Chapter 9

Polynomial Classifier
and
Radial Base Function Networks (RBFN)

Approaches to designing a classifier

There are in principle two different approaches to designing a classifier:

1. Statistical parametrical modelling of class distributions, then MAP
2. Solving a map problem by approximating a function (non-linear regression)

$$\min E\left\{ \left\| \mathbf{f}(\mathbf{x}) - \mathbf{y} \right\|^2 \right\}$$

$\mathcal{X}$ – *feature space*

$\mathcal{Y}$ – *decision space*

Ad 1.) The procedure for designing a classifier described so far is based on approximating the *class specific distribution densities*

$$p(\mathbf{x}|\omega)$$

parametrically by statistical models (by estimating the parameters, e.g. a Gaussian distribution) and by deciding based on a maximum selection. Learning means in this case: improving the parameter-fitting.

Ad 2.) There is a second approach, which is based on evaluating the a-posteriori probability density

$$p(\omega|\mathbf{x})$$

and can be described by problem of *function approximation*.

This function approximation can be performed e.g. by a *non-linear regression with polynoms* or using a *artificial neural net* (NN). This lecture deals with the foundations of both approaches.

Basically the search for a best approximation function is a *variation problem* , which can be reduced to a *parametrical optimization problem* . Learning in this case also means: parameter-fitting.

The equality of both approaches results from the Bayes-theorem:

$$p(\omega_i \mid \mathbf{x}) = \frac{p(\mathbf{x} \mid \omega_i) P(\omega_i)}{\cancel{p(\mathbf{x})}}$$

$p(\mathbf{x})$ — independent on ω

Equality results from the denominatior being independent of ω .

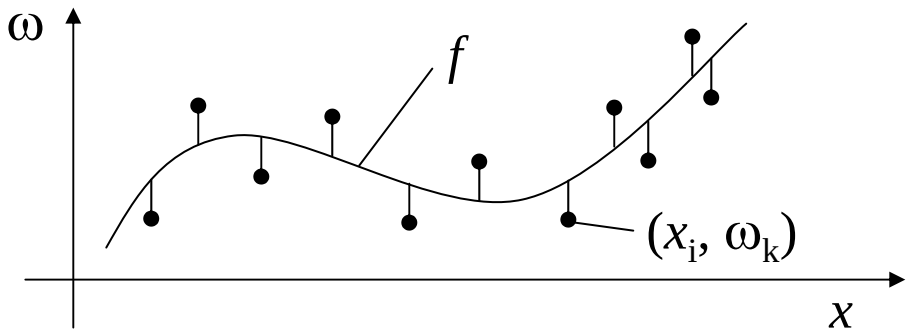
In the following the transfer into a function approximation problem is to be established.

With known a-posteriori-p. $p(\omega|\mathbf{x})$ for every continuous x a ω would be assigned to in the best possible way (functional map/assignment $f:\mathbf{x} \rightarrow \omega$). Since there are simply *samples* given, one searches for a function f, which fits the single experiments in the best possible way and therefore implements a map:

$$\mathbf{f}: \underset{\mathbf{x}}{\text{feature space}} \rightarrow \underset{\omega}{\text{category space}}$$

This task can be solved using a variation calculation. Choosing the minimal square error as a quality factor, it is about minimizing:

$$J = \min_{\mathbf{f}(\mathbf{x})} E\{\|\mathbf{f}(\mathbf{x}) - \mathbf{y}\|^2\}$$



Doing so the target vectors $\{\mathbf{y}_i\}$ in the *decision space* $\mathcal{Y}$ result by simple map of the scalar labels $\{\omega_i\}$

$$\Omega := \{\omega_1, \omega_2, \ldots, \omega_K\}$$

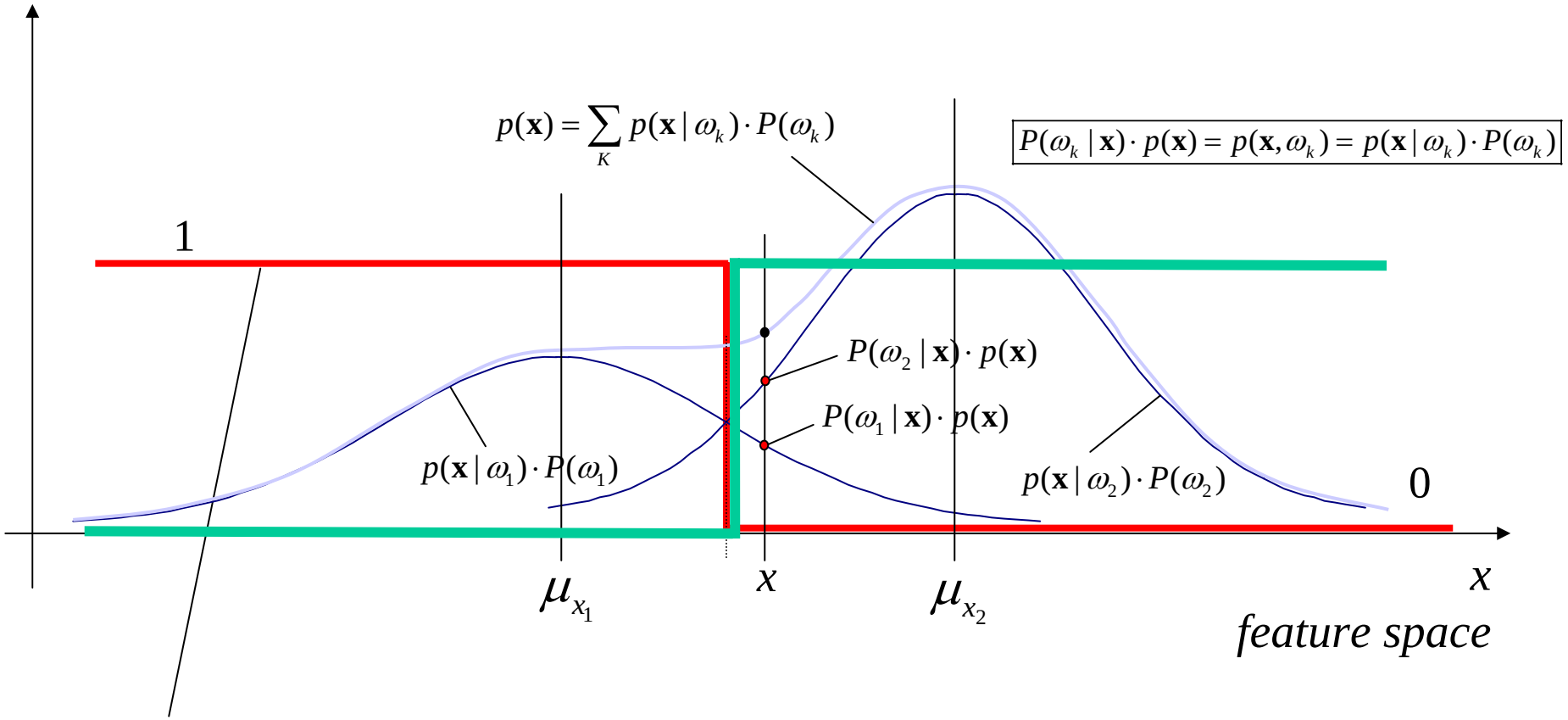
$$\updownarrow \quad \updownarrow \quad \updownarrow$$

