Basis Functions

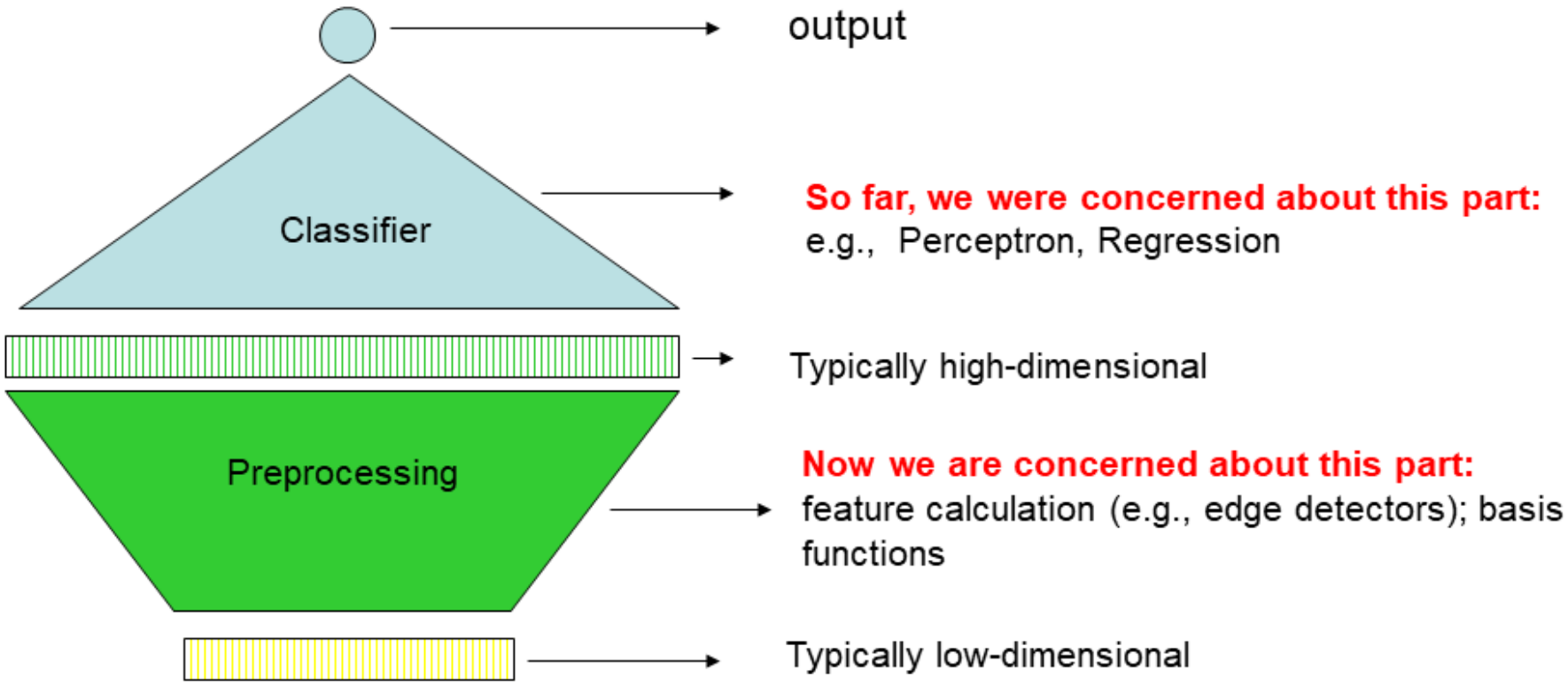
Volker Tresp
Summer 2025

Reference

- This material is related to Bishop, Deep Learning, Sections 4.11, 5.4.2

Nonlinear Mappings and Nonlinear Classifiers

- Regression:
  - Linearity is often a good assumption when many inputs influence the output
  - Some natural laws are (approximately) linear $F = ma$
  - But in general, it is rather unlikely that a true function is linear
- Classification:
  - Linear classifiers also often work well when many inputs influence the output
  - But also for classifiers, it is often not reasonable to assume that the classification boundaries are linear hyperplanes

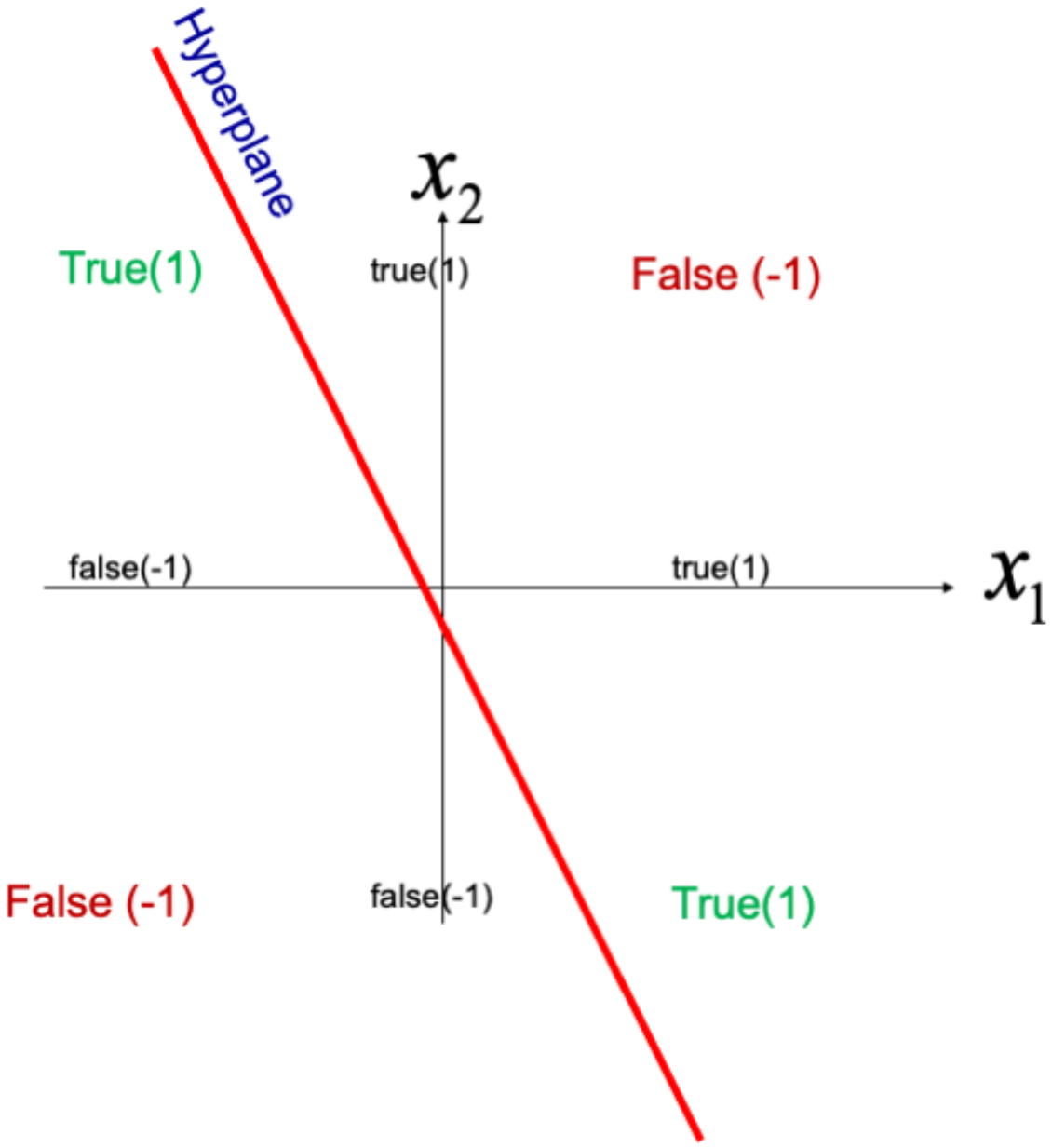

output
Classifier
So far, we were concerned about this part:
e.g., Perceptron, Regression
Typically high-dimensional
Preprocessing
Now we are concerned about this part:
feature calculation (e.g., edge detectors); basis functions
Typically low-dimensional

