Basis Functions

Volker Tresp
Summer 2016

I am an AI optimist. We've got a lot of work in machine learning, which is sort of the polite term for AI nowadays because it got so broad that it's not that well defined.

`Bill Gates (Scientific American Interview, 2004)`

"If you invent a breakthrough in artificial intelligence, so machines can learn," Mr. Gates responded, "that is worth 10 Microsofts." (Quoted in NY Times, Monday March 3, 2004)

Amazon Europe Machine Learning Team Coming To Berlin!

Posted by Victoria Nicholl on Fri, 18/01/2013 - 14:37

Amazon is building a European Machine Learning (ML) team in Berlin! Machine Learning Scientists at Amazon are technical leaders who develop planet-scale platforms for machine learning on the cloud, assist the benchmarking and future development of existing machine learning applications across Amazon, and help develop novel and infinitely-scalable applications that optimize Amazon's systems using cutting edge quantitative techniques. The ML team innovates algorithms that model patterns within data to drive automated decisions at scale in all corners of the company, including our eCommerce site and subsidiaries, Amazon Web Services, Seller & Buyer Services and Digital Media including Kindle. Amazon was one of the first companies to build eCommerce customer recommendations, fraud detection, and product search using machine learning innovations. Being part of the Machine Learning team at Amazon is one of the most exciting machine learning job opportunities in the world today. If you have deep technical knowhow in Machine Learning, know how to deliver, are deeply technical, highly innovative and long for the opportunity to build solutions to challenging problems that directly affect millions of people: there may be no better place than Amazon for you to impact the world!

If you are interested send your CV to strategic-recruiting@amazon.com.

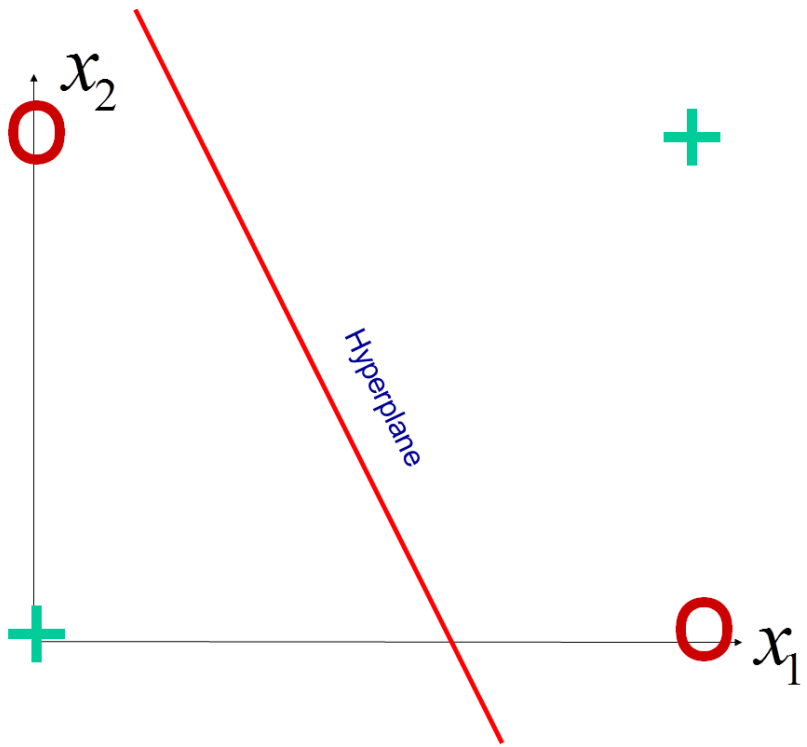
Nonlinear Mappings and Nonlinear Classifiers

- 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:
  - Similarly, 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