$$\mathcal{Y} := \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_K\}$$

with:

$$\mathbf{y}_i = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad \text{i-th unit vector}$$

Two-class problem with Gaussian distribution density



Using the definition equation results:

$$J(\mathbf{f}^* + \delta\mathbf{f}) = E\left\{ \left\| \mathbf{f}^* + \delta\mathbf{f} - \mathbf{y} \right\|^2 \right\} = E\left\{ \langle (\mathbf{f}^* - \mathbf{y}) + \delta\mathbf{f}, (\mathbf{f}^* - \mathbf{y}) + \delta\mathbf{f} \rangle \right\}$$

$$= E\left\{ \left\| \mathbf{f}^* - \mathbf{y} \right\|^2 \right\} + 2E\left\{ \delta\mathbf{f}^T (\mathbf{f}^* - \mathbf{y}) \right\} + E\left\{ \left\| \delta\mathbf{f} \right\|^2 \right\}$$

$$\Rightarrow J(\mathbf{f}^*) = E\left\{ \left\| \mathbf{f}^* - \mathbf{y} \right\|^2 \right\} \quad (\text{für } \delta\mathbf{f} = \mathbf{0})$$

Inserting into the inequality results in:

$$E\left\{ \left\| \delta\mathbf{f} \right\|^2 \right\} + 2E\left\{ \delta\mathbf{f}^T (\mathbf{f}^* - \mathbf{y}) \right\} > 0 \qquad \forall \delta\mathbf{f} \neq 0$$

This is satisfied, when the second term disappears,

because the first term is positive definite (see next slide!).

From that results a sufficient optimality constraint:

$$E\left\{ \delta\mathbf{f}^T (\mathbf{f}^* - \mathbf{y}) \right\} = 0$$

The expected value can be written out in full with the joint distribution density as follows:

$$E\left\{\delta\mathbf{f}^T(\mathbf{f}^* - \mathbf{y})\right\} = \sum_{\mathbf{x}}\sum_{\mathbf{y}} \delta\mathbf{f}^T(\mathbf{f}^* - \mathbf{y}) \cdot p(\mathbf{x},\mathbf{y})$$

$$= \sum_{\mathbf{x}} \delta\mathbf{f}^T \left[\sum_{\mathbf{y}} (\mathbf{f}^* - \mathbf{y}) \cdot p(\mathbf{y}\,|\,\mathbf{x})\right] p(\mathbf{x}) = \mathbf{0}$$

This term disappears, if the term in squared brackets is null.

From that results the following optimality constraint:

$$\sum_{\mathbf{y}} (\mathbf{f}^* - \mathbf{y}) \cdot p(\mathbf{y}\,|\,\mathbf{x}) = \mathbf{f}^* \sum_{\mathbf{y}} p(\mathbf{y}\,|\,\mathbf{x}) - \sum_{\mathbf{y}} \mathbf{y} \cdot p(\mathbf{y}\,|\,\mathbf{x})$$

$$= \mathbf{f}^* - \sum_{\mathbf{y}} \mathbf{y} \cdot p(\mathbf{y}\,|\,\mathbf{x}) = \mathbf{0}$$

The optimal estimation function is the so-called regression function:

$$\mathbf{f}^*(\mathbf{x}) = \sum_{\mathbf{y}} \mathbf{y} \cdot p(\mathbf{y}\,|\,\mathbf{x}) = E\left\{\mathbf{y}\,|\,\mathbf{x}\right\}$$

Polynomial regression

Choosing polynomials as base functions for function approximation.

First of all for a scalar function with vectorial argument:

$$f(\mathbf{x}) = a_0 + a_1 x_1 + a_2 x_2 + \ldots + a_N x_N$$
$$+ a_{N+1} x_1^2 + a_{N+2} x_1 x_2 + a_{N+3} x_1 x_3 + \ldots \qquad \text{with:} \quad N = \dim(\mathbf{x})$$
$$+ a_{\ldots} x_1^3 + a_{\ldots} x_1^2 x_2 + a_{\ldots} x_1^2 x_3 + \ldots$$

This generalized polynom contains a constant a_0, followed by N linear terms, $N(N+1)/2$ square terms and so on.

