Basis Functions

Volker Tresp
Summer 2017

Nonlinear Mappings and Nonlinear Classifiers

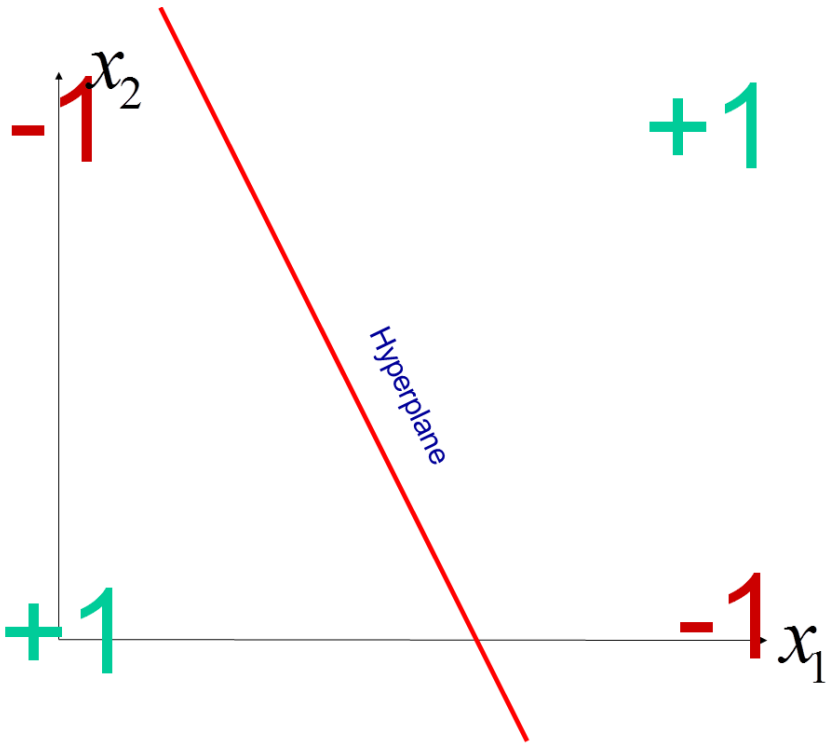
- Regression:
  - Linearity is often a good assumption when many inputs influence the output
  - Some natural laws are (approximately) linear $F = ma$
  - But in general, it is rather unlikely that a true function is linear
- Classification:
  - Linear classifiers also often work well when many inputs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

Trick

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

$$
\begin{array}{cccc}
0 & 0 & \rightarrow & +1 \\
1 & 0 & \rightarrow & -1 \\
0 & 1 & \rightarrow & -1 \\
1 & 1 & \rightarrow & +1
\end{array}
$$

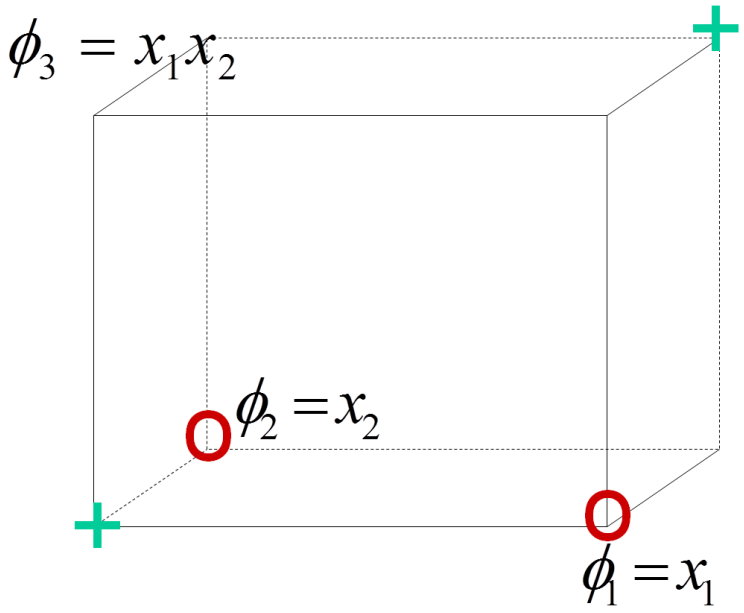
XOR is not linearly separable



Trick: Let's Add Basis Functions

- Linear Model: input vector: $1, x_1, x_2$
- Let's consider $x_1 x_2$ in addition
- The interaction term $x_1 x_2$ couples two inputs nonlinearly

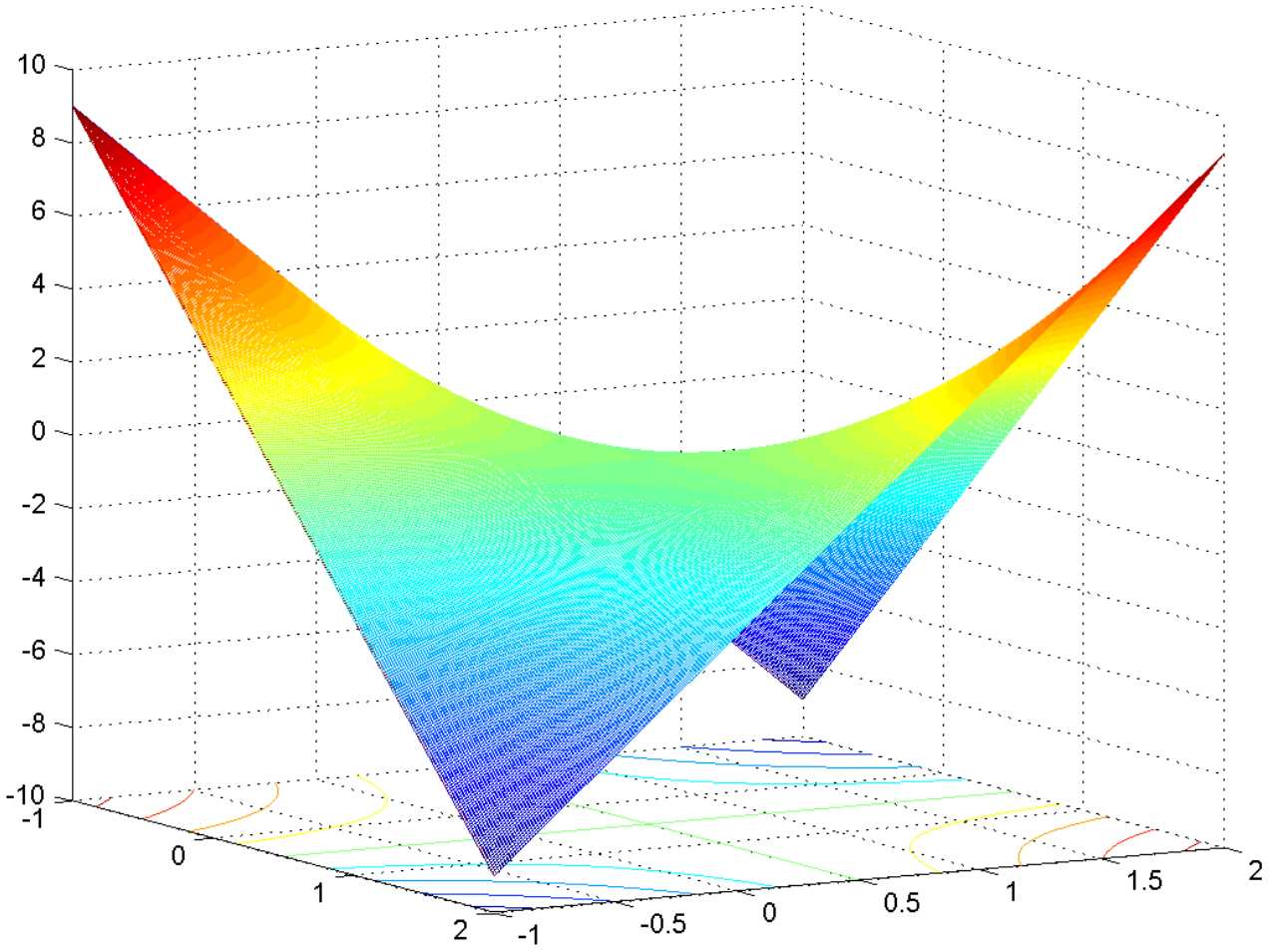
With a Third Input $z_3 = x_1 x_2$ the XOR Becomes Linearly Separable



