Kernels

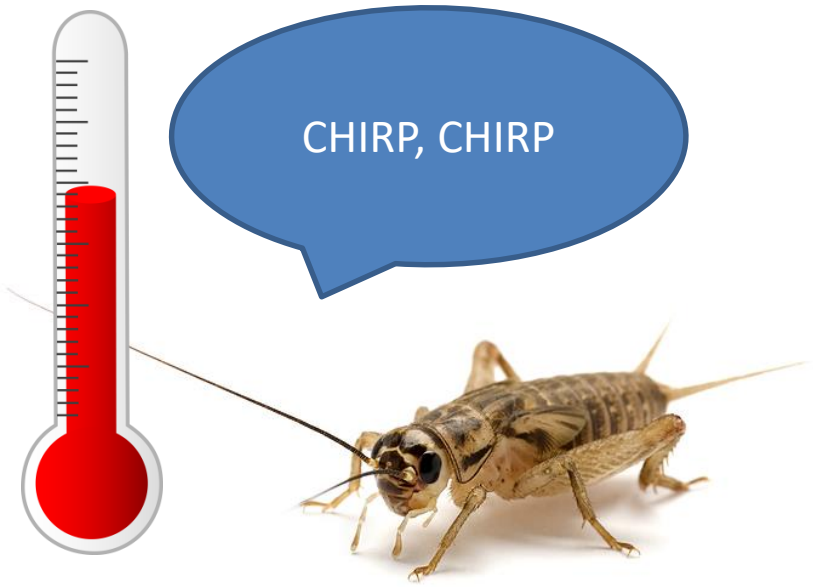
Lecture given by Grady Jensen

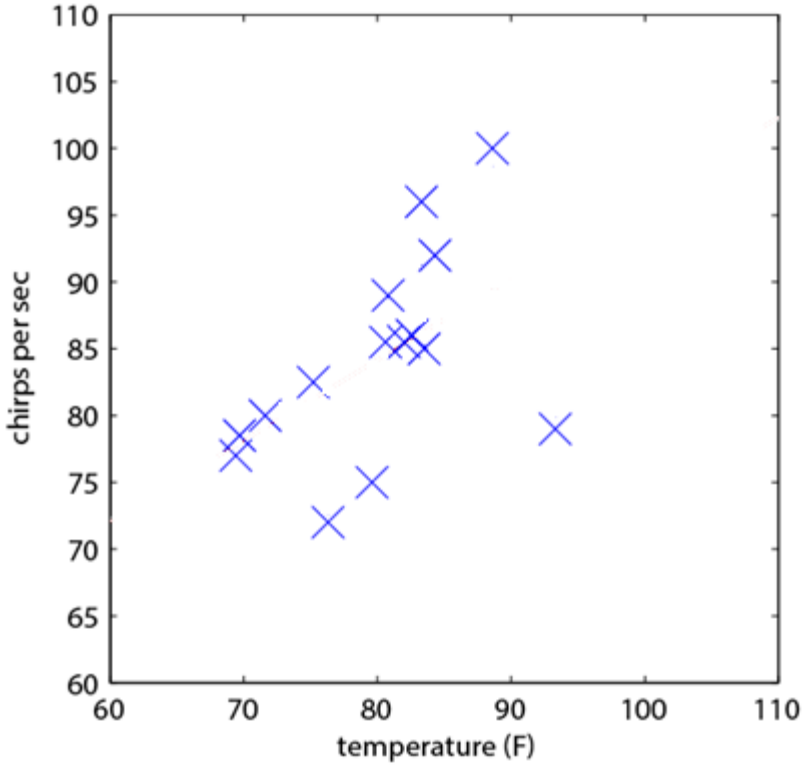
Slides created by Sebastian Urban

Notation

$\mathcal{D}$	$\mathcal{D}$	dataset with N samples
x_n	$x^{(n)}$	n-th input sample
z_n	$y^{(n)}$	target output for n-th sample
Z	$\boldsymbol{y}$	$= \left(y^{(1)}, y^{(2)}, \ldots, y^{(N)}\right)^T$
$y(x)$	$y(x)$	output computed for input x by model
$y(x_n)$	$Y^{(n)}$	output computed for n-th input sample by model
	$\boldsymbol{Y}$	$= \left(Y^{(1)}, Y^{(2)}, \ldots, Y^{(N)}\right)^T$
W	$\boldsymbol{w}$	weights (model parameters)

