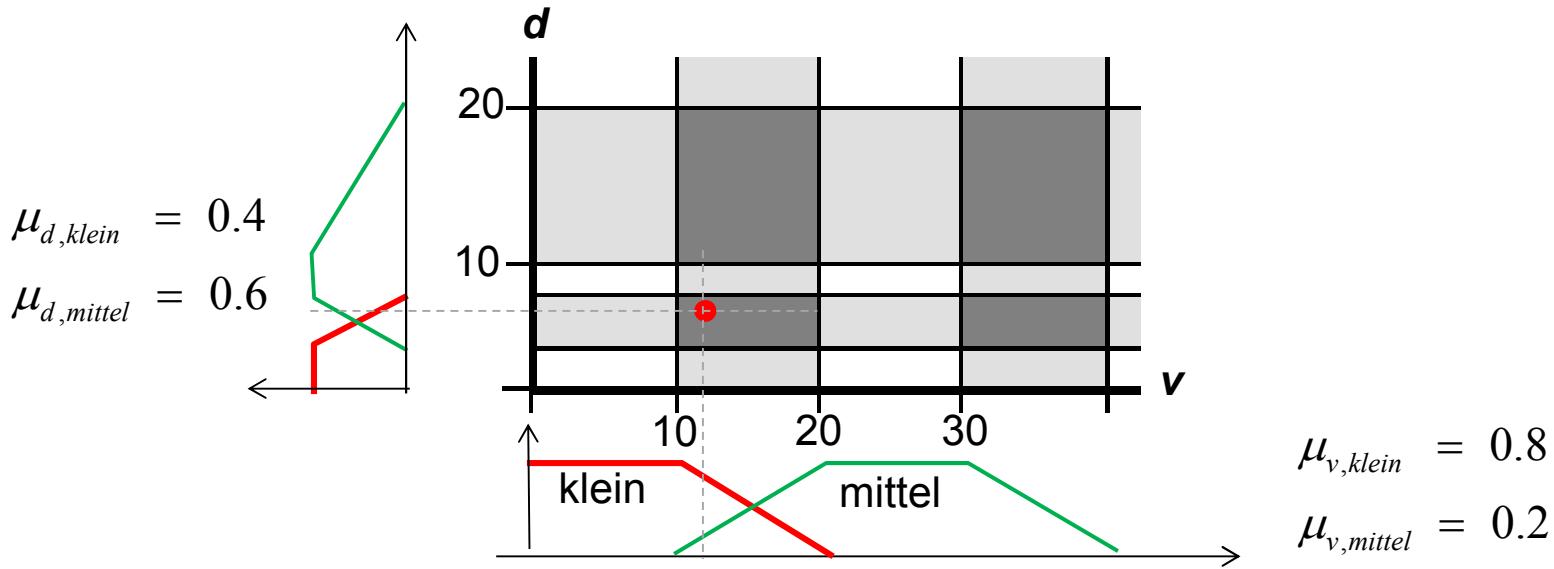
**Schritt 1:** Eingangsaggregation: **Erfüllungsgrad  $\varepsilon$  der Regeln berechnen**



Ein „*Standardverfahren*“ (es gibt auch andere) zur Berechnung des Regel-Erfüllungsgrades  $\varepsilon$  ist es, das Minimum der beteiligten Funktionswerte zu nehmen.  
Hier z.B. :  $\varepsilon_{kg} = \min(0.2, 0.8) = 0.2$  ( $v \rightarrow klein, d \rightarrow groß$ )



Es können auch mehrere Regeln mit unterschiedlichen Erfüllungsgraden erfüllt sein.





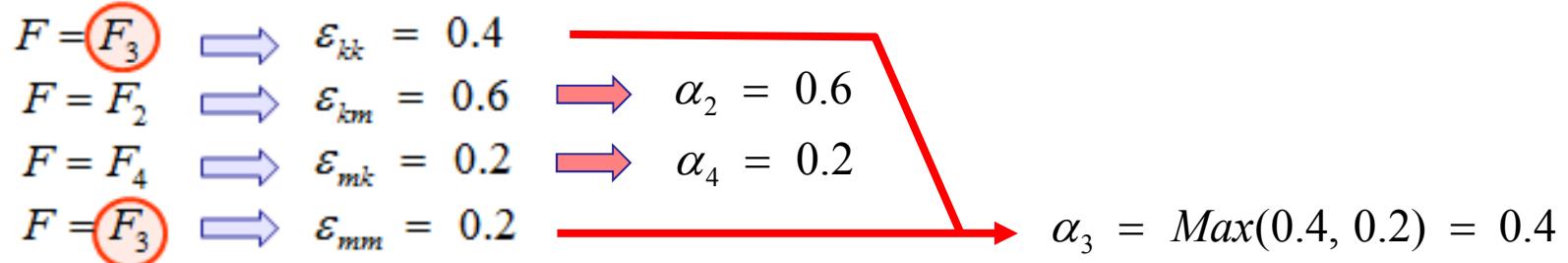
## Schritt 2: Ausgangsaggregation: Erfüllungsgrad $\alpha$ der Ausgangsgrößen bestimmen

WENN  $v \in M_{v,klein}$  UND  $d \in M_{d,klein}$   
 WENN  $v \in M_{v,klein}$  UND  $d \in M_{d,mittel}$   
 WENN  $v \in M_{v,mittel}$  UND  $d \in M_{d,klein}$   
 WENN  $v \in M_{v,mittel}$  UND  $d \in M_{d,mittel}$

DANN  $F = F_3 \rightarrow \varepsilon_{kk} = 0.4$   
 DANN  $F = F_2 \rightarrow \varepsilon_{km} = 0.6$   
 DANN  $F = F_4 \rightarrow \varepsilon_{mk} = 0.2$   
 DANN  $F = F_3 \rightarrow \varepsilon_{mm} = 0.2$

3 Kräfte mit unterschiedlichen Erfüllungsgraden

mehrdeutig



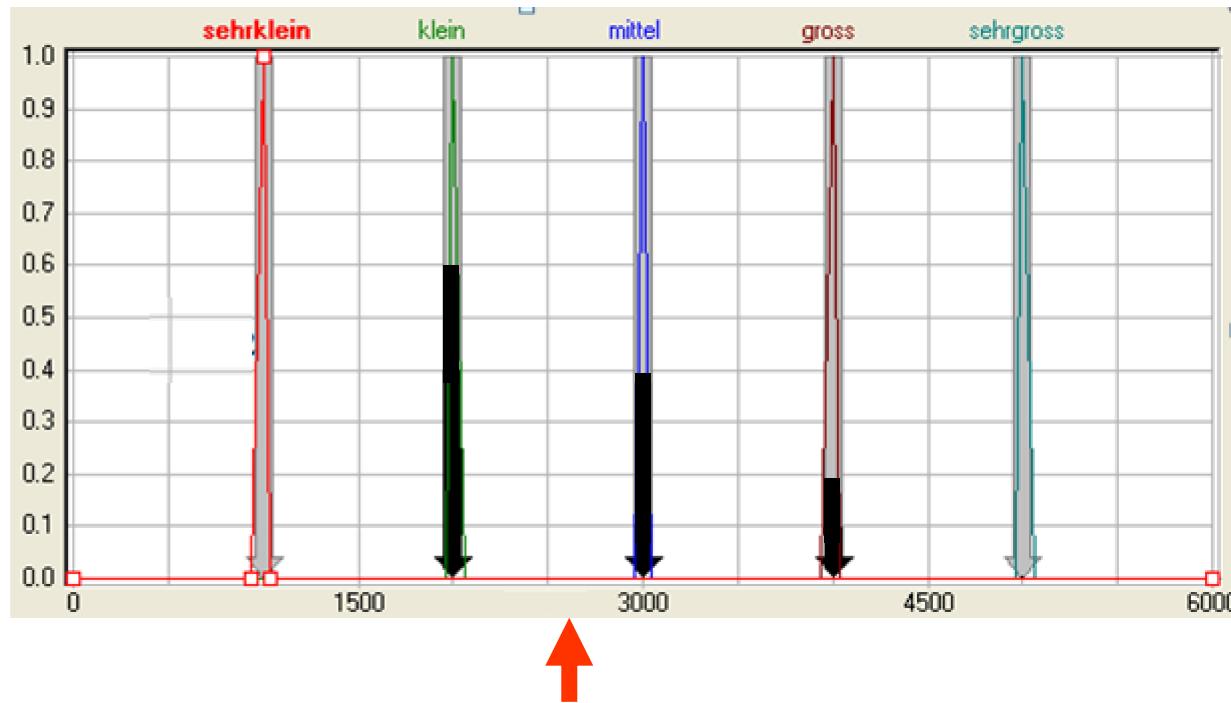
Sonderfall: Mehrere Regeln einer Ausgangsgröße (hier z.B.  $F_3$ ) haben unterschiedliche Erfüllungsgrade.

Lösung: Ein „Standardverfahren“ zur Lösung des Widerspruchs ist es, das Maximum der Erfüllungsgrade dieser Ausgangsgröße zu nehmen.



### Schritt 3: Defuzzifizierung (hier nach der sog. „Singleton-Methode“)

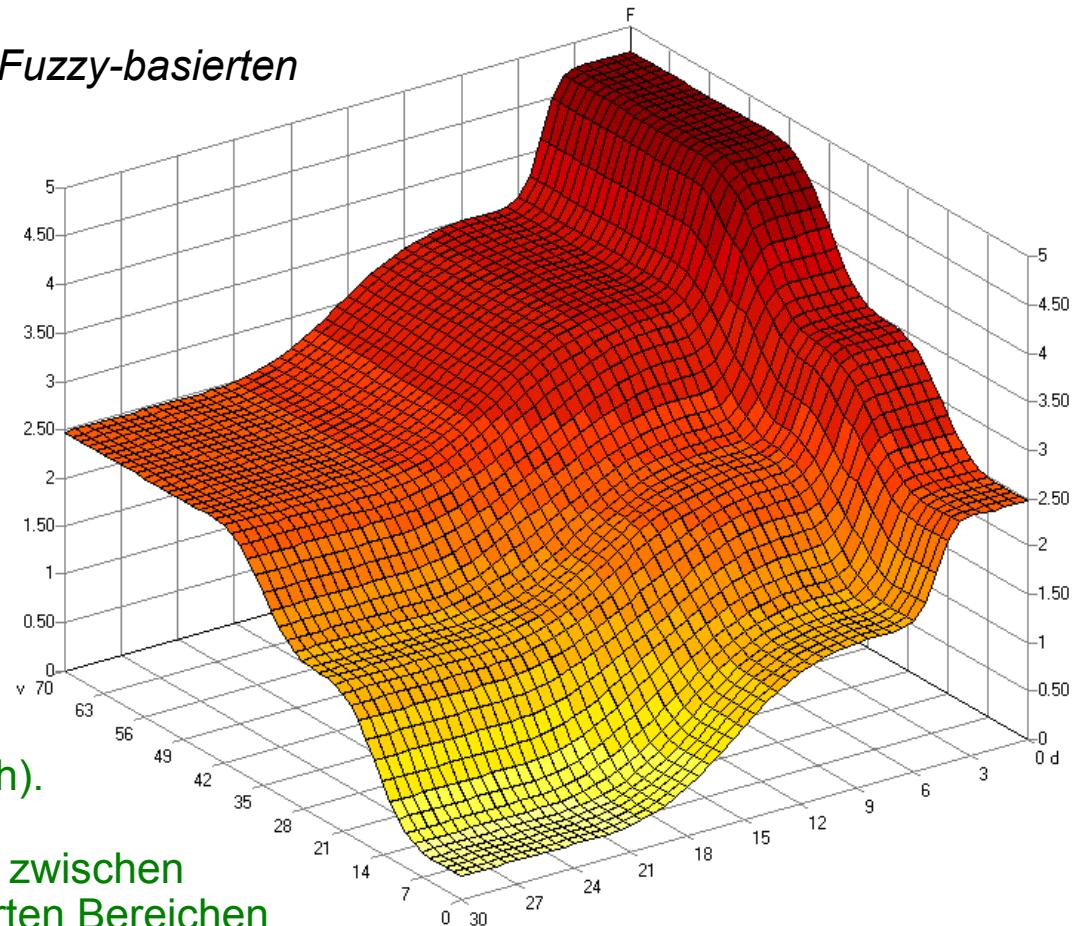
Die beteiligten Ausgangsgrößen ( $F_1, F_2, \dots, F_5$ ) = (1kN, 2kN, ..., 5kN) werden entsprechend ihrem Erfüllungsgrad  $\alpha_{ik}$  gewichtet.



$$F = \frac{0.6 \cdot 2kN + 0.4 \cdot 3kN + 0.2 \cdot 4kN}{0.6 + 0.4 + 0.2} = 2667N$$



## Funktionsgebirge der Fuzzy-basierten Bremskraftsteuerung



### Diskussion:

- (+) Sehr anschaulich (sprachnah).
- (+) Die Glattheit der Übergänge zwischen den von den Regeln definierten Bereichen kann eingestellt werden.
- (-) Im Falle hochdimensionaler Eingangsvektoren kann das Aufstellen der Regeln sehr unübersichtlich werden.
- (-) Der Eingangsraum kann nur in unscharfe (Hyper-)Quader unterteilt werden.



## 5.4 Festlegung des Funktionsblock-Verhaltens durch Beispiele (= Lernen)

### 5.4.1 Einführende Gedanken

#### 5.4.1.1 Was ist Lernen ?

##### **Voraussetzungen:**

Der funktionale Zusammenhang zwischen Ursache (Eingangsgröße) und Wirkung (Ausgangsgröße) ist nicht genau bekannt, d.h.

→ keine Funktionen, Gleichungen oder Regeln bekannt

Es gibt eine genügende Anzahl von Beispielen (Messwerte), die den Zusammenhang zwischen Eingangs- und Ausgangsgröße widerspiegeln.

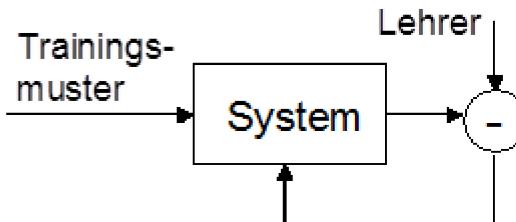
Lernen = Adaption des Funktionsblock-Verhaltens anhand von Beispielen .



### 5.4.1.2 Lernsituationen

#### Überwachtes Lernen

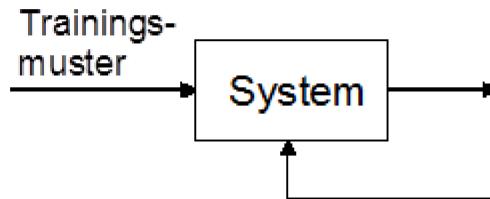
= "supervised learning"  
= Lernen mit Lehrer



Ein Lehrer sagt, was die richtige Ausgabe gewesen wäre.

#### Unüberwachtes Lernen

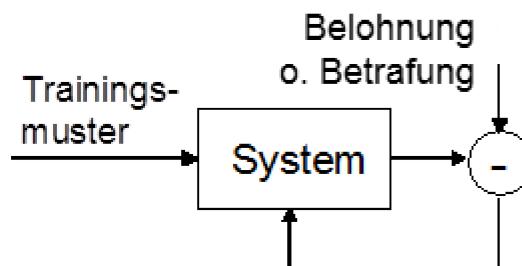
= "unsupervised learning"  
= Lernen ohne Lehrer



Das System bemerkt selbst, dass es unterschiedliche Eingangsklassen gibt.

#### Bestärkendes Lernen

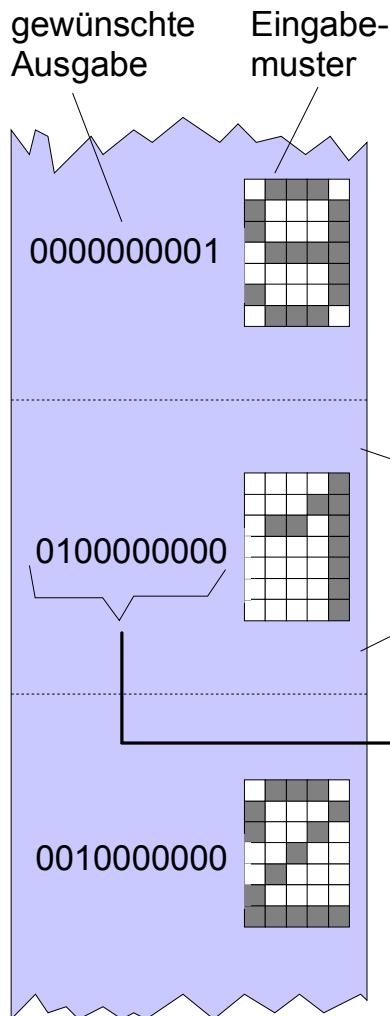
= "reinforcement learning"  
→ Lernen durch Erfahrung



Der Lehrer (Leben / Umgebung) belohnt oder bestraft.

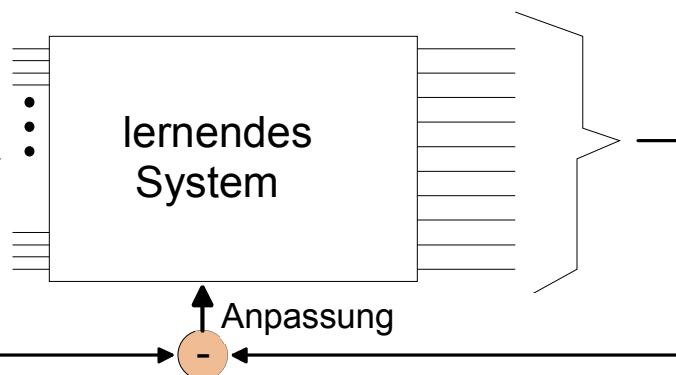


## Beispiel: "überwachtes Lernen," am Beispiel des Zeichenklassifikators



**Prinzip:** Während des Lernvorganges wird dem lernenden System der Reihe nach

- ein Muster (=*Trainingsdaten*) sowie
- die gewünschte Ausgabe (= "*teaching input*") vorgelegt.



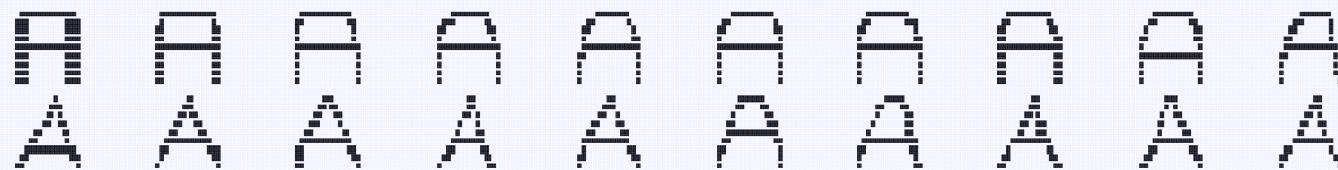
Die Differenz zwischen der Netzausgabe und der gewünschten Ausgabe wird zur Anpassung des Systems verwendet.



### 5.4.1.3 Erwartungen an das angelernte System

1. Das Systemverhalten wird anhand repräsentativer Beispiele (= *Trainingsdaten, Repräsentanten*) trainiert.
2. Das angelernte System erkennt nicht nur die Repräsentanten, sondern auch Varianten davon. → Generalisierungsfähigkeit

*Trainingsdaten :*



*soll z.B. auch erkennen :*





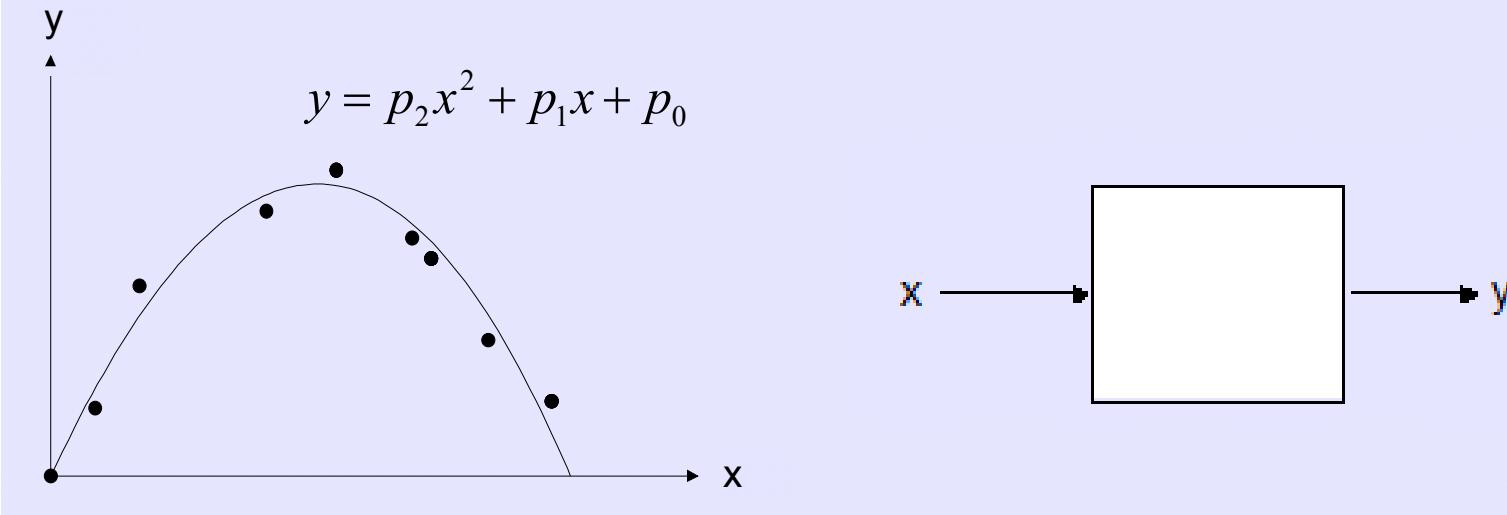
## 5.4.2 Mathematische Perspektive

### 5.4.2.1 Bekanntes Modell - unbekannte Parameter

**Beispiel:** Von einem physikalischen Prozess sei bekannt, daß er parabelförmig verläuft (z.B. Wurfparabel).

Weiter seien n Meßwertpaare  $(x_i, y_i)$  bekannt.