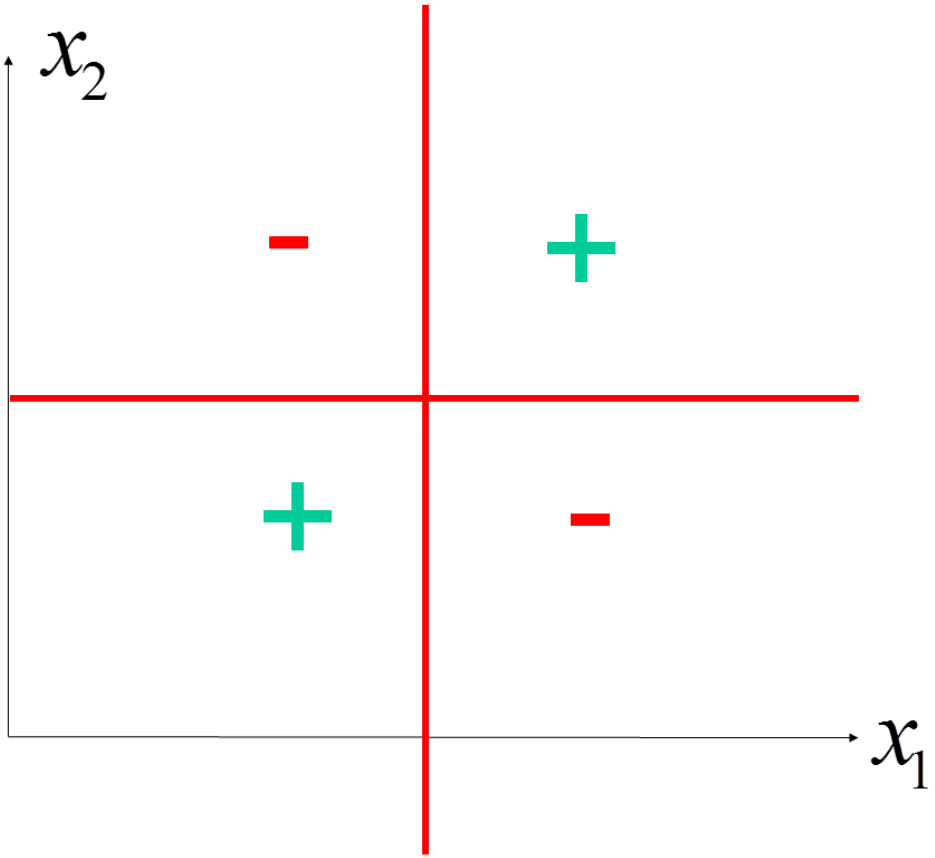
$$f(\mathbf{x}) = 1 - 2x_1 - 2x_2 + 4x_1 x_2 = \phi_1(x) - 2\phi_2(x) - 2\phi_3(x) + 4\phi_4(x)$$

with $\phi_1(x) = 1, \phi_2(x) = x_1, \phi_3(x) = x_2, \phi_4(x) = x_1 x_2$

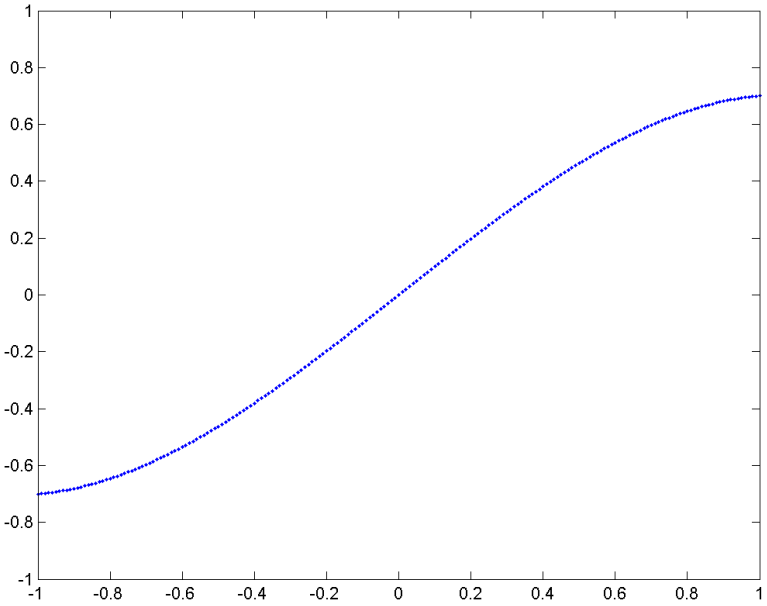
$$f(\mathbf{x}) = 1 - 2x_1 - 2x_2 + 4x_1x_2$$



