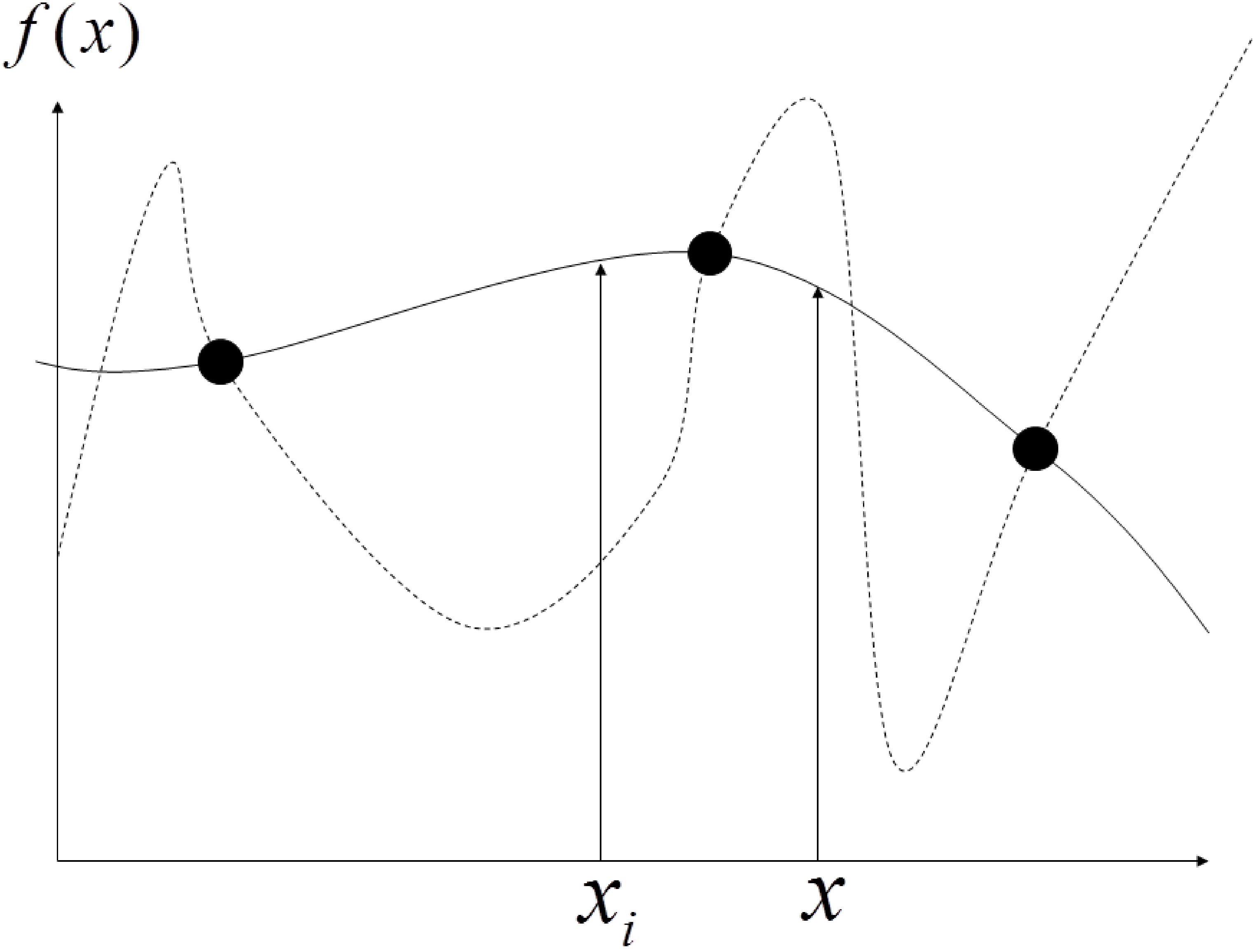
Kernels

Volker Tresp
Summer 2016

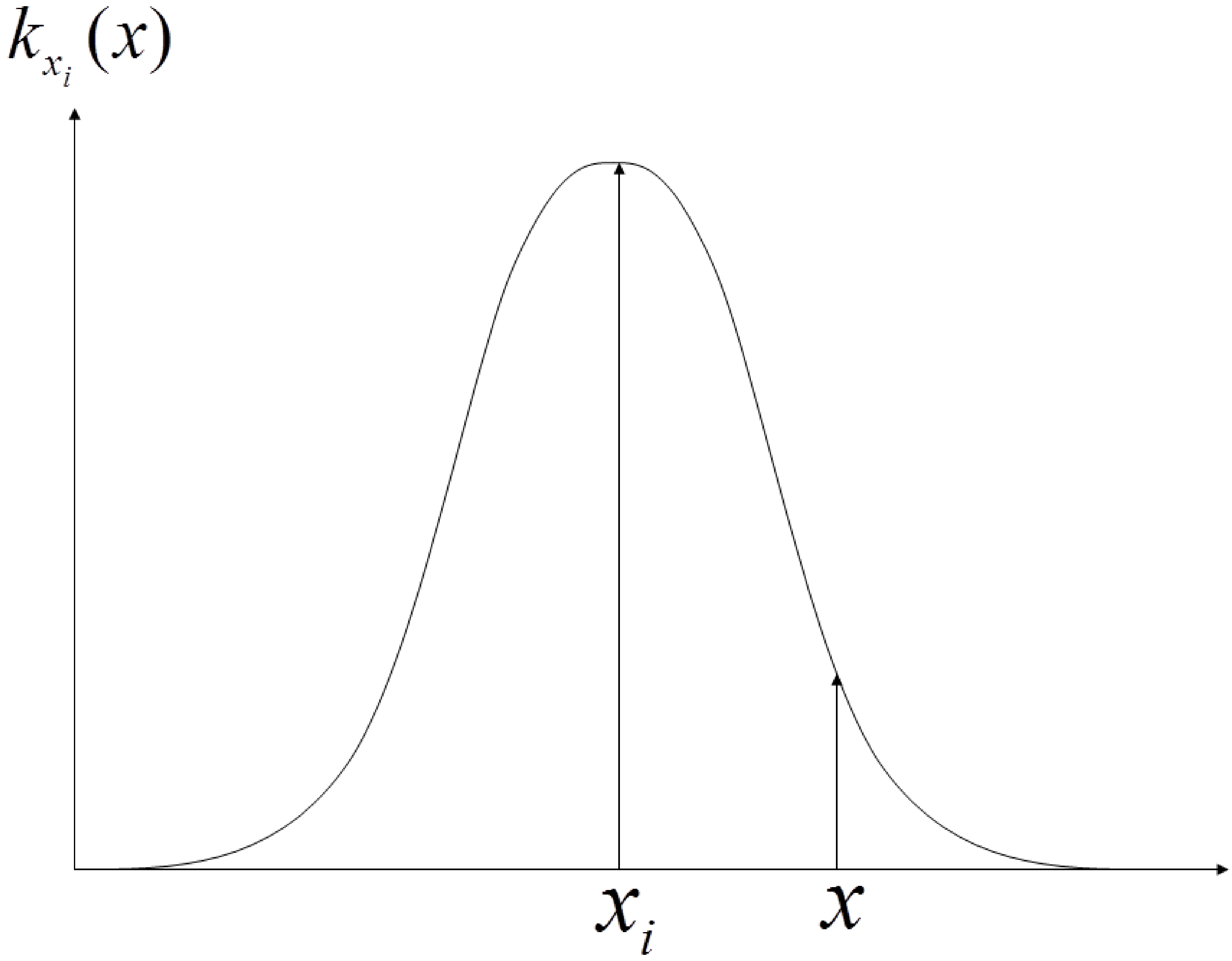
Smoothness Assumption

- So far we used prior knowledge to define the right basis functions: the assumption is that $f(x)$ can be approximated by a weighted sum of basis functions
- Alternatively, it might make sense to have a preference for smooth functions: functional values close in input space should have similar functional values
- In the figure it might make sense that the functional values at $\mathbf{x}_i$ and $\mathbf{x}$ are similar (smoothness assumption)
- Thus, one might prefer the smooth (continuous) function in favor of the dashed function


$f(x)$
x_i
x

Introduction Kernels

- One can implement smoothness assumptions over kernel functions
- A kernel function $k(\mathbf{x}_i, \mathbf{x}) = k_{\mathbf{x}_i}(\mathbf{x})$ determines, how neighboring functional values are influenced when $f(\mathbf{x}_i)$ is given
- Example: Gaussian kernel