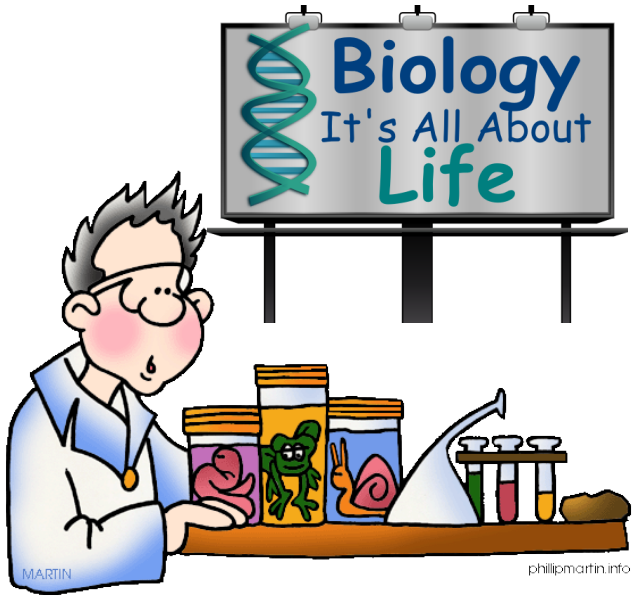
Crickets





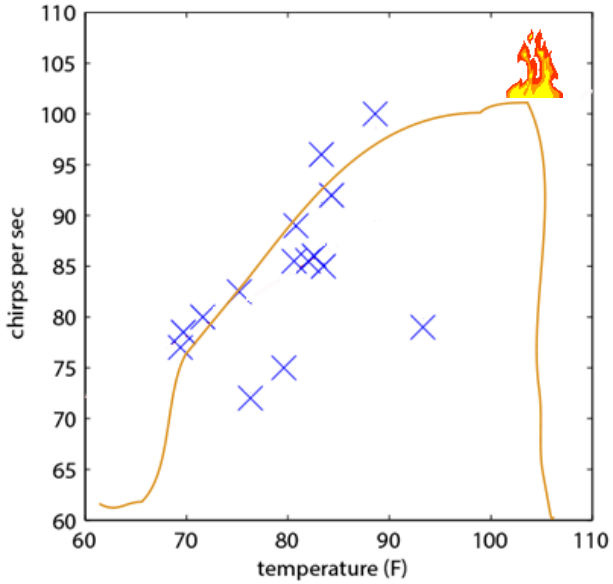
Chirps per second versus temperature.

In an ideal world…



Cricket-Chirp model: $y = f_w(x)$

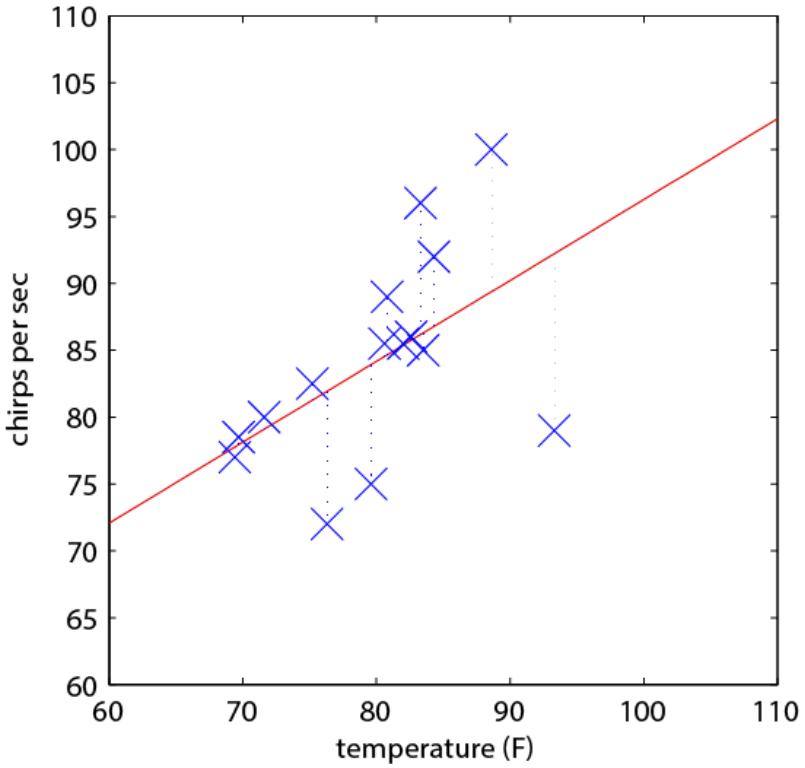
fit model parameters w



Suppose that a biologist is not available.