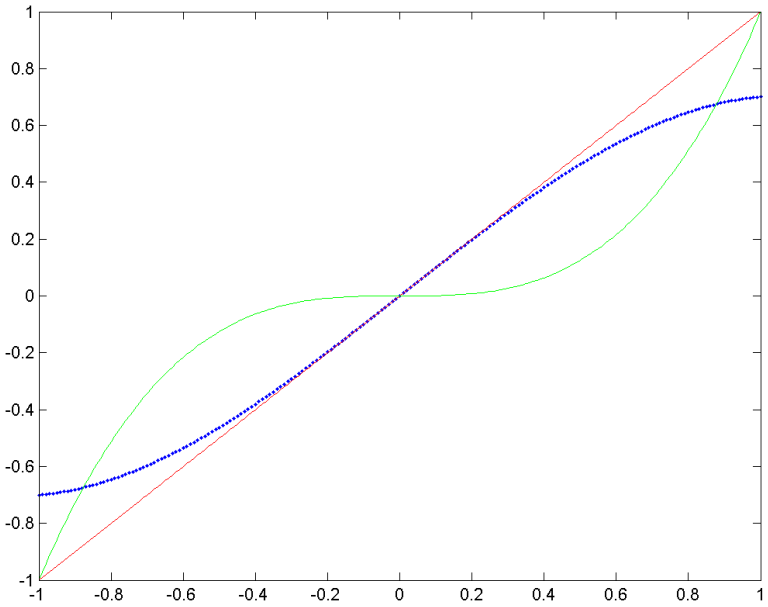
Separating Planes



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

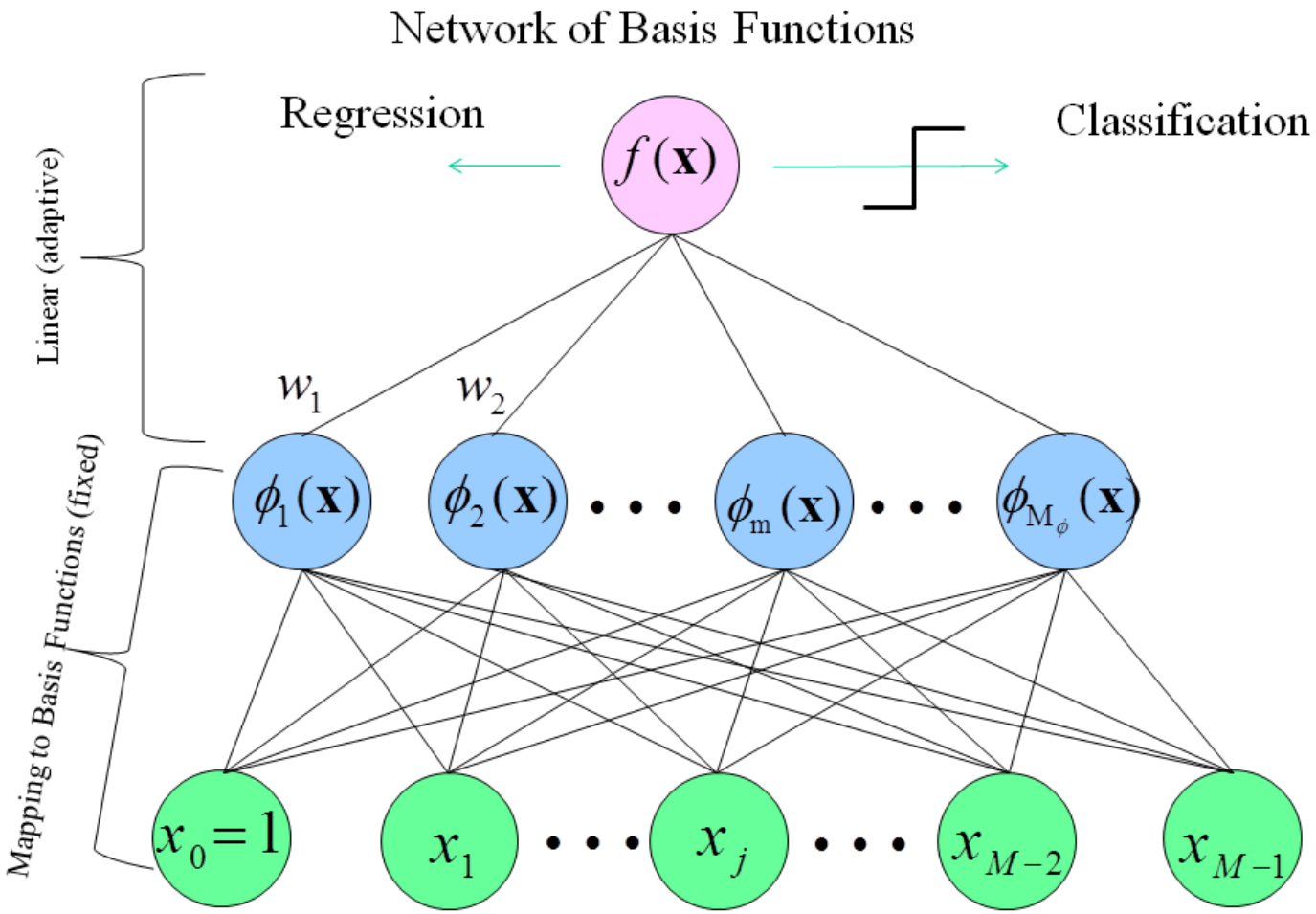
- The simple idea: in addition to the original inputs, we add inputs that are calculated as deterministic functions of the existing inputs, and treat them as additional inputs
- Example: Polynomial Basis Functions

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

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

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

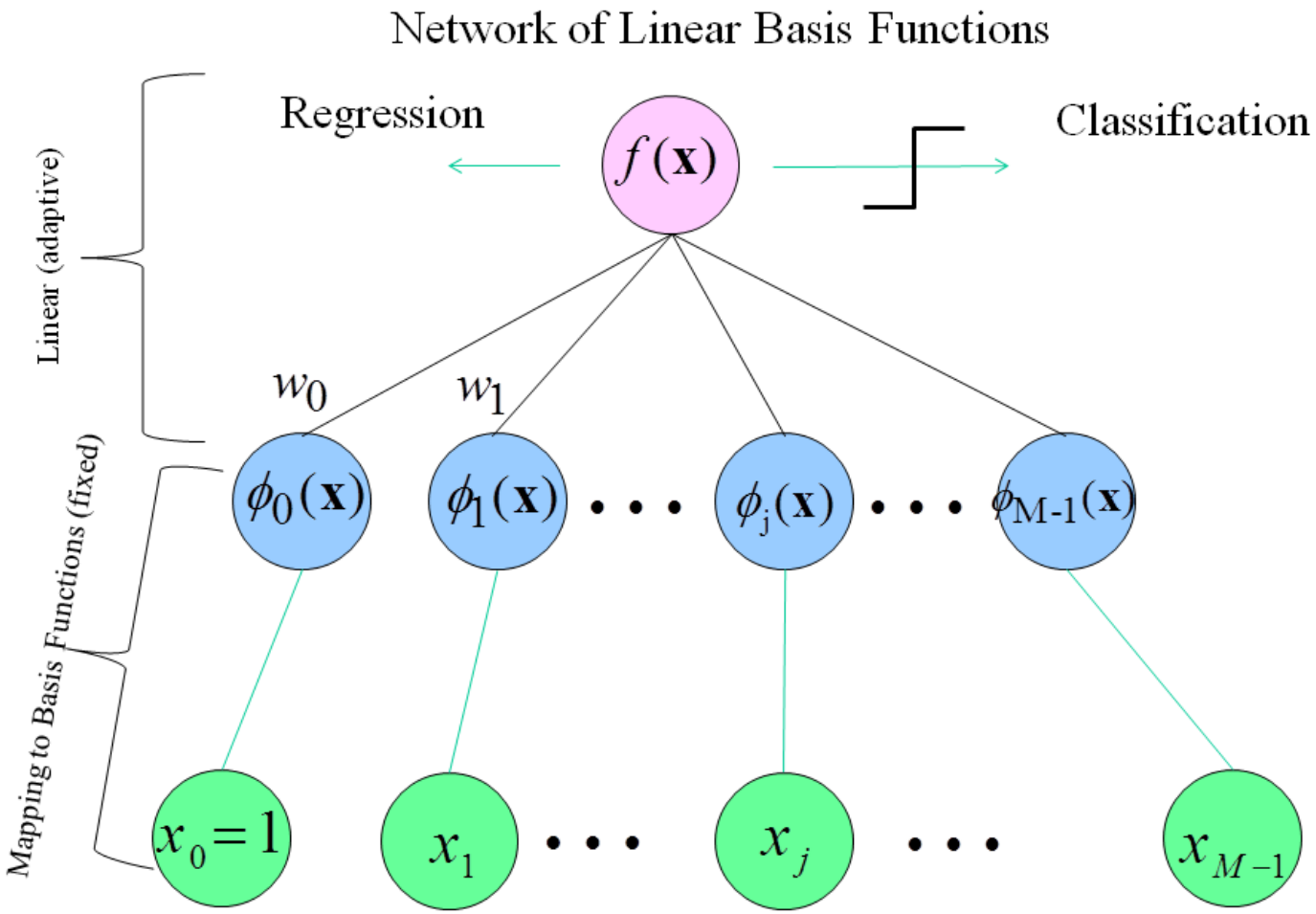
- Independent of the choice of basis functions, the regression parameters are calculated using the well-known equations for linear regression


Network of Basis Functions
Regression
Classification
$f(\mathbf{x})$
Linear (adaptive)
Mapping to Basis Functions (fixed)
w_1
w_2
$\phi_1(\mathbf{x})$
$\phi_2(\mathbf{x})$
$\phi_m(\mathbf{x})$
$\phi_{M_\phi}(\mathbf{x})$
$x_0 = 1$
x_1
x_j
x_{M-2}
x_{M-1}

Linear Model Written as Basis Functions

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

$$\phi_0(\mathbf{x}) = 1 \quad \phi_1(\mathbf{x}) = x_1 \quad \phi_{M-1}(\mathbf{x}) = x_{M-1} \quad \ldots$$


Network of Linear Basis Functions
Regression
Classification
$f(\mathbf{x})$
Linear (adaptive)
w_0
w_1
$\phi_0(\mathbf{x})$
$\phi_1(\mathbf{x})$
$\phi_j(\mathbf{x})$
$\phi_{M-1}(\mathbf{x})$
Mapping to Basis Functions (fixed)
$x_0 = 1$
x_1
x_j
x_{M-1}

Review: Penalized LS for Linear Regression

- Multiple Linear Regression:

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

- Regularized cost function

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

- The penalized LS-Solution gives

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

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, if we replace $1, x_{i,1}, x_{i,2}, \ldots$ with $\phi_1(\mathbf{x}_i), \phi_2(\mathbf{x}_i), \ldots$