A polynom of *degree G* and *dimension N* of the argument vector **x** has

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

polynomial terms, which can be viewed as base functions $f_i(\mathbf{x})$, i=1,2,...,L of a function developement.

The polynom above can be described compactly by introducing a vectorial map of a N- dimensional vector $\mathbf{x}$ to a L-dimensional vector $\mathbf{p}$:

$$\mathbf{p}(\mathbf{x}) = \begin{bmatrix} 1 & x_1 & x_2 & \dots & x_N & x_1^2 & x_1x_2 & x_1x_3 & \dots & x_1^3 & x_1^2x_2 & x_1^2x_3 & \dots \end{bmatrix}^T$$

By introducing a coefficient vector $\mathbf{a}$ results for the scalar polynom :

$$\boxed{f(\mathbf{x}) = \mathbf{a}^T\mathbf{p}(\mathbf{x}) = \sum_{i=0}^{L} a_i\, p_i(\mathbf{x})} \quad \text{with: } L = \dim(\mathbf{p})$$

For the pattern classification we need a polynom function for each class

$$f_k(\mathbf{x}) = \mathbf{a}_k^T\mathbf{p}(\mathbf{x}) \quad \text{responsible for classes}$$
$$k = 1, 2, \dots, K$$

Combining the class specific coefficient vectors $\mathbf{a}_k$ to a matrix

$$\mathbf{A} = \begin{bmatrix} \mathbf{a}_1 & \mathbf{a}_2 & \dots & \mathbf{a}_K \end{bmatrix}$$

results for the vectorial polynom function:

$$\boxed{\mathbf{f}(\mathbf{x}) = \mathbf{A}^T\mathbf{p}(\mathbf{x})}$$

Optimal adjustment of matrix **A**

Now the coefficient matrix A has to be adjusted in the best possible way while optimizing.
The approximation problem can be applied directly to the measures or to features of a subspace (e.g. after a KLT).
Adaption of the coefficient matrix **A**:

$$J = \underbrace{\min}_{\mathbf{f}(\mathbf{x})} E\left\{ \left\| \mathbf{f}(\mathbf{x}) - \mathbf{y} \right\|^2 \right\} \approx \underbrace{\min}_{\mathbf{A}} E\left\{ \left\| \mathbf{A}^T \mathbf{p}(\mathbf{x}) - \mathbf{y} \right\|^2 \right\}$$

The variation problem of searching for an unknown function is reduced to a parameter optimization problem by choosing polynomial base functions!

The quality criterion *J* minimizes the variance of the residual, that is the error between y and the approximation $\mathbf{A}^T\mathbf{p}(\mathbf{x})$.

The solution is determined using the variation approach, which leads to a neccessary constraint:

$$J(\mathbf{A}^* + \delta\mathbf{A}) \geq J(\mathbf{A}^*) \qquad \text{for} \quad \forall \delta\mathbf{A} \neq \mathbf{0}$$

Now it is to adjust the coefficient matrix **A** in the best possible way by a optimizing procedure.

Adaption of the coefficient matrix **A**:

Due to

$$J(\mathbf{A}) = E\left\{ \left\| \mathbf{A}^T\mathbf{p}(\mathbf{x}) - \mathbf{y} \right\|^2 \right\} = E\left\{ \left[\mathbf{A}^T\mathbf{p} - \mathbf{y} \right]^T \left[\mathbf{A}^T\mathbf{p} - \mathbf{y} \right] \right\}$$

and $\underbrace{\mathbf{a}^T\mathbf{b}}_{\text{scalar product}} = \text{trace}(\underbrace{\mathbf{b}\mathbf{a}^T}_{\substack{\text{dyad.} \\ \text{product}}})$

and $\text{trace}(\mathbf{PQ}) = \text{trace}(\mathbf{QP})$

results:

$$J(\mathbf{A}) = E\left\{ \text{trace}\left[\left[\mathbf{A}^T\mathbf{p} - \mathbf{y} \right] \left[\mathbf{A}^T\mathbf{p} - \mathbf{y} \right]^T \right] \right\}$$

$$= \text{trace}\left[E\left\{ \mathbf{y}\mathbf{y}^T \right\} \right] - 2\,\text{trace}\left[\mathbf{A}^T E\left\{ \mathbf{p}\mathbf{y}^T \right\} \right] + \text{trace}\left[\mathbf{A}^T E\left\{ \mathbf{p}\mathbf{p}^T \right\} \mathbf{A} \right]$$

$$= E\left\{ \left\| \mathbf{y} \right\|^2 \right\} - 2\,\text{trace}\left[\mathbf{A}^T E\left\{ \mathbf{p}\mathbf{y}^T \right\} \right] + \text{trace}\left[\mathbf{A}^T E\left\{ \mathbf{p}\mathbf{p}^T \right\} \mathbf{A} \right]$$

As assumed before, the target vector can be w.l.o.g. formulated as unit vector and thus applies:

$$J(\mathbf{A}) = 1 - 2\,\mathrm{trace}\left[\mathbf{A}^T E\left\{\mathbf{p}\mathbf{y}^T\right\}\right] + \mathrm{trace}\left[\mathbf{A}^T E\left\{\mathbf{p}\mathbf{p}^T\right\}\mathbf{A}\right]$$

