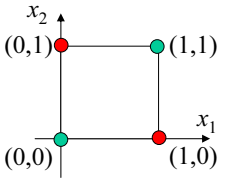
Polynomklassifikator für das XOR-Problem

Wir wählen als Regressionsfunktion ein Polynom mit Termen zweiten Grades in der Hoffnung, dass damit eine Lösung erreicht werden kann:

$$f(\mathbf{x}) = a_1 x_1 + a_2 x_2 + a_3 x_1 x_2 = \mathbf{a}^T \mathbf{p}(\mathbf{x})$$

mit:

$$\mathbf{p} = \left[x_1, x_2, x_1 x_2\right]^T \quad \text{und} \quad \mathbf{a} = \left[a_1, a_2, a_3\right]^T$$

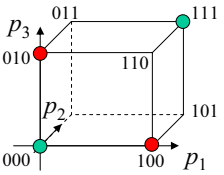


$\mathcal{X}$-Raum

Die Eckpunkte des Quadrats im zweidimensionalen Ursprungsraum $\mathcal{X}$ werden auf die Ecken eines (Hyper-) Würfels einen dreidimensionalen Merkmalsraum $\mathcal{P}$ abgebildet und dieser schließlich auf den eindimensionalen Bedeutungsraum $\mathcal{Y}$:

$$\mathcal{X} \rightarrow \mathcal{P} \rightarrow \mathcal{Y}$$

$\mathcal{X}$	x_1	0	1	0	1
	x_2	0	0	1	1
	p_1	0	1	0	1
$\mathcal{P}$	p_2	0	0	1	1
	p_3	0	0	0	1
$\mathcal{Y}$	y	0	1	1	0



$\mathcal{P}$-Raum

Die optimalen Parameter der Regressionsfunktion **a*** ergeben sich aus:

$$\underbrace{E\left\{\mathbf{p}\mathbf{p}^T\right\}}_{\mathbf{R}_{pp}}\mathbf{a}^* = \underbrace{E\left\{\mathbf{p}\mathbf{y}^T\right\}}_{\mathbf{R}_{py}} \quad \text{bzw:} \quad \mathbf{a}^* = \mathbf{R}_{pp}^{-1} \cdot \mathbf{R}_{py}$$

mit: $\mathbf{R}_{pp} = E\left\{\mathbf{p}\mathbf{p}^T\right\} \approx \frac{1}{4}\sum_{i=1}^{4} \mathbf{p}_i \mathbf{p}_i^T$

$$= \frac{1}{4}\left\{\mathbf{0} + \left[\begin{array}{c|ccc} & 1 & 0 & 0 \\ \hline 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right] + \left[\begin{array}{c|ccc} & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{array}\right] + \left[\begin{array}{c|ccc} & 1 & 1 & 1 \\ \hline 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{array}\right]\right\} = \frac{1}{4}\begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

und: $\mathbf{R}_{py} = E\left\{\mathbf{p}y\right\} \approx \frac{1}{4}\sum_{i=1}^{4} \mathbf{p}_i y_i = \frac{1}{4}\sum_{i=1}^{4} y_i \mathbf{p}_i$

$$= \frac{1}{4}\left\{0 \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + 1 \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} + 1 \cdot \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} + 0 \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}\right\} = \frac{1}{4}\begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

Daraus folgt:

$$\mathbf{a}^* = \mathbf{R}_{pp}^{-1} \cdot \mathbf{R}_{py} = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ -1 & -1 & 3 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ -2 \end{bmatrix}$$

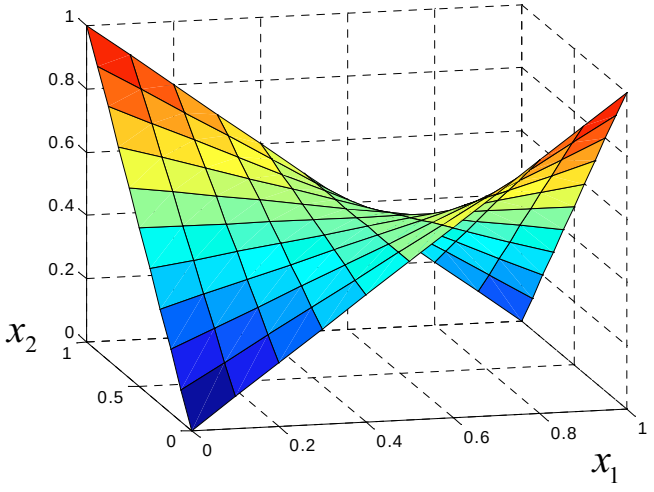
und damit eine Regressionsfunktion

$$\boxed{f(\mathbf{x}) = a_1 x_1 + a_2 x_2 + a_3 x_1 x_2 = x_1 + x_2 - 2x_1 x_2}$$

welche an den 4 Funktionswerten genau die Werte der Zielfunktion annimmt und ansonsten interpoliert!