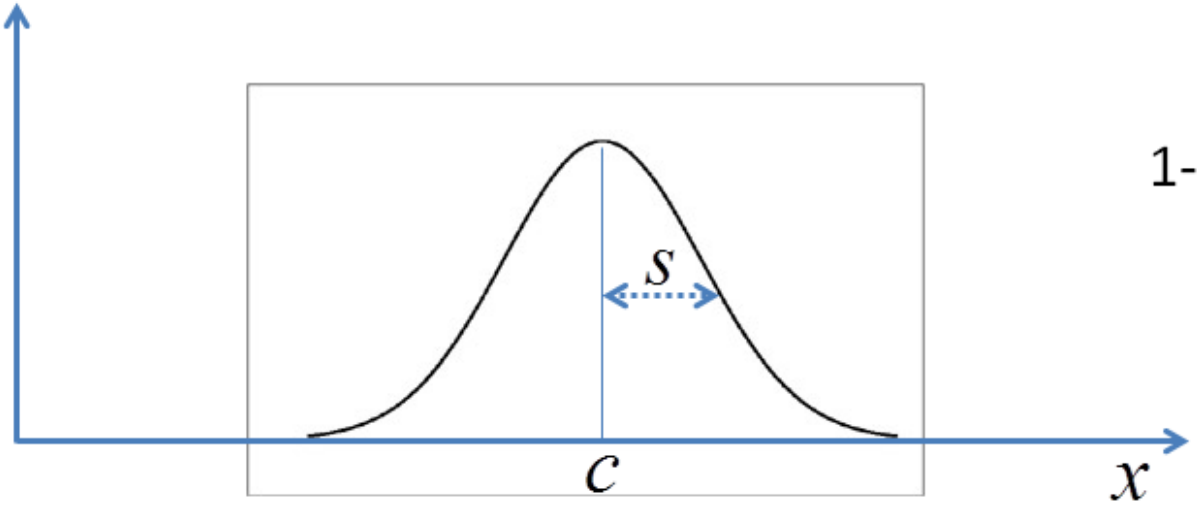
Which Basis Functions?

- 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)

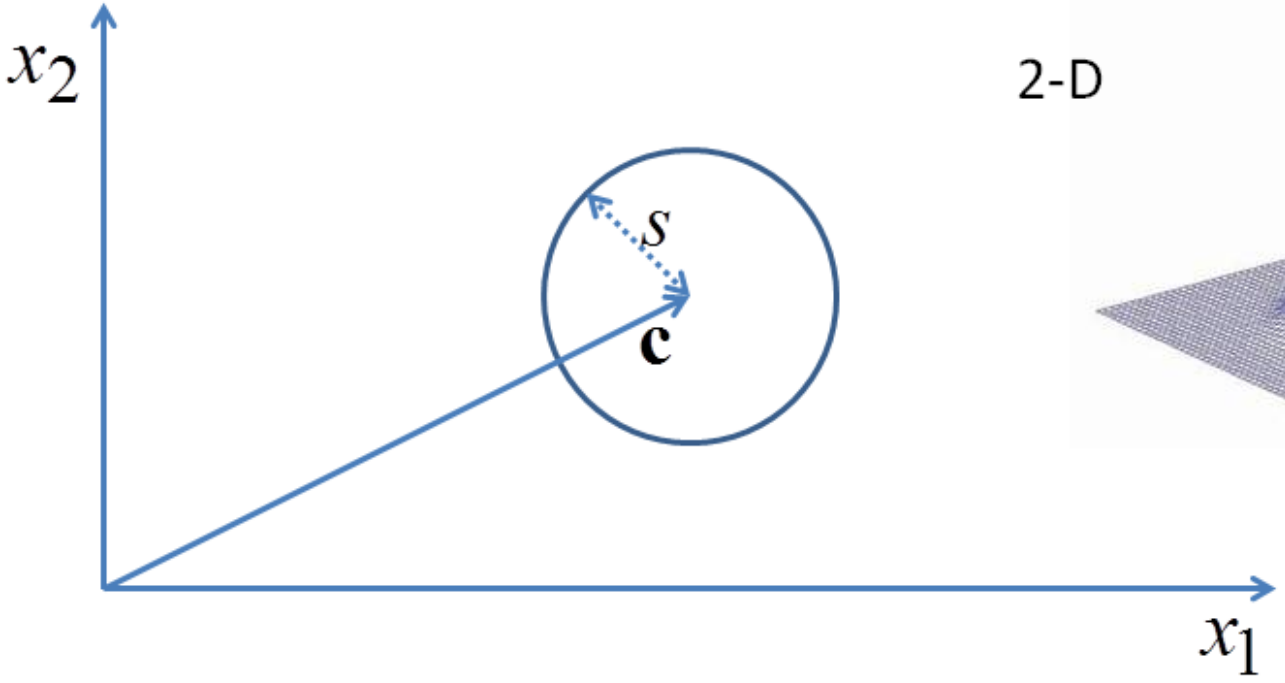
Radial Basis Function (RBF)

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

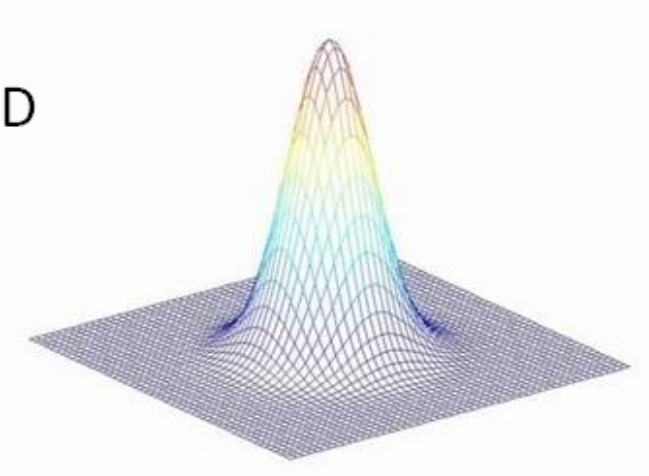
$$\phi_j(\mathbf{x}) = \exp\left(-\frac{1}{2s^2}\|\mathbf{x} - \mathbf{c}_j\|^2\right)$$