Trick

- We simply transform the input into a high-dimensional space where the regression/classification might again be linear!
- Other view: let's define appropriate features (feature engineering)
- Other view: let's define appropriate basis functions
- Challenge: XOR-type problem with patterns

$$\begin{array}{ccccc} -1 & -1 & \rightarrow & -1 \\ +1 & -1 & \rightarrow & +1 \\ -1 & +1 & \rightarrow & +1 \\ +1 & +1 & \rightarrow & -1 \end{array}$$

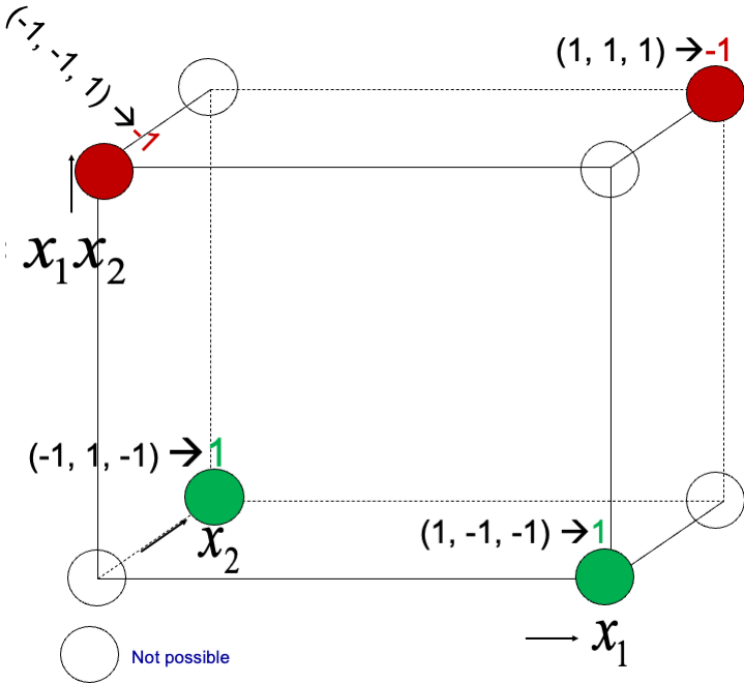
XOR-type problems are not Linearly Separable



Trick: Let's Add Basis Functions

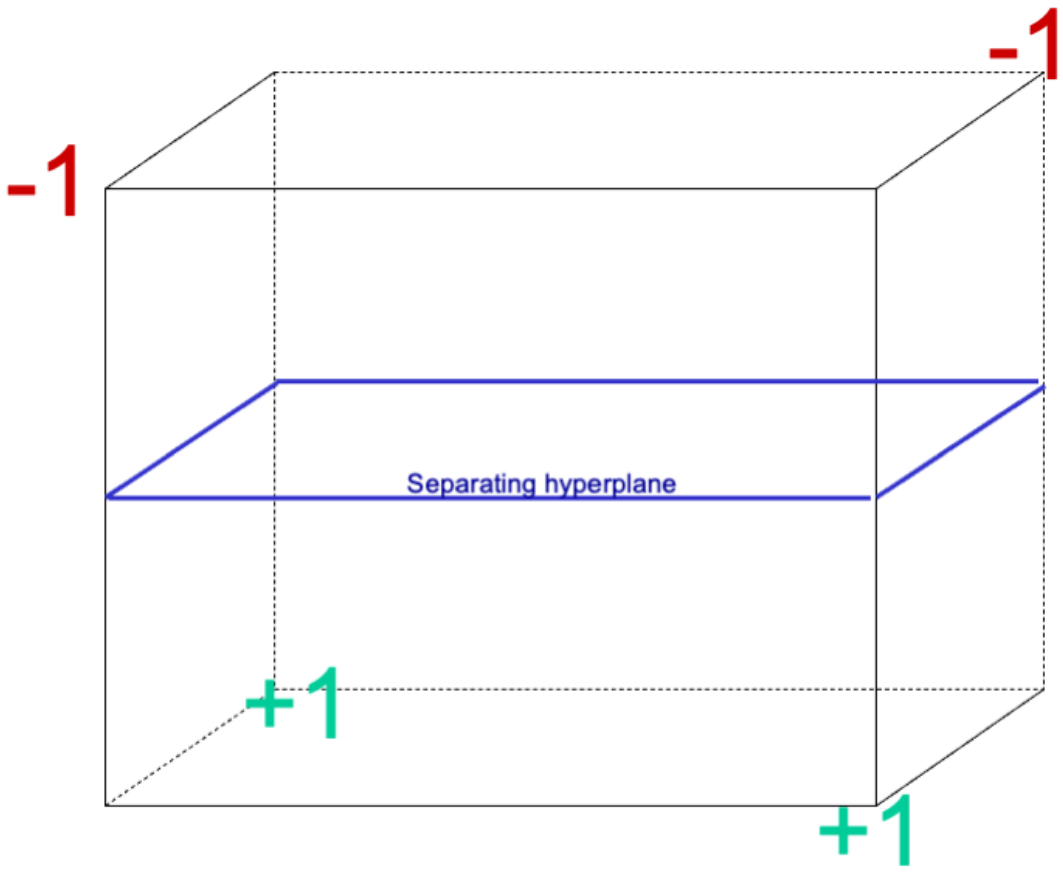
- Linear Model: input variables: x_1, x_2
- Let's consider the product $x_1 x_2$ as additional input
- The interaction term $x_1 x_2$ couples two inputs nonlinearly