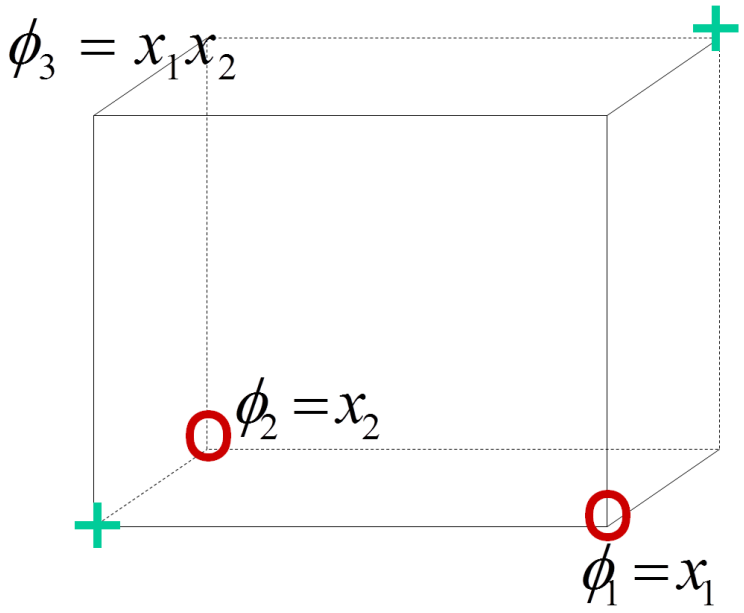
XOR is not linearly separable



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

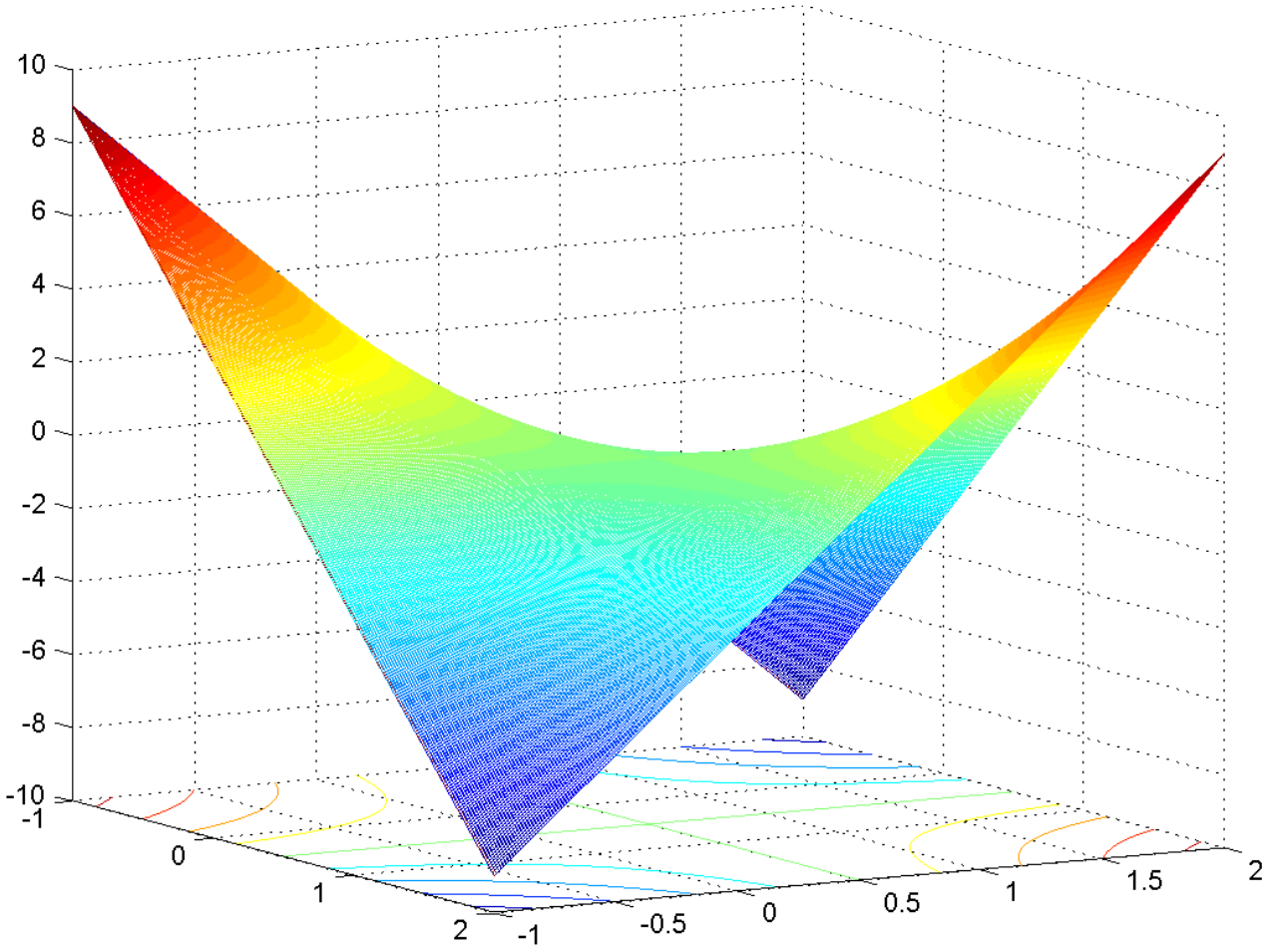
With a Third Input $z_3 = x_1 x_2$ the XOR Becomes Linearly Separable