Zu berechnen: Die unbekannten Parameter  $p_0, p_1, p_2$ .



Das Systemverhalten wird anhand von Beispielen festgelegt.  
→ Interpolation oder Approximation = Lernen



## Lösungsansätze

Fall: wenige unbekannte Parameter

→ lineare bzw. nichtlineare **Ausgleichsrechnung**

Fall: sehr viele unbekannte Parameter

→ **Gradientenabstiegsmethode** effizienter (s.u.)

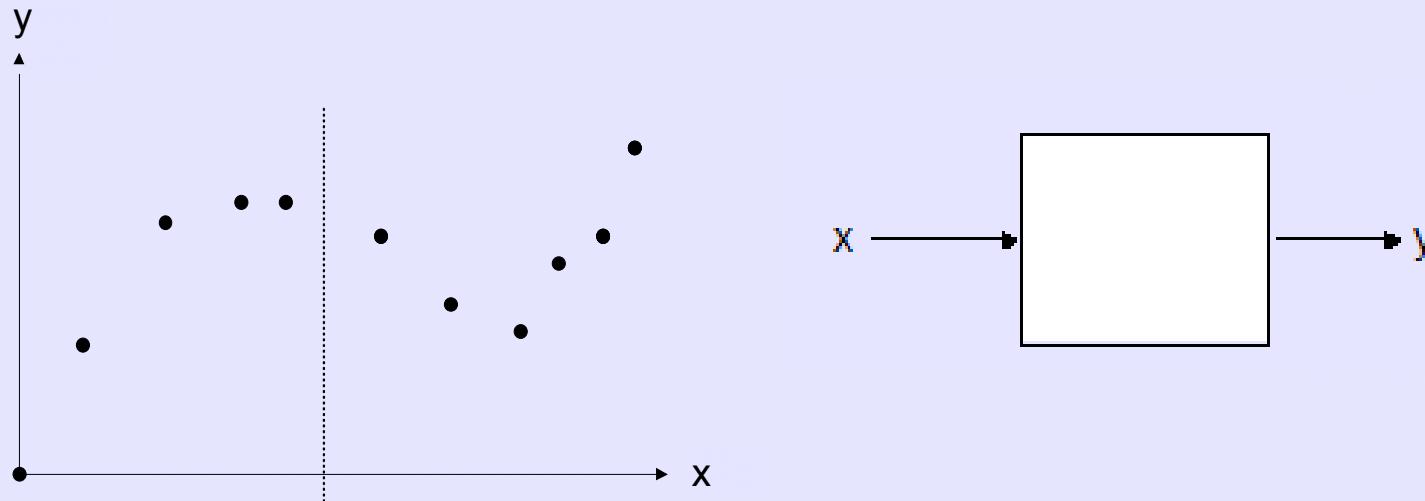


### 5.4.2.2 Unbekanntes Modell

**Beispiel:** Von einem Prozess sei unbekannt, nach welcher Funktion er verläuft.

Weiter seien n Meßwertpaare  $(x_i, y_i)$  bekannt.

Zu berechnen: Die y-Werte für solche x, für die es kein Meßwert gibt.



Das Systemverhalten wird anhand von Beispielen festgelegt.

→ Interpolation oder Approximation = Lernen



## Lösungsansätze

1. Geeignete Funktionen für die Approximation / Interpolation auswählen, z.B.
  - lineare oder radiale Basisfunktionen (s.u.),
  - Polynome
  - ..... usw.
2. Bestimmung der unbekannten Funktionsparameter durch

Fall: wenige unbekannte Parameter

→ lineare bzw. nichtlineare **Ausgleichsrechnung**

Fall: sehr viele unbekannte Parameter

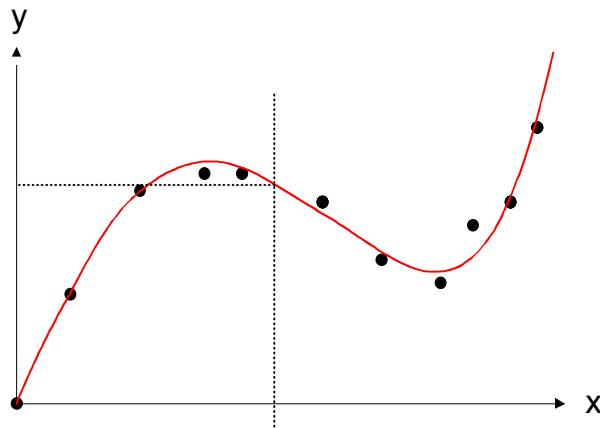
→ **Gradientenabstiegsmethode** effizienter (s.u.)



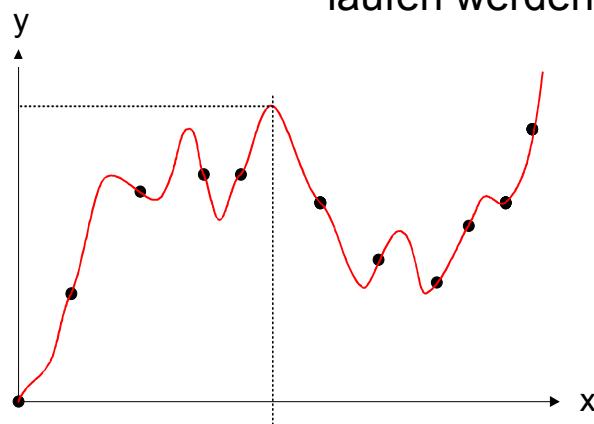
### 5.4.2.3 Interpolation oder Approximation

**Approximation:** Suche einer Funktion in der Art, dass

- a) eine hinreichend glatte Funktion entsteht
- b) die Meßpunkte „bestmöglich“ eingepasst sind.

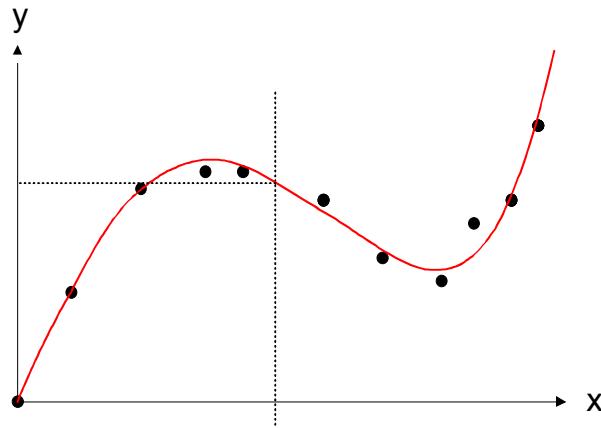


**Interpolation:** Suche einer Funktion in der Art, dass die Messpunkte exakt durchlaufen werden (z.B. für  $n+1$  Messpunkte ein Polynom  $n$ -ten Grades).

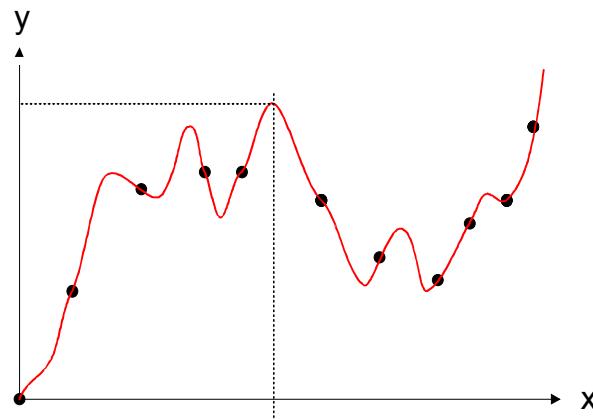




#### 5.4.2.4 Approximationsgüte zwischen den Messpunkten



→ gute "Generalisierung"



→ schlechte "Generalisierung"



## 5.4.3 Ad-hoc-Interpolations-/Approximationsverfahren

### 5.4.3.1 Vorüberlegungen

Die Lernbeispiele (Repräsentanten, Trainingsdaten, Sichproben) legen unmittelbar (ohne Training) das Verhalten des Funktionsblocks fest.

**Vorteil:** - sehr einfach realisierbar

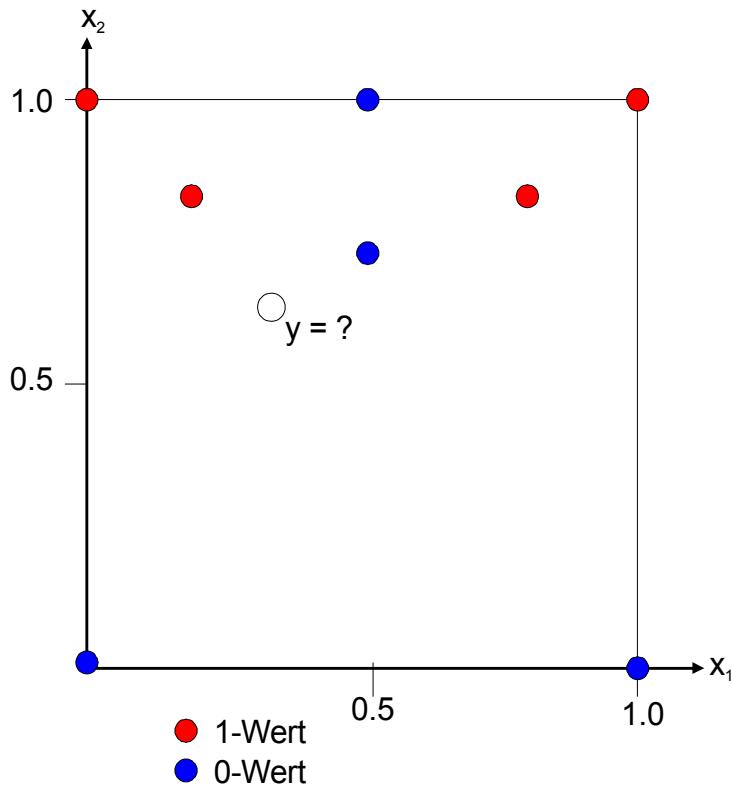
**Nachteil:** - auf einfache Fälle beschränkt (niedrigdimensionale Eingangsvektoren, kleine Stichprobenmengen)



### 5.4.3.2 Nearest-neighbour - Klassifikator

**Beispiel:** Zu entwerfen ist ein Funktionsblock  $y_1 = f(x_1, x_2)$ .  
Gegeben sind 8 Repräsentanten (*Trainingswerte*).

Wie lassen sich die Zwischenwerte bestimmen ?



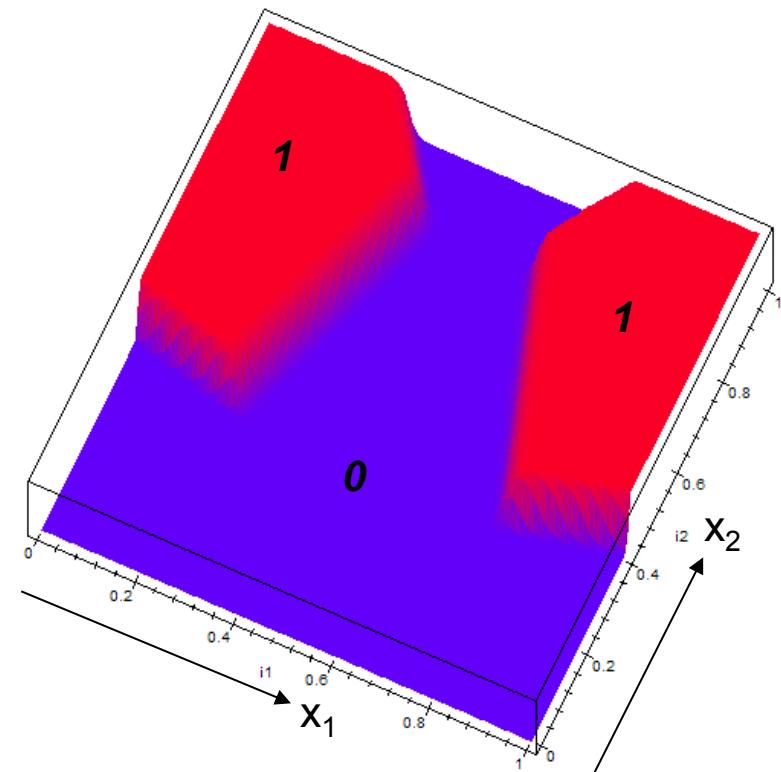
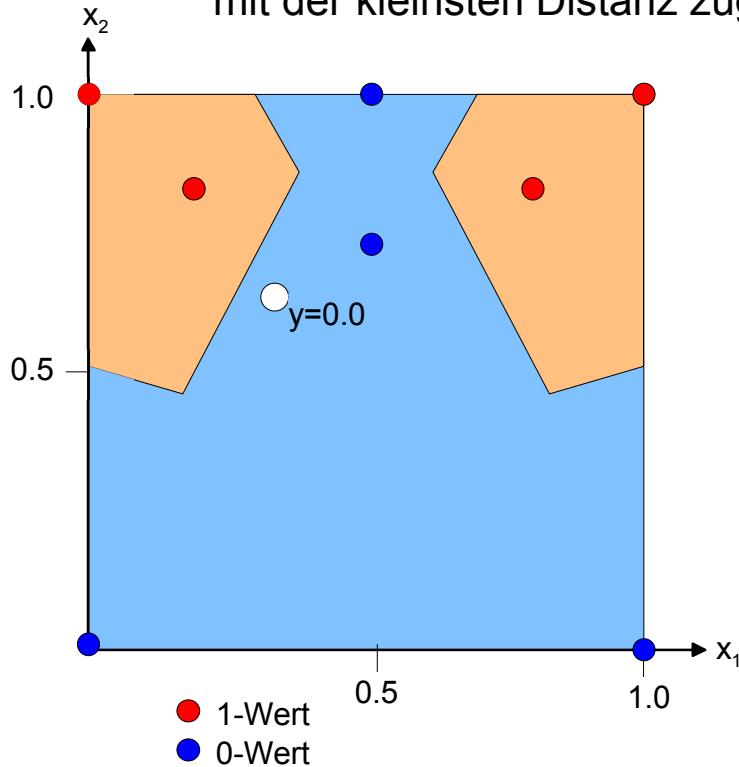
| $x_1$ | $x_2$ | $y$ |
|-------|-------|-----|
| 0.00  | 0.00  | 0.0 |
| 0.00  | 1.00  | 1.0 |
| 0.20  | 0.80  | 1.0 |
| 0.50  | 1.00  | 0.0 |
| 0.50  | 0.75  | 0.0 |
| 0.80  | 0.80  | 1.0 |
| 1.00  | 1.00  | 1.0 |
| 1.00  | 0.00  | 0.0 |





## Funktionsprinzip:

Einem beliebigen Punkt  $(x_1, x_2)$  wird der Wert  $y_1$  des Repräsentanten mit der kleinsten Distanz zugewiesen (euklidische Distanz).





## ÜBUNG: Beispielanwendungen NN-Klassifikator/NN-Interpolator

Diskutieren Sie die folgenden Anwendungsbeispiele:

- a) Funktionsinterpolation  $y=f(x)$
- b) Farbinspektion (z.B. Qualitätskontrolle von Keramikkacheln)



## Eigenschaften:

- (+) naive Suche des nächsten Nachbarn sehr einfach realisierbar
- (+) nicht wie regelbasierte Verfahren auf Hyperquader festgelegt
- (-) sprungförmige Übergänge zwischen den Repräsentanten
- (-) naive Suche bei vielen Repräsentanten oder hochdimensionalen Eingangsvektoren sehr ineffizient

## Verbesserungen und Erweiterungen : (Schlagworte)

- Aussortieren der nichtrelevanten Repräsentanten (z.B. Gabriel Graph)
- effizientere Suche durch Unterteilung des Raumes mit Baumsuche
  - k-d-Tree für niedrigdimensionale Suchräume
  - approximate k-d-Tree für hochdimensionale Suchräume

## Bibliotheken verfügbar:

- ANN (C++), FANN (C++, Python, Matlab), .....



### 5.4.3.3 Approximation durch radiale Basisfunktionen vom Gauß-Typ (Ad-hoc-Ansatz)

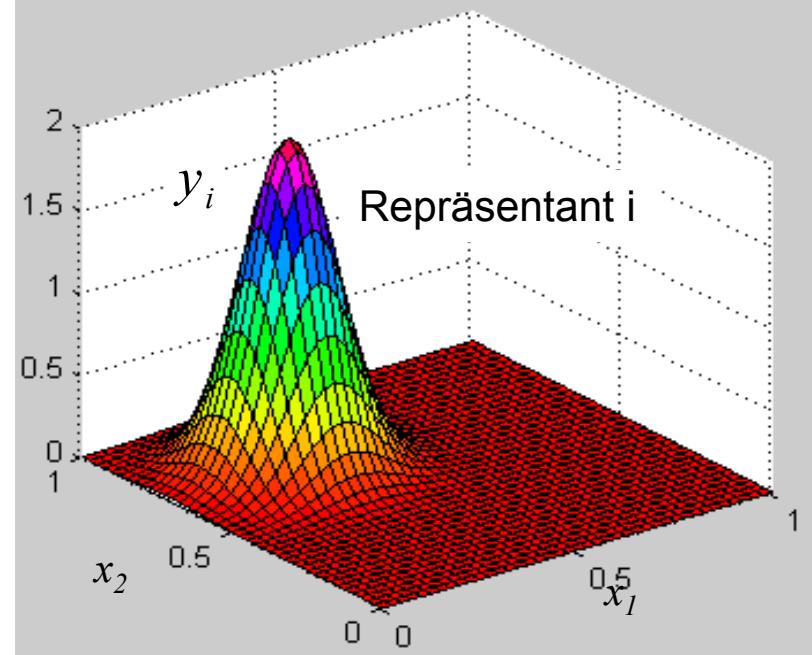
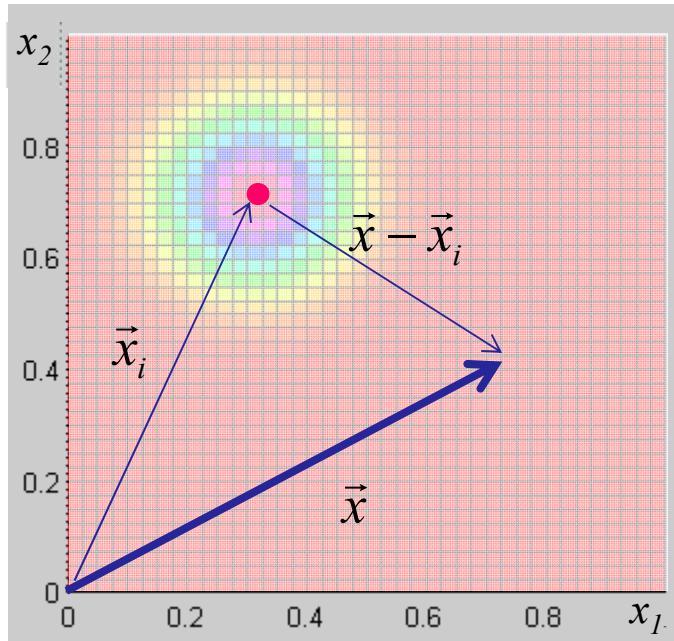
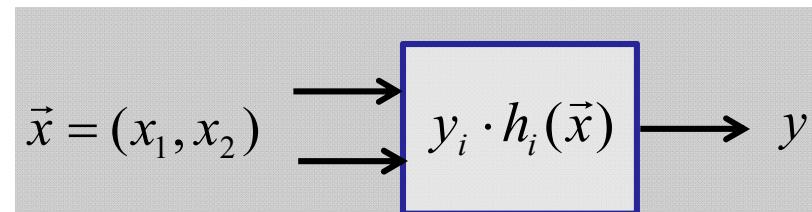
Pro Repräsentant  $i$  gibt es eine Teilfunktion :

$\vec{x}_i$  : Ort des Repräsentanten  $i$

$y_i$  : Wert des Repräsentanten  $i$

$\vec{x}$  : Eingangsvektor

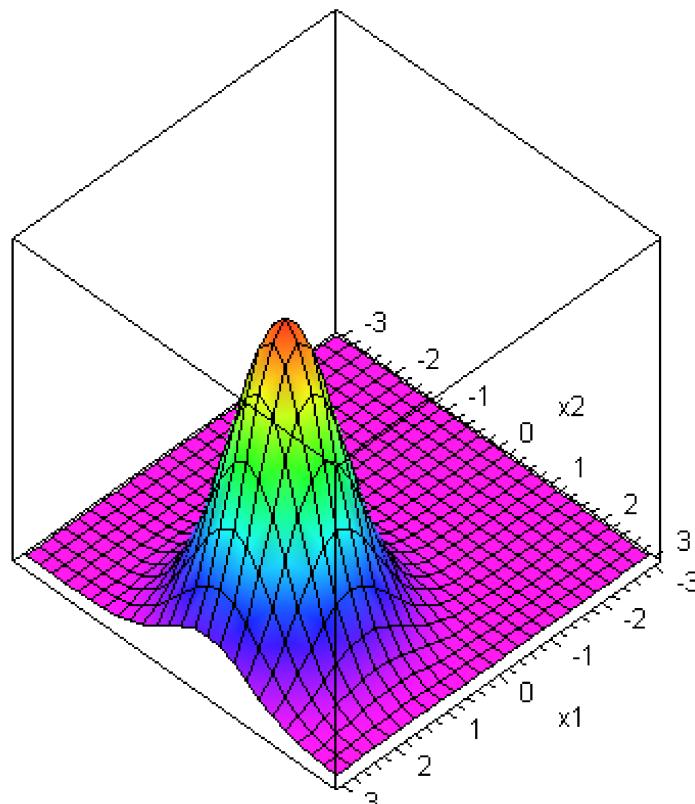
$$h_i(\vec{x}) = e^{-\frac{(\vec{x} - \vec{x}_i)^2}{2\sigma^2}}$$