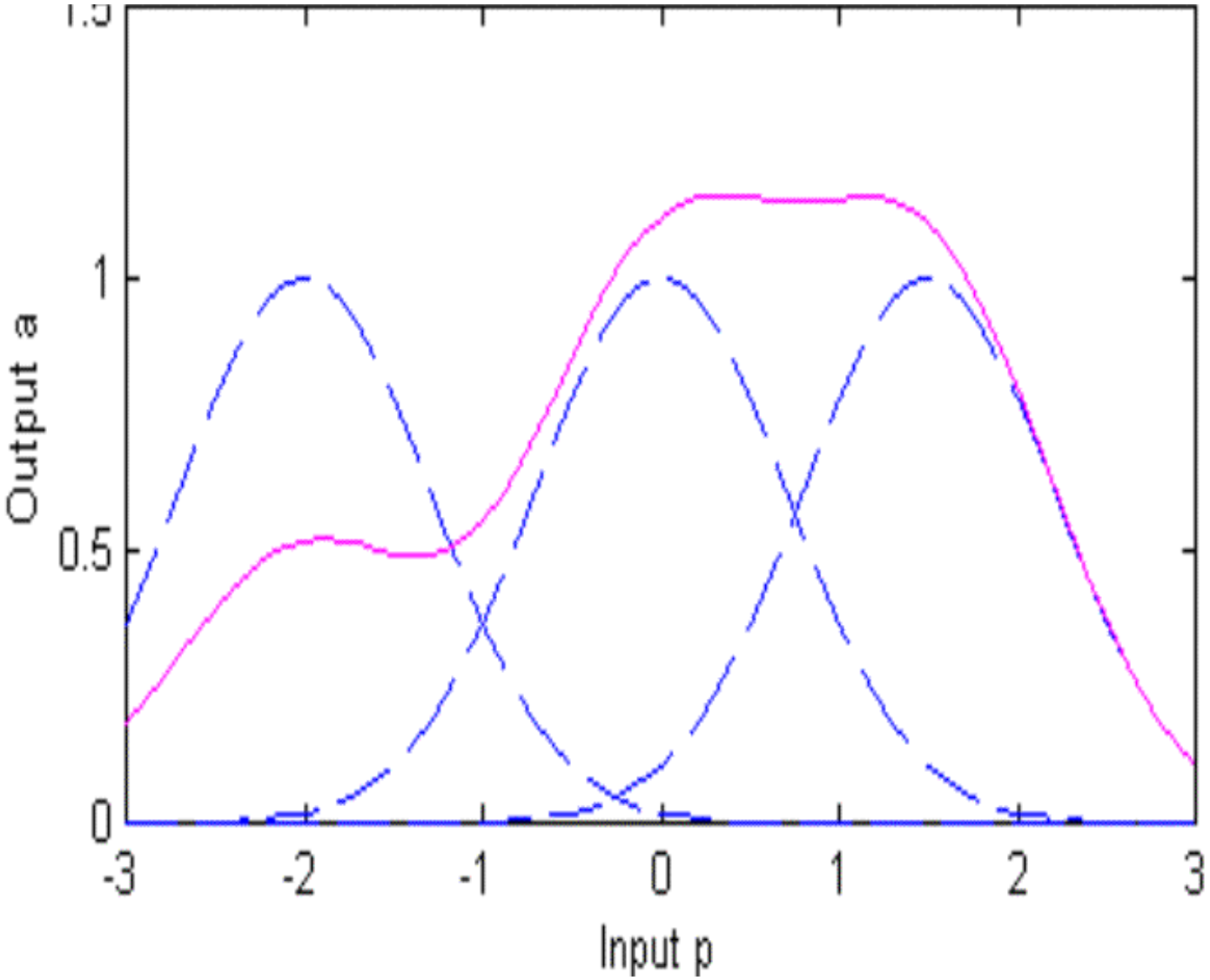
1-D



2-D



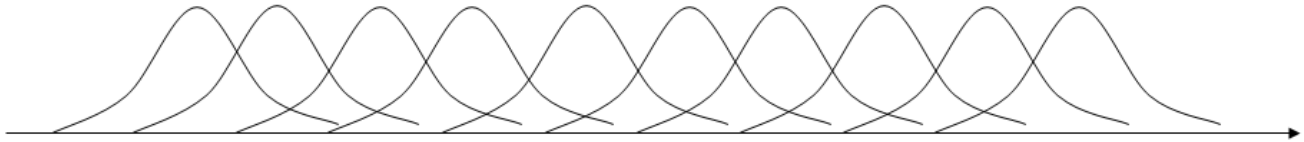
Three RBFs (blue) form $f(\mathrm{x})$ (pink)

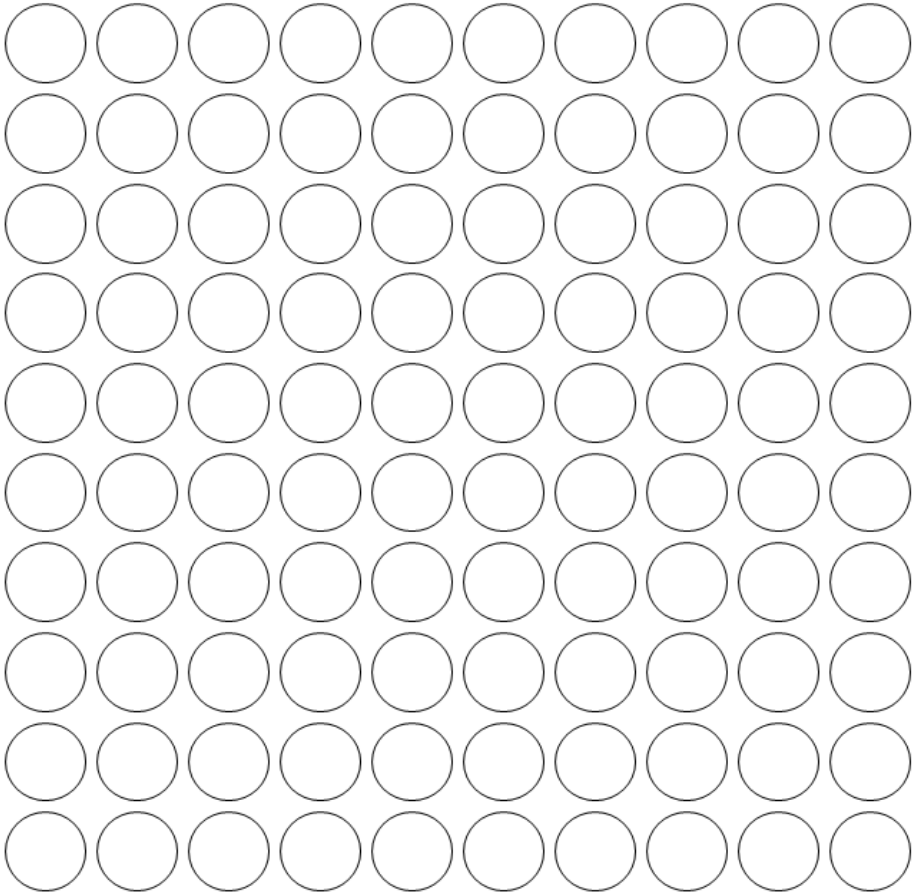


Optimal Basis Functions

- So far all seems to be too simple
- Here is the catch: the number of "sensible" basis functions increases exponentially with the number of inputs
- If I am willing to use K RBF-basis functions "per dimension". then I need K^M RBFs in M dimensions
- 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

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: Polynomklassifikatoren (OCR, J. Schürmann, AEG)
  - 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

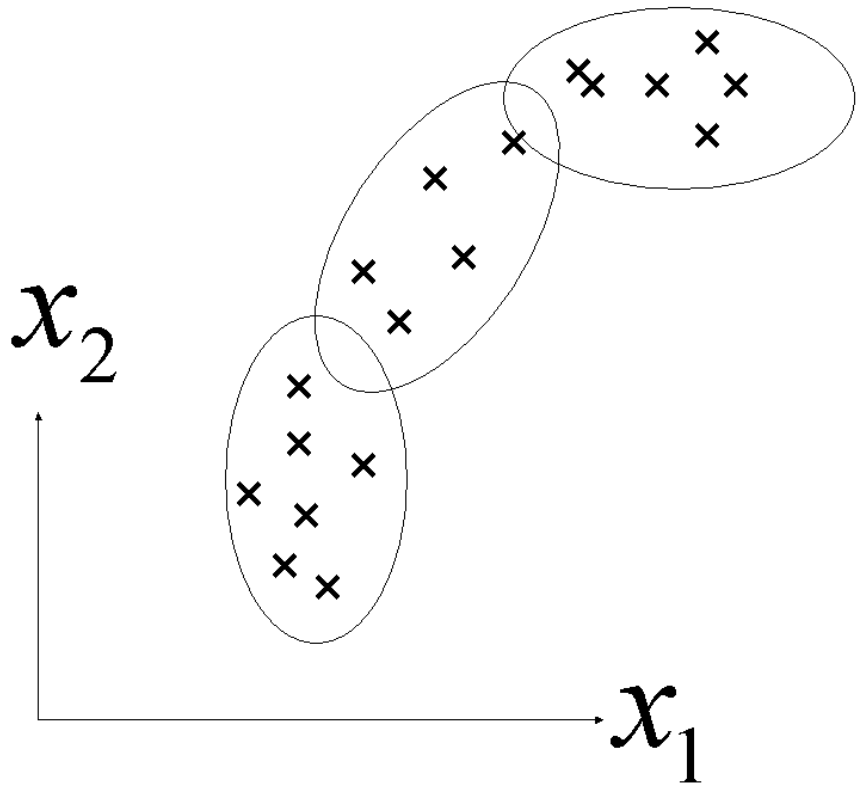
Backward Selection: Stepwise Decrease of Model Class Complexity (Model Pruning)