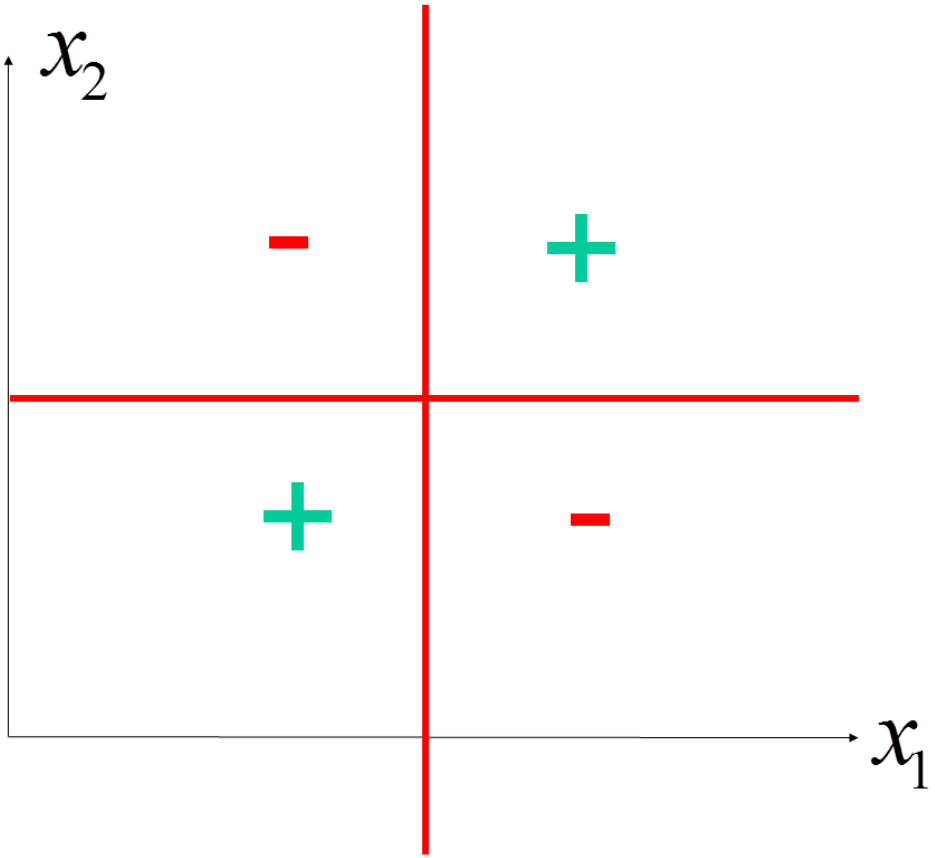
$$f(\mathbf{x}) = 1 - 2x_1 - 2x_2 + 4x_1x_2 = \phi_1(x) - 2\phi_2(x) - 2\phi_3(x) + 4\phi_4(x)$$

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

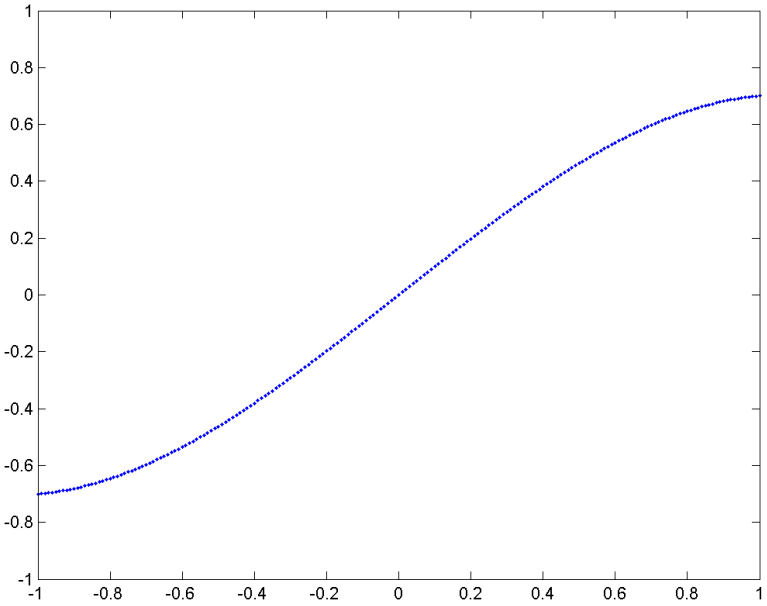
$$f(\mathbf{x}) = 1 - 2x_1 - 2x_2 + 4x_1x_2$$