With a Third Input x_1x_2 the XOR Becomes Linearly Separable

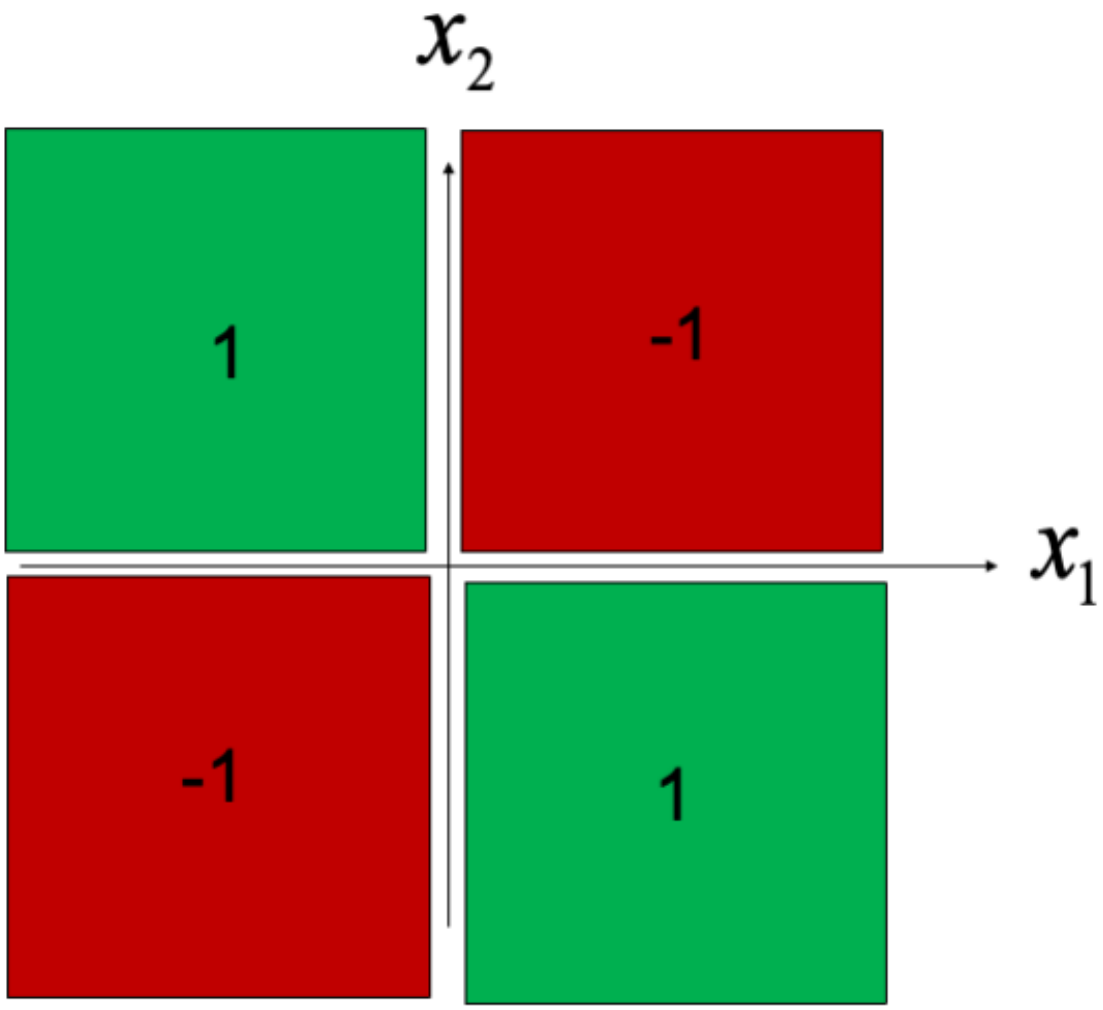


$$h(\mathbf{x}) = 0 \times \phi_1(\mathbf{x}) + 0 \times \phi_2(\mathbf{x}) + 0 \times \phi_3(\mathbf{x}) - 1 \times \phi_4(\mathbf{x})$$

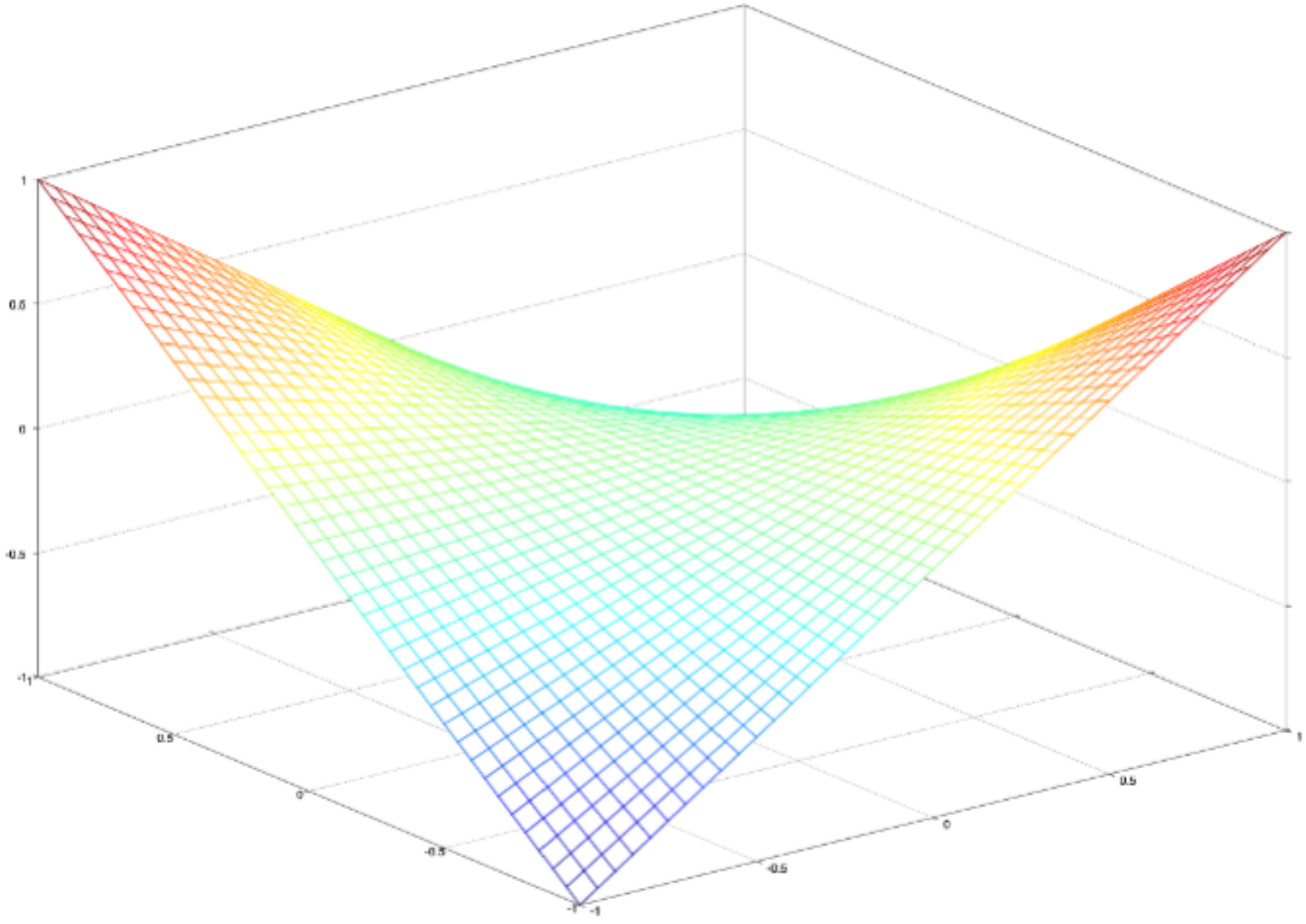
with $\phi_1(\mathbf{x}) = 1, \phi_2(\mathbf{x}) = x_1, \phi_3(\mathbf{x}) = x_2, \phi_4(\mathbf{x}) = x_1x_2$