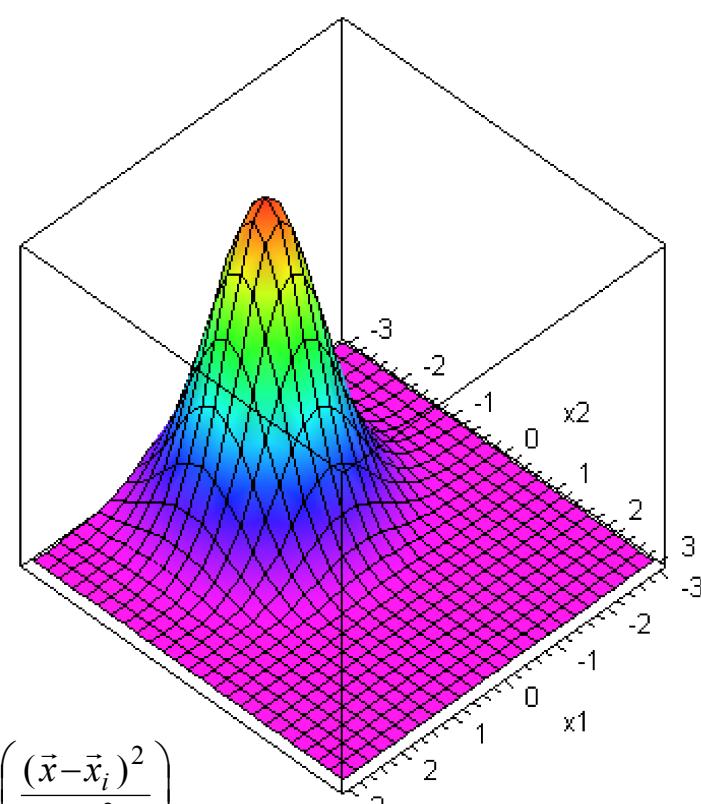


## BEISPIEL: Ort der Repräsentanten i

$$\vec{x}_i = (x_{i1}, x_{i2})^T = (1.5, 0.5)^T$$
$$\sigma = 1$$



$$\vec{x}_i = (x_{i1}, x_{i2})^T = (0.0, -1.5)^T$$
$$\sigma = 1$$

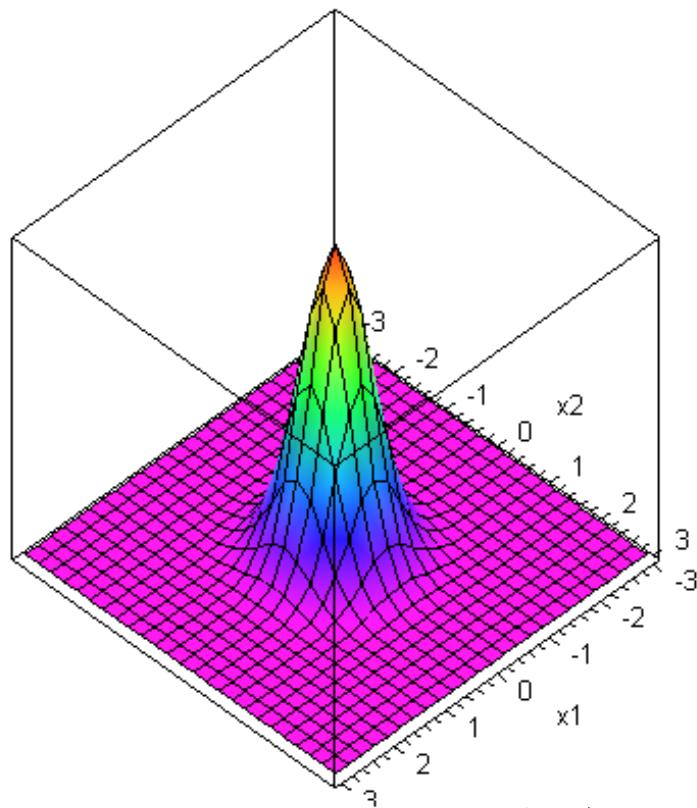


$$h(\vec{x}) = 1 \cdot e^{-\left( \frac{(\vec{x} - \vec{x}_i)^2}{2\sigma^2} \right)}$$

BEISPIEL: Einfluss von  $\sigma$ 

$$\vec{x}_i = (x_{i1}, x_{i2})^T = (0.0, 0.0)^T$$

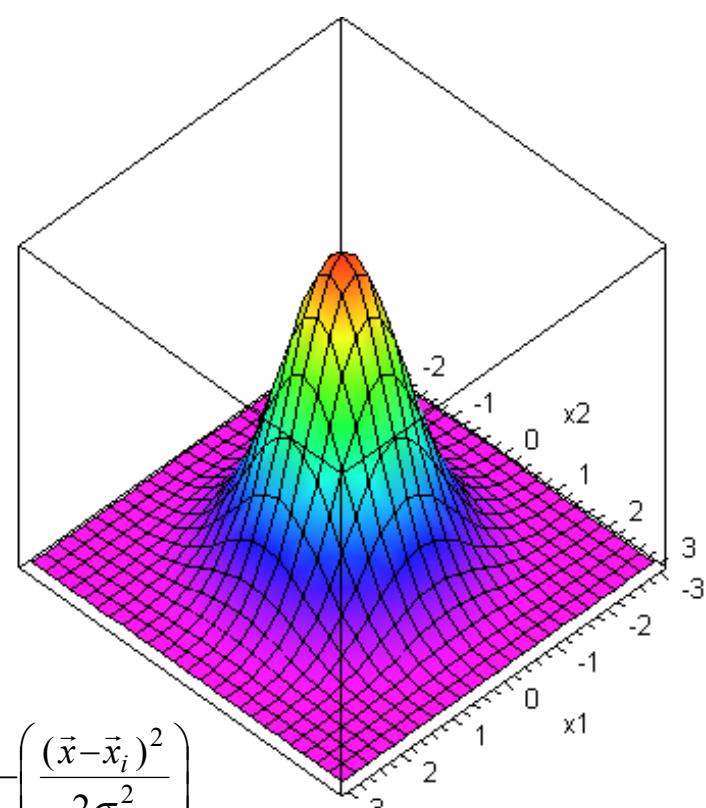
$$\sigma = 0.5$$



$$h(\vec{x}) = 1 \cdot e^{-\left(\frac{(\vec{x}-\vec{x}_i)^2}{2\sigma^2}\right)}$$

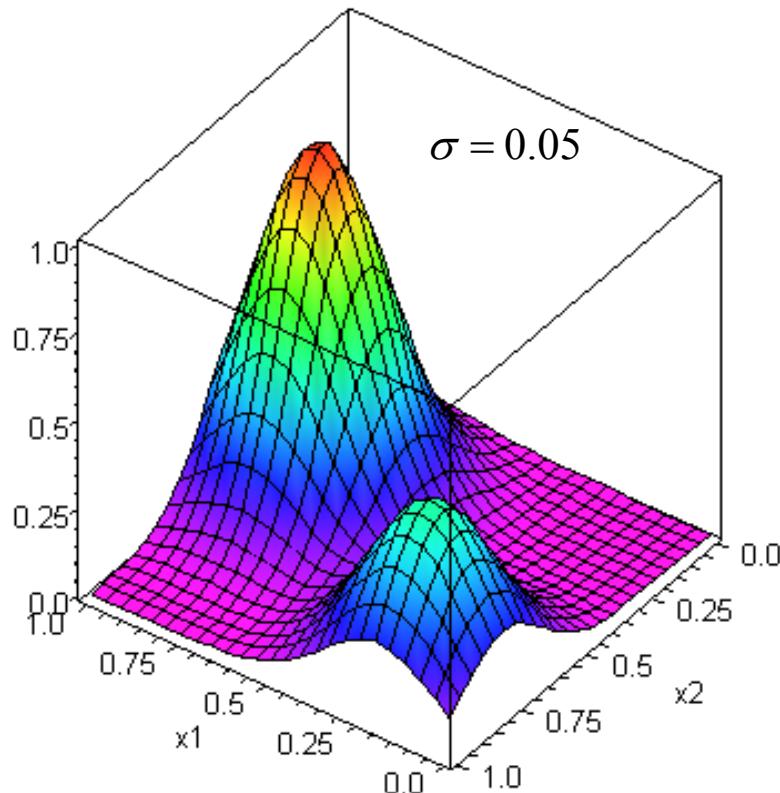
$$\vec{x}_i = (x_{i1}, x_{i2})^T = (0.0, 0.0)^T$$

$$\sigma = 1.5$$





## BEISPIEL: Summe mehrerer gewichteter Basisfunktionen



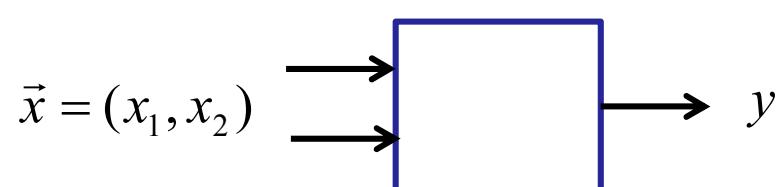
### 2 Repräsentanten

$$\vec{x}_1 = (x_{1,1}, x_{1,2})^T = (0.2, 0.8)^T$$

$$y_1 = 0.5$$

$$\vec{x}_2 = (x_{2,1}, x_{2,2})^T = (0.8, 0.4)^T$$

$$y_2 = 1.0$$



$$y(\vec{x}) = 0.5 \cdot e^{-\left(\frac{(\vec{x}-\vec{x}_1)^2}{2\sigma^2}\right)} + 1.0 \cdot e^{-\left(\frac{(\vec{x}-\vec{x}_2)^2}{2\sigma^2}\right)}$$

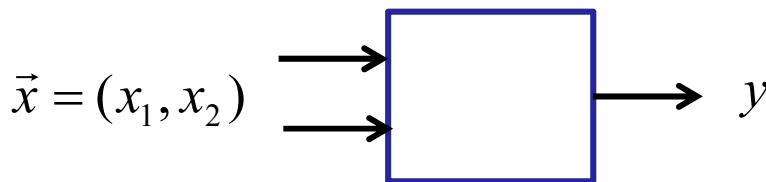


## ÜBUNG: Gauss-RBF

Welchen Wert hat die Funktion  $r(\vec{x})$  bei den Eingangswerten:  $\vec{x} = (0.1, 0.1)$

$$\vec{x} = (0.2, 0.6)$$

$$\vec{x} = (0.3, 0.7)$$



$$r(\vec{x}) = y_1 \cdot h_1(\vec{x}) = y_1 \cdot e^{-\frac{(\vec{x}-\vec{x}_1)^2}{2\sigma^2}}$$

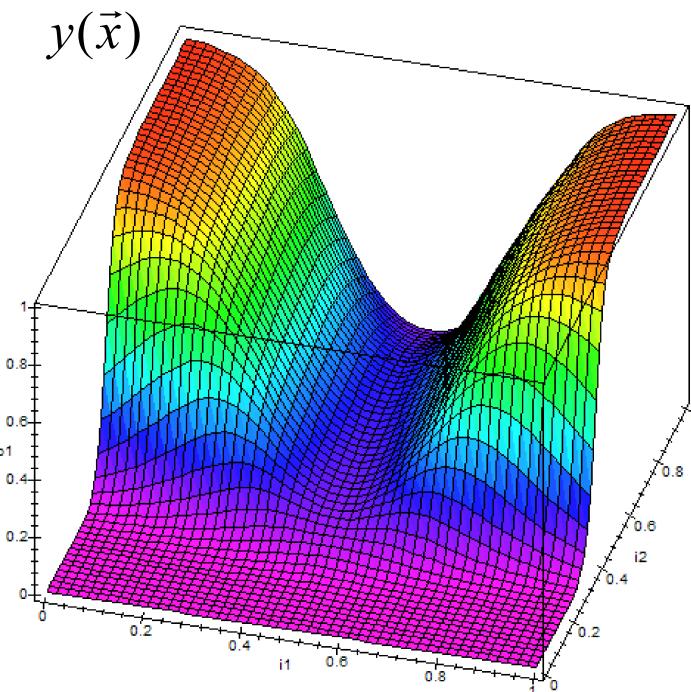
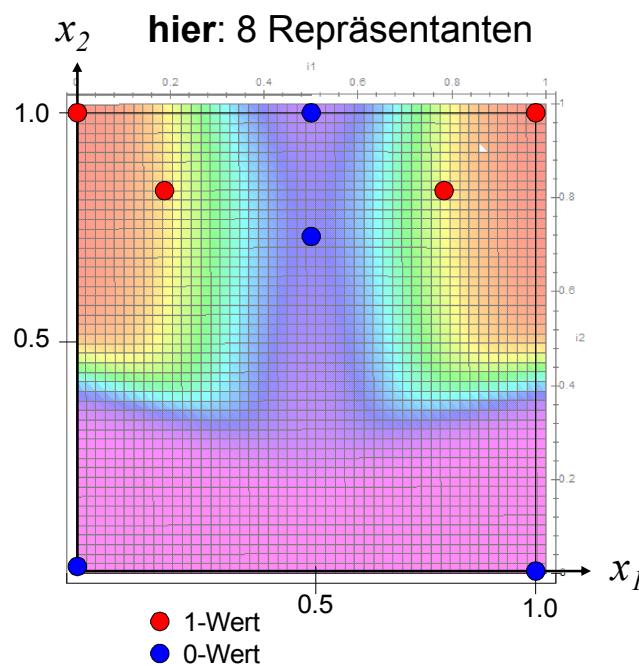
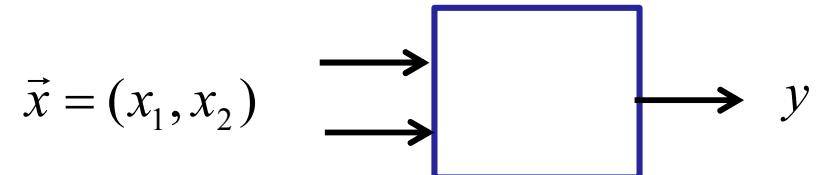
Parameter des  
Repräsentanten i

$$\begin{aligned}\vec{x}_1 &= (0.3, 0.7) \\ y_1 &= 2 \\ \sigma &= 0.1\end{aligned}$$



Die Approximationsfunktion wird mit Hilfe der pro Repräsentant erzeugten Teilstrukturen wie folgt zusammengesetzt:

$$y(\vec{x}) = \frac{\sum_i [ y_i \cdot h_i(\vec{x}) ]}{\sum_i h_i(\vec{x})}$$

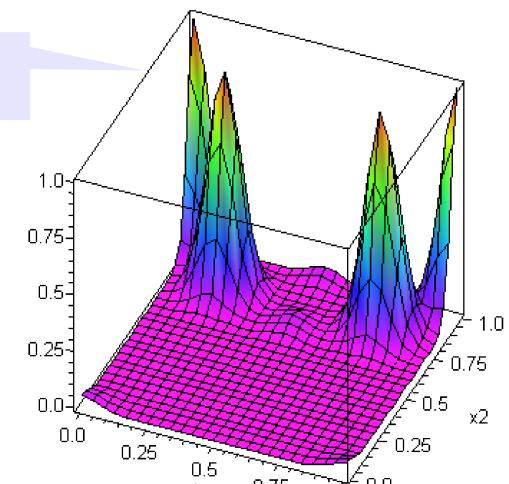
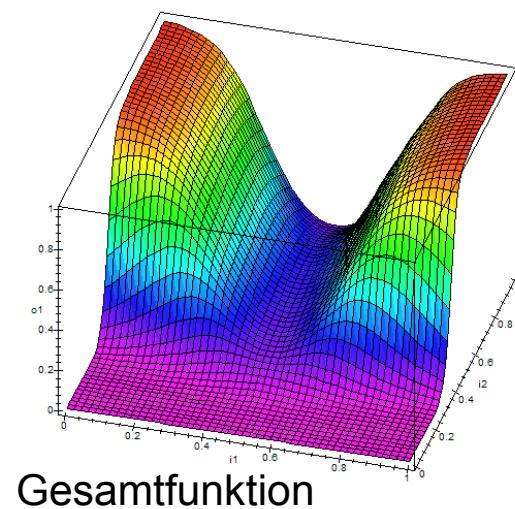
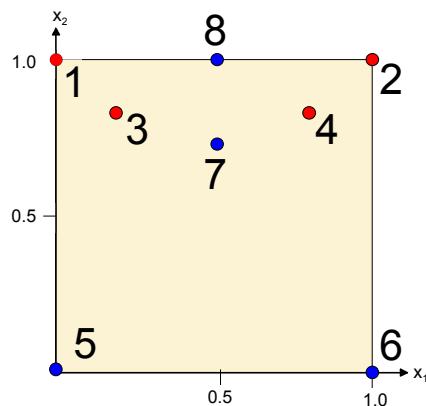




## Realisierung des Beispiels im Detail

$$\begin{aligned}
 h_1(x_1, x_2) &= e^{-\left(\frac{(x_1-0.0)^2}{2\sigma^2} + \frac{(x_2-1.0)^2}{2\sigma^2}\right)}, & h_2(x_1, x_2) &= e^{-\left(\frac{(x_1-1.0)^2}{2\sigma^2} + \frac{(x_2-1.0)^2}{2\sigma^2}\right)}, & h_3(x_1, x_2) &= e^{-\left(\frac{(x_1-0.2)^2}{2\sigma^2} + \frac{(x_2-0.8)^2}{2\sigma^2}\right)} \\
 h_4(x_1, x_2) &= e^{-\left(\frac{(x_1-0.8)^2}{2\sigma^2} + \frac{(x_2-0.8)^2}{2\sigma^2}\right)}, & h_5(x_1, x_2) &= e^{-\left(\frac{(x_1-0.0)^2}{2\sigma^2} + \frac{(x_2-0.0)^2}{2\sigma^2}\right)}, & h_6(x_1, x_2) &= e^{-\left(\frac{(x_1-1.0)^2}{2\sigma^2} + \frac{(x_2-0.0)^2}{2\sigma^2}\right)} \\
 h_7(x_1, x_2) &= e^{-\left(\frac{(x_1-0.5)^2}{2\sigma^2} + \frac{(x_2-0.75)^2}{2\sigma^2}\right)}, & h_8(x_1, x_2) &= e^{-\left(\frac{(x_1-0.5)^2}{2\sigma^2} + \frac{(x_2-1.0)^2}{2\sigma^2}\right)}
 \end{aligned}$$

$$y(x_1, x_2) = \frac{1 \cdot h_1 + 1 \cdot h_2 + 1 \cdot h_3 + 1 \cdot h_4 + 0 \cdot h_5 + 0 \cdot h_6 + 0 \cdot h_7 + 0 \cdot h_8}{h_1 + h_2 + h_3 + h_4 + h_5 + h_6 + h_7 + h_8}$$





## ÜBUNG: Radiale Basisfunktionen

Geben Sie eine Funktion basierend auf radialen Basisfunktionen an, mit folgenden Eigenschaften:

- 2 Eingänge / 1 Ausgang

- XOR-Verhalten

$$0.0 / 0.0 \rightarrow 0.0$$

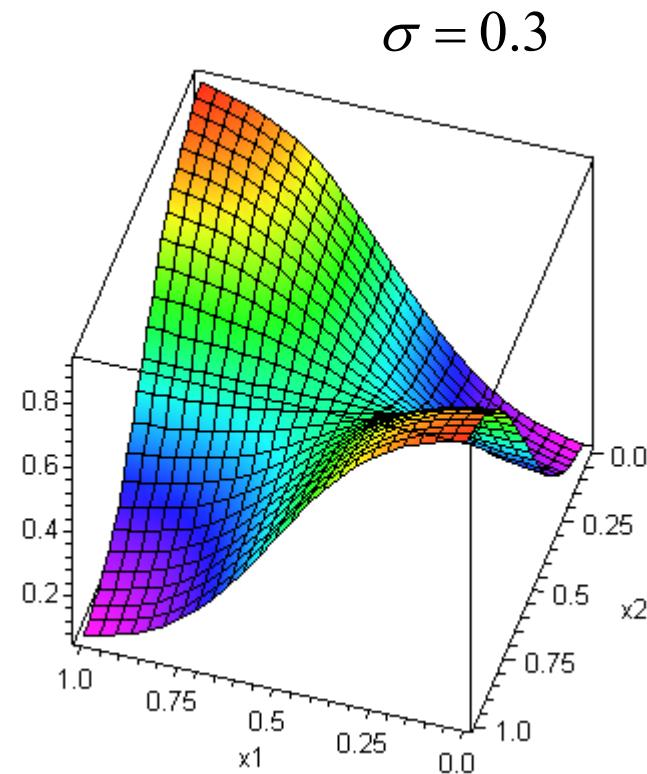
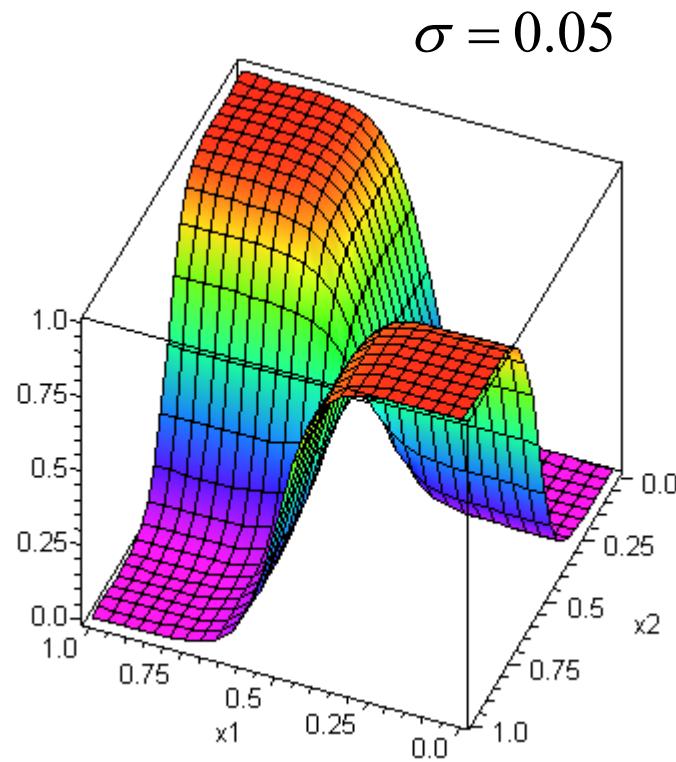
$$1.0 / 1.0 \rightarrow 0.0$$

$$1.0 / 0.0 \rightarrow 1.0$$

$$0.0 / 1.0 \rightarrow 1.0$$



## Fortsetzung ..... : Ergebnis → XOR mit RBF (Gauss-Kernel)





## ÜBUNG: Beispielanwendungen RBF

Diskutieren Sie die folgenden Anwendungsbeispiele:

- a) Linsenverzeichnungskorrektur
- b) Farbinspektion (z.B. Qualitätskontrolle von Keramikkacheln)
- c) Grauwertinterpolation (ähnlich bilineare Interpolation)
- d) Indoor-Navigation durch Intensitätsmessung von Schallquellen  
(alternativ: Feldstärken von Sendern, Intensität von Farblichtquellen)



- Eigenschaften:**
- (+) sehr einfach realisierbar
  - (+) gleitende Übergänge zwischen den Repräsentanten
  - (+) nicht wie Fuzzy-Logic auf Hyperquader festgelegt
  - (-) sehr viele Repräsentanten haben einen hohen Rechenaufwand bei der Berechnung des Ausgabevektors zur Folge
  - (-)  $\sigma$  Festlegung unklar

**Verbesserungen und Erweiterungen :** (Schlagworte)

- Aussortieren der nichtrelevanten Repräsentanten (z.B. Gabriel Graph)
- erst Clustern (k-Means, LVQ, ...) und dann pro Cluster eine oder mehrere RBF verwenden
- autom. und individuelle Anpassung der  $\sigma$

**Bibliotheken verfügbar:**

- RBF\_INTERP\_ND (C++), NN-Toolbox (MATLAB), .....