$k_{x_i}(x)$
x_i
x

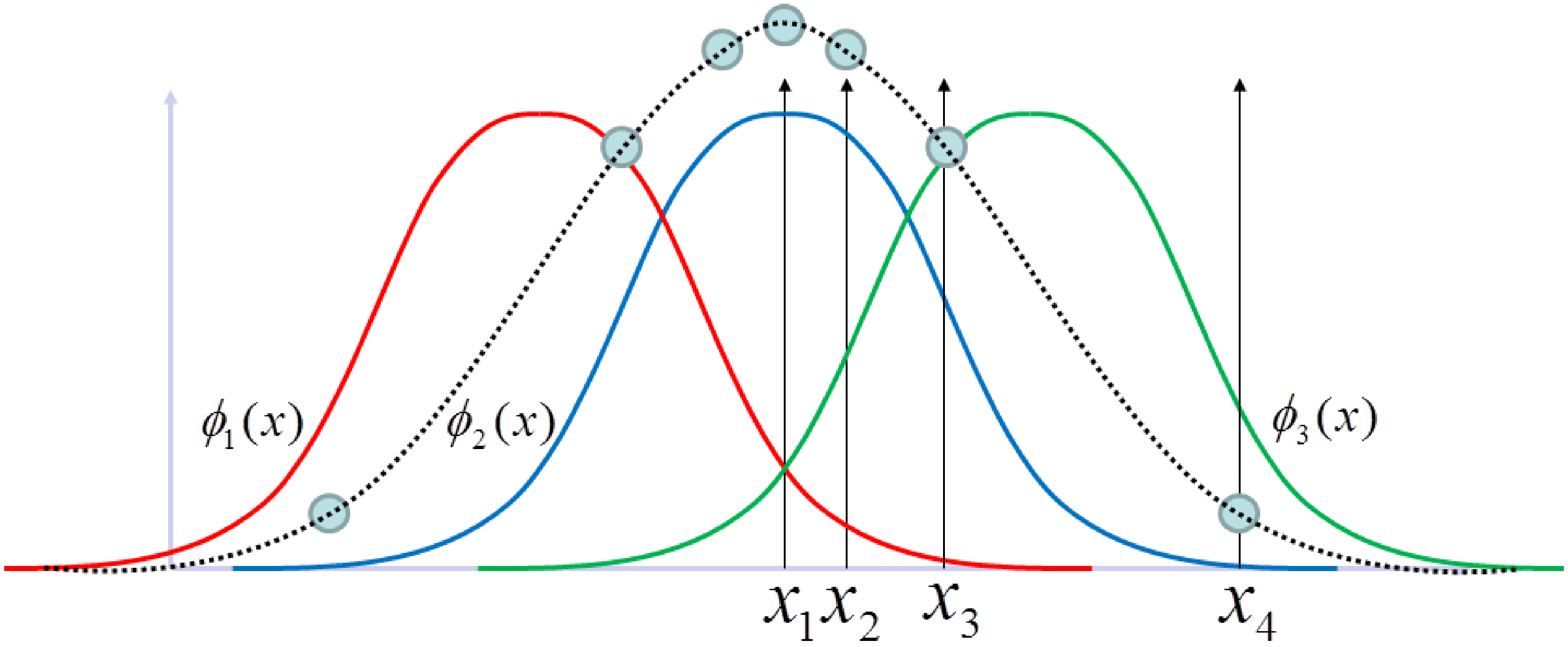
Kernels and Basis Functions

- It turns out that there is a close relationship between kernels and basis functions:

$$k(\mathbf{x}_i, \mathbf{x}) = k_{\mathbf{x}_i}(\mathbf{x}) = \sum_{j=1}^{M_\phi} \phi_j(\mathbf{x}_i)\phi_j(\mathbf{x})$$

- It follows the symmetry: $k(\mathbf{x}_i, \mathbf{x}_j) = k_{\mathbf{x}_i}(\mathbf{x}_j) = k(\mathbf{x}_j, \mathbf{x}_i) = k_{\mathbf{x}_j}(\mathbf{x}_i)$
- Thus: given the M_ϕ basis functions, this equation gives you the corresponding kernel
- Note the kernel is a function of weighted basis functions. The weight $\phi_j(\mathbf{x}_i)$ are the amplitudes of the basis functions at $\mathbf{x}_i$
- As we see later: For positive definite kernels, we can also go the other way: given the kernels I can give you a corresponding set of basis functions (not unique)

Gaussian basis functions (continuous)
Kernel: dotted



$$\vec{\phi}(x_1) = (0.25, 1.00, 0.25)^T \qquad k(x_1, x_1) = \vec{\phi}^T(x_1)\vec{\phi}(x_1) = 1.12$$

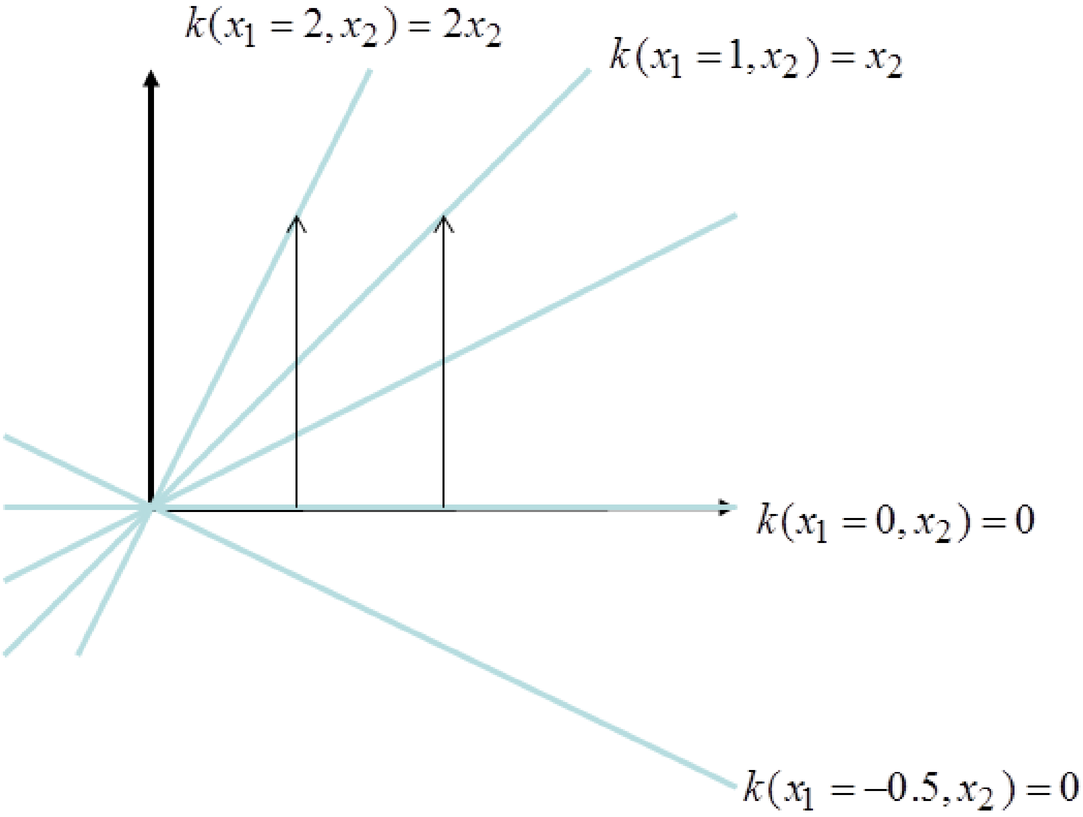
$$\vec{\phi}(x_2) = (0.10, 0.90, 0.50)^T \qquad k(x_1, x_2) = \vec{\phi}^T(x_1)\vec{\phi}(x_2) = 1.05$$