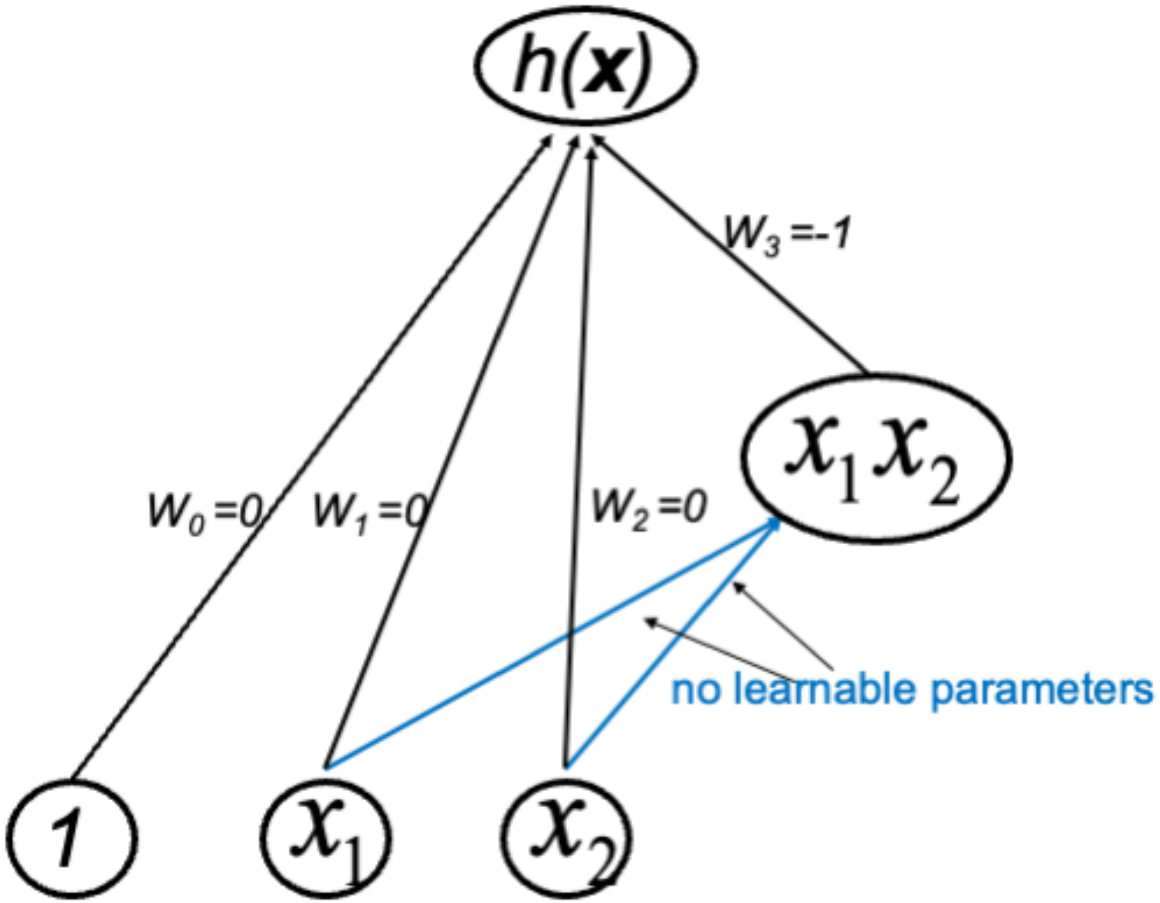

-1
-1
Separating hyperplane
+1
+1

Class Separation (after sign)

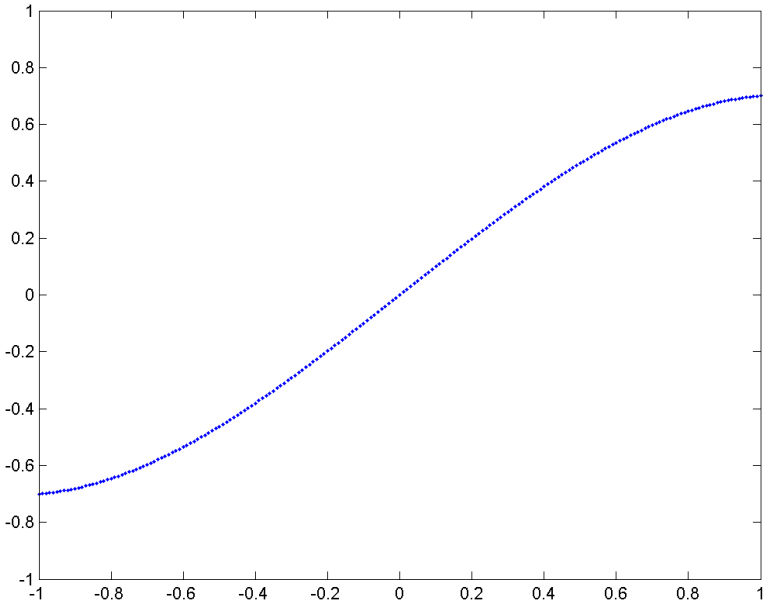


$$h(x_1,x_2) = - x_1 x_2$$

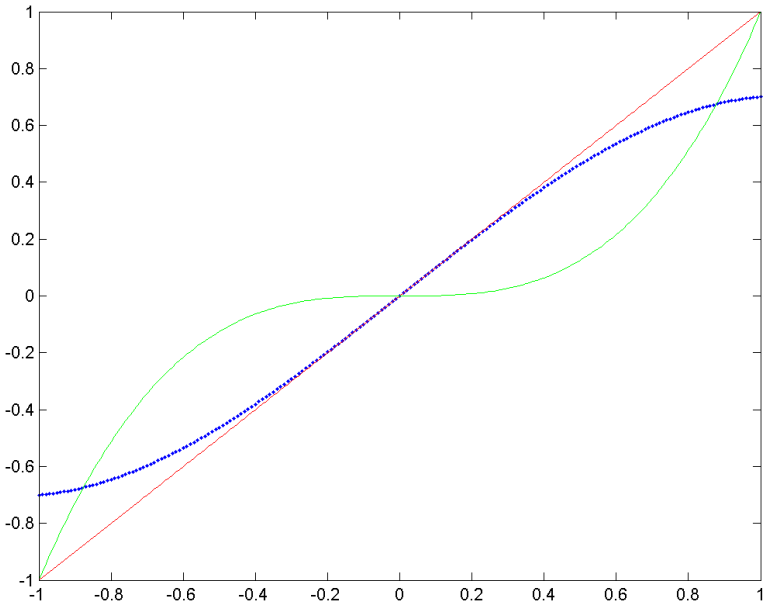



h(x)
$w_3 = -1$
$x_1 x_2$
$w_0 = 0$
$w_1 = 0$
$w_2 = 0$
no learnable parameters
1
x_1
x_2

A Nonlinear Function



$$f(x) = x - 0.3x^3$$



Basis functions $\phi_1(x) = 1, \phi_2(x) = x, \phi_3(x) = x^2, \phi_4(x) = x^3$ und $\mathbf{w} = (0, 1, 0, -0.3)$