Can we make predictions without a scientific model?

Linear regression



Chirps per second versus temperature.

Given the dataset

$$\mathcal{D} = \{(x^{(n)}, y^{(n)}), n = 1, \dots, N\}$$

we want to fit the linear model

$$y(x) = ax + b\,.$$

We define the *feature map*

$$\boldsymbol{\phi}: \mathbb{R} \to \mathbb{R}^2, \qquad \boldsymbol{\phi}(x) = (x, 1)^T$$

and can formulate our linear model as

$$y(x) = \boldsymbol{w}^T \boldsymbol{\phi}(x)$$

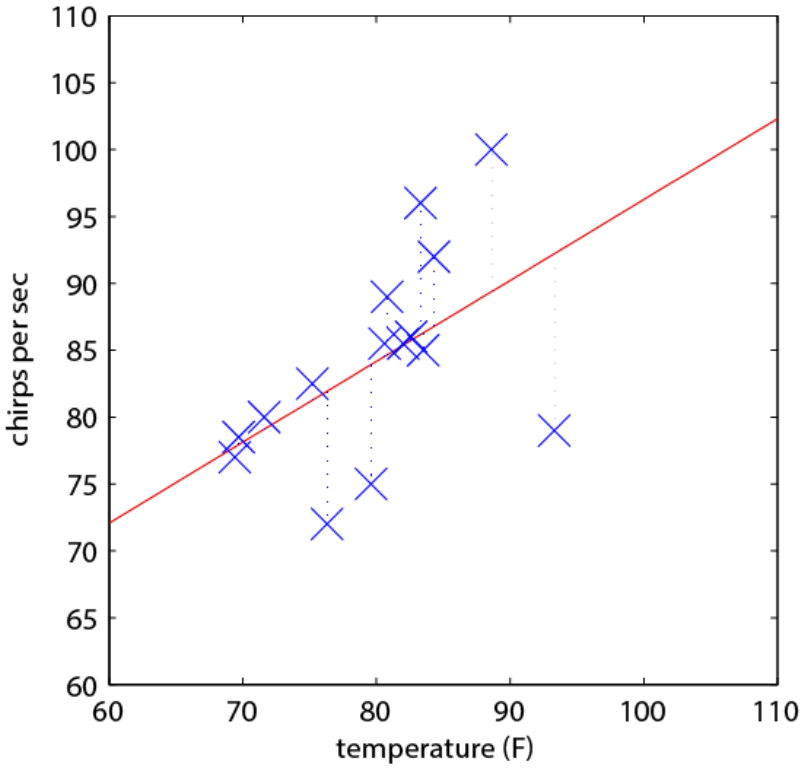
with $\boldsymbol{w} = (a, b)^T$.

Using the squared prediction error we obtain the objective function

$$E(\boldsymbol{w}) = \sum_{n=1}^{N} \left[y^{(n)} - \boldsymbol{w}^T \boldsymbol{\phi}^{(n)} \right]^2$$

where $\boldsymbol{\phi}^{(n)} := \boldsymbol{\phi}(x^{(n)})$.

Linear regression in matrix form



Chirps per second versus temperature.

In matrix form the error function is

$$E(\boldsymbol{w}) = (\boldsymbol{y}^T - \boldsymbol{w}^T\boldsymbol{\Phi})^{\mathrm{T}}(\boldsymbol{y}^T - \boldsymbol{w}^T\boldsymbol{\Phi})$$

with $\boldsymbol{y}_n = y^{(n)}$ and $\boldsymbol{\Phi}_{ij} = \phi_i\left(\boldsymbol{x}^{(j)}\right)$.