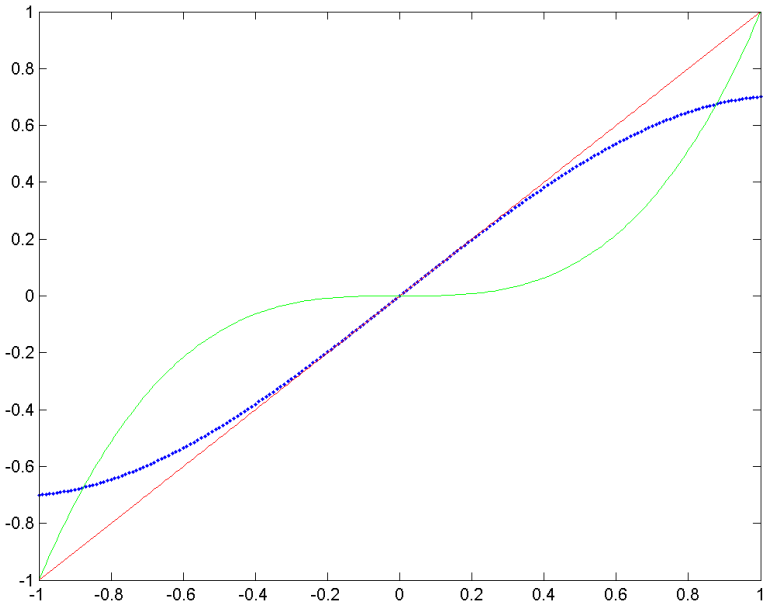
Separating Planes



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

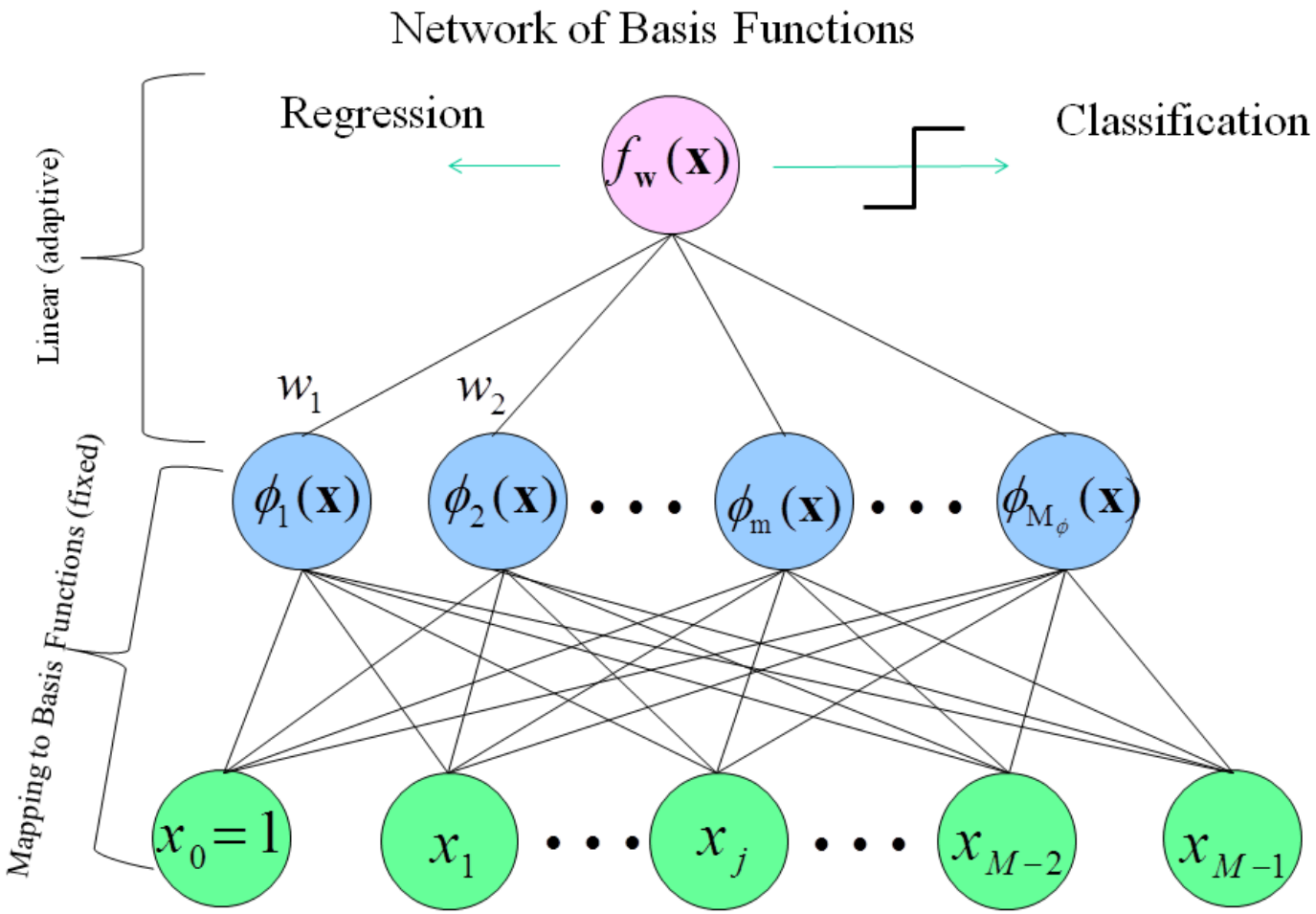
- The simple idea: in addition to the original inputs, we add inputs that are calculated as deterministic functions of the existing inputs and treat them as additional inputs
- Example: Polynomial Basis Functions