Basic Idea

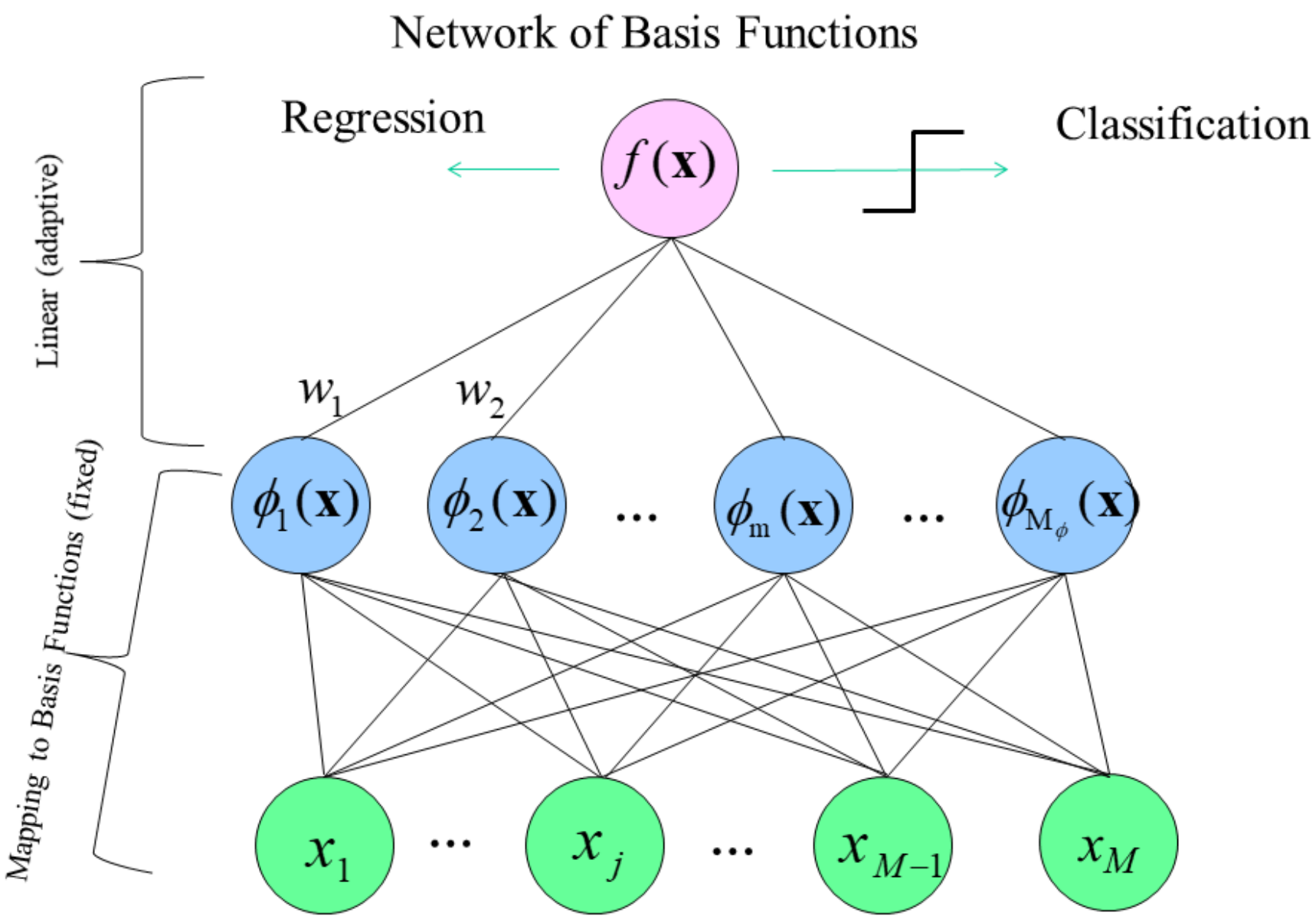
- The simple idea: in addition to the original inputs, we add inputs that are calculated as deterministic functions of the existing inputs, and treat them as additional inputs
- Example: Polynomial Basis Functions

$$\{1, x_1, x_2, x_3, x_1x_2, x_1x_3, x_2x_3, x_1^2, x_2^2, x_3^2\}$$

- Basis functions $\{\phi_m(\mathbf{x})\}_{m=1}^{M_\phi}$
- In the example:

$$\phi_1(\mathbf{x}) = 1 \quad \phi_2(\mathbf{x}) = x_1 \quad \phi_6(\mathbf{x}) = x_1x_3 \quad ...$$

- Independent of the choice of basis functions, the regression parameters are calculated using the well-known equations for linear regression


Network of Basis Functions
Regression
Classification
$f(\mathbf{x})$
Linear (adaptive)
Mapping to Basis Functions (fixed)
w_1
w_2
$\phi_1(\mathbf{x})$
$\phi_2(\mathbf{x})$
...
$\phi_m(\mathbf{x})$
...
$\phi_{M_\phi}(\mathbf{x})$
x_1
...
x_j
...
x_{M-1}
x_M

The Constant Basis Function

- Note that we write $\phi_1(\mathbf{x}) = 1$, so the first basis function is the constant (if required)
- Thus, for the rest of the discussion, the number of basis functions M_ϕ is identical to the number of adaptable parameters M_p, i.e.,

$$M_\phi = M_p$$

  Typically, $M_\phi >> M$, where M is the number of inputs

Linear Model Written as Basis Functions

- We can also write a linear model as a sum of basis functions with

$$\phi_1(\mathbf{x}) = 1, \quad \phi_2(\mathbf{x}) = x_1, \quad \ldots \quad \phi_{M_\phi}(\mathbf{x}) = x_M$$

Review: Penalized LS for Linear Regression

- Multiple Linear Regression:

$$f_{\mathbf{w}}(\mathbf{x}) = w_0 + \sum_{j=1}^{M} w_j x_j = \mathbf{x}^T \mathbf{w}$$

- Regularized cost function

$$\text{cost}^{pen}(\mathbf{w}) = \sum_{i=1}^{N} (y_i - f_{\mathbf{w}}(\mathbf{x}_i))^2 + \lambda \sum_{j=0}^{M} w_j^2$$

- The penalized LS-Solution gives

$$\hat{\mathbf{w}}_{pen} = \left(\mathbf{X}^T\mathbf{X} + \lambda I\right)^{-1} \mathbf{X}^T \mathbf{y} \quad \text{with} \quad \mathbf{X} = \begin{pmatrix} x_{1,0} & \dots & x_{1,M} \\ \dots & \dots & \dots \\ x_{N,0} & \dots & x_{N,M} \end{pmatrix}$$

Regression with Basis Functions

- Model with basis functions:

$$f_{\mathbf{w}}(\mathbf{x}) = \sum_{m=1}^{M_\phi} w_m \phi_m(\mathbf{x})$$

- Regularized cost function with weights as free parameters

$$\text{cost}^{pen}(\mathbf{w}) = \sum_{i=1}^{N} \left(y_i - \sum_{m=1}^{M_\phi} w_m \phi_m(\mathbf{x}_i) \right)^2 + \lambda \sum_{m=1}^{M_\Phi} w_m^2$$

- The penalized least-squares solution

$$\widehat{\mathbf{w}}_{pen} = \left(\mathbf{\Phi}^T \mathbf{\Phi} + \lambda I \right)^{-1} \mathbf{\Phi}^T \mathbf{y}$$

with

$$\mathbf{\Phi} = \begin{pmatrix} \phi_1(\mathbf{x}_1) & \dots & \phi_{M_\phi}(\mathbf{x}_1) \\ \dots & \dots & \dots \\ \phi_1(\mathbf{x}_N) & \dots & \phi_{M_\phi}(\mathbf{x}_N) \end{pmatrix}$$