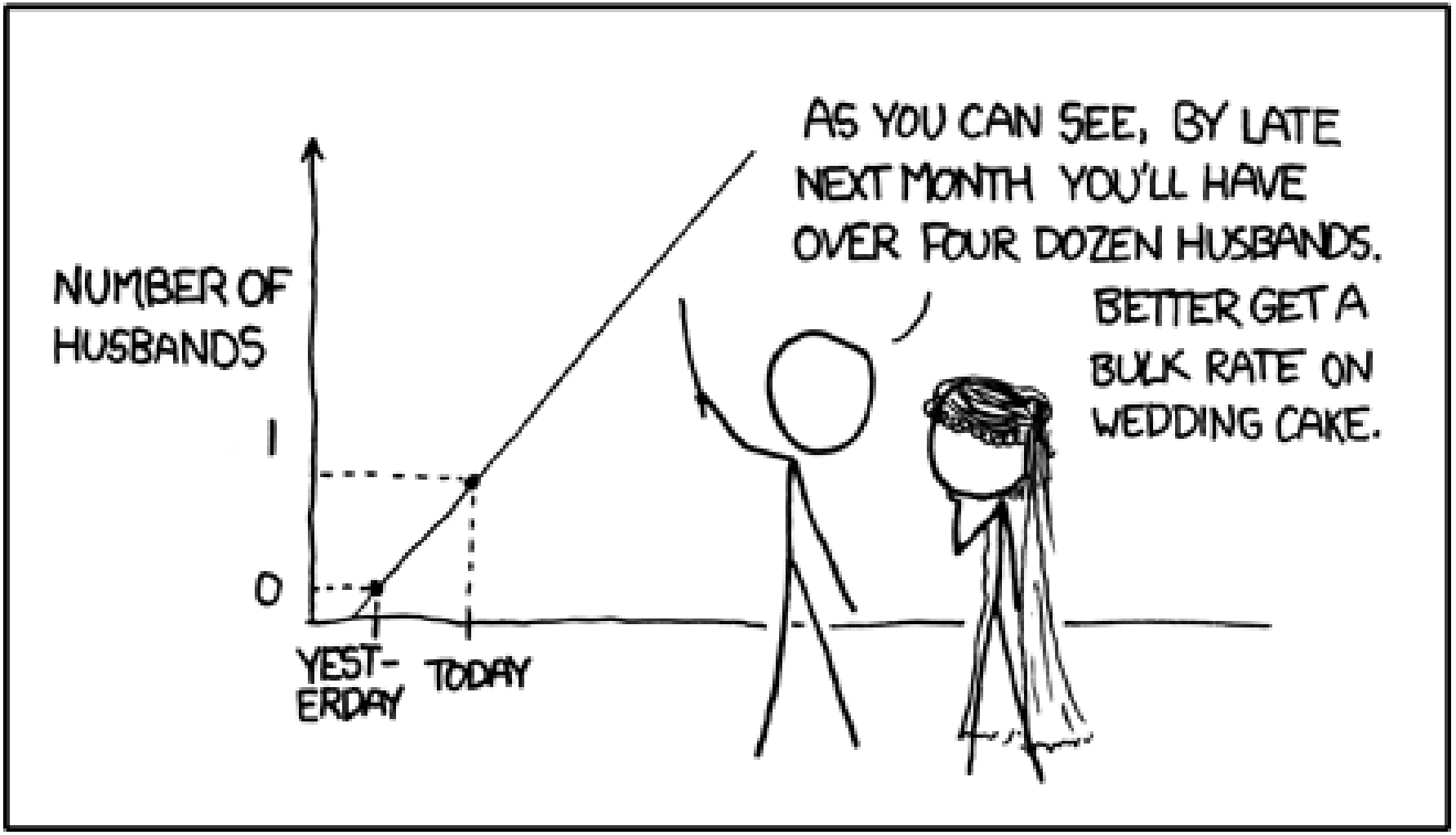
Minimum error is obtained by

$$\boldsymbol{w} = (\boldsymbol{\Phi}\boldsymbol{\Phi}^T)^{-1}\boldsymbol{\Phi}\,\boldsymbol{y}$$