$$\vec{\phi}(x_3) = (0.02, 0.60, 0.90)^T \qquad k(x_1, x_3) = \vec{\phi}^T(x_1)\vec{\phi}(x_3) = 0.83$$

$$\vec{\phi}(x_4) = (0.00, 0.01, 0.30)^T \qquad k(x_1, x_4) = \vec{\phi}^T(x_1)\vec{\phi}(x_4) = 0.08$$

Kernels do not need to look symmetrical: linear kernel in 1-D

$$k(x_1, x_2) = x_1 x_2$$



Kernel Prediction

- Regression

$$\widehat{y}(\mathbf{x}) = \sum_{i=1}^{N} v_i k(\mathbf{x}, \mathbf{x}_i)$$

- Classification

$$\widehat{y}(\mathbf{x}) = \mathsf{sign}\left(\sum_{i=1}^{N} v_i k(\mathbf{x}, \mathbf{x}_i)\right)$$

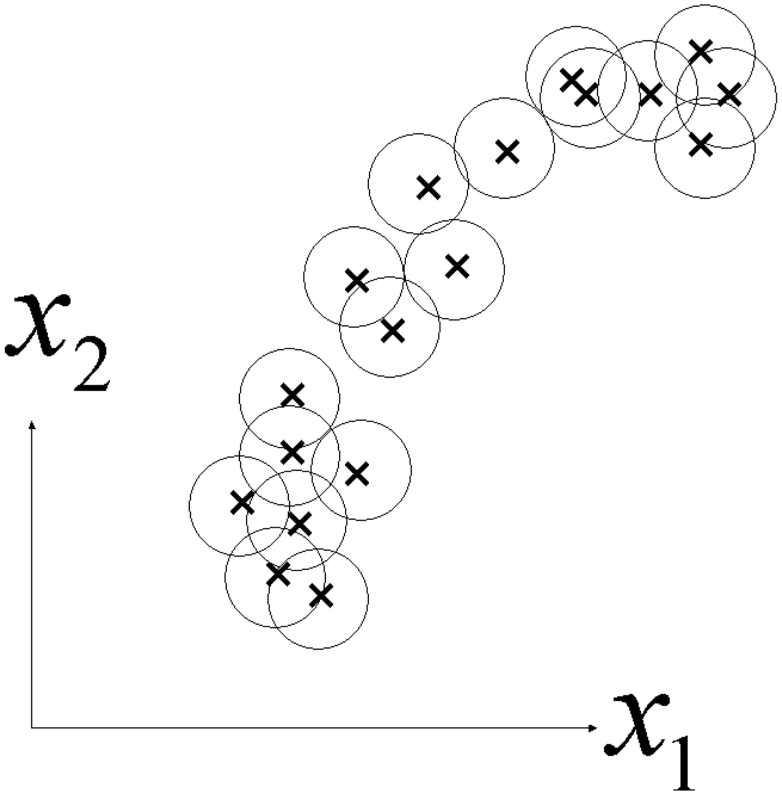
- The solution contains as many kernels as there are data points N (independent on the number of underlying basis functions M)