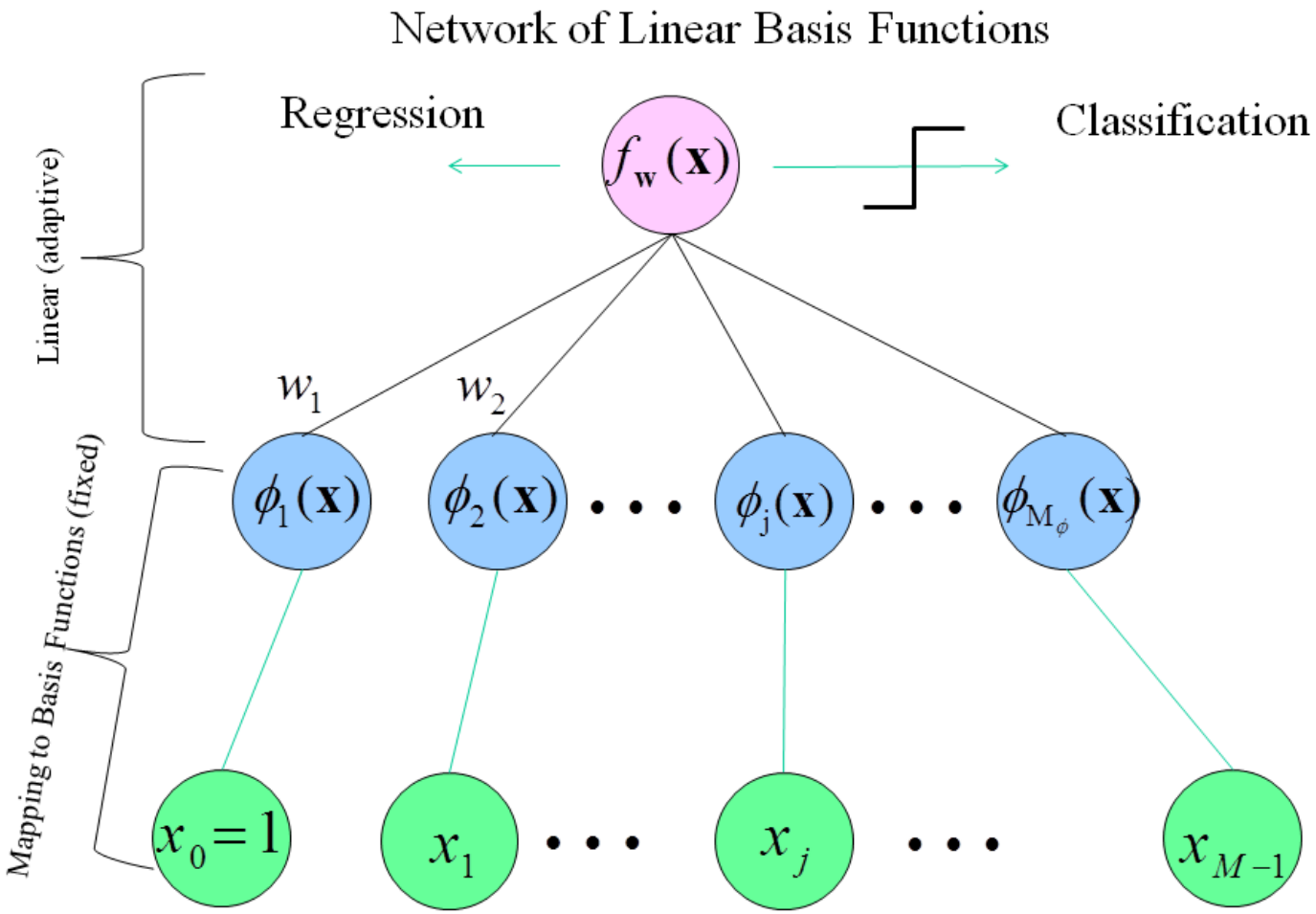
$$\{1, x_1, x_2, x_3, x_1x_2, x_1x_3, x_2x_3, x_1^2, x_2^2, x_3^2\}$$

- 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
$f_{\mathrm{w}}(\mathbf{x})$
Linear (adaptive)
Mapping to Basis Functions (fixed)
w_1
w_2
$\phi_1(\mathbf{x})$
$\phi_2(\mathbf{x})$
$\phi_{\mathrm{m}}(\mathbf{x})$
$\phi_{\mathrm{M}_\phi}(\mathbf{x})$
$x_0 = 1$
x_1
x_j
x_{M-2}
x_{M-1}


Network of Linear Basis Functions
Regression
Classification
$f_{\mathbf{w}}(\mathbf{x})$
Linear (adaptive)
Mapping to Basis Functions (fixed)
w_1
w_2
$\phi_1(\mathbf{x})$
$\phi_2(\mathbf{x})$
$\phi_j(\mathbf{x})$
$\phi_{M_\phi}(\mathbf{x})$
$x_0 = 1$
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

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

- Die penalized LS-Solution

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

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 only basis function weights as free parameters (version 1)

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

- The penalized least-squares solution

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

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?