With the variation approach results:

$$\begin{aligned}
J(\mathbf{A}^* + \delta\mathbf{A}) &= 1 - 2\,\mathrm{trace}\left[(\mathbf{A}^* + \delta\mathbf{A})^T E\left\{\mathbf{p}\mathbf{y}^T\right\}\right] \\
&\quad + \mathrm{trace}\left[(\mathbf{A}^* + \delta\mathbf{A})^T E\left\{\mathbf{p}\mathbf{p}^T\right\}(\mathbf{A}^* + \delta\mathbf{A})\right] \\
&= 1 - 2\,\mathrm{trace}\left[\mathbf{A}^{*T} E\left\{\mathbf{p}\mathbf{y}^T\right\}\right] - 2\,\mathrm{trace}\left[\delta\mathbf{A}^T E\left\{\mathbf{p}\mathbf{y}^T\right\}\right] \\
&\quad + \mathrm{trace}\left[\mathbf{A}^{*T} E\left\{\mathbf{p}\mathbf{p}^T\right\}\mathbf{A}^*\right] + 2\,\mathrm{trace}\left[\delta\mathbf{A}^T E\left\{\mathbf{p}\mathbf{p}^T\right\}\mathbf{A}^*\right] \\
&\quad + \mathrm{trace}\left[\delta\mathbf{A}^T E\left\{\mathbf{p}\mathbf{p}^T\right\}\delta\mathbf{A}\right]
\end{aligned}$$

Inserting the inequation of the neccessary constraint for an optimun results in:

$$\mathrm{trace}\left[\delta\mathbf{A}^T E\left\{\mathbf{p}\mathbf{p}^T\right\}\delta\mathbf{A}\right] - 2\,\mathrm{trace}\left[\delta\mathbf{A}^T\left(E\left\{\mathbf{p}\mathbf{y}^T\right\} - E\left\{\mathbf{p}\mathbf{p}^T\right\}\mathbf{A}^*\right)\right] \geq 0$$

Due to the positive definiteness of the first term and the moment matrix $E\{\mathbf{p}\mathbf{p}^T\}$, the second term has to disappear:

$$\text{trace}\left[\delta\mathbf{A}^T\left(E\left\{\mathbf{p}\mathbf{y}^T\right\}-E\left\{\mathbf{p}\mathbf{p}^T\right\}\mathbf{A}^*\right)\right]=0 \qquad \forall\delta\mathbf{A}$$

and thus the equation linear in **A*** in for the optimal polynomial classifier:

$$\boxed{E\left\{\mathbf{p}\mathbf{p}^T\right\}\mathbf{A}^*=E\left\{\mathbf{p}\mathbf{y}^T\right\}}$$

this equation guarantees a minimal residuum of the error vector:

$$\Delta\mathbf{f}(\mathbf{x})=\mathbf{f}(\mathbf{x})-\mathbf{y}=\mathbf{A}^{*T}\mathbf{p}(\mathbf{x})-\mathbf{y}$$

Estimation of both moment matrices $E\{\mathbf{p}\mathbf{p}^T\}$ and $E\{\mathbf{p}\mathbf{y}^T\}$ from the training set of data and solving a linear matrix equation leads to the solution of the problem.

The remaining error vector calculates as:

$$\boxed{J(\mathbf{A}^*)=1-\text{trace}\left[\mathbf{A}^{*T}E\left\{\mathbf{p}\mathbf{p}^T\right\}\mathbf{A}^*\right]}$$

Orthogonality of the estimate error vector

The best approximation requires according to the projection theorem, that the error vector $\Delta\mathbf{f}$ is orthogonal to the subspace, which is spanned by the polynomial base functions in $\mathbf{p}(\mathbf{x})$.

That means in statistical notation, that the moment matrix, which is spanned by $\mathbf{p}$ and $\Delta\mathbf{f}$, must be a zero matrix:

$$E\left\{ \mathbf{p}\Delta\mathbf{f}^T \right\} = E\left\{ \mathbf{p}\left(\mathbf{A}^{*T}\mathbf{p} - \mathbf{y}\right)^T \right\} = E\left\{ \mathbf{p}\mathbf{p}^T \right\}\mathbf{A}^* - E\left\{ \mathbf{p}\mathbf{y}^T \right\} = \mathbf{0}$$

This property is inherited from $\mathbf{p}$ to $\mathbf{f}$, since $\mathbf{f}$ is built from linear combinations of the base functions in $\mathbf{p}(\mathbf{x})$.

$$E\left\{ \mathbf{f}^*\Delta\mathbf{f}^T \right\} = E\left\{ \mathbf{A}^{*T}\mathbf{p}\Delta\mathbf{f}^T \right\} = \mathbf{A}^{*T}E\left\{ \mathbf{p}\Delta\mathbf{f}^T \right\} = \mathbf{0}$$

Unbiasedness of the approximation function

The estimate function is unbiased. The estimate error (residuum) has expected value **0**.

$$E\left\{\Delta\mathbf{f}\right\} = E\left\{\mathbf{f}(\mathbf{x}) - \mathbf{y}\right\} = \mathbf{0}$$

From that follows, that **f**(**x**) is an unbiased estimate of **y**:

$$E\left\{\mathbf{f}(\mathbf{x})\right\} = E\left\{\mathbf{y}\right\}$$

Recursive learning rule for the polynom classifier

(in case new values can be added in the experiment)

For the classifier that is built on the minimal error square results the following recursive learning strategy for the coefficient matrix **A**:

$$\mathbf{A}_n = \mathbf{A}_{n-1} - \alpha \left[\sum_{i=1}^{n} \mathbf{p}_i \mathbf{p}_i^T \right]^{-1} \mathbf{p}_n \left[\mathbf{A}_{n-1}^T \mathbf{p}_n - \mathbf{y}_n \right]^T$$

This learning rule consists of the following steps:

- Choosing a starting value for **A**
- The current sample $[\mathbf{x}, \mathbf{y}]$ is mapped using the polynom base $\mathbf{p}(\mathbf{x})$ to $[\mathbf{p}, \mathbf{y}]$
- Calculate the estimate **f** based on the given observations **p** and the current coefficient matrix **A**: $\mathbf{f} = \mathbf{A}^T \mathbf{p}$
- Calculate the residuum: $\Delta \mathbf{f} = \mathbf{f} - \mathbf{y}$
- Adjust $\mathbf{A}_n$ based on the recursion above

$$\mathbf{A}_n = \mathbf{A}_{n-1} - \alpha \mathbf{G}^{-1} \mathbf{p}_n \Delta \mathbf{f}^T$$

The weight matrix **G** is in this case the moment matrix $E\{\mathbf{p}\mathbf{p}^T\}$ based on the *n* available samples.