- 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

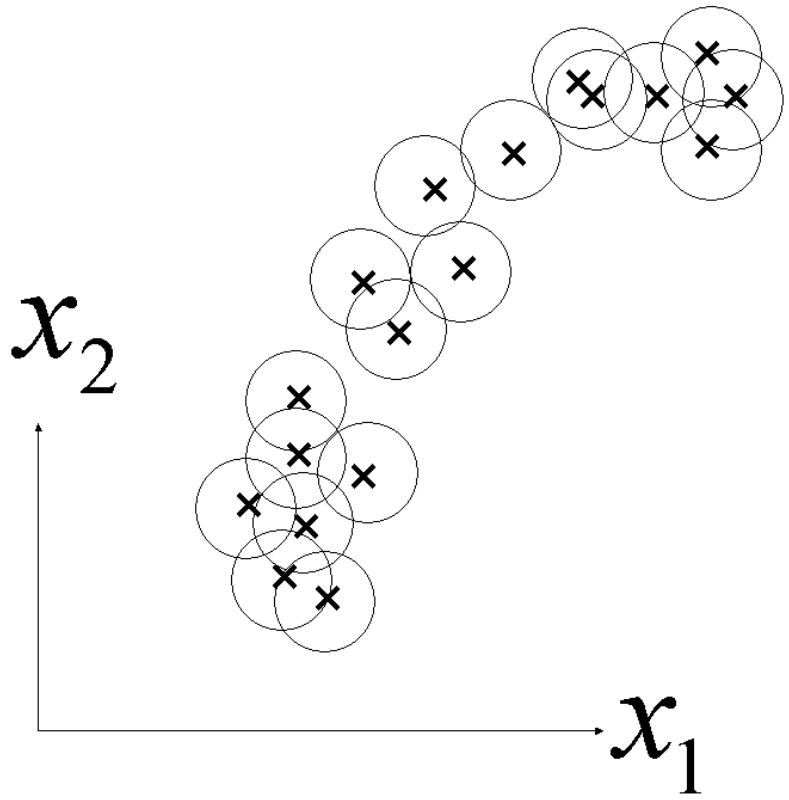
Model Selection: RBFs

- Sometimes it is sensible to first group (cluster) data in input space and to then use the cluster centers as positions for the Gaussian basis functions
- The widths of the Gaussian basis functions might be derived from the variances of the data in the cluster
- An alternative is to use one RBF per data point. The centers of the RBFs are simply the data points themselves and the widths are determined via some heuristics (or via cross validation, see later lecture)

RBFs via Clustering



One Basis Function per Data Point



Application-Specific Features

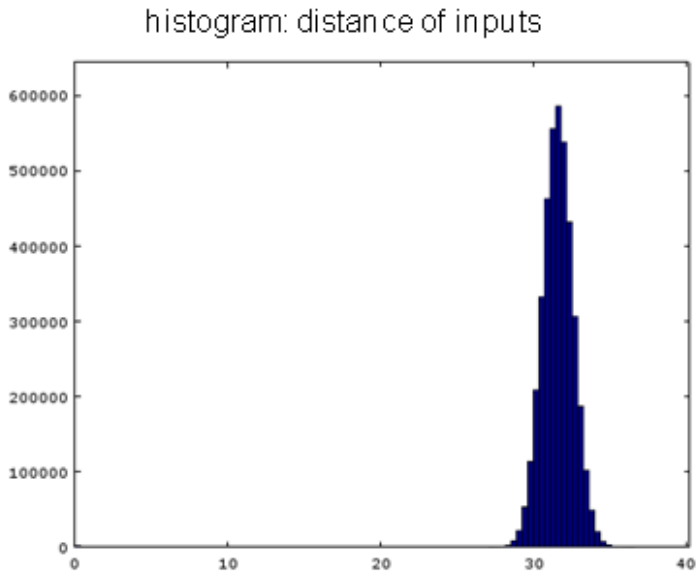
- Often the basis functions can be derived from sensible application features
  - Given an image with $256 \times 256 = 65536$ pixels. The pixels form the input vector for a linear classifier. This representation would not work well for face recognition
  - With fewer than 100 appropriate features one can achieve very good results (example: PCA features, see later lecture)
- The definition of suitable features for documents, images, gene sequences, ... is a very active research area
- If the feature extraction already delivers many features, it is likely that a linear model will solve the problem and no additional basis functions need to be calculated
- This is quite remarkable: learning problems can become simpler in high-dimensions, in apparent contradiction to the famous "curse of dimensionality" (Bellman) (although

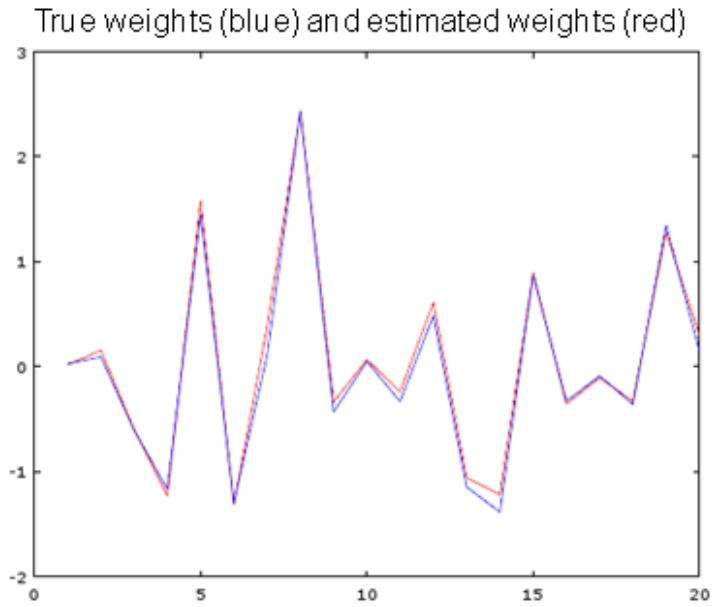
there still is the other "curse of dimensionality" since the number of required basis functions might increase exponentially with the number of inputs!)

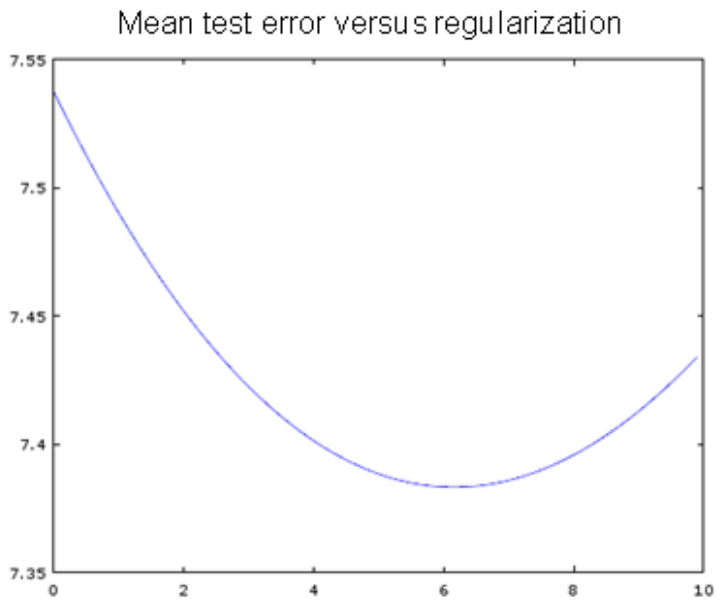
- If inputs are random, then data points become equidistant in high dimensions
- This is not the case for a high-dimensional space generated by transformations (basis functions): the data are on a low dimensional manifold!
- Thus one often applies first **dimensionality reduction (PCA)** to remove noise on the input and then **increases dimensionality** again by using basis functions!