## 5.4.4 Trainingsbasierte Approximation mit Basisfunktionen

### 5.4.4.1 Vorüberlegungen

Anhand der Lernbeispiele (Repräsentanten, Trainingsdaten, Sichproben) werden in einem speziellen Verarbeitungsschritt (=Training) möglichst einfache Ersatzfunktionen gesucht, welche die Aufgabe (Klassifikation, Parameterbestimmung) hinreichend gut erfüllt.

#### **Ziel:**

- bessere Laufzeit-Effizienz als die trainingsfreien Methoden (ad-hoc-Ansätze)
- Eignung auch für hochdimensionale Eingangsvektoren und viele Lernbeispiele

#### **Ansatz:**

- Reduktion der Basisfunktionsanzahl  
d.h. viel weniger Basisfunktionen als Repräsentanten



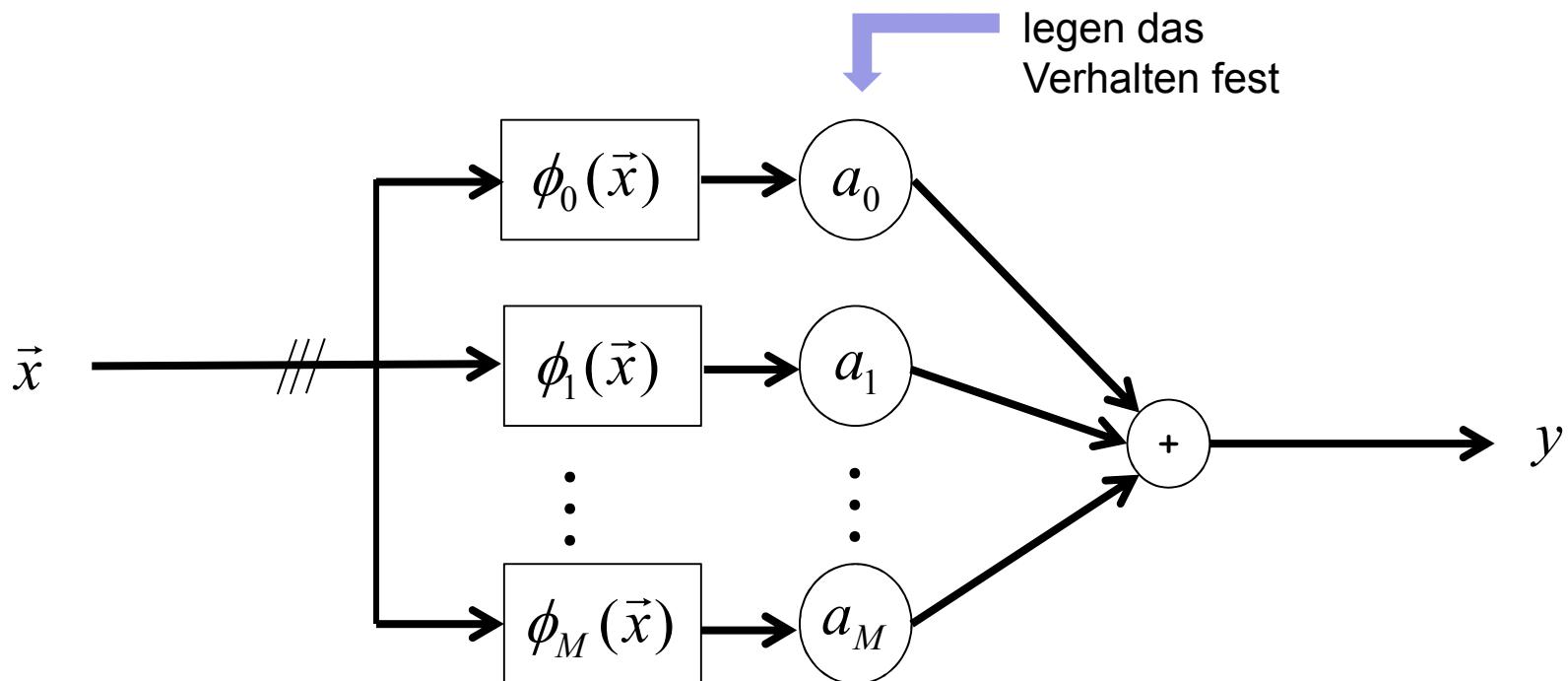
#### 5.4.4.2 Approximation mit linearen Basisfunktionen

Der Ausgangswert  $y$  wird durch eine Funktion des folgenden Typs approximiert:

$$y = \sum_{m=0}^M a_m \cdot \phi_m(\vec{x})$$

Die Basisfunktionen  $\phi_m(\vec{x})$  können beliebige Funktionen von  $\vec{x}$  sein.

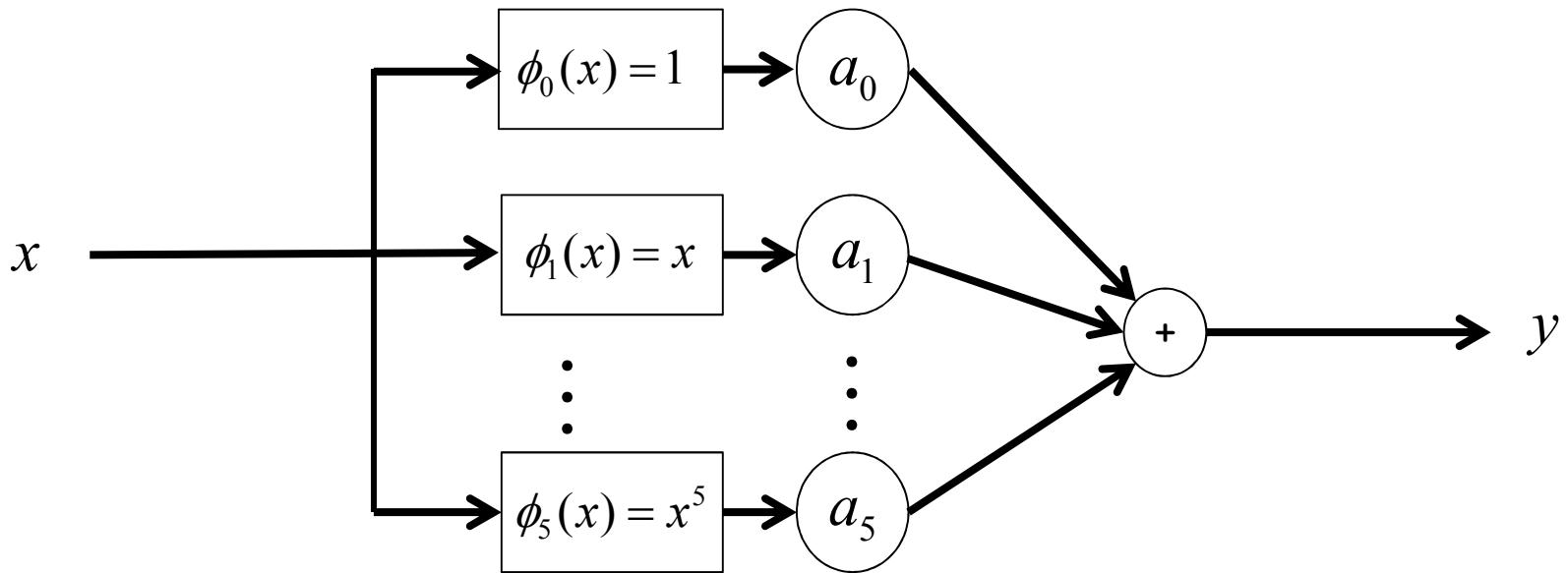
Die Parameter  $a_m$  werden so festgelegt, dass die Repräsentanten bestmöglich (least square) eingepasst werden.



**Beispiel: Polynom-LBF (hier : 1-dim., Polynom 5. Grades)**

$$y = a_0 + a_1 \cdot x + a_2 \cdot x^2 + a_3 \cdot x^3 + a_4 \cdot x^4 + a_5 \cdot x^5$$

Basisfunktionen :  $(\phi_0(x), \dots, \phi_5(x)) = (1, x, x^2, x^3, x^4, x^5)$

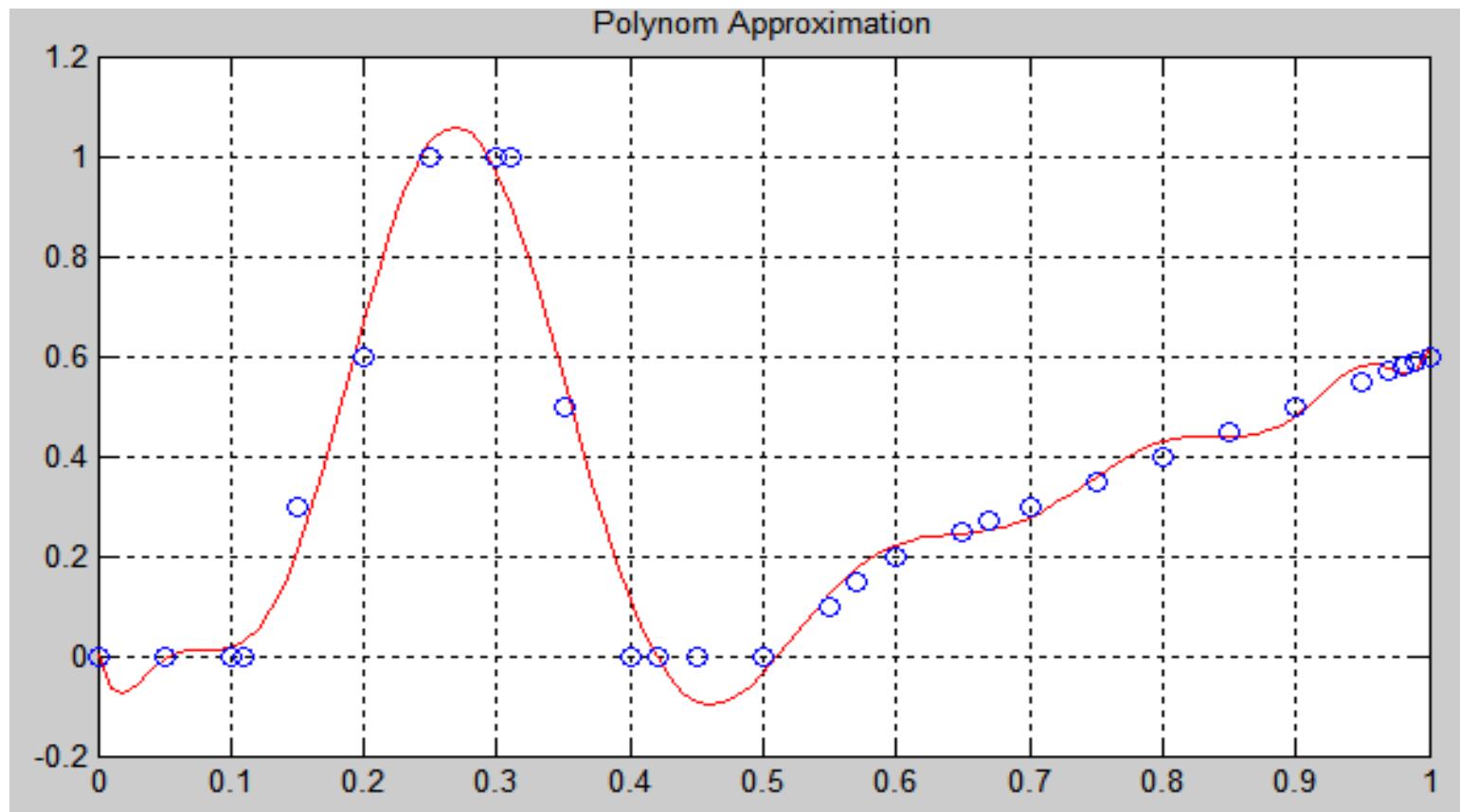




## Fortsetzung ..... : Lineare Basisfunktionen des Polynom-Typs

$$y = a_0 + a_1 \cdot x + a_2 \cdot x^2 + \dots + a_{13} \cdot x^{13} + a_{14} \cdot x^{14}$$

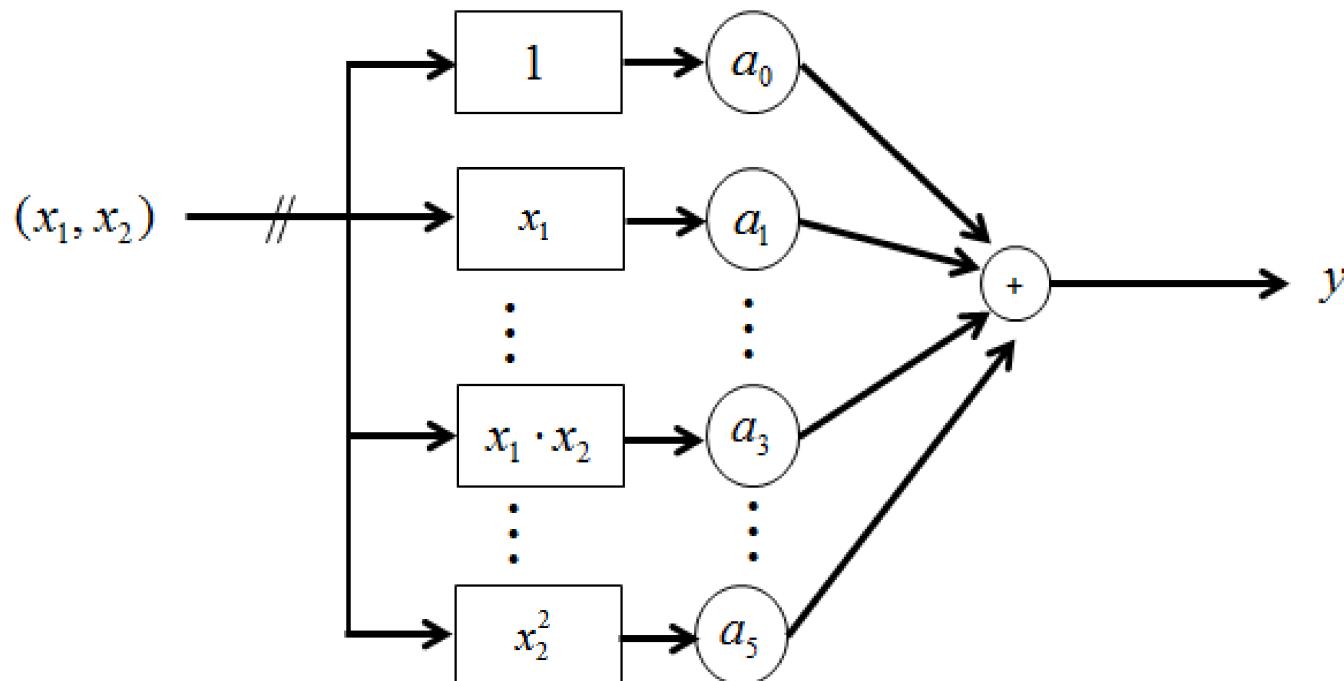
least-square-Einpassung (LSQ = Ausgleichsrechnung)



**Beispiel: Polynom-LBF** (hier : 2-dimensional, Polynom 2. Grades)

$$y = a_0 + a_1 \cdot x_1 + a_2 \cdot x_2 + a_3 \cdot x_1^2 + a_4 \cdot x_1 x_2 + a_5 \cdot x_2^2$$

Basisfunktionen :  $(1, x_1, x_2, x_1^2, x_1 x_2, x_2^2)$

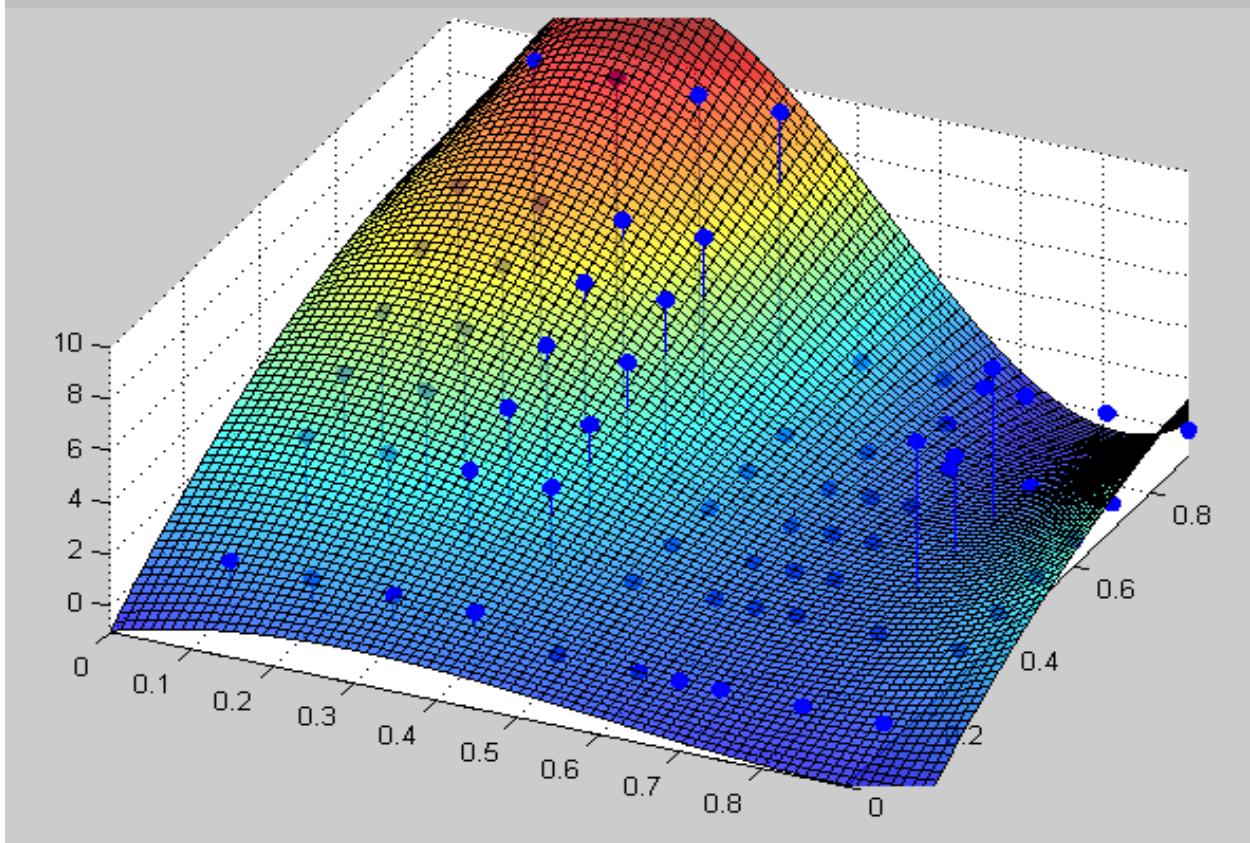




## Fortsetzung ..... : Lineare Basisfunktionen des Polynom-Typs

$$y = a_0 + a_1 \cdot x_1 + a_2 \cdot x_2 + a_3 \cdot x_1^2 + a_4 \cdot x_1 x_2 + a_5 \cdot x_2^2 + \\ \dots + a_6 \cdot x_1^3 + a_7 \cdot x_1^2 x_2 + a_8 \cdot x_1 x_2^2 + a_9 \cdot x_2^3 + \dots + a_{15} \cdot x_1^3 x_2^3$$

Repräsentanten mit eingepasster Approx.-fkt. (16 Parameter)



least-square-  
Einpassung  
(LSQ)