A strict consideration of the recursion above would require another recursion for the inverse of the weight matrix according to:

$$\mathbf{G}_n^{-1} = \frac{1}{1-\alpha}\left[\mathbf{G}_{n-1}^{-1} - \alpha\frac{\mathbf{G}_{n-1}^{-1}\mathbf{p}_n\mathbf{p}_n^T\mathbf{G}_{n-1}^{-1}}{1+\alpha\left(\mathbf{p}_n^T\mathbf{G}_{n-1}^{-1}\mathbf{p}_n - 1\right)}\right]$$

G or the true additional iteration don‘t really change the result. G can even be simplified to:

$$\mathbf{G} = E\left\{\ \mathbf{p}\mathbf{p}^T\right\} = \sum_{i=1}^{n}\mathbf{p}_i\mathbf{p}_i^T$$

without impairment of convergence properties (Schürmann p. 174).

Derivation of the recursive learnimg rule

$E\left\{\mathbf{p}\mathbf{p}^T\right\}\mathbf{A} = E\left\{\mathbf{p}\mathbf{y}^T\right\}$ opt. solution of the polynomkl.

Introducing estimate values:

$$\underbrace{\left(\frac{1}{n}\sum_{i=1}^{n}\mathbf{p}_i\mathbf{p}_i^T\right)}_{\mathbf{G}_n}\mathbf{A}_n = \underbrace{\frac{1}{n}\sum_{i=1}^{n}\mathbf{p}_i\mathbf{y}_i^T}_{\mathbf{H}_n}$$

$\mathbf{G}_n\mathbf{A}_n = \mathbf{H}_n$ and

$$\mathbf{G}_n = (1-\alpha)\mathbf{G}_{n-1} + \alpha\mathbf{p}_n\mathbf{p}_n^T$$
$$\mathbf{H}_n = (1-\alpha)\mathbf{H}_{n-1} + \alpha\mathbf{p}_n\mathbf{y}_n^T$$

$$\Rightarrow \mathbf{G}_n\mathbf{A}_n = (1-\alpha)\mathbf{H}_{n-1} + \alpha\mathbf{p}_n\mathbf{y}_n^T$$
$$= (1-\alpha)\mathbf{G}_{n-1}\mathbf{A}_{n-1} + \alpha\mathbf{p}_n\mathbf{y}_n^T$$
$$= \left(\mathbf{G}_n - \alpha\mathbf{p}_n\mathbf{p}_n^T\right)\mathbf{A}_{n-1} + \alpha\mathbf{p}_n\mathbf{y}_n^T$$
$$= \mathbf{G}_n\mathbf{A}_{n-1} - \alpha\mathbf{p}_n(\mathbf{p}_n^T\mathbf{A}_{n-1} - \mathbf{y}_n^T)$$
$$\Rightarrow \boxed{\mathbf{A}_n = \mathbf{A}_{n-1} - \alpha\mathbf{G}_n^{-1}\mathbf{p}_n\left[\mathbf{A}_{n-1}^T\mathbf{p}_n - \mathbf{y}_n\right]^T}$$

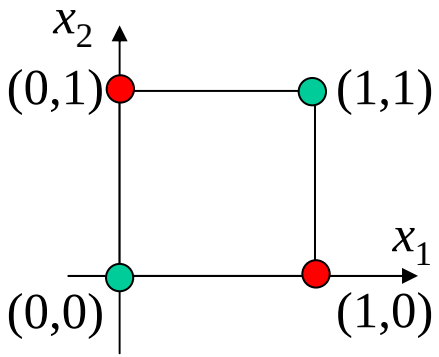
Polynom classifier for the XOR-problem

We choose a polynom with terms of second degree as regression function hoping to find a solution by that (a linear solution does not exist!):

$$f(\mathbf{x}) = a_1 x_1 + a_2 x_2 + a_3 x_1 x_2 = \mathbf{a}^T \mathbf{p}(\mathbf{x}) = \sum_{i=0}^{L} a_i p_i(\mathbf{x})$$

with:

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

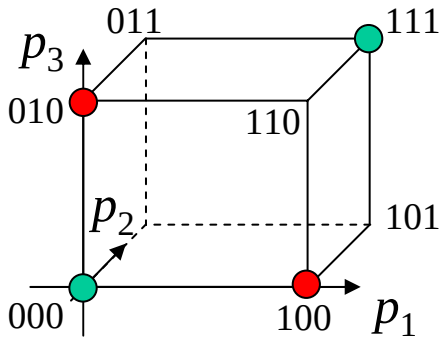


$\mathcal{X}$ -space

The vertices of the square in the two-dimensional origin space $\mathcal{X}$ are mapped to the vertices of a (hyper-) cube of a three-dimensional feature space $\mathcal{P}$ and this space finally to a one-dimensional category space $\mathcal{Y}$:

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

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



$\mathcal{P}$ -space

The optimal parameter of the regression function **a*** result from:

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

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

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

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

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

Thus follows:

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

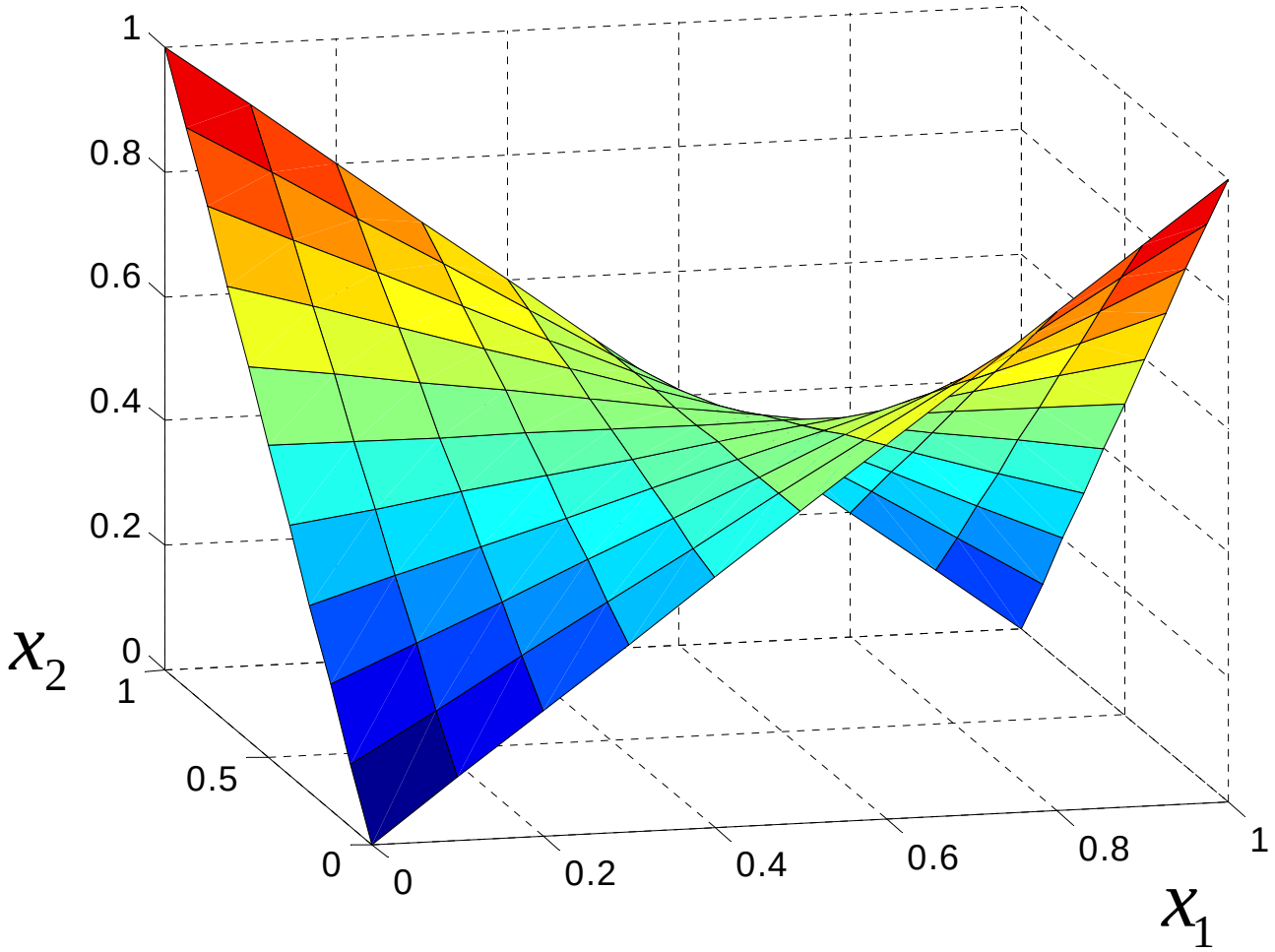
and thus a regression function

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

which takes exactly the values of the target function at the 4 function values and interpolates otherwise!