Kernels: Infinite Number of Basis Functions

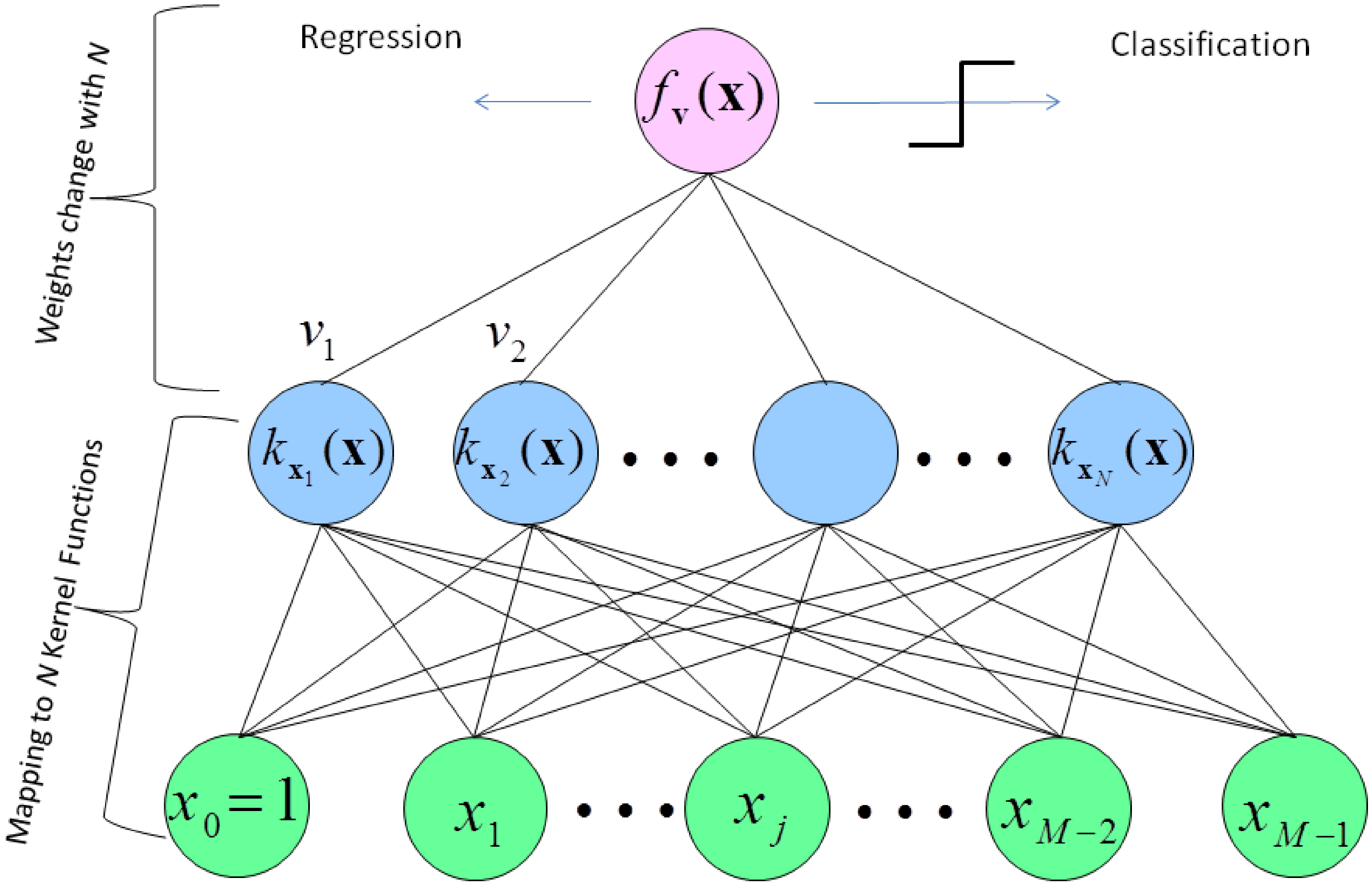
- Thus with $M_\phi \to \infty$: I can work with a finite number N of kernels, instead of an infinite number of basis functions

- So in neural networks, one makes a model of basis functions more flexible by introducing hidden parameters for tuning the basis functions, with kernels one makes the model more flexible by working with an infinite number of fixed basis functions

One Kernel for Each Data Point



Kernel System


Regression
$f_{\mathbf{v}}(\mathbf{x})$
Classification
Weights change with N
v_1
v_2
$k_{\mathbf{x}_1}(\mathbf{x})$
$k_{\mathbf{x}_2}(\mathbf{x})$
$k_{\mathbf{x}_N}(\mathbf{x})$
Mapping to N Kernel Functions
$x_0=1$
x_1
x_j
x_{M-2}
x_{M-1}

Starting with the Cost Function

- We start with the penalized least squares cost function for models with basis functions
- Regularized cost function

$$\mathrm{cost}^{pen}(\mathbf{w}) = \sum_{i=1}^{N}(y_i - \sum_j w_j \phi_j(x_i))^2 + \lambda \sum_{i=0}^{M} w_i^2$$

$$= (\mathbf{y} - \mathbf{\Phi w})^T(\mathbf{y} - \mathbf{\Phi w}) + \lambda \mathbf{w}^T\mathbf{w}$$

where Φ is the design matrix design with $(\mathbf{\Phi})_{i,j} = \phi_j(\mathbf{x}_i)$.