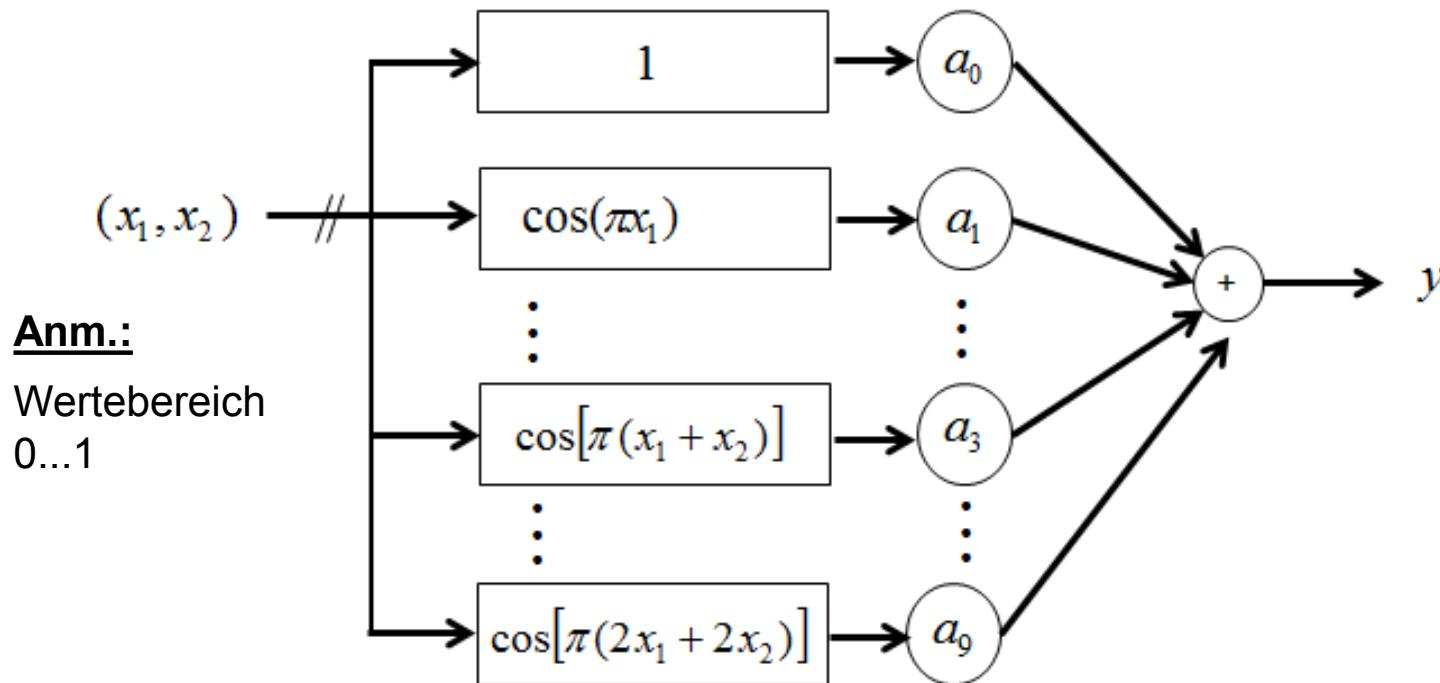
70 Repräsentanten  
16 LBFs



### Beispiel: Fourier-LBF ( hier 2-dim. )

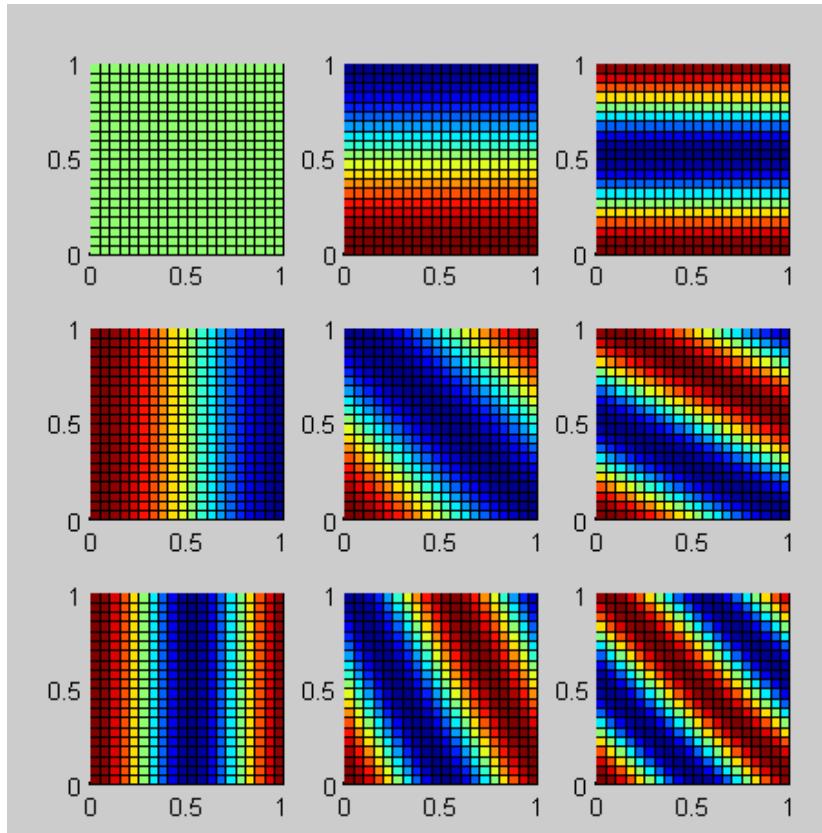
$$y = a_0 + a_1 \cdot \cos(\pi x_1) + a_2 \cdot \cos(\pi x_2) + a_3 \cdot \cos[\pi(x_1 + x_2)] + \\ a_4 \cdot \cos(\pi 2x_1) + a_5 \cdot \cos(\pi 2x_2) + a_6 \cdot \cos[\pi(2x_1 + x_2)] + \dots$$

Basisfunktionen : (1,  $\cos(\pi x_1)$ ,  $\cos(\pi x_2)$ ,  $\cos(\pi x_1 + \pi x_2)$ ,  $\cos(\pi 2x_1)$ , ....)





## Fortsetzung ..... : Lineare Basisfunktionen des Fourier-Typs



abgebildete Basisfunktionen

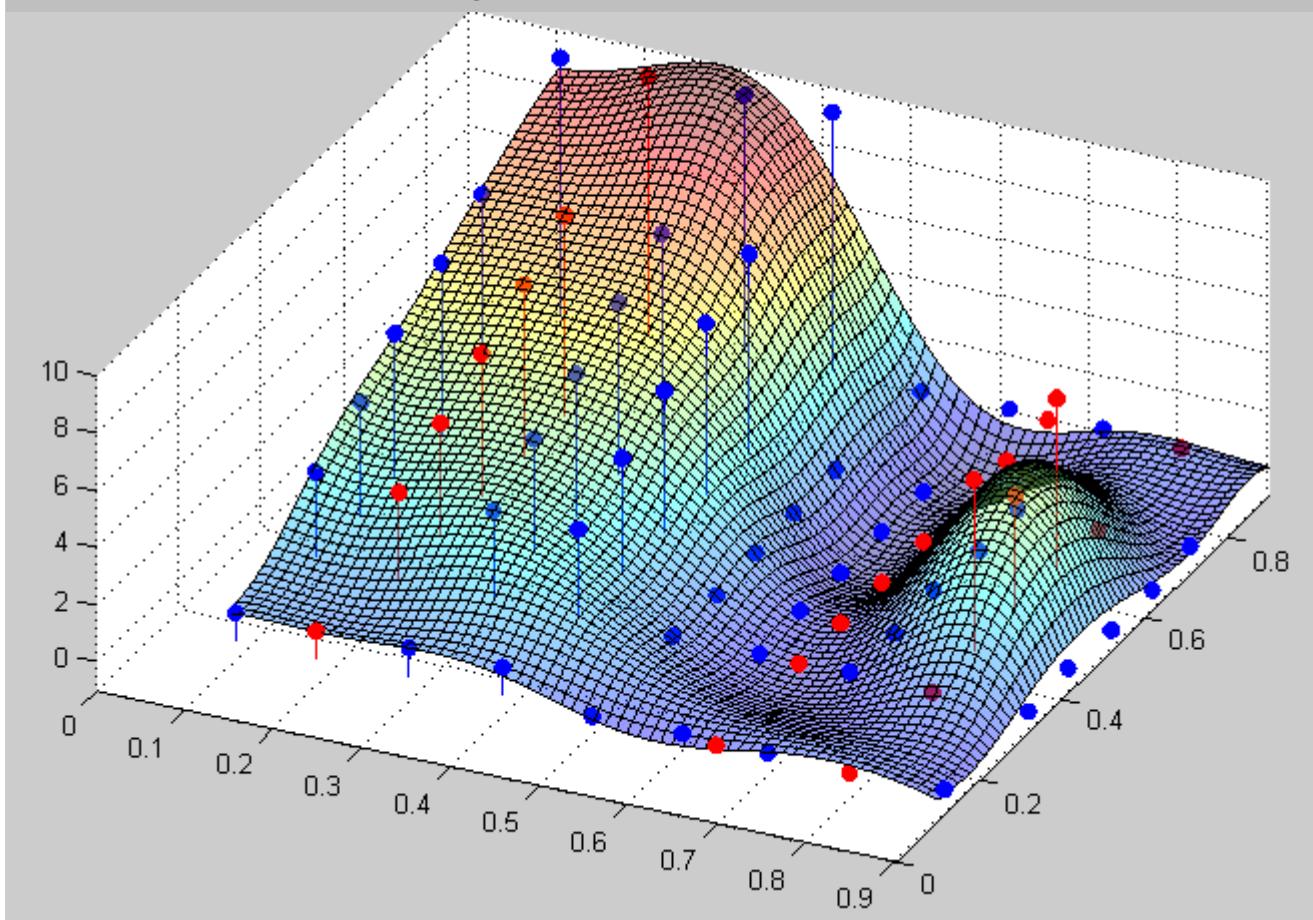
|                  |                         |                          |
|------------------|-------------------------|--------------------------|
| 1                | $\cos(\pi x_1)$         | $\cos(\pi 2x_1)$         |
| $\cos(\pi x_2)$  | $\cos[\pi(x_1 + x_2)]$  | $\cos[\pi(2x_1 + x_2)]$  |
| $\cos(\pi 2x_2)$ | $\cos[\pi(x_1 + 2x_2)]$ | $\cos[\pi(2x_1 + 2x_2)]$ |



## Fortsetzung .... : Fourier-LBF-Approximationsfunktion mit Repräsentanten

$$y = a_0 + a_1 \cdot \cos(\pi x_1) + \dots + a_{35} \cdot \cos[\pi(5x_1 + 5x_2)]$$

Repräsentanten mit eingepasster Approx.-fkt (36 Parameter)



least-square-  
Einp assung  
(LSQ)

70 Repräsentanten  
36 LBFs



## Abschließende Fragen zu lin. Basisfunktionen

Wie bestimmt man die Parameter  $a_0, a_1, a_2, \dots$  ? (folgt später)

Wie wählt man den Basisfunktionstyp (Polynom, Fourier-Typ, ...) aus ?

Wie legt man die Anzahl der Basisfunktionen fest ?

Wie viele Messwerte (Samples) werden für eine brauchbare Approximation benötigt ?

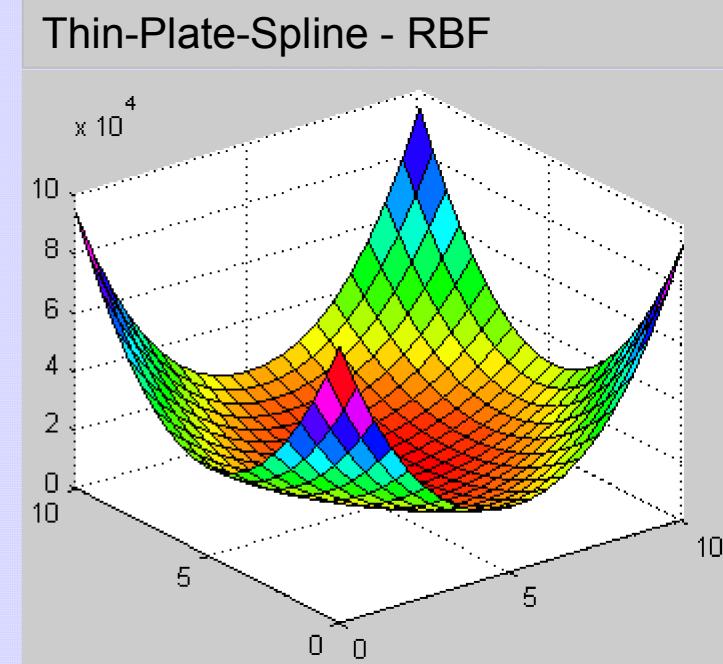
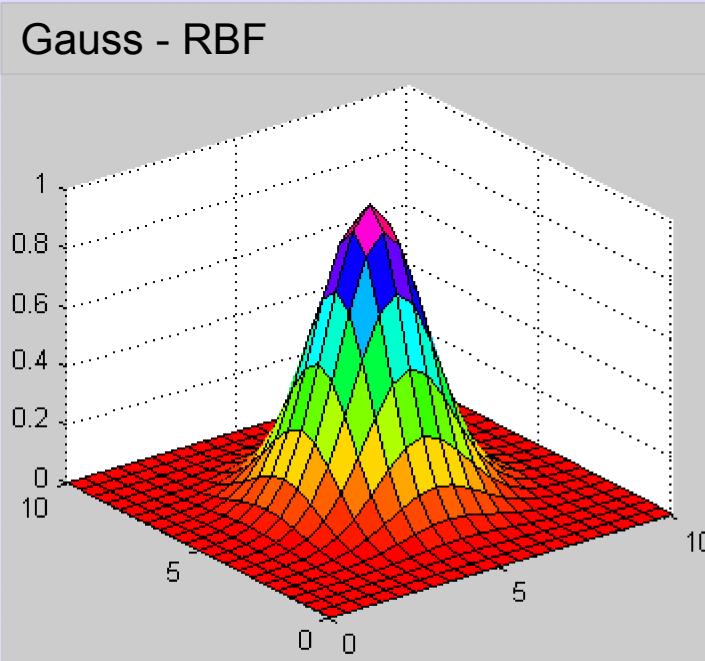
Was passiert außerhalb des durch die Messwerte aufgespannten Bereiches ?



#### 5.4.4.3 Approximation durch radiale Basisfunktionen

Radiale Basisfunktionen zeichnen sich dadurch aus, dass sie radialsymmetrisch zu einem Zentrum sind.

**Beispiele:** Zentrum in beiden Fällen  $\vec{x}_m = (5,5)$





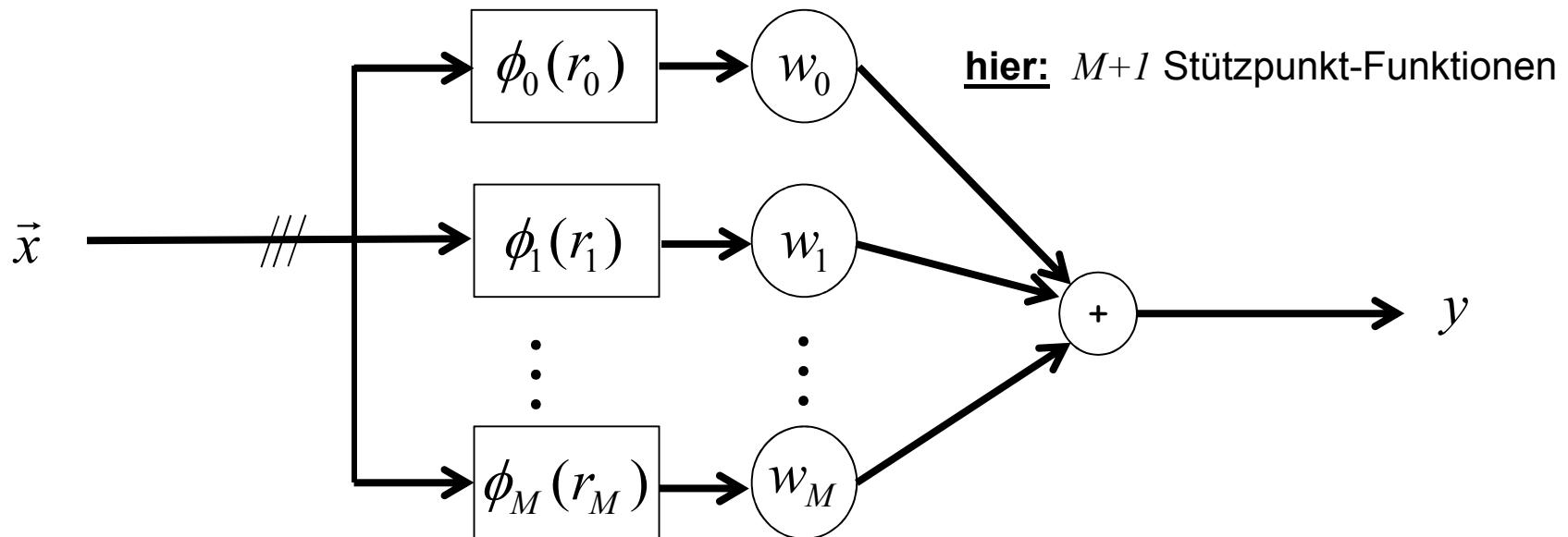
Die Funktion  $y(\vec{x})$  wird approximiert durch gewichtete Überlagerung mehrerer radialer Basisfunktionen, die an unterschiedlichen Stellen (Stützpunkten)  $\vec{x}_m$  platziert werden.

$$y = \sum_{m=1}^M w_m \cdot \phi(r_m)$$

s. Beispiel: nä. Seite

mit  $r_m = r_m(\vec{x}) = \|\vec{x} - \vec{x}_m\|$  Abstandsnorm

z.B.  $r_m(\vec{x}) = \sqrt{(\vec{x} - \vec{x}_m)^2}$  d.h. euklid. Abstand eines Eingangswertes  $\vec{x}$  vom Stützpunkt  $\vec{x}_m$





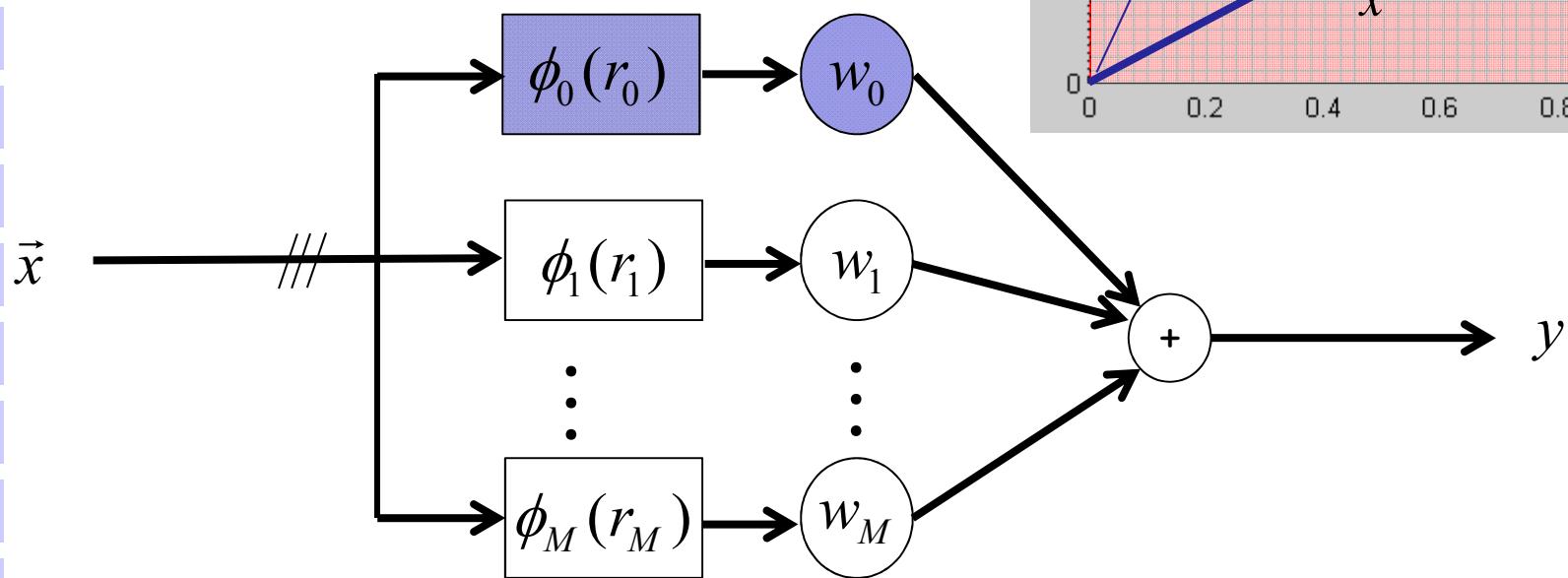
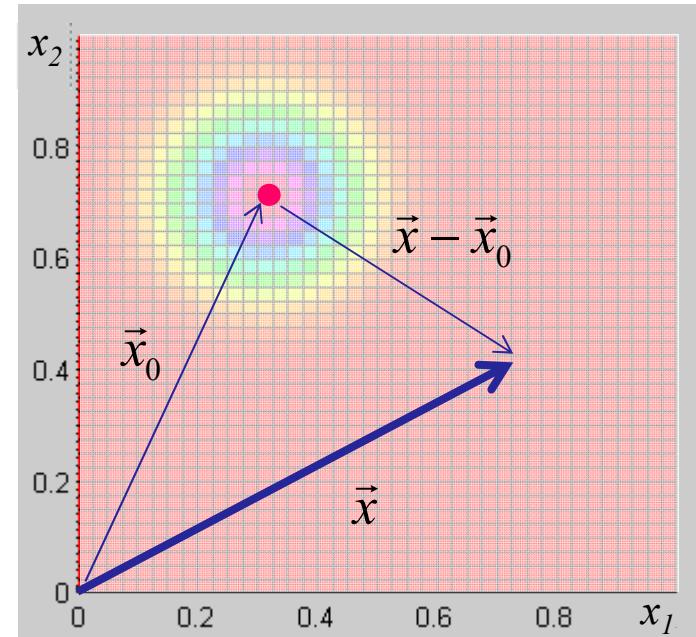
## Beispiel: eine gewichtete radiale Basisfunktion vom Gauss-Typ