Regressionsfunktion

$$f(\mathbf{x}) = x_1 + x_2 - 2x_1 x_2$$

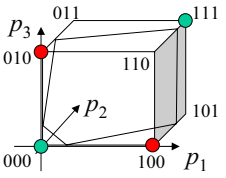


Trennflächen für die beiden Klassen

Die Trennfläche für die beiden Bedeutungsklassen verläuft dort, wo die Regressionsfunktion $f(\mathbf{x})$ den arithmetischen Mittelwert der Zielfunktion, nämlich $c=(0+1)/2=0{,}5$ annimmt. Das nichtlinear separierbare XOR-Problem kann in dem dreidimensionalen Merkmalsraum durch eine *lineare* (Hyper-)Fläche separiert werden, was auch leicht geometrisch an dem dreidimensionalen Kubus nachvollziehbar ist:

$$f(\mathbf{x}) = \langle \mathbf{a}, \mathbf{p} \rangle - 0{,}5 = p_1 + p_2 - 2p_3 - 0{,}5 \begin{cases} > 0 \Rightarrow \mathbf{x} \in \mathcal{A} \\ < 0 \Rightarrow \mathbf{x} \in \mathcal{B} \end{cases}$$

Gleichung für *lineare* Trennfläche



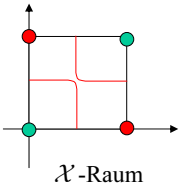
$\mathcal{P}$-Raum

Nichtlineare Trennfläche im Originalraum: (Kegelschnitt)

$$f(\mathbf{x}) = x_1 + x_2 - 2x_1x_2 - 0{,}5 \begin{cases} > 0 \Rightarrow \mathrm{x} \in \mathrm{A} \\ < 0 \Rightarrow \mathrm{x} \in \mathrm{B} \end{cases}$$

$$\Rightarrow (x_1 + x_2 - 2x_1x_2 - 0{,}5) = -2(x_1 - 0{,}5)(x_2 - 0{,}5) = 0$$

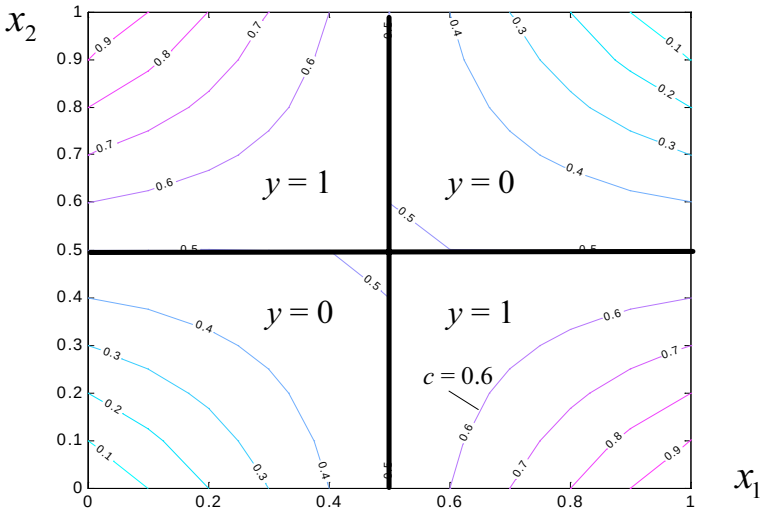
gilt für: $\begin{cases} x_1 = 0{,}5 & x_2 \text{ beliebig} \\ x_2 = 0{,}5 & x_1 \text{ beliebig} \end{cases}$ gerade Trennlinien



$\mathcal{X}$-Raum

Verlauf der Trennkurven im Originalraum (Hyperbeln) für unterschiedliche Schwellwerte c

$$f(\mathbf{x}) = x_1 + x_2 - 2x_1x_2 = c$$



Polynomklassifikator für ein etwas komplexeres Beispiel

$$f(\mathbf{x}) = a_0 + \sum_{i=1}^{N} a_i x_i + \sum_{i=1}^{N-1} \sum_{m=i+1}^{N} a_{im} x_i x_m + \ldots = P(x_1) \times P(x_2) \times \ldots \times P(x_N)$$

kartesisches Produkt der eindimensionalen Polynome mit:

$\dim(\mathbf{x}) = N$ und Grad des Polynoms: r

d.h. der N-dimensionale Originalraum wird in einen L-dimensionalen Merkmalsraum abgebildet. Dieser enthält Terme der Art:

$$x_1^{p_1} x_2^{p_2} \cdots x_N^{p_N} \quad \text{mit:} \quad p_1 + p_2 + \ldots + p_N \leq r$$

Für ein Polynom vom Grade r und ein Merkmalsvektor der Dimension N ergibt sich eine Dimension des Ausgangsraums L von:

$$L = \binom{N+r}{r} = \frac{(N+r)!}{r!N!}$$

Demo mit MATLAB
(Klassifikationgui.m)

- Öffnen von Matlab
  - zuerst setmypath.m aufrufen, dann
  - C:\Home\ppt\Lehre\ME_2002\matlab\KlassifikatorEntwurf-WinXX\Klassifikationgui
  - Datensatz xor2.mat laden
  - Polynomklassifikator vom Grad 2 anwenden
- Zweiklassenproblem mit Bananenshape eingeben
  - Datensatz Samples/banana_test_samples.mat laden
  - Versuch mit Polynomklassifikator vom Grad 2 und Grad 3

Start der Matlab-Demo
matlab-Klassifikation_gui.bat

Eigenschaften des Polynomklassifikators:

- Vorteile:
  - Lineare Entwurfsgleichungen mit expliziter Lösung für ein globales Minimum des Approximationsfehlers
- Nachteile:
  - Die Entwicklung nach den global wirkenden Polynombasen zeigen eine schlechte Konvergenz. Lokale Besonderheiten von Merkmalsclustern können nur sehr schlecht approximiert werden, ohne dabei gleichzeitig nicht zuviel Qualität an Generalisierungsfähigkeit zu verlieren.

Funktionsapproximation mit einem Radiale-Basis-Funktionen-Netzwerk (RBFN)

- Anstatt Polynome für die Regressionsaufgabe zu verwenden, kann man auch ein eine lineare Entwicklung in Radiale Basisfunktionen (RBF) in Erwägung ziehen. Damit erhält man zwar wiederum ein nichtlineares Parameterschätzproblem (wie bei NN), welches typischerweise nur iterativ gelöst werden kann, aber es ergeben sich bessere Konvergenzeigenschaften.
- RBFs als Interpolationsfunktionen sind Funktionen der Form:

$$p(\|\mathbf{x}-\mathbf{c}_i\|)$$

- D.h. im Argument einer jeden RBF steht die Euklidsche Distanz zwischen dem Eingangsvektor $\mathbf{x}$ und einem Zentrum $\mathbf{c}_i$ (daraus erklärt sich der Name. Die RBFs wirken nur in *lokalen* Umgebungen.

Beispiele für Radiale-Basis-Funktionen (RBFN)

$$p(\mathbf{x}) = \exp\left(-\frac{1}{2\sigma_i^2} \left\| \mathbf{x} - \mathbf{c}_i \right\|^2 \right) \quad \text{(Gauss-RBFs)}$$

$$p(\mathbf{x}) = \frac{\sigma_i^2}{\sigma_i^2 + \left\| \mathbf{x} - \mathbf{c}_i \right\|^2}$$

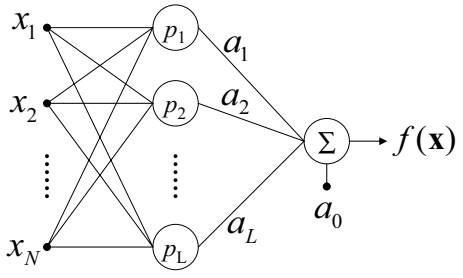
- Man kann nun zeigen, dass mit der folgenden Summe jede Funktion f(x) für hinreichend große L beliebig genau approximiert werden kann:

$$f(\mathbf{x}) = \mathbf{a}^T \mathbf{p}(\mathbf{x}) = a_0 + \sum_{i=1}^{L} a_i p_i(\mathbf{x})$$

- Unbekannt sind die Parameter:

$$\{a_i\}, \{c_i\}, \{\sigma_i\}$$

RBF-Netzwerk als verallgemeinerter linearer Klassifikator



$$f(\mathbf{x}) = \mathbf{a}^T \mathbf{p}(\mathbf{x}) = \sum_{i=0}^{L} a_i p_i(\mathbf{x})$$

Beim Perceptron wird der Originalraum in Hyperebenen aufgeteilt. Der in den neuen Variablen p_i lineare Klassifikator unterteilt den Originalraum in Hyperspären, da die $p_i(\mathbf{x})$ nichtlineare Funktionen von **x** sind.
Der Polynomklassifikator läßt sich in der gleichen Art darstellen.