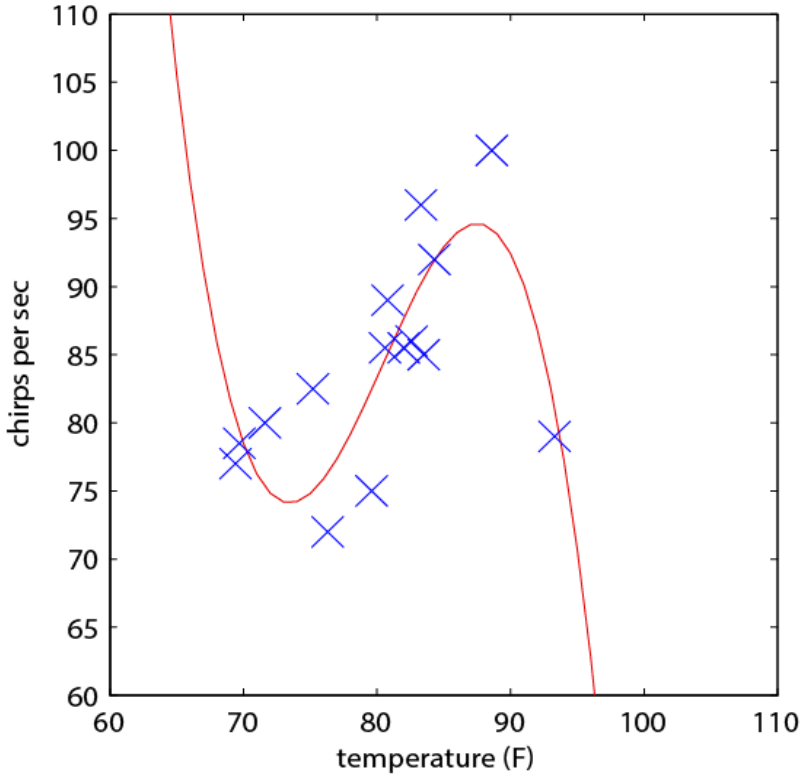
where $(\boldsymbol{\Phi}\boldsymbol{\Phi}^T)^{-1}\boldsymbol{\Phi}$ is called the Moore-Penrose pseudo-inverse of $\boldsymbol{\Phi}$.

$(\boldsymbol{\Phi}\boldsymbol{\Phi}^T)$ is a MxM matrix where M is the number of basis functions.
$\Rightarrow O(M^3)$ operations to calculate $(\boldsymbol{\Phi}\boldsymbol{\Phi}^T)^{-1}$.

MY HOBBY: EXTRAPOLATING



More complex basis functions



Chirps per second versus temperature.

We can also use a more complex model, for example a 3th order polynomial

$$y(x) = w_1 + w_2 x + w_3 x^2 + w_4 x^3 \, .$$

This yields the feature map

$$\boldsymbol{\phi}(x) = \begin{pmatrix} 1 \\ x \\ x^2 \\ x^3 \end{pmatrix} \, .$$

Quality of solution? **Worse**.
Model complexity? **Higher**.
Why? Model does not match data.