Nonlinear Models for Regression and Classification

- Regression:

$$f_{\mathbf{w}}(\mathbf{x}) = \sum_{m=1}^{M_\phi} w_m \phi_m(\mathbf{x})$$

  As discussed, the weights can be calculated via penalized LS

- Classification:

$$\widehat{y} = \text{sign}(f_{\mathbf{w}}(\mathbf{x})) = \text{sign}\left(\sum_{m=1}^{M_\phi} w_m \phi_m(\mathbf{x})\right)$$

  The Perceptron learning rules can be applied, or some other learning rules for linear classifiers, if we replace $1, x_{i,1}, x_{i,2}, \ldots$ with $\phi_1(\mathbf{x}_i), \phi_2(\mathbf{x}_i), \ldots$

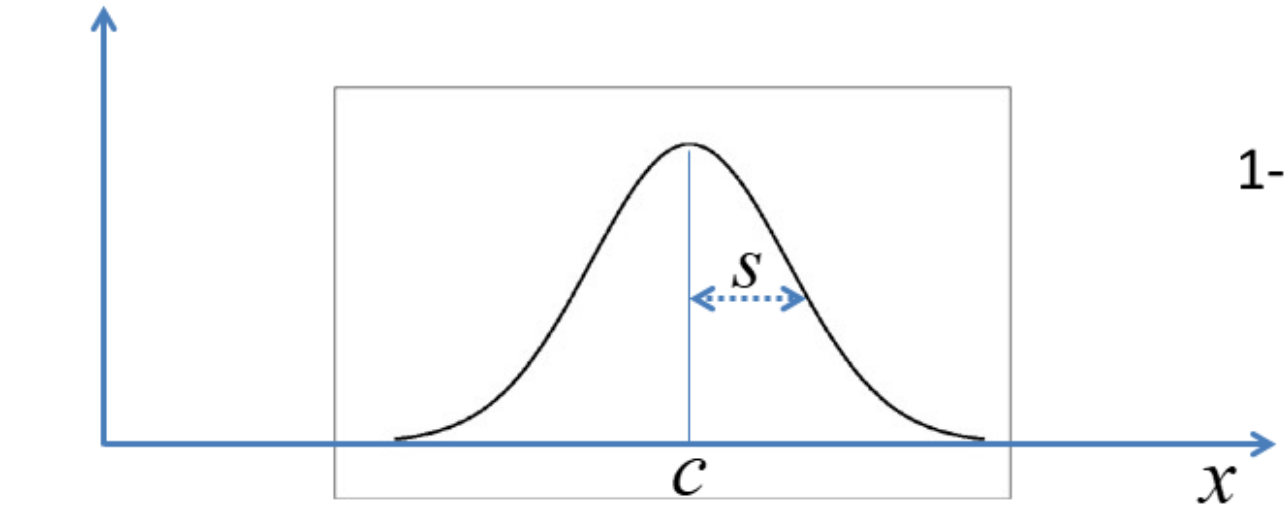
Which Basis Functions?

- The challenge is to find problem specific basis functions which are able to effectively model the true mapping, resp. that make the classes linearly separable; in other words we assume that the true dependency $f(\mathbf{x})$ can be modelled by at least one of the functions $f_{\mathbf{w}}(\mathbf{x})$ that can be represented by a linear combination of the basis functions, i.e., by one function in the function class under consideration
- If we include too few basis functions or unsuitable basis functions, we might not be able to model the true dependency
- If we include too many basis functions, we need many data points to fit all the unknown parameters (This sound very plausible, although we will see in the lecture on kernels that it is possible to work with an infinite number of basis functions)

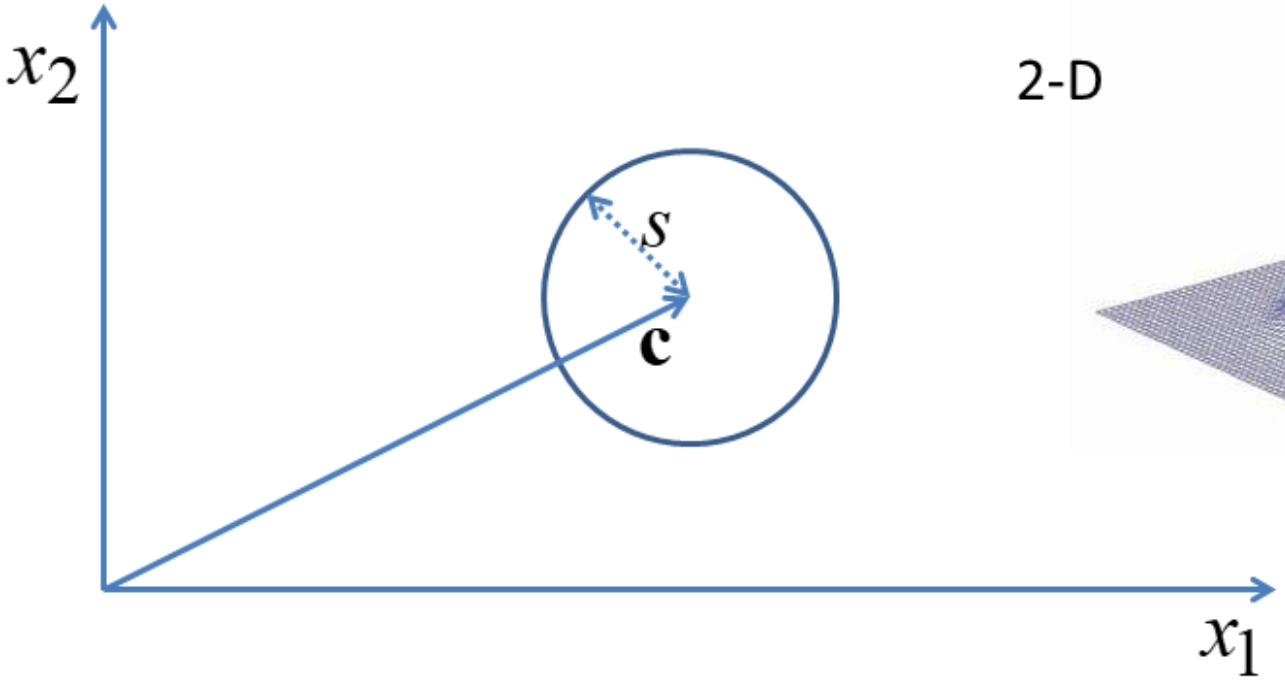
Radial Basis Function (RBF)

- We already have learned about polynomial basis functions
- Another class are radial basis functions (RBF). Typical representatives are Gaussian basis functions

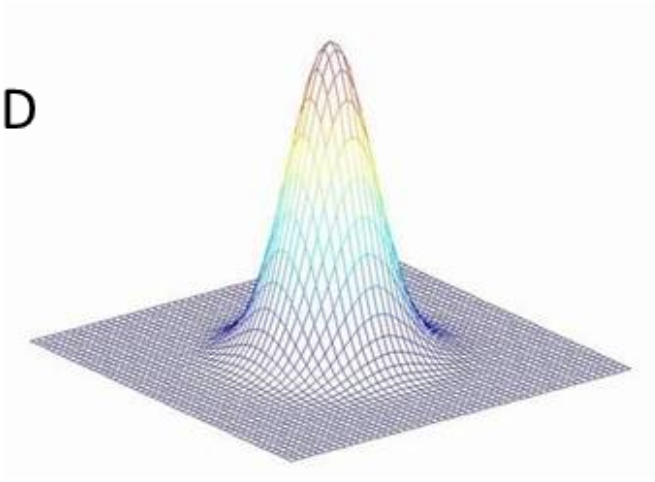
$$\phi_j(\mathbf{x}) = \exp\left(-\frac{1}{2s^2}\|\mathbf{x} - \mathbf{c}_j\|^2\right)$$