Curse or Blessing of Dimensionality

- With $M = 500$ inputs, we generated random inputs $\{\mathbf{x}_i\}_{i=1}^{5}00$
- Curse of dimensionality: near equidistance between data points (see next figure): distance-based methods, such as nearest-neighbor classifiers, might not work well
- Blessing of dimensionality: A linear regression model with $N = 1000$ training data points gives excellent results


histogram: distance of inputs
600000
500000
400000
300000
200000
100000
0
0
10
20
30
40

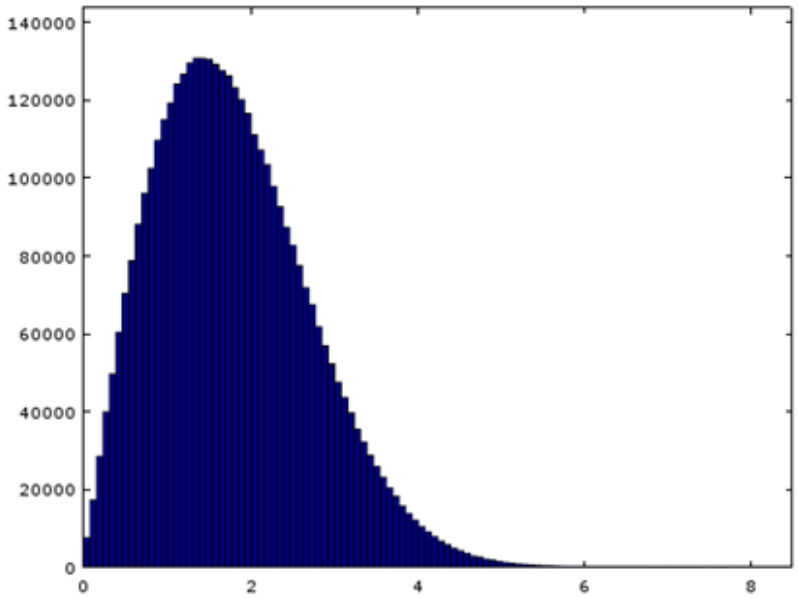

True weights (blue) and estimated weights (red)
3
2
1
0
-1
-2
0
5
10
15
20


Mean test error versus regularization
7.55
7.5
7.45
7.4
7.35
0
2
4
6
8
10

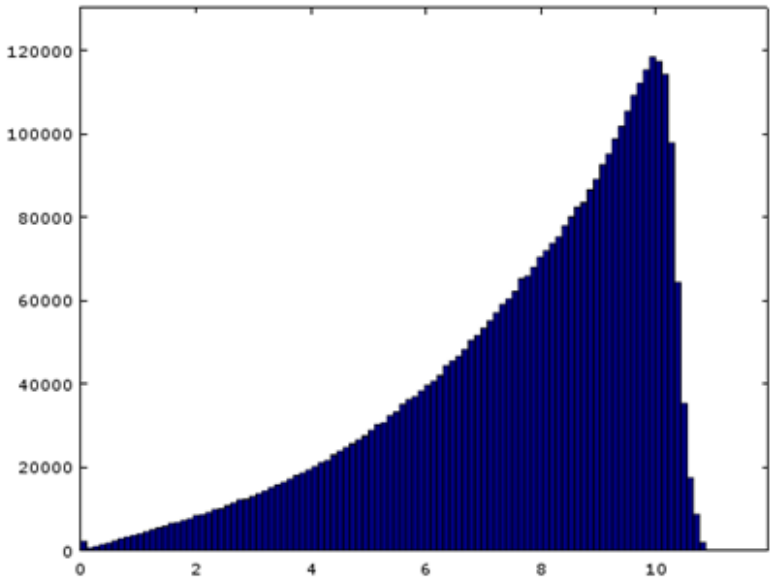
Curse or Blessing of Dimensionality in RBF Space

- We start with random data points in $M = 2$. The left figure shows the distribution of data points distances
- We calculate M_ϕ RBF-basis functions $\{\phi_m(\mathbf{x})\}_{m=1}^{500}$
- In the original space but also in the RBF space, distances are not peaked: in the RBF space, data lives in a 2-dim manifold in 500-dim space

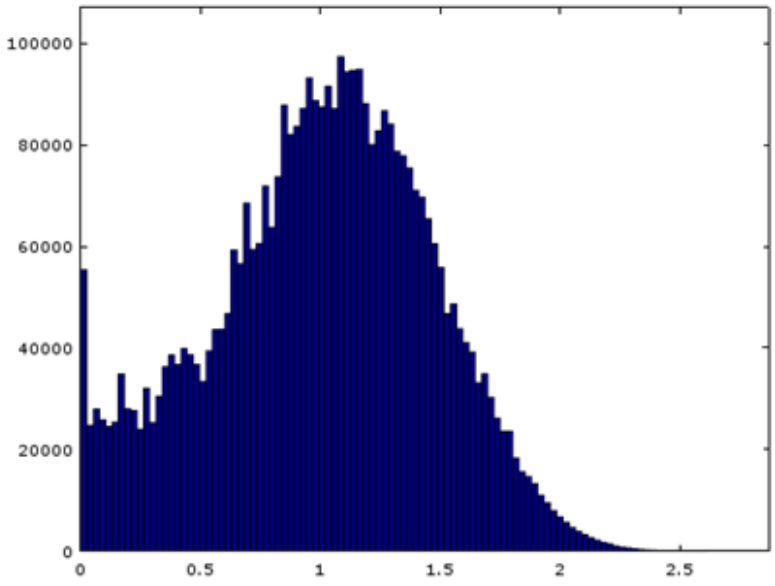
histogram: distance of original inputs with M = 2



histogram: distance of RBF-space (s=1)



histogram: distance of RBF-space (s=0.1)



Interpretation of Systems with Fixed Basis Functions

- 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 imply specific correlations between (mostly near-by) function values, implementing a smoothness prior

Function Spaces (Advanced)

- Consider the function $f(\mathbf{x})$
- Often $\mathbf{x}$ is assumed to be an element in a vector space. Example: $\mathbf{x} \in \mathbb{R}^M$
- A function $f(\cdot)$ can also be an element of a vector space

Is an Image a 2-D Function or an N^2 Vector?

- *greyValue*$(i, j) = f(i, j)$, with $i = 1, \ldots, N, j = 1, \ldots, N$
- We can model with basis functions

$$f(i, j) = \sum_{m=1}^{M_\phi} w_m \phi_m(i, j)$$

- Typical 2-D basis functions used in image analysis are Fourier basis functions and cosine basis functions with $M_\phi = N^2$
- A discrete pixel image has finite support (is only defined at the N^2 pixels)