Dealing with unknown basis functions

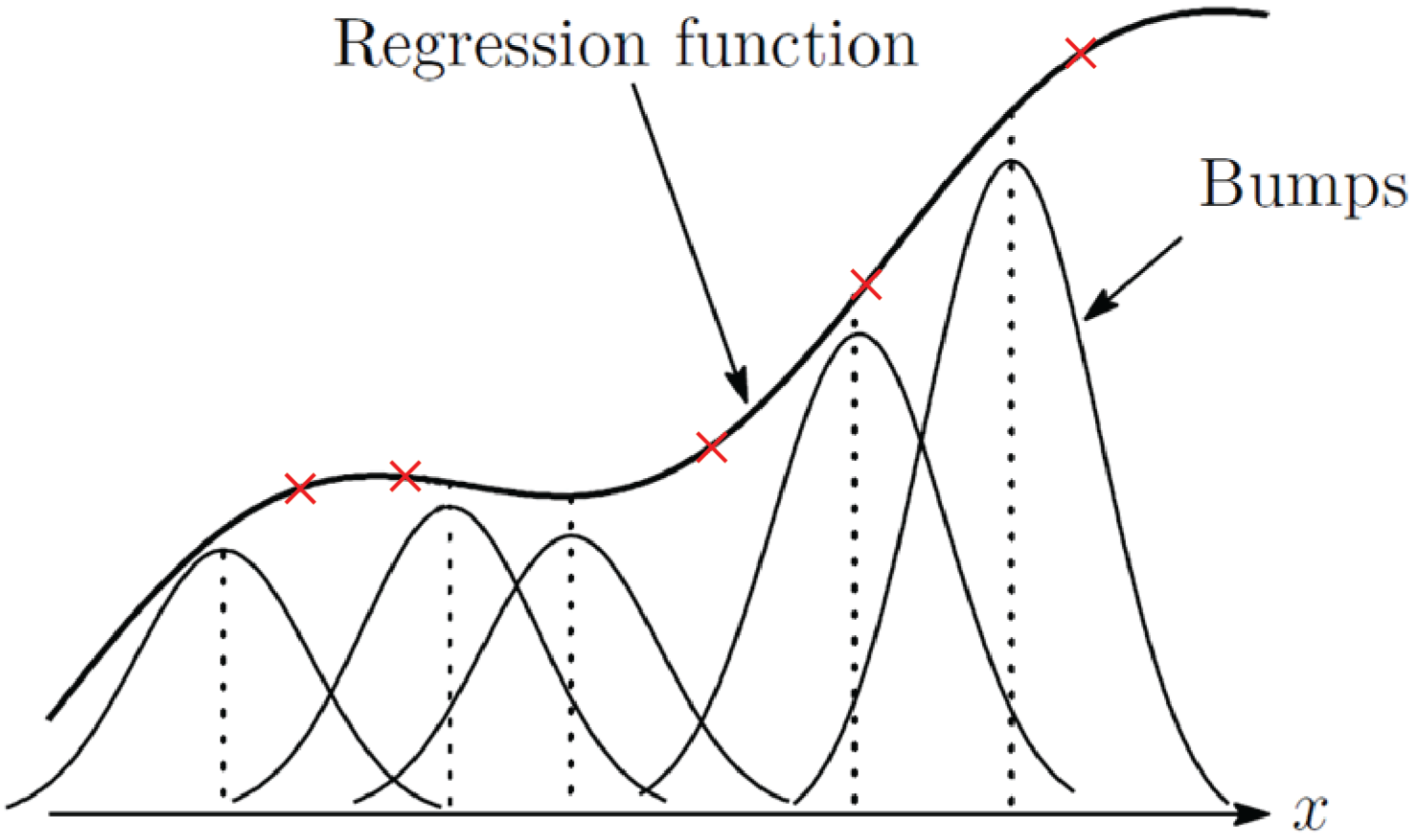
One way to deal with overfitting is by using regularization. However it will not help much if the basis functions do not match the data.

So, what should we do if we do not know the model?

Is there a **universal set of basis functions** that can approximate arbitrary functions reasonably well?

Linear regression using “bump” functions

Approximate the function with a sum of bump-shaped functions.



H. Takeda. Kernel Regression for Image Processing and Reconstruction. 2006.

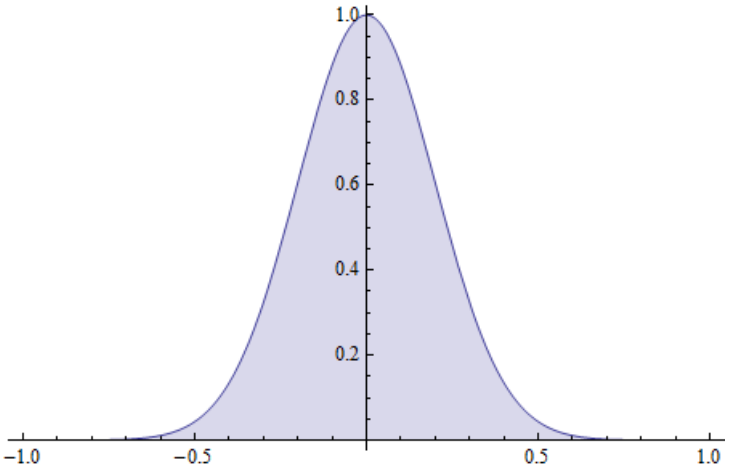
Radial basis functions (RBFs)

Gaussian Radial Basis Functions (RBFs):