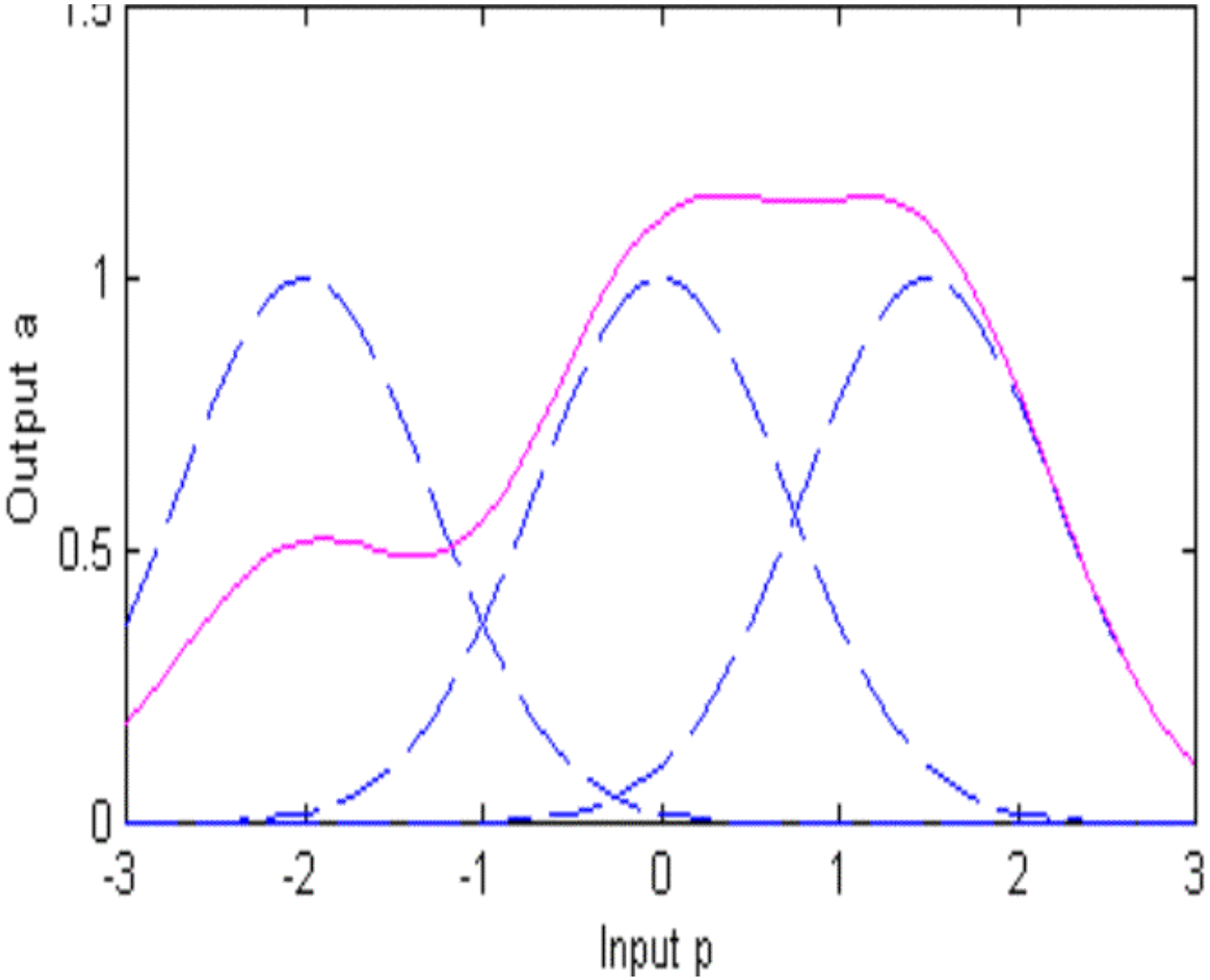
1-D

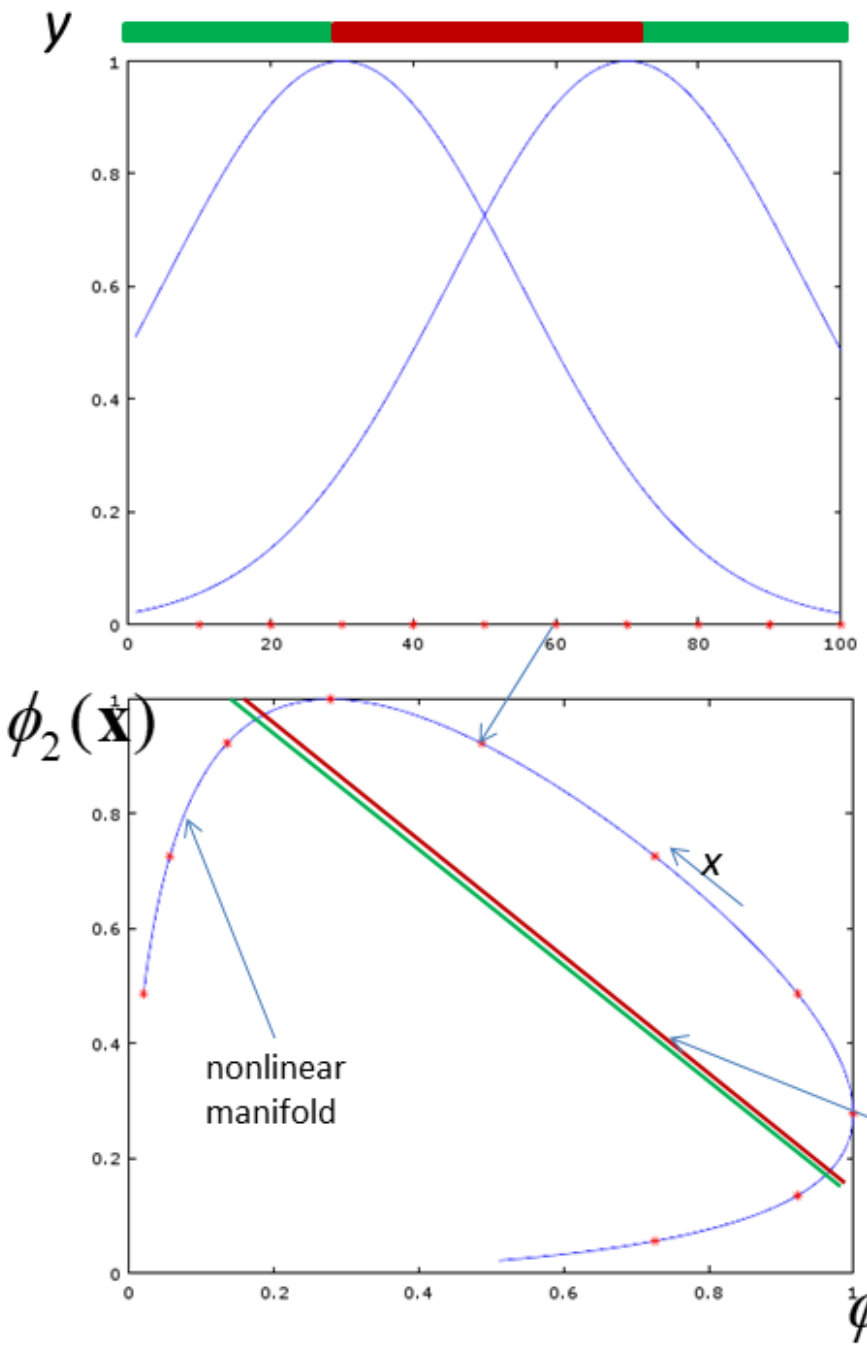


2-D



Three RBFs (blue) form $f(\mathrm{x})$ (pink)





Class labels (green, red, green)

In the 1-D input space, a linear classifier would not be able to separate the two classes

From a linear 1-D input space (top) to a nonlinear 1-D manifold in 2-D basis function space (bottom)

In basis function space, classes can linearly be separated!