Implicit Solution

- We calculate the first derivatives and set them to zero,

$$\frac{\partial \mathrm{cost}^{pen}(\mathbf{w})}{\partial \mathbf{w}} = -2\mathbf{\Phi}^T(\mathbf{y} - \mathbf{\Phi}\mathbf{w}) + 2\lambda \mathbf{w} = 0$$

It follows that one can write,

$$\mathbf{w}_{pen} = \frac{1}{\lambda}\mathbf{\Phi}^T(\mathbf{y} - \mathbf{\Phi}\mathbf{w}_{pen})$$

Approach

- This is not an explicit solution ($\mathbf{w}_{pen}$ appears on both sides of the equation). But we know now, that we can write the solution as a linear combination of the input vectors

$$\mathbf{w}_{pen} = \mathbf{\Phi}^T \mathbf{v} = \sum_{i=1}^{N} v_i \vec{\phi}(\mathbf{x}_i)$$

- Note that we have a sum over N data points (and not M basis functions)

Kernel Model

- We immediately get,

$$\widehat{f}(\mathbf{x}) = \sum_{j=1}^{M_\phi} w_{j,pen} \phi_j(\mathbf{x}_i) = \vec{\phi}(\mathbf{x})^T \mathbf{w}_{pen}$$

$$= \vec{\phi}(\mathbf{x})^T \mathbf{\Phi}^T \mathbf{v} = \sum_{i=1}^{N} v_i k(\mathbf{x}, \mathbf{x}_i)$$

with $\mathbf{v} = (v_1, \ldots, v_N)^T$ and

$$k(\mathbf{x}, \mathbf{x}_i) = \vec{\phi}(\mathbf{x})^T \vec{\phi}(\mathbf{x}_i) = \sum_{k=1}^{M_\phi} \phi_k(\mathbf{x}) \phi_k(\mathbf{x}_i)$$

- But note that not all functions that can be represented by the basis functions can be written in this form, only the functions that minimize the cost function!

A New Form of the Cost Function

- We can substitute the constraints, and obtain as a cost function with kernel weights as free parameters

$$\text{cost}^{pen}(\mathbf{v}) = (\mathbf{y} - \mathbf{\Phi}\mathbf{\Phi}^T\mathbf{v})^T(\mathbf{y} - \mathbf{\Phi}\mathbf{\Phi}^T\mathbf{v}) + \lambda\mathbf{v}^T\mathbf{\Phi}\mathbf{\Phi}^T\mathbf{v}$$

$$= (\mathbf{y} - K\mathbf{v})^T(\mathbf{y} - K\mathbf{v}) + \lambda\mathbf{v}^T K\mathbf{v}$$

Explicitly

$$\text{cost}^{pen}(\mathbf{v}) = \sum_{i=1}^{N}\left(y_i - \sum_{j=1}^{N} v_j k(x_i, x_j)\right)^2 + \lambda\sum_{i=1}^{N}\sum_{j=1}^{N} v_i v_j k(x_i, x_j)$$

Here K is an $N \times N$ matrix with elements

$$k_{i,j} = \vec{\phi}(\mathbf{x}_i)^T \vec{\phi}(\mathbf{x}_j) = \sum_{k=1}^{M_\phi} \phi_k(\mathbf{x}_i)\phi_k(\mathbf{x}_j)$$

- An important result: **We can write the cost function, such that only dot products of the basis functions appear (i.e., the kernels), but not the basis functions themselves!**

Kernel Parameters

- Now we can take the derivative of the cost function with respect to $\mathbf{v}$ (note, that $K = K^T$)

$$\frac{\partial \mathbf{cost}^{pen}(\mathbf{v})}{\partial \mathbf{v}} = 2K(\mathbf{y} - K\mathbf{v}) + 2\lambda K\mathbf{v}$$

such that

$$\mathbf{v}_{pen} = (K + \lambda I)^{-1}\mathbf{y}$$

- For comparison: *If we would have treated the kernels at the data points as fixed basis functions*, the solution would have been

$$\mathbf{v}_{pen} = (K^T K + \lambda I)^{-1} K^T \mathbf{y}$$

Kernel Prediction

- A prediction can be written as

$$\hat{f}(\mathbf{x}) = \vec{\phi}(\mathbf{x})^T \mathbf{w} = \vec{\phi}(\mathbf{x})^T \mathbf{\Phi}^T \mathbf{v}_{pen} = \sum_{i=1}^{N} v_i k(\mathbf{x}, \mathbf{x}_i)$$