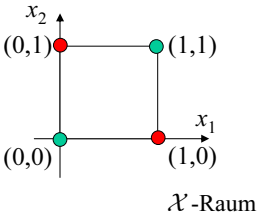
Lösung des XOR-Problems mit einem Gauss-RBFN

Wir wählen als Regressionsfunktion ein Gauss-RBFN der Dimension L=2:

$$f(\mathbf{x}) = \mathbf{a}^T \mathbf{p}(\mathbf{x}) = a_0 + \sum_{i=1}^{L} a_i p_i(\mathbf{x})$$

$$= a_0 + \sum_{i=1}^{L} a_i \exp\left(-\frac{1}{2\sigma_i^2} \|\mathbf{x} - \mathbf{c}_i\|^2 \right)$$

$$= a_0 + \sum_{i=1}^{L} a_i \exp\left(-\frac{(\mathbf{x} - c_i)^T (\mathbf{x} - c_i)}{2\sigma_i^2} \right)$$



$\mathcal{X}$-Raum

Mit $L = 2,$ den beiden Zentren

$\mathbf{c}_1 = [1 \quad 1]^T$ sowie $\mathbf{c}_2 = [0 \quad 0]^T$

sowie $\sigma_i = 1$ ergibt sich:

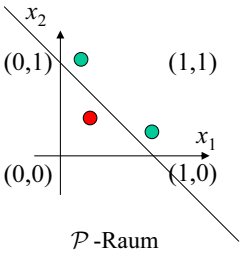
$$\mathbf{p}(\mathbf{x}) = \begin{bmatrix} \exp\left(-\|\mathbf{x} - \mathbf{c}_1\|^2\right) \\ \exp\left(-\|\mathbf{x} - \mathbf{c}_2\|^2\right) \end{bmatrix}$$

$\mathcal{X} \rightarrow \mathcal{P}$

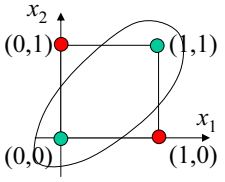
x_1	0	1	0	1
x_2	0	0	1	1
p_1	$e^{-2} \approx 0.135$	$e^{-1} \approx 0.368$	$e^{-1} \approx 0.368$	1
p_2	1	$e^{-1} \approx 0.368$	$e^{-1} \approx 0.368$	$e^{-2} \approx 0.135$

Lösung des XOR-Problems mit einem Gauss-RBFN

Die Vektoren $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ und $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ werden auf den gleichen Punkt abgebildet.



$\mathcal{P}$-Raum



$\mathcal{X}$-Raum

Im $\mathcal{P}$-Raum sind die beiden Klassen nun offensichtlich linear separierbar:

Trenngerade im $\mathcal{P}$-Raum: $f(\mathbf{p}) = p_1 + p_2 - 1 = 0$

Trennkurve im $\mathcal{X}$-Raum: $f(\mathbf{x}) = \exp\left(-\|\mathbf{x} - \mathbf{c}_1\|^2\right) + \exp\left(-\|\mathbf{x} - \mathbf{c}_2\|^2\right) - 1 = 0$

Eigenschaften von RBFNs:

- Vorteile:
  - Lokale Besonderheiten von Merkmalsclustern können sehr gut approximiert werden, ohne Qualitätsverlust bei der Generalisierungsfähigkeit, d.h. man hat ein sehr gutes Konvergenzverhalten.
- Nachteile:
  - Es handelt sich um einen in den Parametern nichtlinearen Entwurf, welcher nur iterativ gelöst werden kann mit all den negativen Begleiterscheinungen evtl. schlechter Konvergenz und der Gefahr nur Nebenminima und damit nur suboptimale Lösungen zu finden.

Demo mit MATLAB

- Funktionsapproximation mit RBFNs:
  - Gute Wahl der σ_i (generalisierfähig)
    (Radial basis approximation, demorb1.m)
  - Wahl der σ_i zu klein (overfitting, d.h. neu hinzukommende Punkte werden nicht erreicht)
    (Radial basis underlapping neurons, demorb3.m)
  - Wahl der σ_i zu groß; die Basisfunktionen spannen den Raum nur sehr schlecht auf, da sie nahezu linaer abhängig voneinander sind.
    (Radial basis overlapping neurons, demorb4.m)
- Klassifikationsbesipiel mit RBFNs (PNN classification, demopnn1.m).

Start der Matlab-Demo
matlab-RBFNs.bat