Abstand des Ein-gangsvektors vom Zentrum der RBF

$$r_0(\vec{x}) = \sqrt{(\vec{x} - \vec{x}_0)^2}$$

Funktionswert im Abstand  $r_0$

$$\phi_0(\vec{x}) = e^{-\varepsilon \cdot r_0^2}$$





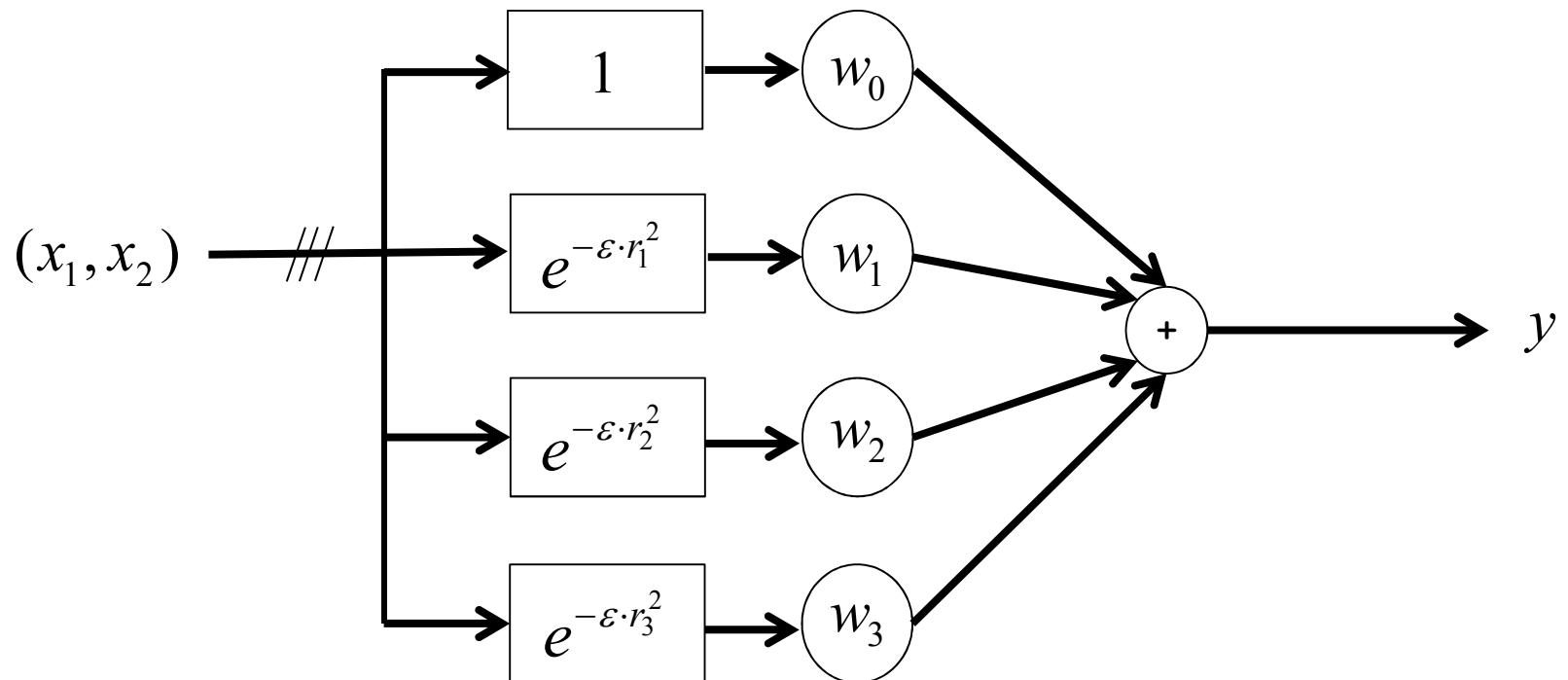
## Beispiel : Gauss – RBF

$$y = w_3 \cdot e^{-\varepsilon \cdot r_3^2} + w_2 \cdot e^{-\varepsilon \cdot r_2^2} + w_1 \cdot e^{-\varepsilon \cdot r_1^2} + w_0$$

z.B. mit  $r_m^2(\vec{x}) = (\vec{x} - \vec{x}_m)^2$

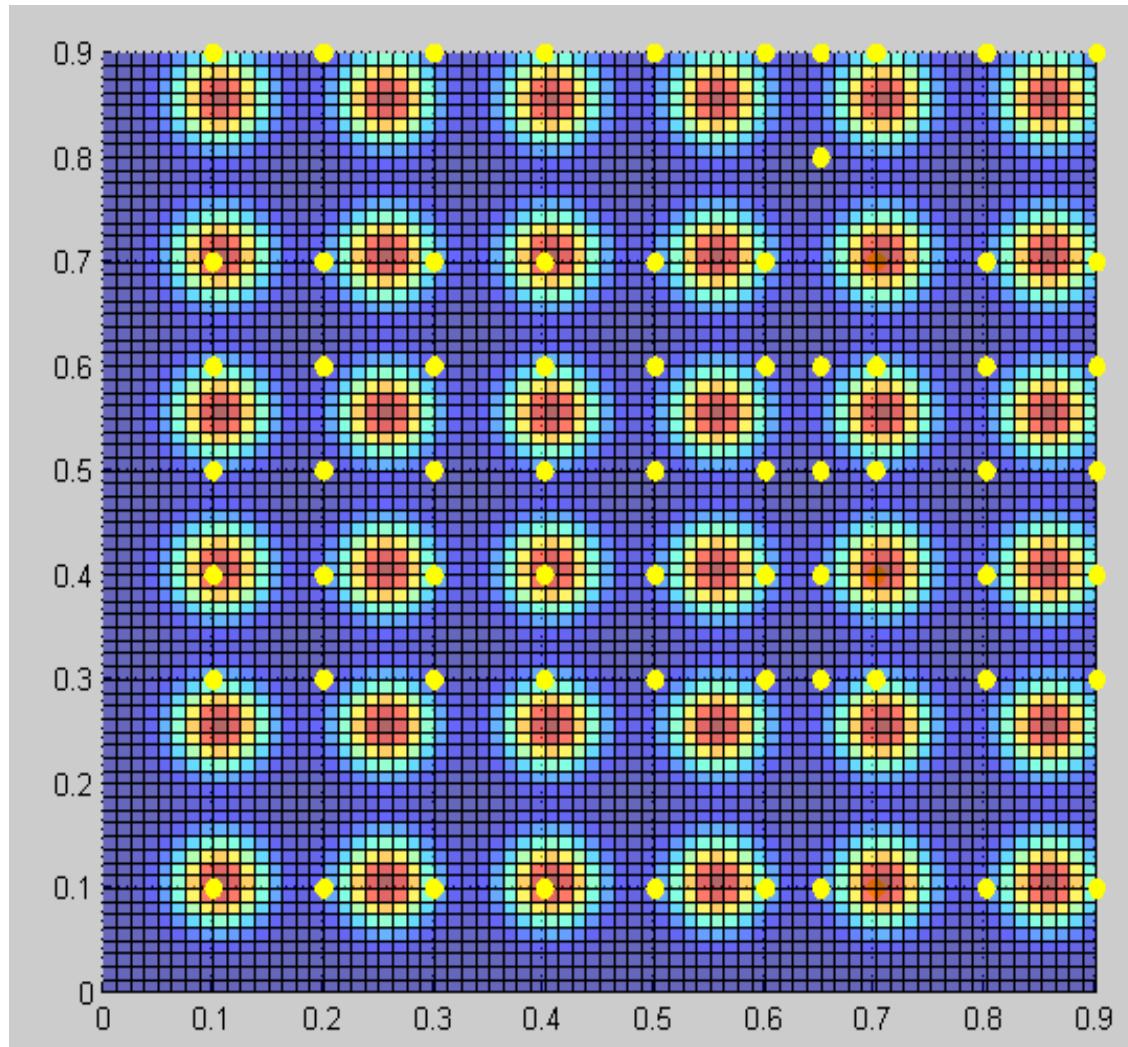
Zentrum der RBF  $m$  :  $(\vec{x}_m)$

Glätteparameter (frei festlegbar) :  $\varepsilon$





## Fortsetzung .... : RBF-Raster und Ort der Repräsentanten (gelbe Punkte)



**hier gewählt:**  
äquidistantes Funktionsraster (0.15 Schritte)  
→ 36 RBF's

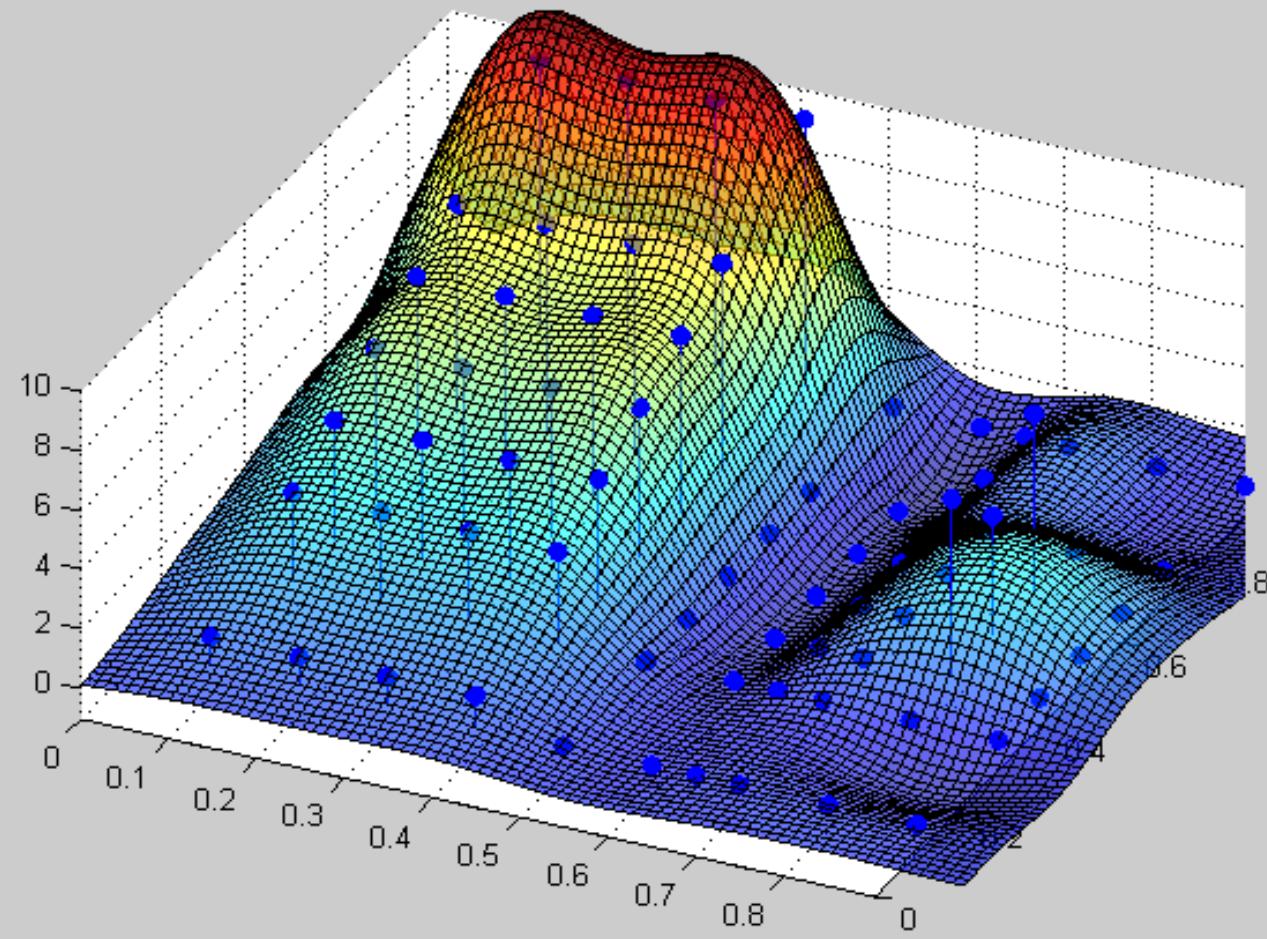
$\varepsilon = 24$   
(aus Gründen der besseren Darstellung sind sehr schmale RBFs angezeigt)

Man beachte den Unterschied zu den ad-hoc-RBF's.  
Dort gab es pro Repräsentant genau eine RBF !



## Fortsetzung .... : Gauss-RBF-Approximationsfunktion mit Repräsentanten

70 Messwerte mit eingepasster Approximationsfunktion (36 RBF's)



least-square-  
Einpassung  
(LSQ)

$\varepsilon = 8$   
breitere RBF's

70 Repräsentanten  
36 RBFs

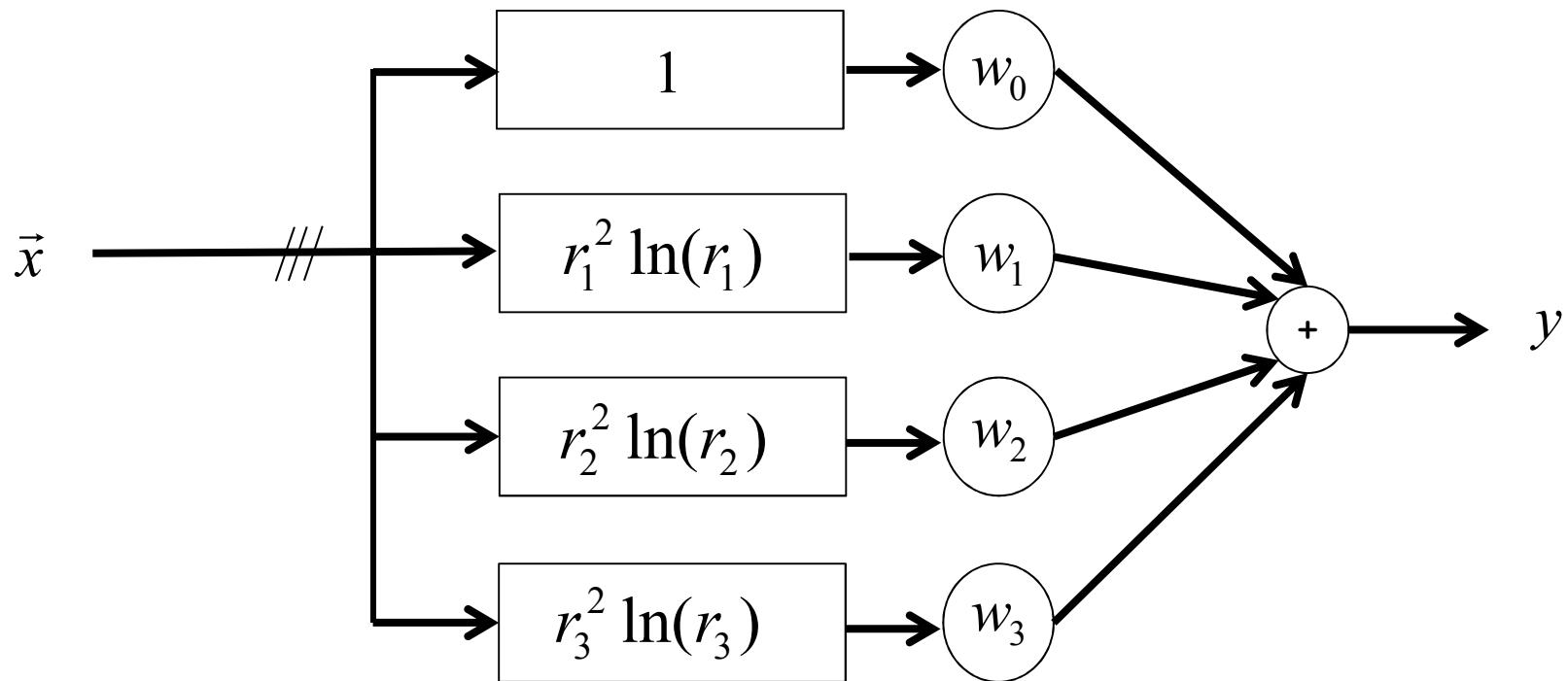


## Beispiel : Thin-Plate-Spline – RBF

$$y = w_3 \cdot r_3^2 \ln(r_3) + w_2 \cdot r_2^2 \ln(r_2) + w_1 \cdot r_1^2 \ln(r_1) + w_0$$

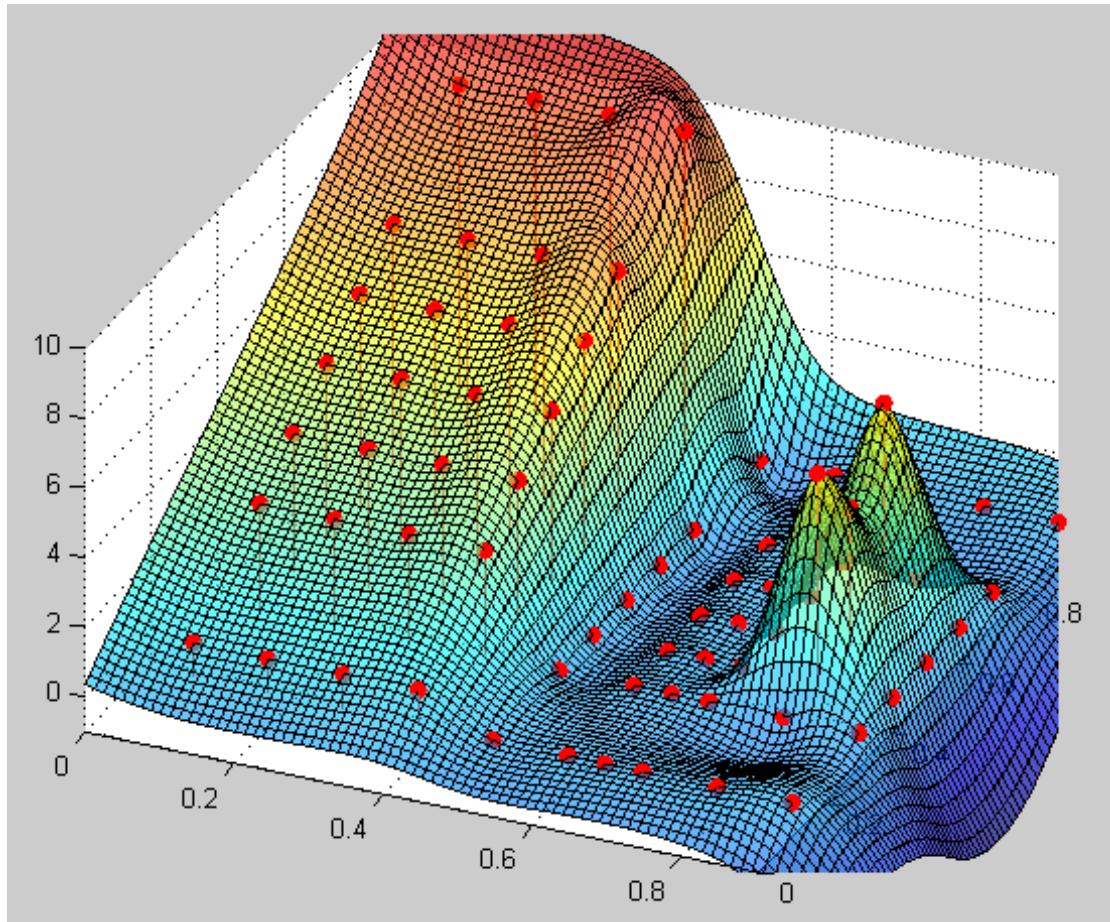
z.B. mit  $r_m^2(\vec{x}) = (\vec{x} - \vec{x}_m)^2$

Zentrum der RBF  $m$ :  $(\vec{x}_m)$





## Fortsetzung ..... : Thin-plate-Approximationfunktion mit Repräsentanten



Hier pro Messwert eine radiale Basisfunktion gesetzt (muss aber nicht sein).

Die Approximationsfunktion verhält sich wie eine dünne Membran durch die Punkte (min. Biegeenergie).

### Großer Vorteil:

Kein manuell einzustellender Parameter (besser als Gauß-Typ →  $\varepsilon$  bzw.  $\sigma$ )



## Abschließende Fragen zu radialen Basisfunktionen

Wie bestimmt man die Parameter  $w_0, w_1, w_2, \dots$  ? (folgt)

Wie wählt man den Basisfunktionstyp (Gauss, Thin-Plate, ....) aus ?

Wie legt man die Anzahl der Basisfunktionen fest ?

Wo platziert man die Basisfunktionen ?

- a) Im einfachsten Fall gibt es eine Basisfunktion pro Messwert:  
In diesem Fall legen die Meßwerte den Ort fest (s.o.).
- b) Möglicherweise sollen ja erheblich weniger Basisfunktionen als Messwerte verwendet werden. Dann stellt sich die Frage nach deren Platzierung (z.B. äquidistant, durch Clustering).

Wie viele Messwerte (Samples) werden für eine brauchbare Approximation benötigt ?

Was passiert außerhalb des durch die Messwerte aufgespannten Bereiches ?



#### 5.4.4.4 Vorüberlegungen zum Training

Die auf linearen und radialen Basisfunktionen basierenden Systeme sind linear bezüglich der zu bestimmenden Parameter ( $a_k$ - bzw.  $w_k$ -Parameter).