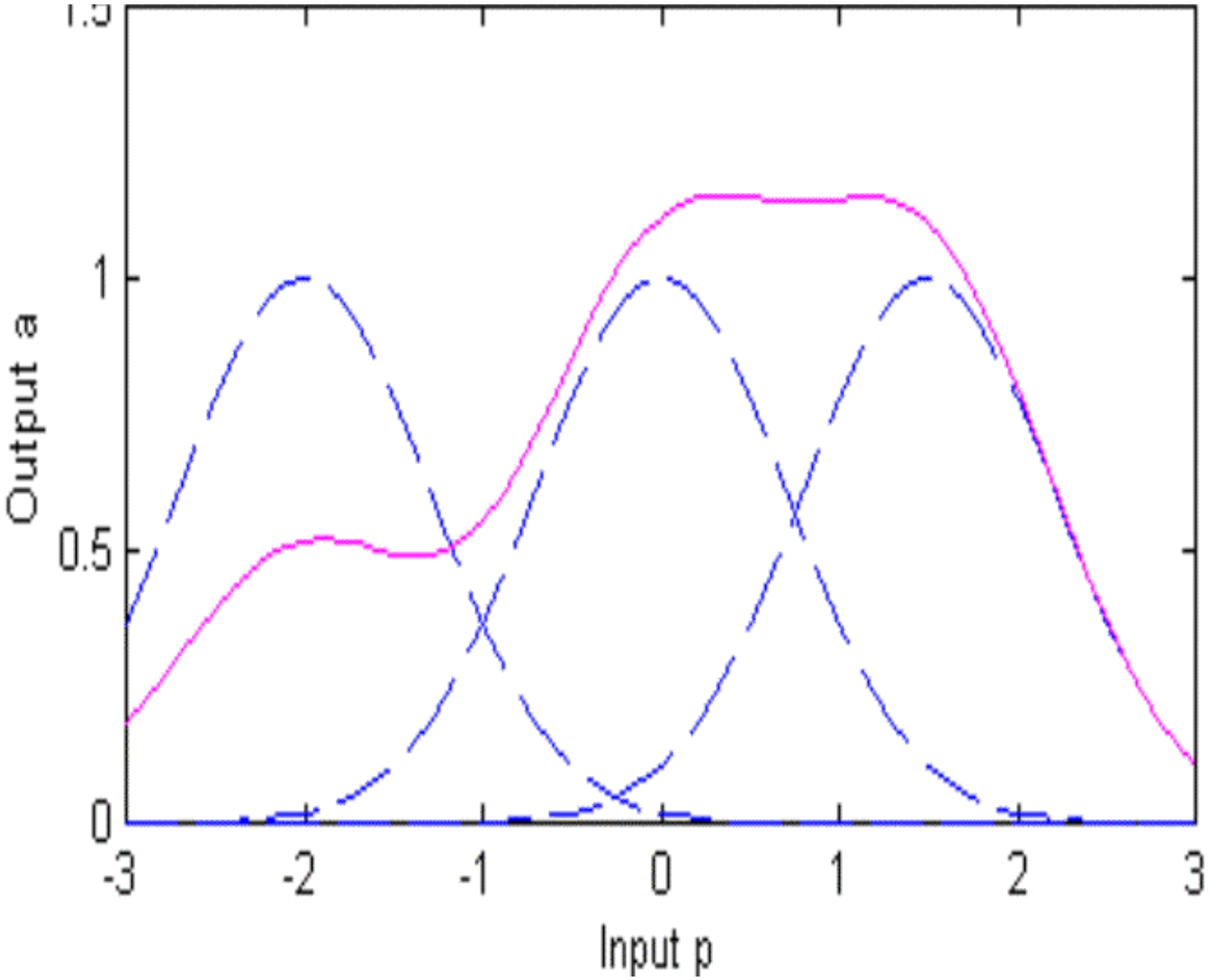
- The challenge is to find problem specific basis functions which are able to effectively model the true mapping, resp. that make the classes linearly separable; in other words we assume that the true dependency $f(\mathbf{x})$ can be modelled by at least one of the functions $f_{\mathbf{w}}(\mathbf{x})$ 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

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

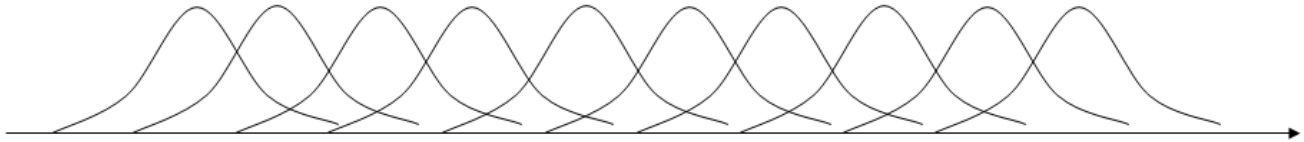
Three RBFs (blue) form $f(\mathrm{x})$ (pink)