The image of the 1-D input data space is a 1-D **nonlinear manifold**

separating hyperplane

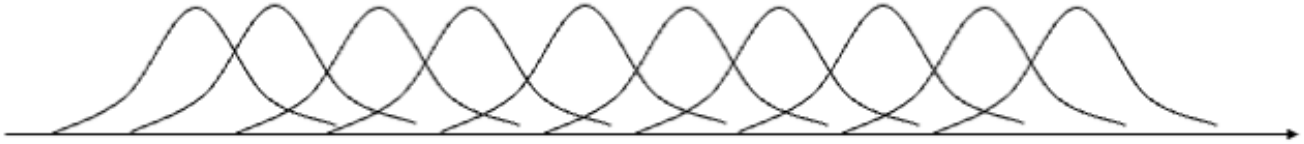
Nonlinear Manifold

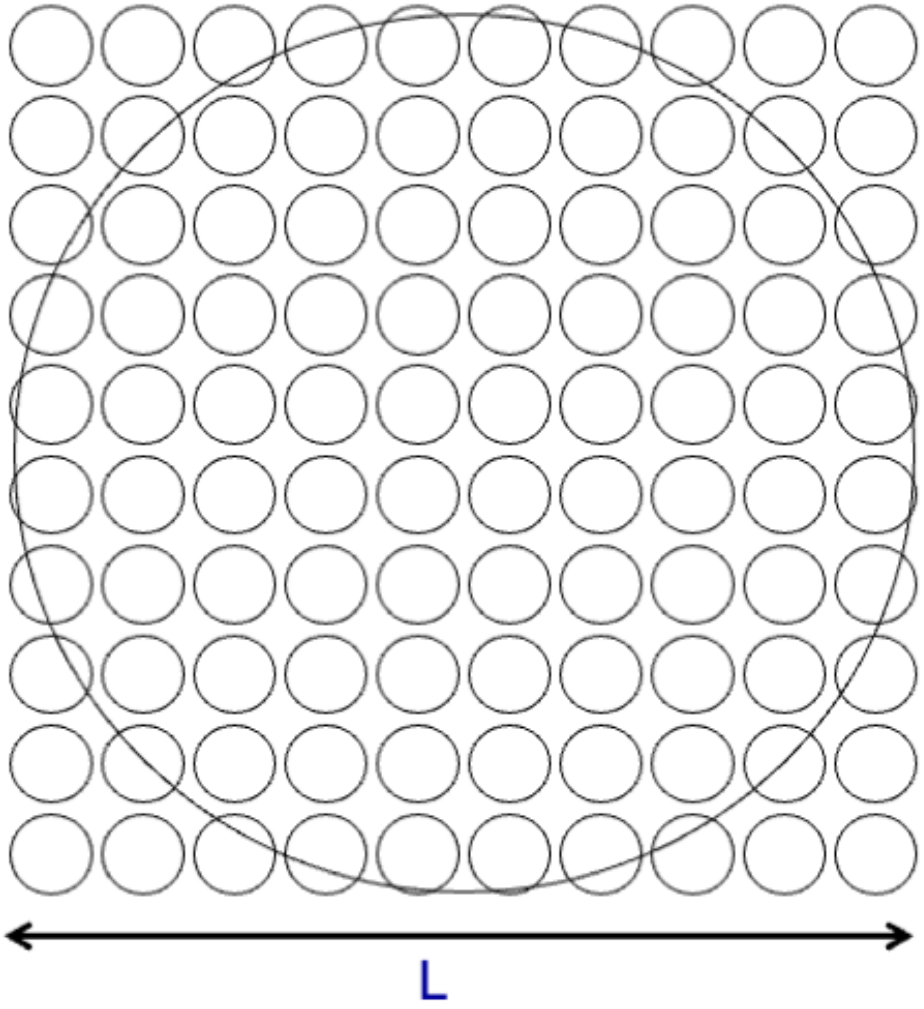
- Note that if data points in $\mathbf{x}$-space are uniformly or Gaussian distributed, in basis function space, data can be on a nonlinear manifold, so neither uniformly nor Gaussian distributed!

Optimal Basis Functions

- So far all seems to be too simple
- Here is the catch: in some cases, the number of "sensible" basis functions increases exponentially with the number of inputs
- If in one dimensions, we need $M_\phi^{(1)}$ RBFs (e.g., $M_\phi^{(1)} = 10$), and we want to maintain the same complexity in higher dimensions, then we need $(M_\phi^{(1)})^M$ RBFs in M dimensions (e.g., 10^M) (tensor product of basis functions)
- We get a similar exponential increase for polynomial basis functions; the number of polynomial basis functions of a given degree increases quickly with the number of dimensions (x^2); (x^2, y^2, xy); $(x^2, y^2, z^2, xy, xz, yz), \ldots$
- *The most important challenge: How can I get a small number of relevant basis functions*, i.e., a small number of basis functions that define a function class that contains the true function (true dependency) $f(\mathbf{x})$?

10 RBFs in one dimension





100 RBFs in
two dimensions

d, s

Forward Selection: Stepwise Increase of Model Class Complexity

- Start with a linear model
- Then we stepwise add basis functions; at each step add the basis function whose addition decreases the training cost the most (greedy approach)
- Examples: classification with polynomial basis functions (OCR, J. Schuermann, AEG, later Siemens)
  - Pixel-based image features (e.g., of hand written digits)
  - Dimensional reduction via PCA (see later lecture)
  - Start with a linear classifier and add polynomials that significantly increase performance
  - Apply a linear classifier

Backward Selection: Stepwise Decrease of Model Class Complexity (Model Pruning)

- Start with a model class which is too complex and then incrementally decrease complexity
- First start with many basis functions
- Then we stepwise remove basis functions; at each step remove the basis function whose removal increases the training cost the least (greedy approach)
- A stepwise procedure is not optimal. The problem of finding the best subset of K basis functions is NP-hard

Example: Ad Placements

- The decision of which ad to place for which user in which context is defined as a classification or regression problem in a high-dimensional feature space
- Here the features are often handcrafted and new features are continuously added and removed, optimizing the prediction in the long run
- Speed in training and recall and a small memory trace are important criteria

Conclusions

- The best way to think about models with fixed basis functions is that they implement a form of prior knowledge: we make the assumption that the true function can be modelled by the set of weighted basis function
- The data then favors certain members of the function class
- In the lecture on kernel systems we will see that the set of basis functions can be translated in assuming certain correlations between (mostly near-by) function values, implementing a smoothness prior