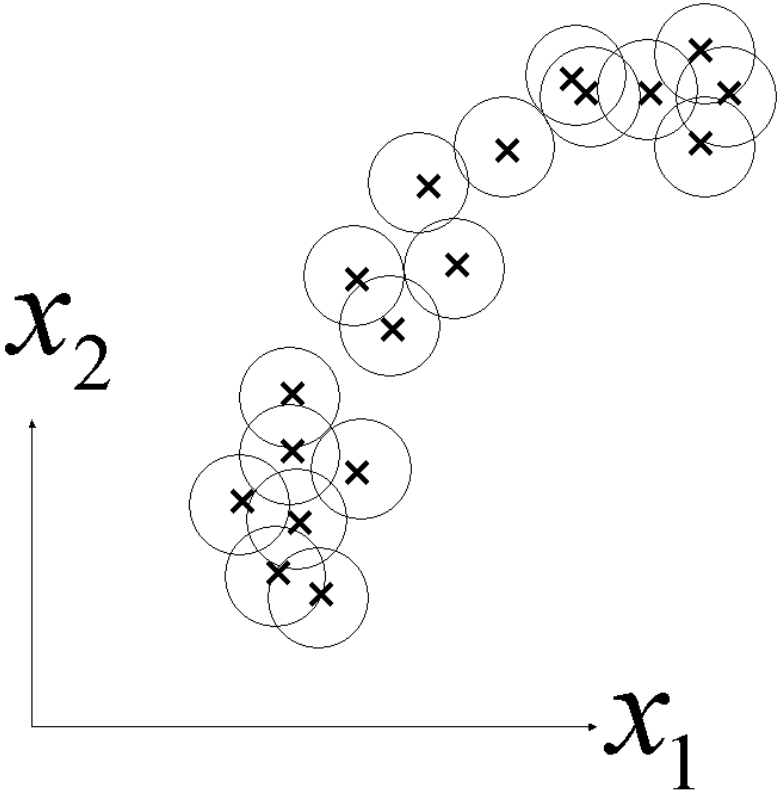
with

$$k(\mathbf{x}, \mathbf{x}_i) = \vec{\phi}(\mathbf{x})^T \vec{\phi}(\mathbf{x_i})$$

- Another important result: we can write the solution such that only dot products are used; **the solution can be written as a weighted sum of N kernels**.

- We want to point out again, that *not* each function that can be written as $\sum_j w_j \phi_j(x_i)$ can be expressed in this way, only a subset of the functions and in particular that one which minimizes the cost function based on the specific N training data points!

One Kernel for Each data Point

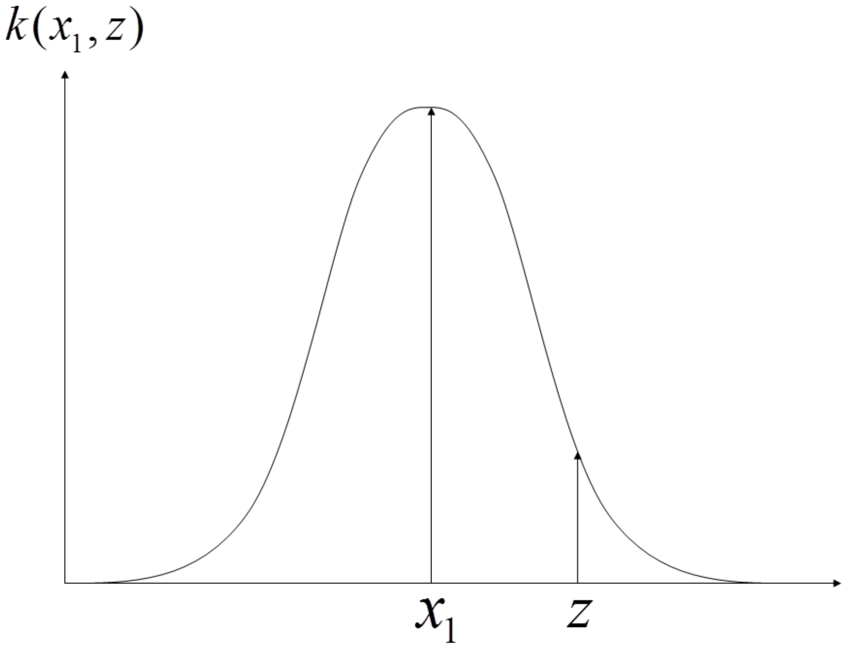


With only One Training Data Point

- With only one training data point we get

$$f(\mathbf{x}) = v_1 k(\mathbf{x}, \mathbf{x}_1)$$

- As discussed previously:



Comments and Interpretation of a Kernel

- This is interesting, since there can be more basis functions than data points; in particular this result is valid, even if we work with an **infinite number of basis functions**!
- It is even possible to start with the kernels, without knowing exactly, what the underlying basis functions are
- Different interpretations of the kernel
  - As a dot product $k(\mathbf{x}_i, \mathbf{x}) = \vec{\phi}^T(\mathbf{x})\vec{\phi}(\mathbf{x})$
  - As covariance: how strong is the correlation of the functional values

$$\textbf{cov}(f(\mathbf{x}_i), f(\mathbf{x}_j)) = \textbf{cov}(\vec{\phi}(x_i)\mathbf{w}, \vec{\phi}(x_j)\mathbf{w}) = \vec{\phi}(x_i)\ \textbf{cov}(\mathbf{w})\vec{\phi}(x_i)$$

$$= k(\mathbf{x}_i, \mathbf{x}_j)$$

  when we assume that *a priori* $\textbf{cov}(\mathbf{w}) = I$.