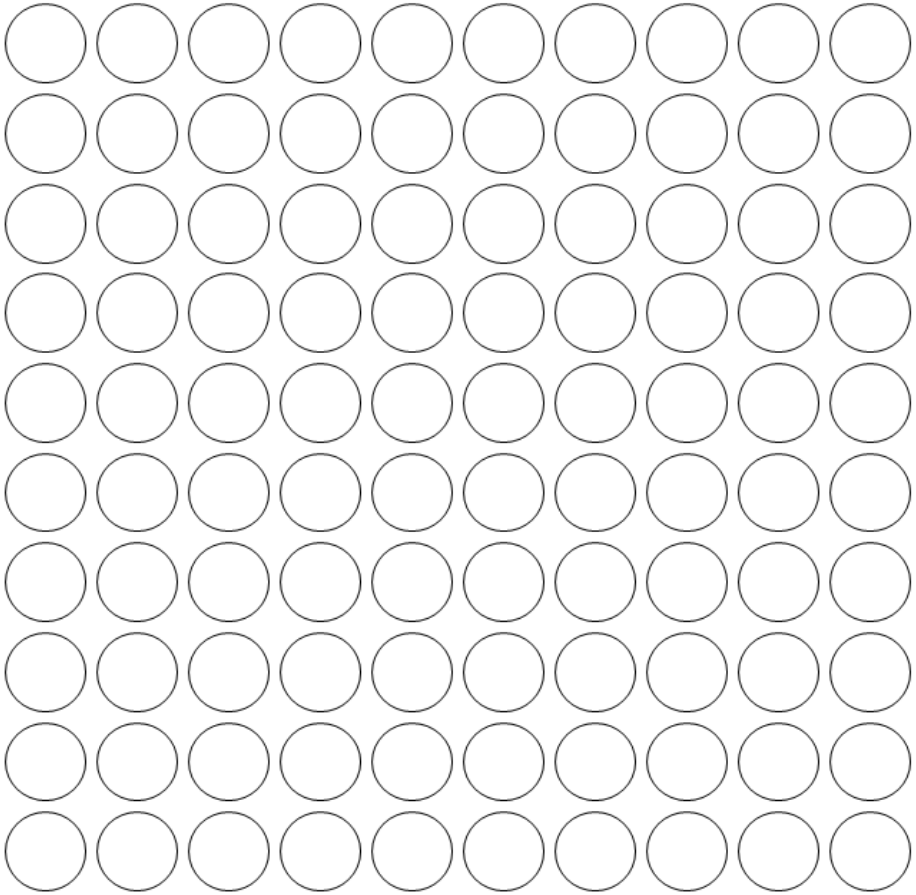


Optimal Basis Functions

- So far all seems to be too simple
- Here is the catch: the number of "sensible" basis functions increases exponential 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

Strategy: Stepwise Increase of Model Class Complexity

- Start with a model class which is too simple and then incrementally add complexity
- First we only work with the original inputs and form a linear model
- Then we stepwise add basis functions that improve the model significantly
- For example we explore all quadratic basis functions. We include the quadratic basis function that mostly decreases the training cost; then we explore the remaining basis functions and, again, include the basis function that mostly decreases the training cost, and so on
- 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

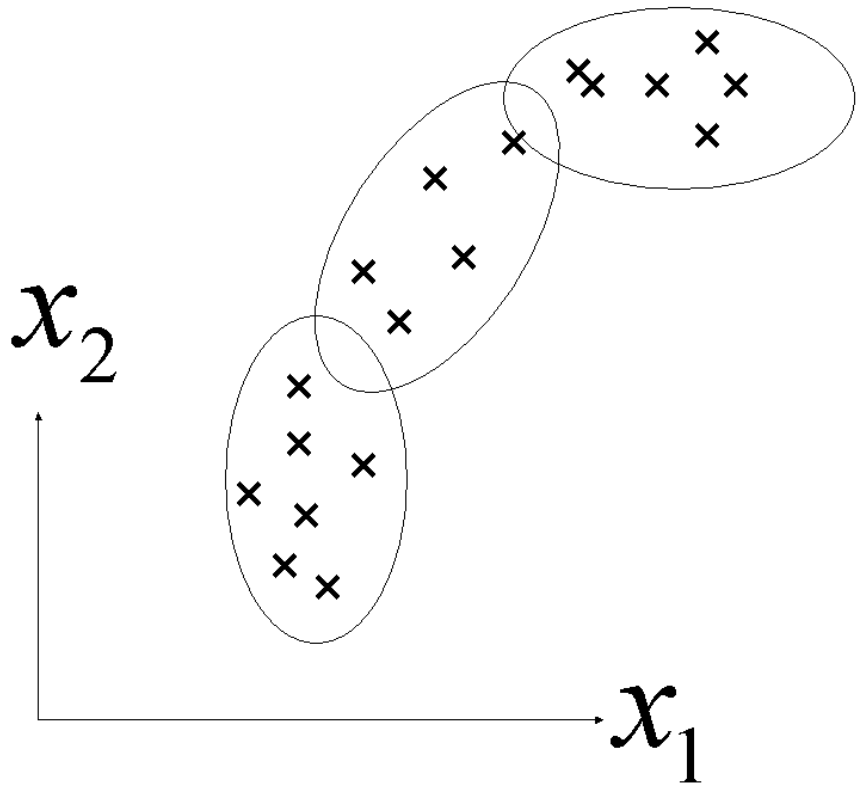
Strategy: Stepwise Decrease of Model Class Complexity (Model Pruning)

- 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 that increase the training cost the least
- A stepwise procedure is not optimal. Better: what is the best subset of K basis functions. Unfortunately, this problem is NP-hard

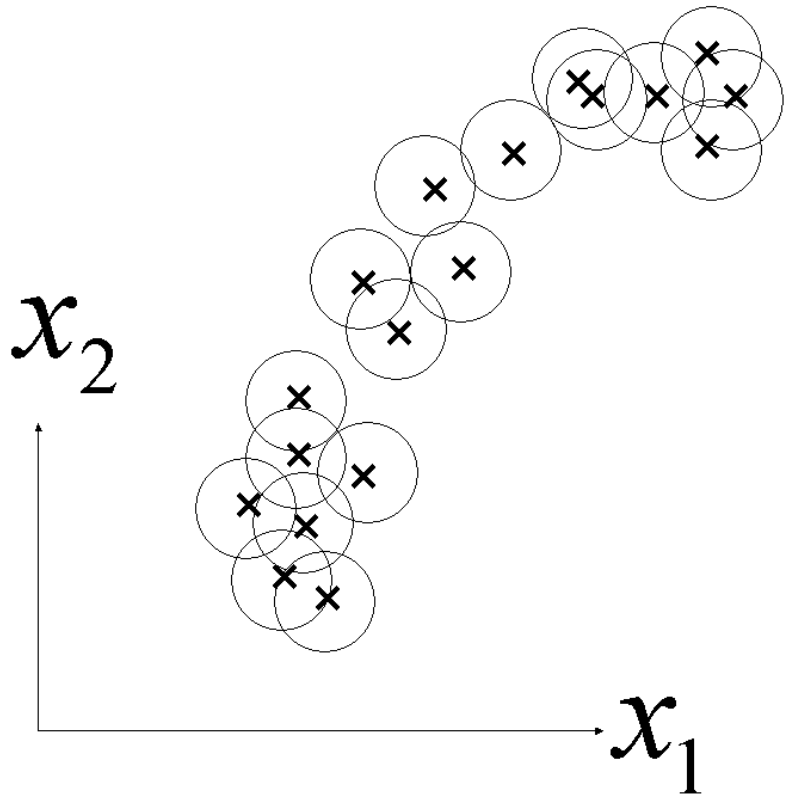
Model Selection: RBFs

- 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!)