Or is an Image an N^2-dimensional Vector

- $\mathbf{f}$ is an N^2-dimensional vector with

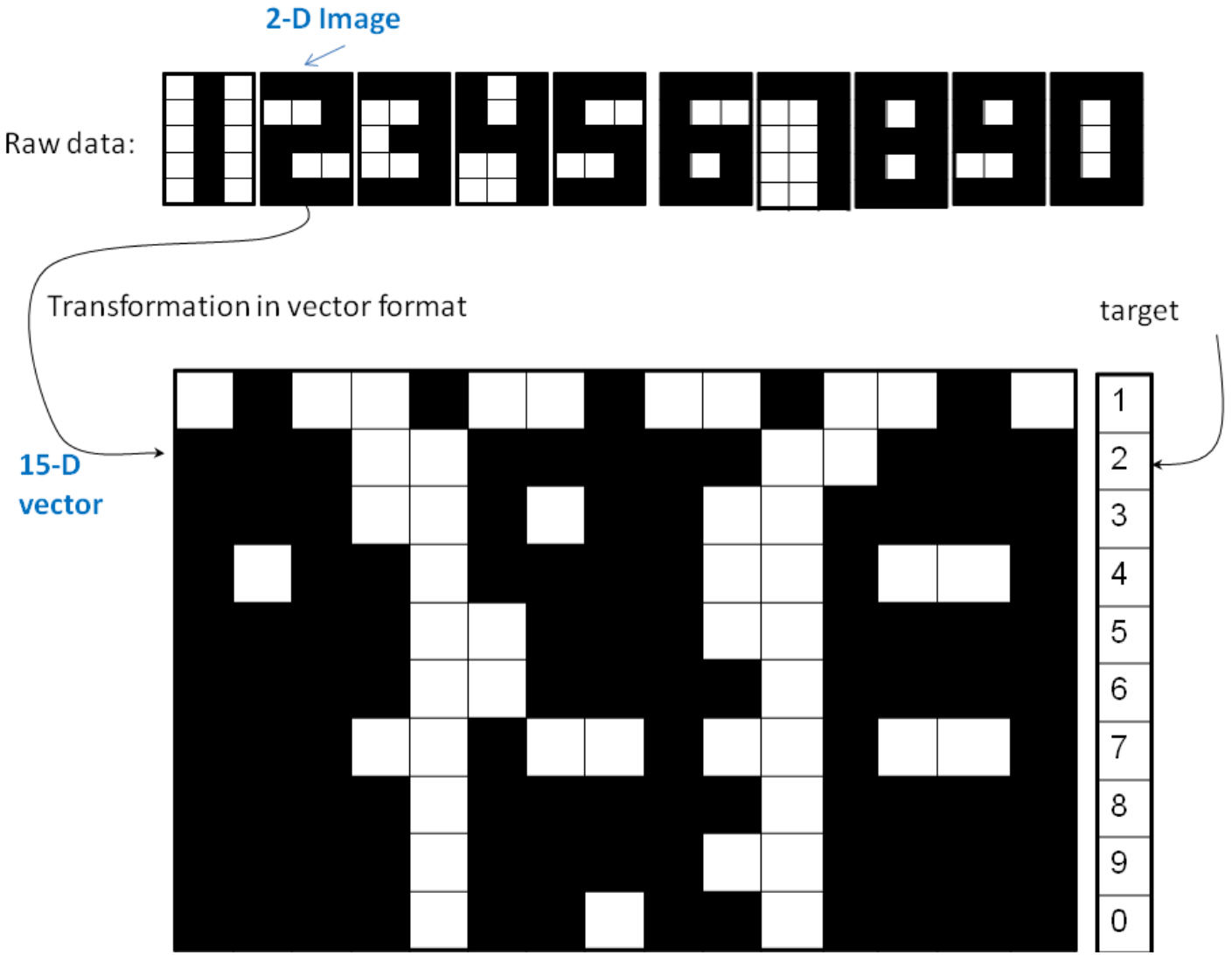
$$f_{i(N-1)+j} = f(i,j) \quad i = 1,\ldots,N, j = 1,\ldots,N$$

- We can model using vector algebra

$$\mathbf{f} = \sum_{m=1}^{M_\phi} w_m \vec{\phi}_m$$

With $M_\phi = N^2$ linearly independent basis functions, we can approximate any image, and an image becomes an element in an N^2-dimensional **vector** space

Illustration



Functions with Infinite Support

- Also a continuous **function** $f(\mathbf{x})$ with infinite support (e.g., $\mathbf{x} \in \mathbb{R}^2$) can be interpreted as an element in an **infinite** dimensional **vector** space (Hilbert space)
- We can model with (a finite or an infinite number of) basis functions

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

- Special case: the basis functions are δ functions

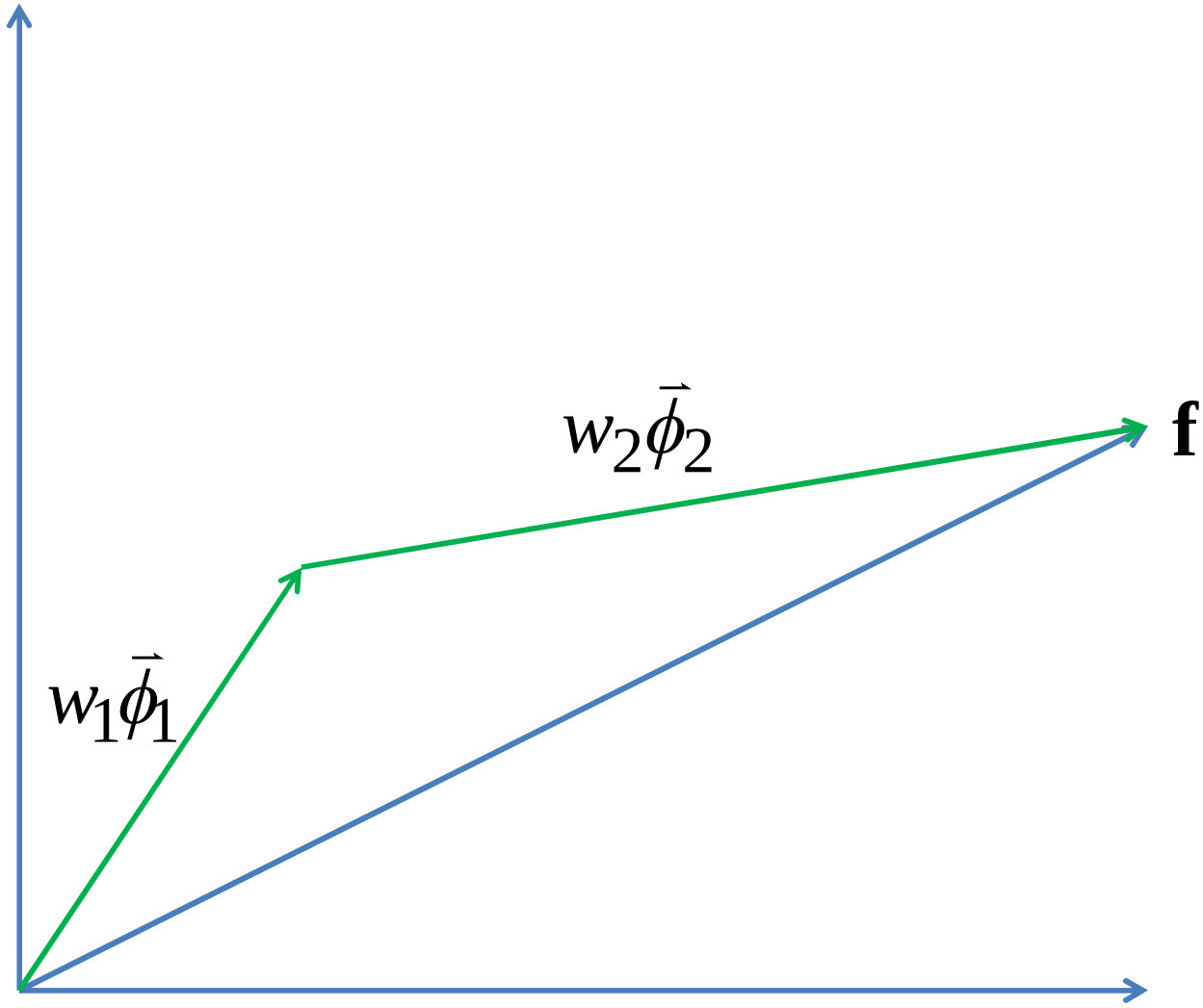
$$f(\mathbf{x}) = \int f(\mathbf{x}')\delta(\mathbf{x} - \mathbf{x}')\, d\mathbf{x}'$$

- Recall that the dot product (scalar product) $\langle \mathbf{a}, \mathbf{b} \rangle = \sum_i a_i b_i = \mathbf{a}^T \mathbf{b}$ defines an inner product for vectors
- Similarly, we can derive a basis-function specific inner products as

$$\langle \mathbf{f}, \mathbf{g} \rangle = \mathbf{w}_f^T \mathbf{w}_g$$

- With square integrable functions and δ functions as basis functions we get

$$\langle \mathbf{f}, \mathbf{g} \rangle = \int f(\mathbf{x}) g(\mathbf{x}) \, d\mathbf{x}$$


$w_2\vec{\phi}_2$
$\mathbf{f}$
$w_1\vec{\phi}_1$