  - This identity is used in Gaussian process regression

  - Note that the dot product of the activations of the basis functions at two locations ("empirical correlation") defined the *a priori* covariance of the corresponding functional values

- When $N >> M$ it is computationally more efficient to work with basis functions (requiring $M^3 + M^2N$ operations). When $M >> N$, the kernel version is more efficient, requiring $N^3 + N^2M$ operations. If the kernels are known a priori (i.e., if they do not need to be calculates via dot product), the kernel solution requires N^3 operations.

- Still, not all functions are valid kernel functions. We need the following theorem ...

Mercer's Theorem

- (From Vapnik: The nature of statistical learning theory. Springer, 2000)

- *Mercer's Theorem:* To guarantee, that the symmetric functions $k(\mathbf{x}, \mathbf{z}) = k(\mathbf{z}, \mathbf{x})$ from L_2 permits an expansion as

  $$k(\mathbf{z}, \mathbf{x}) = \sum_{h=1}^{\infty} \lambda_h \phi_h^T(\mathbf{z}) \phi_h(\mathbf{x})$$

  with positive coefficients $\lambda_h > 0$, it is necessary and sufficient, that

  $$\int \int k(\mathbf{z}, \mathbf{x}) g(\mathbf{x}) g(\mathbf{z}) d\mathbf{x} d\mathbf{z} > 0$$

  for all $g \neq 0$, for which

  $$\int g^2(\mathbf{x}) d\mathbf{x} < \infty$$

- The theorem says, that for so-called positive-definite kernels ("Mercer kernels"), a decomposition in basis functions is possible!

- Each kernel-matrix K is then also positive definite, $\mathbf{a}^T K \mathbf{a} > 0$, for all vectors $\mathbf{a} \neq 0$. A symmetric matrix is positive definite iff all its eigenvalues are positive
- The results also generalize to the non-negative (positive-semidefinite) case

Kernel Design

- Linear Kernel

$$k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$$

The kernel matrix is then $K = XX^T$

- Polynomial kernel (1)

$$k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j)^d$$

The basis functions are all ordered polynomials of order d

- Polynomial kernel (2)

$$k(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j + R)^d$$

The corresponding basis functions are all polynomials of order d **or smaller**. R is a tuning parameter

- Gauß-kernels (RBF-kernels)

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

These kernels correspond to infinitely many Gaussian basis functions

- Sigmoid (“neural network”) kernels

$$k(\mathbf{x}_i, \mathbf{x}_j) = \mathsf{sig}\left(\mathbf{x}_i^T \mathbf{x}_j\right)$$

Comment on Valid Kernels

- A necessary condition is that $k(\mathbf{x}_i, \mathbf{x}_j) = k(\mathbf{x}_j, \mathbf{x}_i)$!
- So any function of $\|\mathbf{x}_i - \mathbf{x}_j\|$ would be a good candidate. These kernels also appear symmetrical, like a Gaussian kernel
- But note that also any function of $\|\mathbf{x}_i^T \mathbf{x}_j\|$ would be a good candidate. They don't necessarily look symmetrical, like the linear kernel or the polynomial kernel
- Here is an example of a kernel that violates the necessary condition

$$k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j + \alpha \|\mathbf{x}_i\|^2$$

- The kernels discussed here are called dot-product kernels, Mercer kernels, or kernels in a reproducing kernel Hilbert space
- Kernels are widely used in mathematics. The kernels used here should, for example, not to be confused with the kernels used in kernel smoothing!

Sometimes it is Easier to Define Sensible Kernels than it is to Define Sensible Basis Functions

- Example: Classification of chemical graphs
  - Molecules can be described as graphs (structural formula, chemical graph theories)
  - Task: I know from N molecules, if these have a particular medical effect (training data). Can I predict the medical effect of a new molecule?
  - Features which describe a chemical structure formula are difficult to describe; it is easier to define graph kernels
- Example: Classification of a person in a social network
  - Kernels reflect similarity with respect to a network topology. For example, one can define a kernel based on the number of overlapping substructures of two persons in their mutual neighborhoods

Representer Theorem

- *Representer Theorem:* Let Ω be a strictly monotonously increasing function and let $\mathsf{loss}()$ be an arbitrary loss function, then the minimizer of the loss function

$$\sum_{i=1}^{N} \mathsf{loss}(y_i, f(\mathbf{x}_i)) + \Omega(\|\mathbf{f}\|_\phi)$$

  can be represented as

$$f(\mathbf{x}) = \sum_{i=1}^{N} v_i k(\mathbf{x}_i, \mathbf{x})$$

- $\|\mathbf{f}\|_\phi = \sqrt{\langle \mathbf{f}, \mathbf{f} \rangle_\phi}$ is a norm in a *reproducing kernel Hilbert space* (RKHS) and includes $\|\mathbf{f}\|_\phi = \mathbf{w}^T\mathbf{w}$

- So kernel solutions are possible for **all cost functions we are considering**!