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 can be translated in assuming certain correlations between (mostly near-by) function values, implementing a smoothness prior

Consider an Image as a Function of Two Dimensions

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

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

- Typical basis functions used in image analysis are Fourier basis functions and cosine basis functions

Or is an Image a N^2-dimensional Vector

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

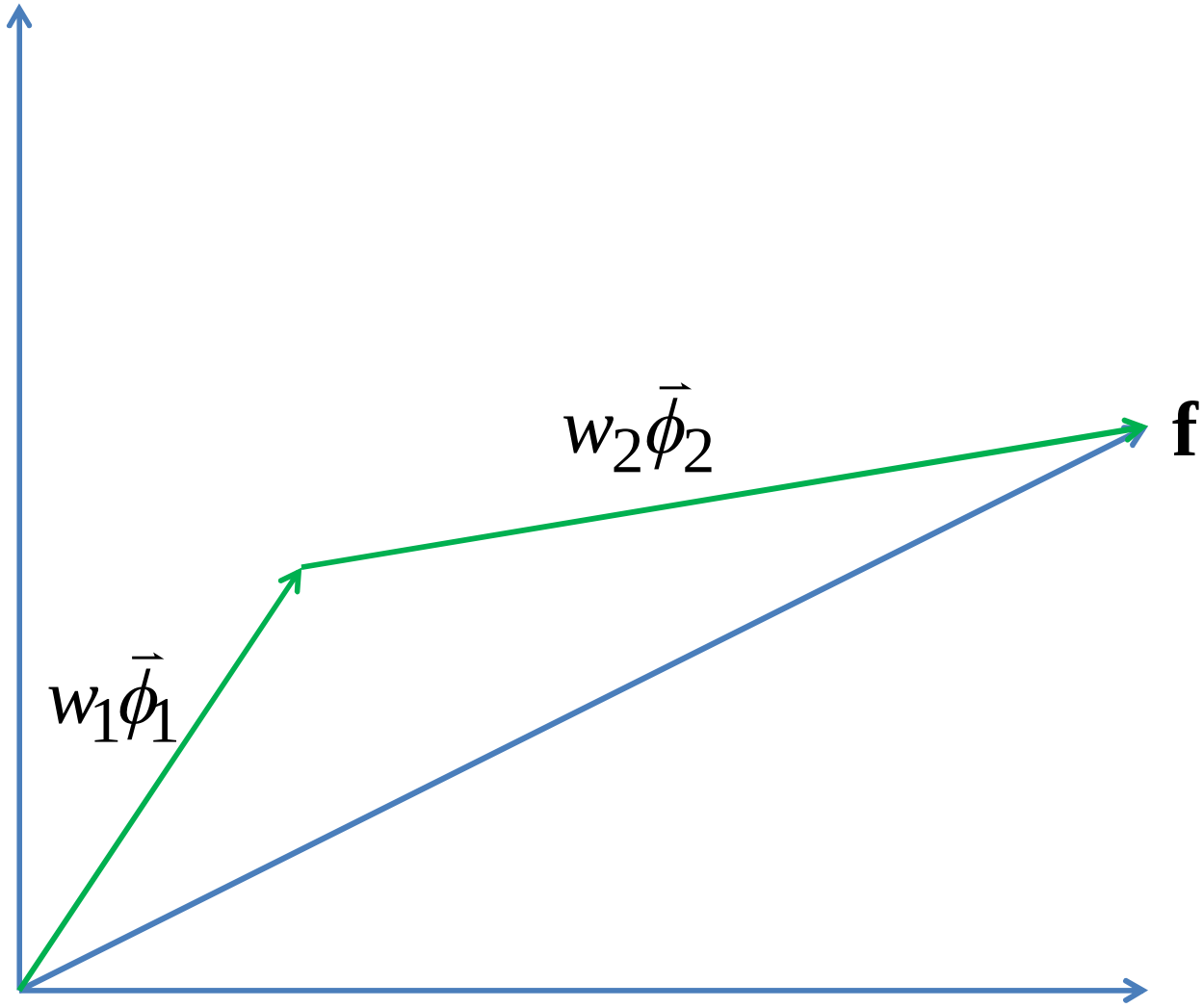
$$f_{i(N-1)+j} = f(i,j) \quad i = 1,\ldots,N, j = 1,\ldots,N$$

- We can model using vector algebra

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

- A continuous function becomes an element in an infinite dimensional vector (Hilbert)
- One can define inner products for functions! Recall that $\langle \mathbf{f}, \mathbf{g} \rangle = \sum_i f_i g_i$ defines an inner product for vectors. A generalization for square integrable function is

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


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