Regression function

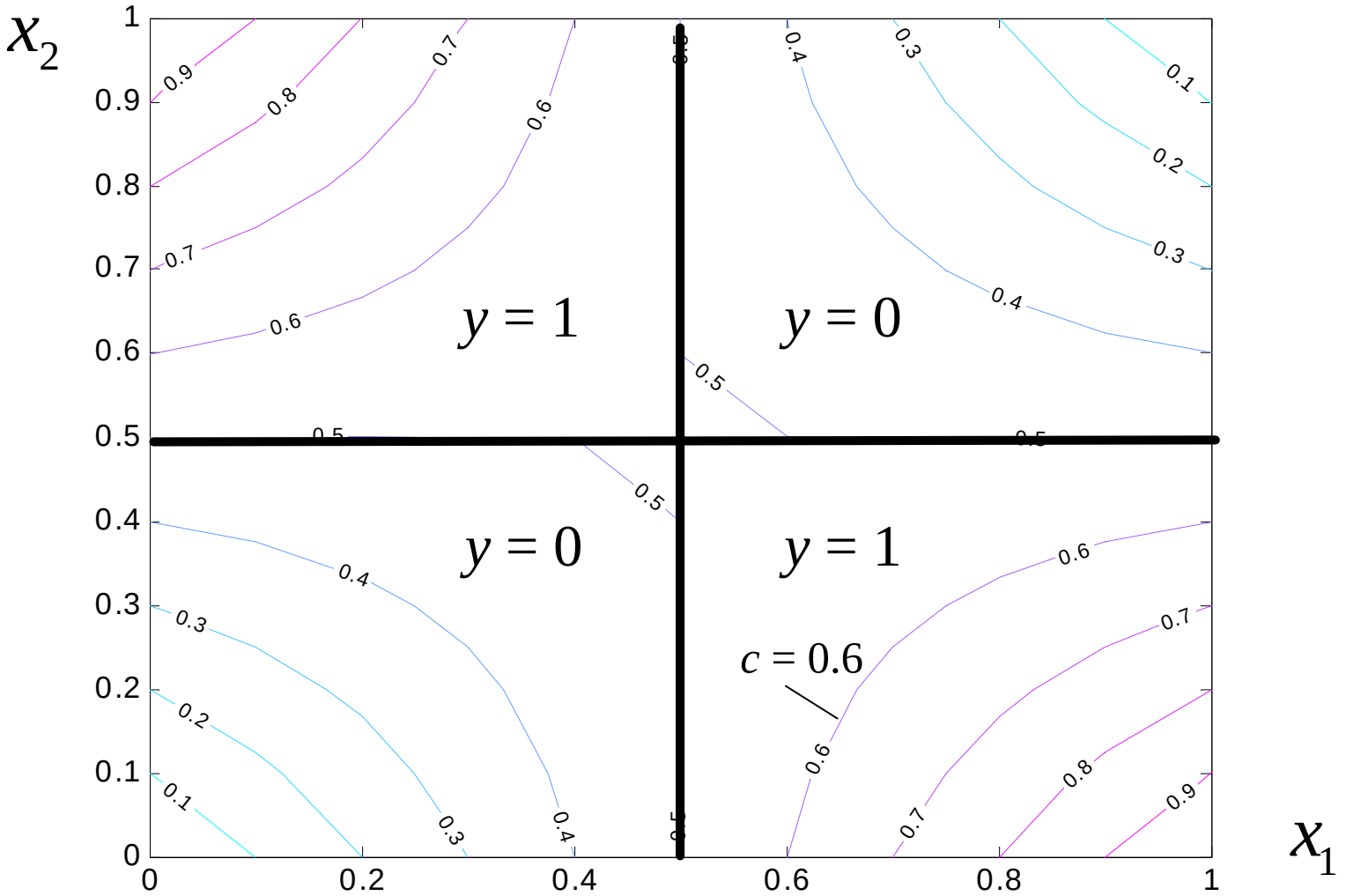
$$f(\mathbf{x}) = x_1 + x_2 - 2x_1x_2$$



Matlab-demo: XOR_poly.bat

Contours of the separation lines in the original space (hyperbolas) for different thresholds c

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



Separation areas for both classes

We obtain a

- *linear* classifier in the **p**-space

and a

- *non-linear* classifier in the **x**-space

Polynom classifier for a slightly more complex example

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

cartesian product of the one-dimensional polynoms with:

$\dim(\mathbf{x}) = N$ and degree of polynom: G

i.e. the N-dimensional origin space is mapped to a L-dimensional feature space, which contains terms like:

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

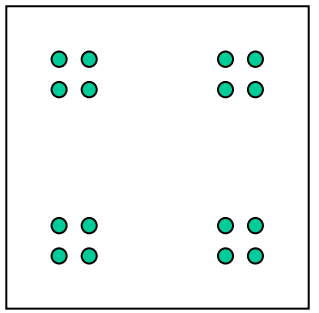
For a polynom of degree G and a feature vector of dimension N results a dimension of the output space L of:

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

Demo with MATLAB
(Klassifikationgui.m)

- Open Matlab
  - first call setmypath.m, then
  - C:\Home\ppt\Lehre\ME_2002\matlab\KlassifikatorEntwurf-WinXX\Klassifikationgui
  - Set of data: load samples/xor_test_exakt.mat
  - use polynom classifier of degree 2
- Two-class problem with banana shape
  - Set of data: load samples/banana_train_samples.mat
  - Experiment with polynom classifier of degree 2 and degree 3

for obtaining enough independent samples for the polynom :



Starting Matlab-demo
matlab-Klassifikation_gui.bat

Properties of the polynom classifiers:

- Advantages:
  - Linear design equations with explicit solution for a global minimun of the approximation error
- Disadvantages:
  - The developement using Polynomial-basis functions with global domain shows poor convergence. Local specifics of feature clusters can be approximated only very poorly, otherwise they lead to a loss of generalization ability.

Function approximation with a radial-base-function-network (RBFN)

- Instead of using polynoms for the regression task, a linear developement in radial base function (RBF) can be considered. Although another non-linear parameter estimation problem results (just like for NNs), which typically can only be solved iteratively, better convergence properties result.
- RBFs as interpolation functions are functions of form:

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

- i.e. the argument of each RBF is the Euclidean distance between input vector $\mathbf{x}$ and a center $\mathbf{c}_i$ (which explains the name. The RBFs have an effect only in *local* enviroments (as opposed to polynoms).

Example for radial-base-functions (RBFN)

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

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

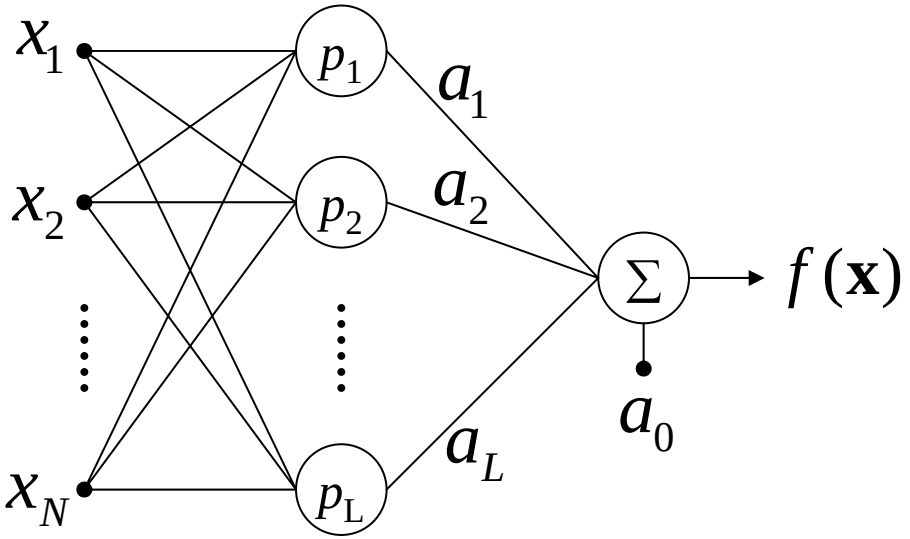
- It can be shown, that each function f(x) can be approximated arbitrarily exact for sufficiently big L with the following sum:

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

- Unknown are the parameters:

$$\{a_i\}, \{\mathbf{c}_i\}, \{\sigma_i\}$$

RBF-network as a generalized linear classifier



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

The original space is divided into *hyper planes* for the perceptron.

RBF-network: The classifier that is linear in the new variables p_i divides the original space into *hyper spheres* , since the $p_i(\mathbf{x})$ are non-linear function of **x**.
The polynom classifier can be characterized in the same way.

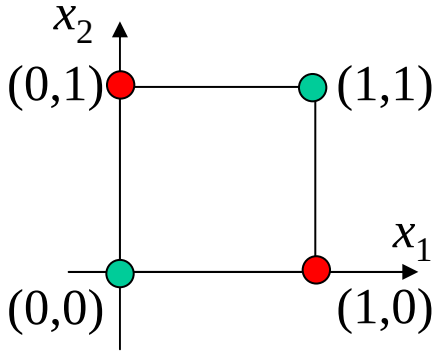
Solving the XOR-problem with Gaussian-RBFN

We choose a Gaussian-RBFN of dimension L=2 as regression function:

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

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

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



$\mathcal{X}$-space

With $L = 2,$ the two centers

$\mathbf{c}_1 = \begin{bmatrix} 1 & 1 \end{bmatrix}^T$ and $\mathbf{c}_2 = \begin{bmatrix} 0 & 0 \end{bmatrix}^T$

and $\sigma_i = 1$ results:

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

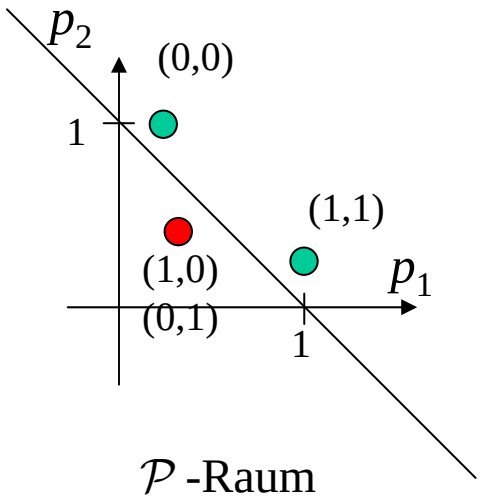
$\mathcal{X} \rightarrow \mathcal{P}$

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	$e^{-1} \approx 0.368$	$e^{-1} \approx 0.368$	$e^{-2} \approx 0.135$
$f(\mathbf{p})$	0.135	−0.264	−0.264	0.135

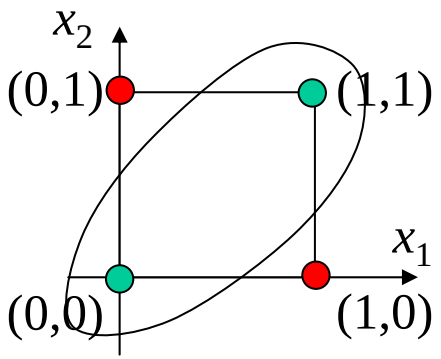
$$f(\mathbf{p}) = p_1 + p_2 - 1$$

Solving the XOR-problem with Gaussian-RBFN

The vectors $\mathbf{x} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ and $\mathbf{x} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ are mapped to the same point $\mathbf{p} = \begin{bmatrix} e^{-1} \\ e^{-1} \end{bmatrix}$.



$\mathcal{P}$ -Raum



$\mathcal{X}$ -Raum

In the $\mathcal{P}$-space both classes are obviously linearly separable:

separation line in $\mathcal{P}$ -space: $f(\mathbf{p}) = p_1 + p_2 - 1 = 0$

separation curve in $\mathcal{X}$ -space: $f(\mathbf{x}) = \exp\left(-\|\mathbf{x} - \mathbf{c}_1\|^2\right) + \exp\left(-\|\mathbf{x} - \mathbf{c}_2\|^2\right) - 1 = 0$

Properties of RBFNs:

- Advantages:
  - Local specifics of feature clusters can be approximated very good, without loss of quality for the generalization ability, i.e. very good convergence performance.
- Disadvantages:
  - Since the design is non-linear in the parameters, the problem can only be solved iteratively with all negative concomitants like bad convergance and the possibility to find only auxiliary minima and thus suboptimal solutions.

Demo with MATLAB

- Function approximation with RBFNs:
  (Demo/Toolboxes/Neural Network/Radial Basis Networks)
  - Good choice of σ_i (able to generalize)
    (Radial basis approximation, demorb1.m)
  - Choice of σ_i too small (overfitting, i.e. added point cannot be reached)
    (Radial basis underlapping neurons, demorb3.m)
  - Choice of σ_i too big; the base functions span the space not properly, since they are almost linearly dependent on each other.
    (Radial basis overlapping neurons, demorb4.m)
- Classification example with RBFNs (PNN classification, demopnn1.m).

Starting Matlab-Demo
matlab-RBFNs.bat