Die Bestimmung der Funktionsparameter ( $a_k$ - bzw.  $w_k$ ) mit den Lösungsmethoden für lin. Gleichungssysteme (bzw. Ausgleichungssysteme) ist daher prinzipiell möglich.

aber ....

Problem bei vielen Parametern (typ. bei hochdimensionalem Eingangsvektor):

- extrem viele Unbekannte → riesiges Gleichungssystem
- in vertretbarer Zeit nicht lösbar

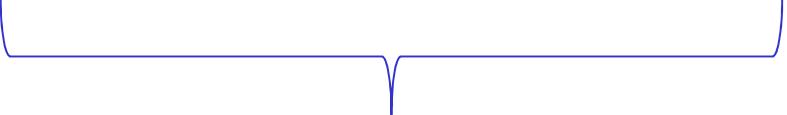
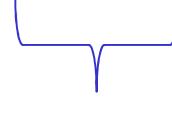
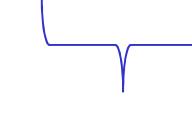
In diesem Fall werden andere Methoden benötigt (z.B. Gradientenabstiegsverfahren).



#### 5.4.4.5 Bestimmung der Parameter durch Ausgleichsrechnung (LSQ)

Darstellung in Matrixform: pro Repräsentant (Messwert, Sample, ....) eine Zeile

$$\begin{pmatrix} \phi_M(\vec{x}_1) & \phi_{M-1}(\vec{x}_1) & \dots & \phi_0(\vec{x}_1) \\ \phi_M(\vec{x}_2) & \phi_{M-1}(\vec{x}_2) & \dots & \phi_0(\vec{x}_2) \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ \phi_M(\vec{x}_N) & \phi_{M-1}(\vec{x}_N) & \dots & \phi_0(\vec{x}_N) \end{pmatrix} \cdot \begin{pmatrix} a_M \\ a_{M-1} \\ \vdots \\ a_0 \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix}$$




  
 $A$        $\vec{\xi}$        $\vec{b}$

Die Ausgleichslösung (also die Parameter  $a_0, a_1, a_2, a_3, \dots$ ) erhält man dann durch Lösen des Gleichungssystems :

$$\underline{A}^T \underline{A} \cdot \vec{\xi} = \underline{A}^T \cdot \vec{b}$$



## ÜBUNG : XOR mit Polynom-LBF

Geben Sie eine Funktion basierend auf linearen Basisfunktionen (Polynom-Typ) an, mit folgenden Eigenschaften:

- 2 Eingänge / 1 Ausgang
- XOR-Verhalten

$$0.0 / 0.0 \rightarrow 0.0$$

$$1.0 / 1.0 \rightarrow 0.0$$

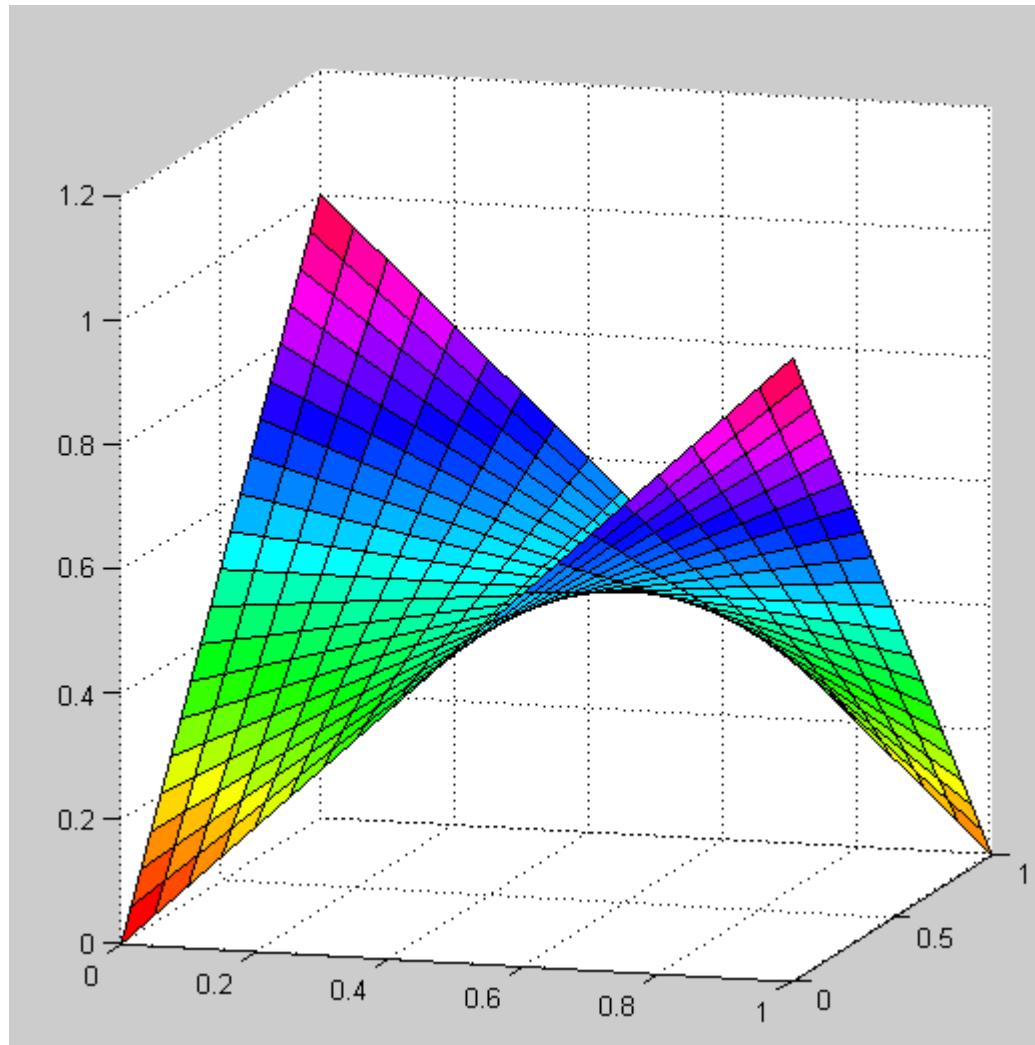
$$1.0 / 0.0 \rightarrow 1.0$$

$$0.0 / 1.0 \rightarrow 1.0$$

Verwenden Sie das Polynom:  $y = a_0 + a_1 \cdot x_2 + a_2 \cdot x_1 + a_3 \cdot x_1 x_2$



## Fortsetzung der Übung : Ergebnis → XOR mit Polynom-LBF





## ÜBUNG: Image-Warping mit Thin-Plate-Splines

Diskutieren Sie Realisierungsmöglichkeiten eines Image-Warping-Verfahrens auf Basis der Thin-Plate-Splines.





## Einfluss der Eingangsvektor-Dimension auf die Zahl der Parameter

$$y = a_0 + a_1 \cdot \cos(\pi x) + a_2 \cdot \cos(2\pi x) + \dots + a_5 \cdot \cos(5\pi x)$$

1-dimens. Eingangsvektor → 6 Parameter (= Unbekannte)

$$y = a_0 \cdot \cos(0\pi x_1 + 0\pi x_2) + a_1 \cdot \cos(0\pi x_1 + 1\pi x_2) + \dots + a_{35} \cdot \cos[\pi(5x_1 + 5x_2)]$$

2-dimens. Eingangsvektor → 36 Parameter (= Unbekannte)

$$\begin{aligned} y = & a_0 \cdot \cos[\pi \cdot (0,0,0,0,0) \cdot \vec{x}] + a_1 \cdot \cos[\pi \cdot (0,0,0,0,1) \cdot \vec{x}] + \dots \\ & + a_{7774} \cdot \cos[\pi \cdot (5,5,5,5,4) \cdot \vec{x}] + a_{7775} \cdot \cos[\pi \cdot (5,5,5,5,5) \cdot \vec{x}] \end{aligned}$$

5-dimens. Eingangsvektor → 7776 Parameter (= Unbekannte) !!!



## Diskussion: Bestimmung der Parameter durch Ausgleichsrechnung (LSQ)

**Eigenschaften:** (+) bestmögliche Einpassung der Funktion in die Messwerte (LSQ)  
im Rahmen der Möglichkeiten der gewählten Funktion

(+) Online-Ausgleichsrechnung (= inkrementelle Hinzunahme  
weiterer Meßwerte) möglich

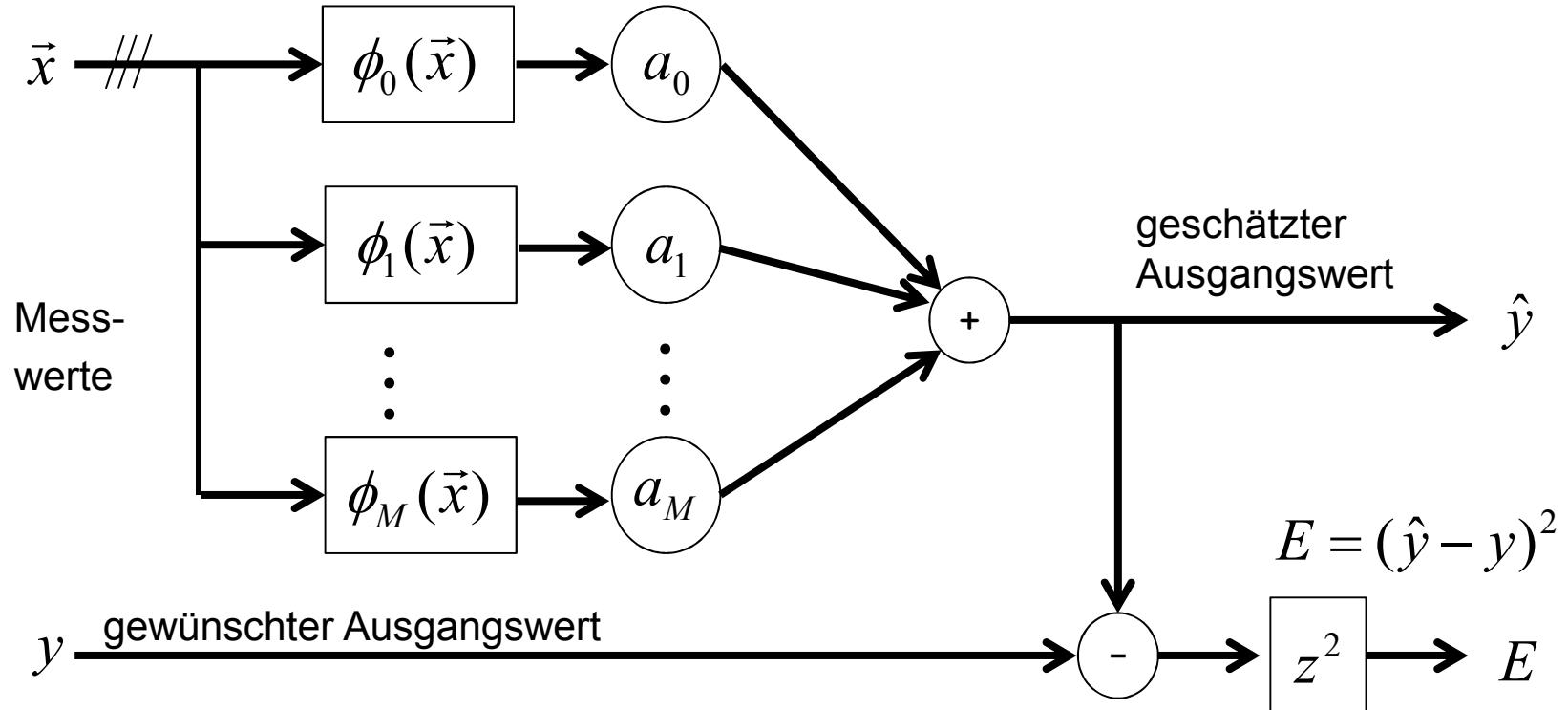
(-) bei hochdimensionalen Eingangsgrößen ist die  
Parameterbestimmung ineffizient bis undurchführbar:  
 $k$  Parameter → Komplexität  $\geq O(k^2)$



#### 5.4.4.6 Iterative Parameterbestimmung durch Gradientenabstieg

Wie kann das direkte Lösen von Gleichungssystemen vermieden werden ?

**Ansatz:** Ausgehend von einem Schätzwert für die Parameter  $a_0, a_1, \dots$  werden die Parameter in kleinen Schritten so modifiziert, dass der Fehler über alle Trainingsdaten kleiner wird.

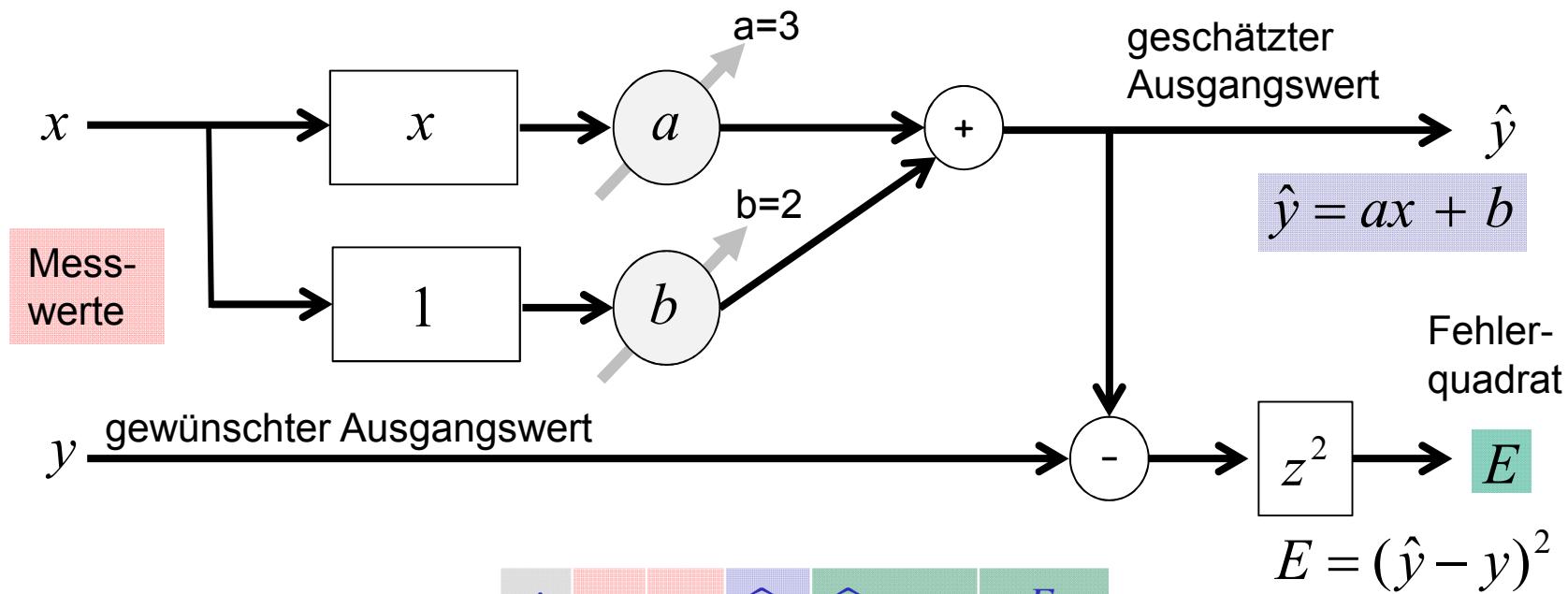




## Beispiel: Bestimmung der Parameter a und b der Gerade $y=ax+b$

**Gegeben:** Eine Menge von Messwerten  $(x,y)$

**Gesucht:** Die Geradenparameter a und b, die das Fehlerquadrat E über alle Messwerte minimieren



für die Startwerte

$$a=3, b=2$$

und die angegebenen  
Messwerte werden fol-  
gende Werte gemessen :

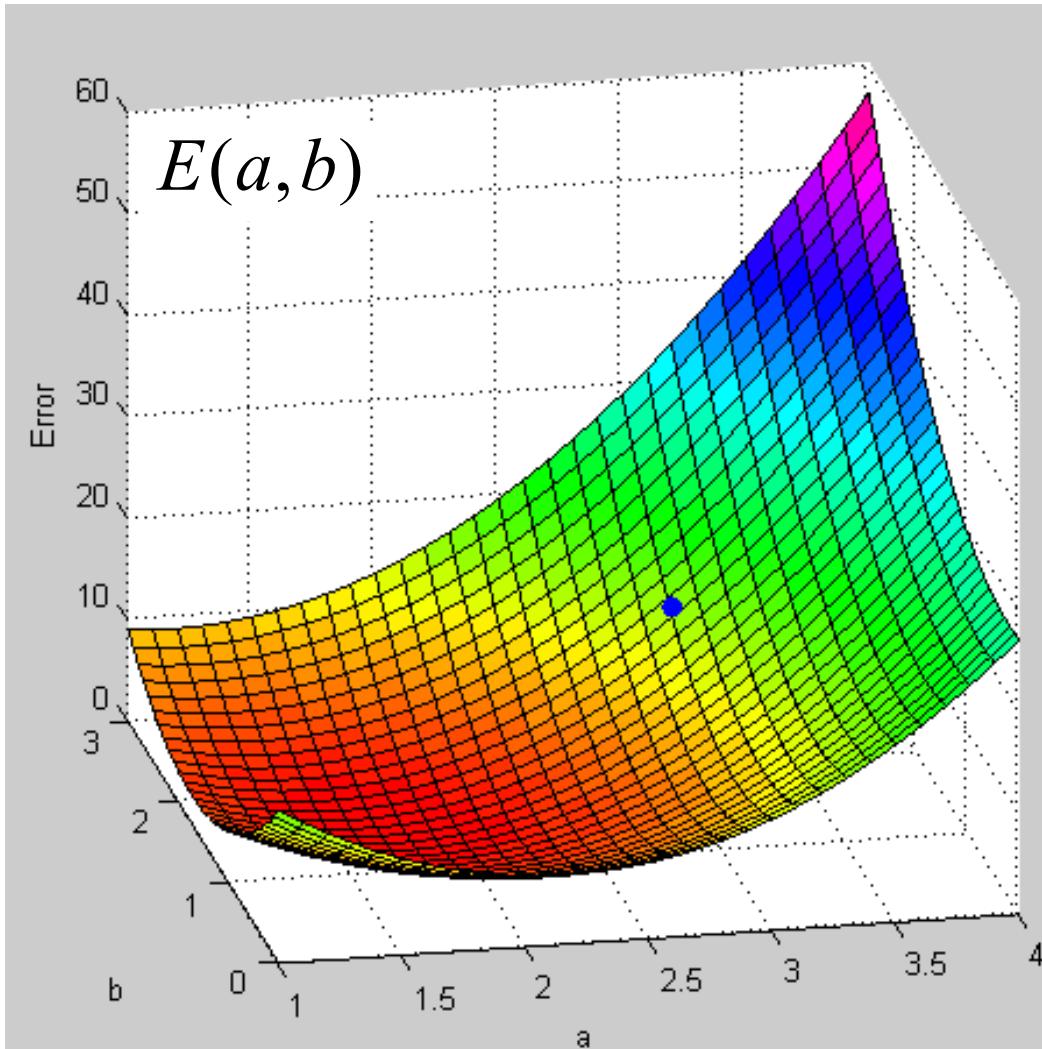
| $i$ | $x$ | $y$ | $\hat{y}$ | $\hat{y} - y$ | $E_i$ |
|-----|-----|-----|-----------|---------------|-------|
| 1   | 1   | 3   | 5         | 2             | 4     |
| 2   | 2   | 5   | 8         | 3             | 9     |
| 3   | 0   | 1   | 2         | 1             | 1     |

bei der gegebenen  
Parametereinstellung  
ist der Gesamtfehler:

$$E = 4 + 9 + 1 = 14$$



## Fortsetzung ....: Fehlergebirge über den Parametern a und b



### Frage:

Mit welcher Strategie findet man diejenigen Parameter a und b, die den Fehler über alle Messwerte minimieren?

- a) alles absuchen  
→ i.A. zu aufwendig
- b) immer bergab gehen



## Fortsetzung ....: Bewegen in Richtung des steilsten Abstiegs

Der Gradient einer Funktion mit mehreren Variablen zeigt in Richtung des steilsten Aufstiegs.

$$\vec{\nabla}E = \begin{pmatrix} \frac{\partial E}{\partial a} & \frac{\partial E}{\partial b} \end{pmatrix}^T$$

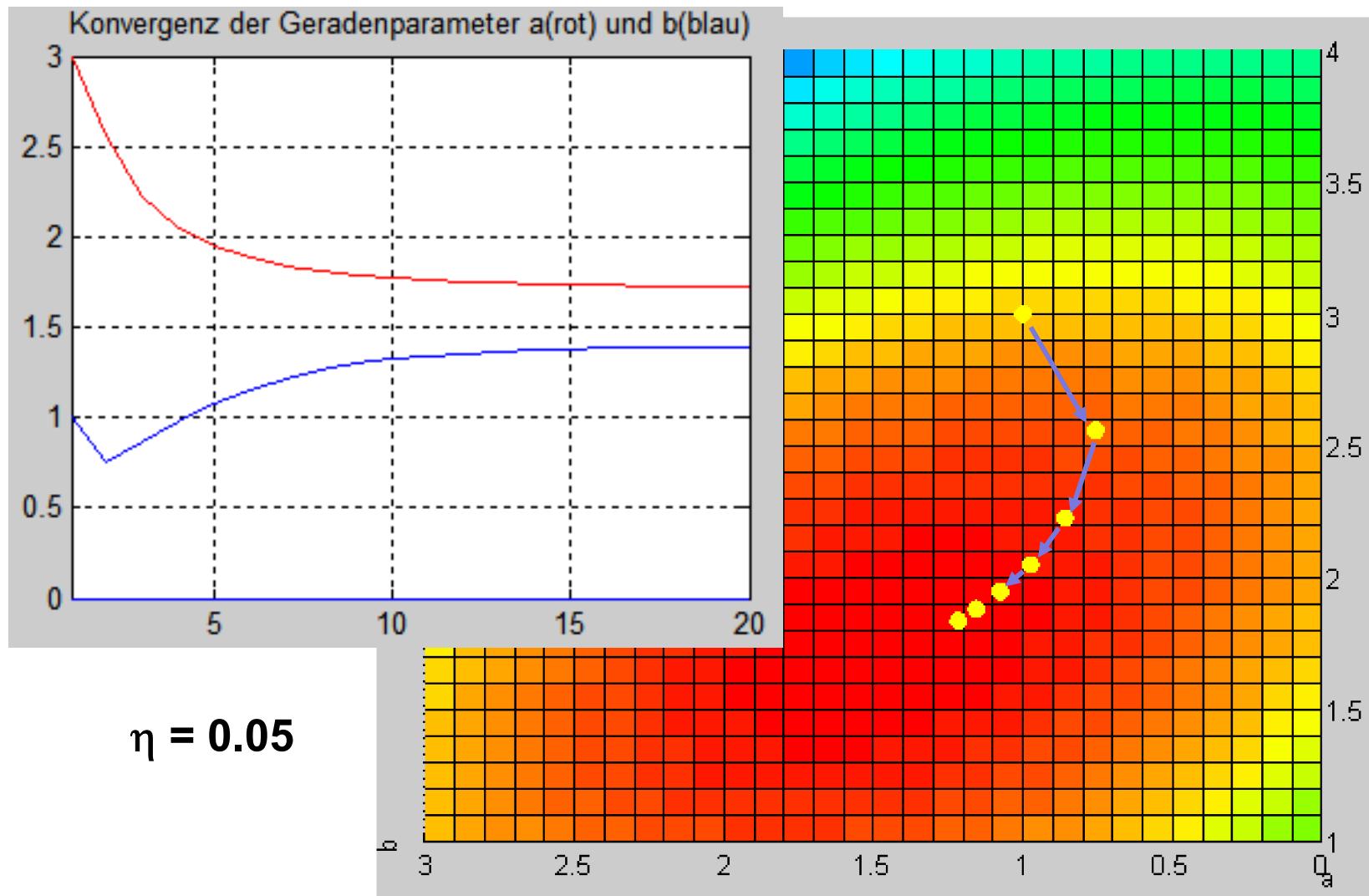
Eine Strategie zum finden des Fehlerminimums könnte daher sein, in Richtung des negativen Gradienten (= steilster Abstieg, steepest descent) zu gehen.

$$\begin{pmatrix} a_{n+1} \\ b_{n+1} \end{pmatrix} = \begin{pmatrix} a_n \\ b_n \end{pmatrix} - \eta \cdot \vec{\nabla}E(a, b)$$

mit  $\eta$  : frei wählbarer Schrittweitenfaktor

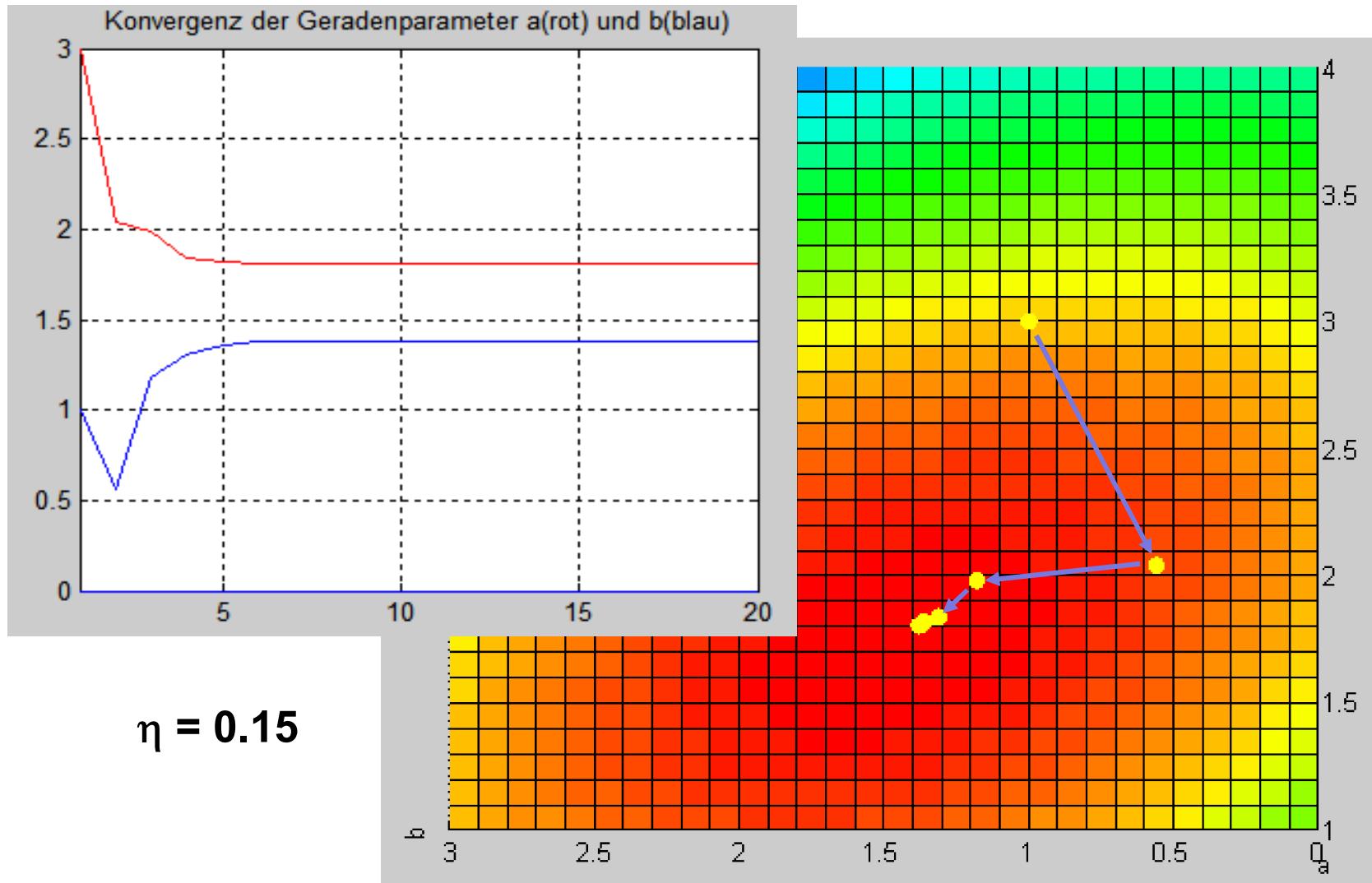


## Fortsetzung ....: Konvergenzverlauf der Parameter a und b über dem Fehlergebirge



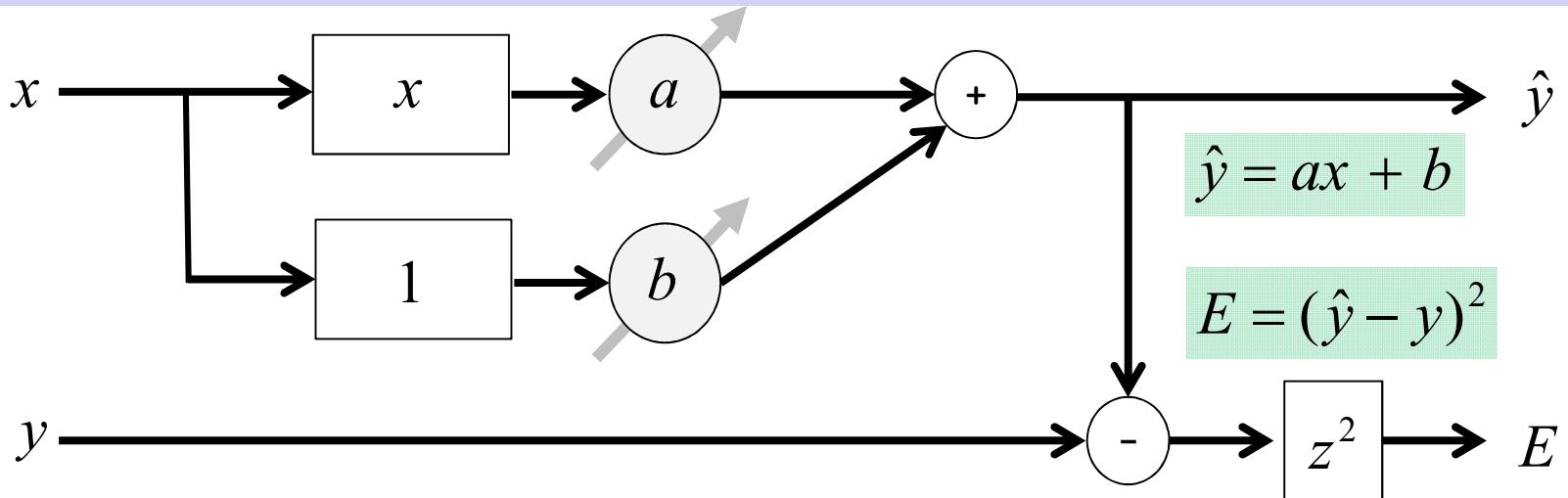


## Fortsetzung ....: Konvergenzverlauf der Parameter a und b über dem Fehlergebirge





## Fortsetzung ....: Berechnung des Gradienten



Mit Hilfe der Kettenregel lassen sich die beiden Komponenten des Gradienten leicht bestimmen:

$$\frac{\partial E}{\partial a} = \frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial a} = 2(\hat{y} - y) \cdot x$$

$$\frac{\partial E}{\partial b} = \frac{\partial E}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial b} = 2(\hat{y} - y) \cdot 1$$

Die Iterationsgleichung für die Suche des Fehlerminimums lautet daher

$$\begin{pmatrix} a_{n+1} \\ b_{n+1} \end{pmatrix} = \begin{pmatrix} a_n \\ b_n \end{pmatrix} - \eta \cdot 2 \cdot (\hat{y} - y) \cdot \begin{pmatrix} x \\ 1 \end{pmatrix}$$



## Fortsetzung ....: Matlab-Code

```
a=3; b=2; % initial value

% measurements
x = [1, 2, 0];
y= [3, 5, 1];

Eta = 0.1; % stepfactor
N = 20; % iterations

V = [a b]'; % parameter vector

for n = 1:N % iterations
    for k=1:size(x,2) % all measurements

        a = V(1); b = V(2);

        yE = a * x(k) + b; % est. output value
        % grad. descent
        V = V + Eta*2*(y(k)-yE)*[x(k) 1]';

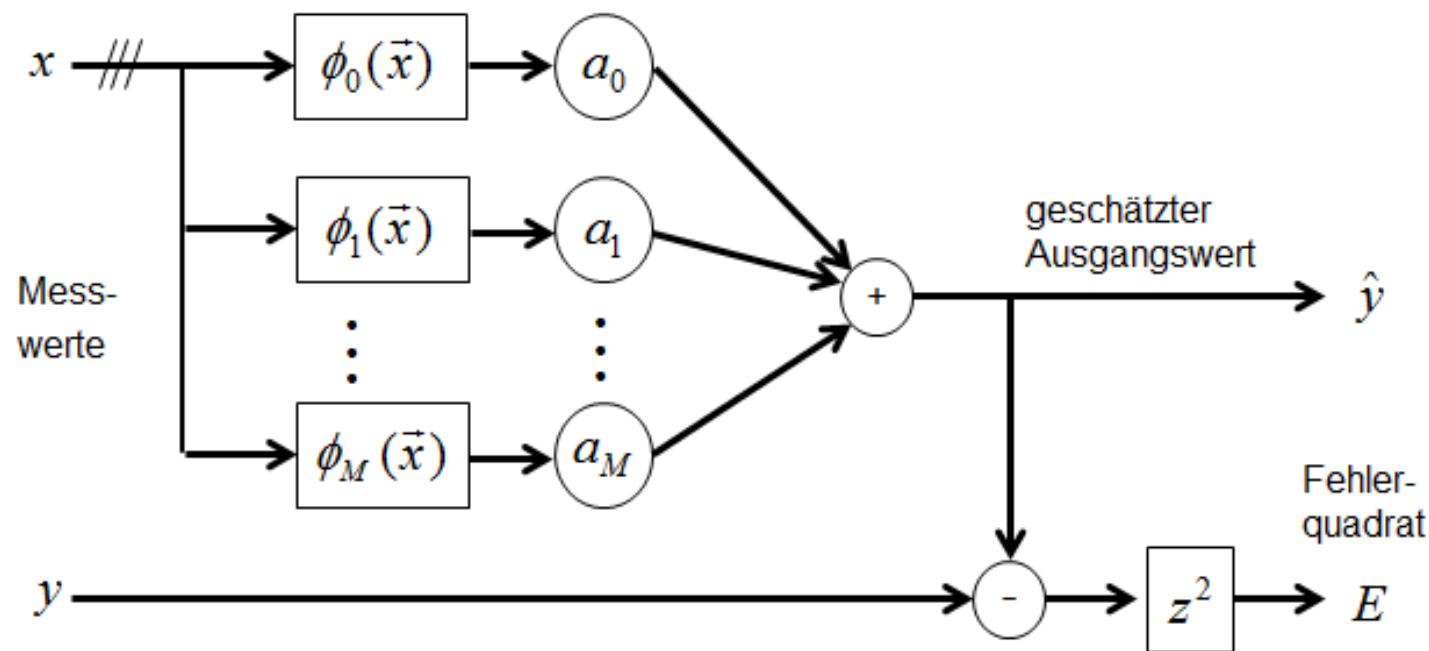
    end
end
```



## ÜBUNG : Herleitung der allg. Lösung

Es ist zu zeigen, dass für Basisfunktionen die folgende Iterationsgleichung gilt:

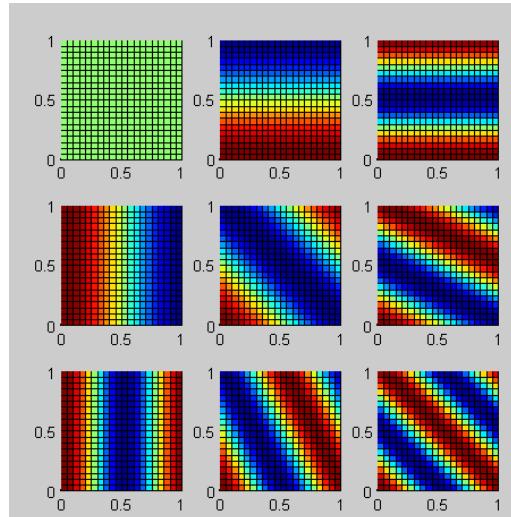
$$\begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_M \end{pmatrix}_{n+1} = \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_M \end{pmatrix}_n - \eta \cdot 2 \cdot (\hat{y} - y) \cdot \begin{pmatrix} \phi_0(\vec{x}) \\ \phi_1(\vec{x}) \\ \vdots \\ \phi_M(\vec{x}) \end{pmatrix}$$





**Beispiel :** 2-dimensionaler Eingangsvektor  
+ Fourier-Approximation (bis Ordnung 5)

$$\hat{y} = a_0 \cdot \cos[\pi \cdot (0,0) \cdot \vec{x}] + \dots + a_6 \cdot \cos[\pi \cdot (0,5) \cdot \vec{x}] \dots \\ + a_7 \cdot \cos[\pi \cdot (1,0) \cdot \vec{x}] + \dots + a_{12} \cdot \cos[\pi \cdot (1,5) \cdot \vec{x}] \dots \\ + a_{31} \cdot \cos[\pi \cdot (5,0) \cdot \vec{x}] + \dots + a_{36} \cdot \cos[\pi \cdot (5,5) \cdot \vec{x}]$$



usw.

$$\begin{pmatrix} a_0 \\ \vdots \\ a_{36} \end{pmatrix}_{n+1} = \begin{pmatrix} a_0 \\ \vdots \\ a_{36} \end{pmatrix}_n - \eta \cdot 2 \cdot (\hat{y} - y) \cdot \begin{pmatrix} \cos[\pi \cdot (0,0) \cdot \vec{x}] \\ \vdots \\ \cos[\pi \cdot (5,5) \cdot \vec{x}] \end{pmatrix}$$



## Fortsetzung des Beispiels : Algorithmus

init. Parameter

**foreach** Trainingsinput  $\vec{x}, y$

**for** i=1:Iterations

$$\begin{aligned}\hat{y} = & a_0 \cdot \cos[\pi \cdot (0,0) \cdot \vec{x}] + \dots + a_6 \cdot \cos[\pi \cdot (0,5) \cdot \vec{x}] \dots \\ & + a_7 \cdot \cos[\pi \cdot (1,0) \cdot \vec{x}] + \dots + a_{12} \cdot \cos[\pi \cdot (1,5) \cdot \vec{x}] \dots \\ & + a_{31} \cdot \cos[\pi \cdot (5,0) \cdot \vec{x}] + \dots + a_{36} \cdot \cos[\pi \cdot (5,5) \cdot \vec{x}]\end{aligned}$$

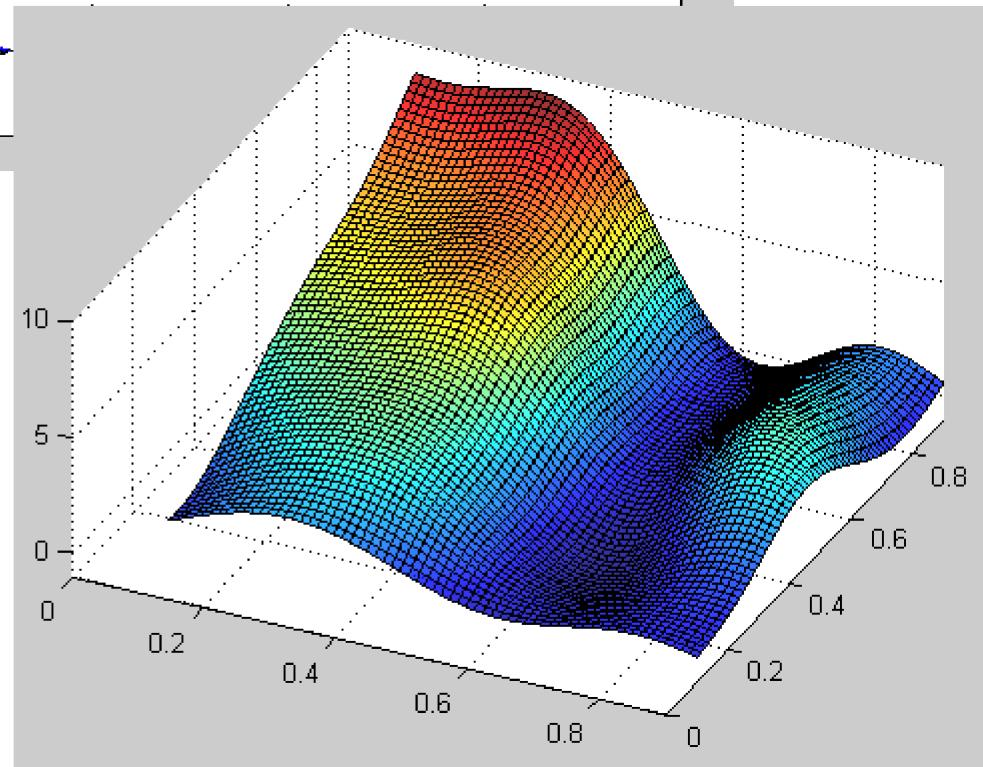
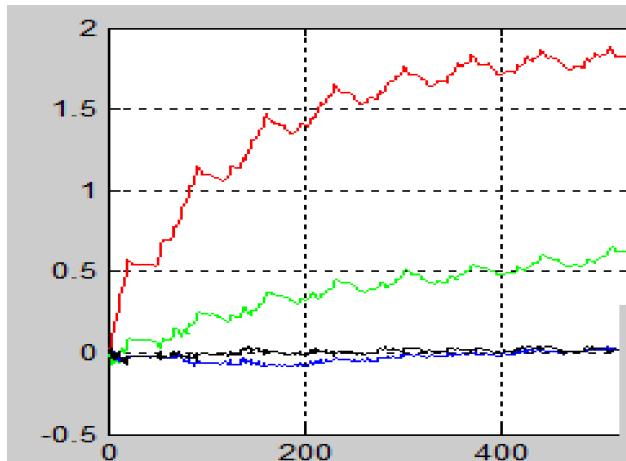
$$\begin{pmatrix} a_0 \\ \vdots \\ a_{36} \end{pmatrix}_{n+1} = \begin{pmatrix} a_0 \\ \vdots \\ a_{36} \end{pmatrix}_n - \eta \cdot 2 \cdot (\hat{y} - y) \cdot \begin{pmatrix} \cos[\pi \cdot (0,0) \cdot \vec{x}] \\ \vdots \\ \cos[\pi \cdot (5,5) \cdot \vec{x}] \end{pmatrix}$$

**end for**

**end foreach**



## Fortsetzung des Beispiels : Konvergenzverlauf einiger Parameter (4 von 36)



approximierte Funktion  
(Fourier bis Ordnung 5)



## Diskussion: Bestimmung der Parameter durch Gradientenabstieg (LSQ)

**Eigenschaften:** (+) bei sehr vielen Funktionsparametern deutlich effizienter als das Lösen eines Gleichungssystems:  
 $k$  Unbekannte  $\rightarrow$  Komplexität  $\geq O(k)$

(+) sehr gut für Onlinetraining geeignet, d.h.  
wenn die Daten nacheinander eintreffen

(-) Steuerparameter  $\eta$

(-) i. Allg. wird nur ein lokales Fehlerminimum getroffen