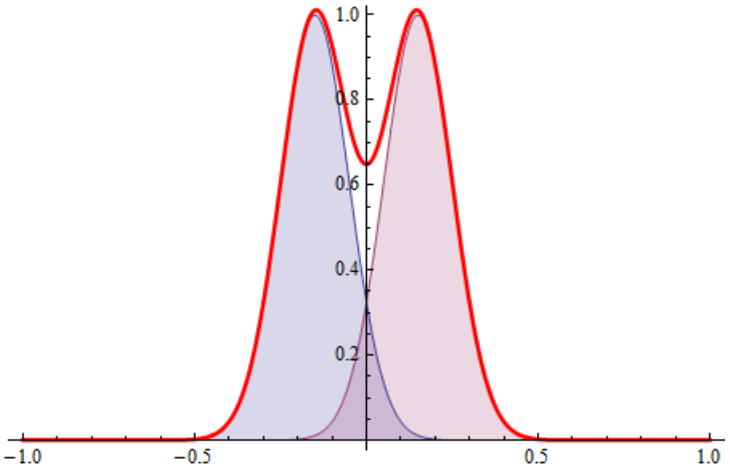
$$\phi_i(\boldsymbol{x}) = \exp\left(-\frac{\left(\boldsymbol{x} - \boldsymbol{m}^{(i)}\right)^2}{2\alpha^2}\right)$$

The center is at $\boldsymbol{m}^{(i)}$ and α determines the width of the bump.

Advantage: They decrease smoothly to zero and do not show osciallatory behavior unlike higher order polynomials.

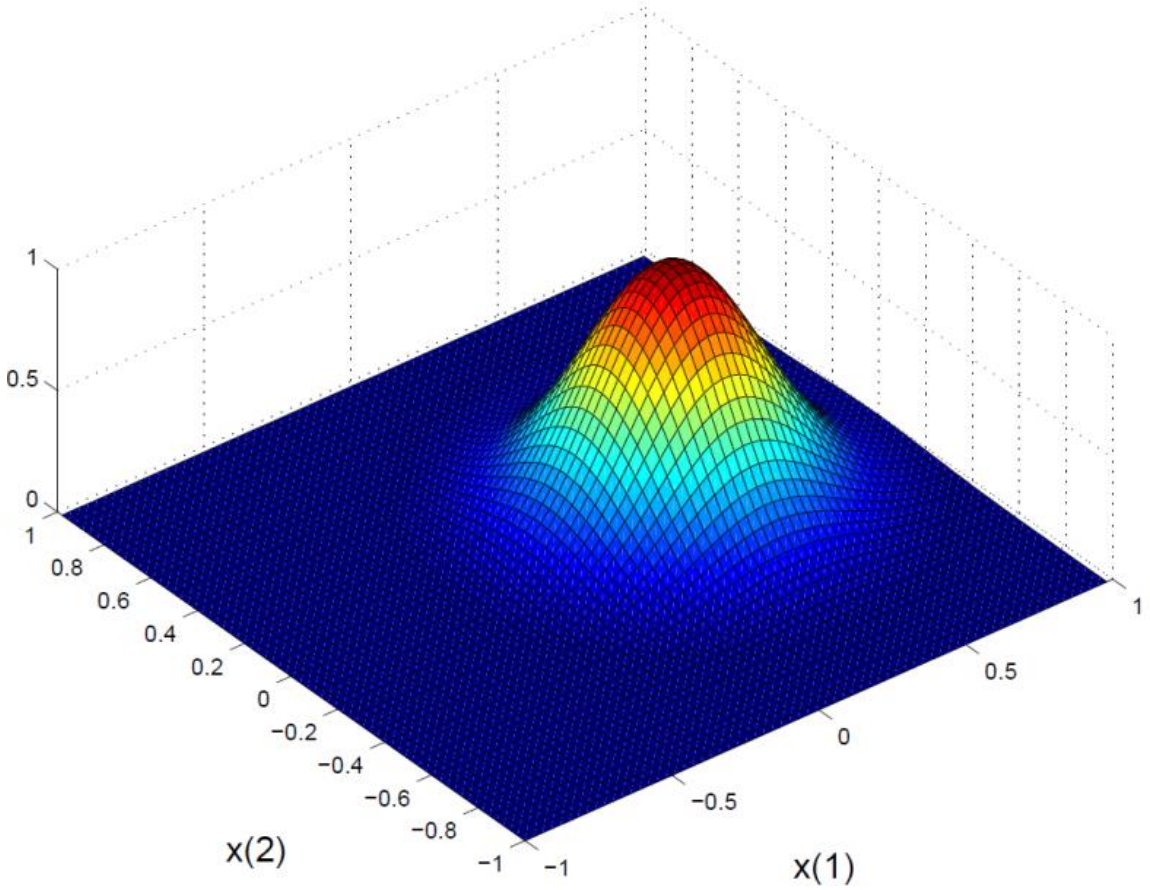


$\alpha = 0.2, m = 0$



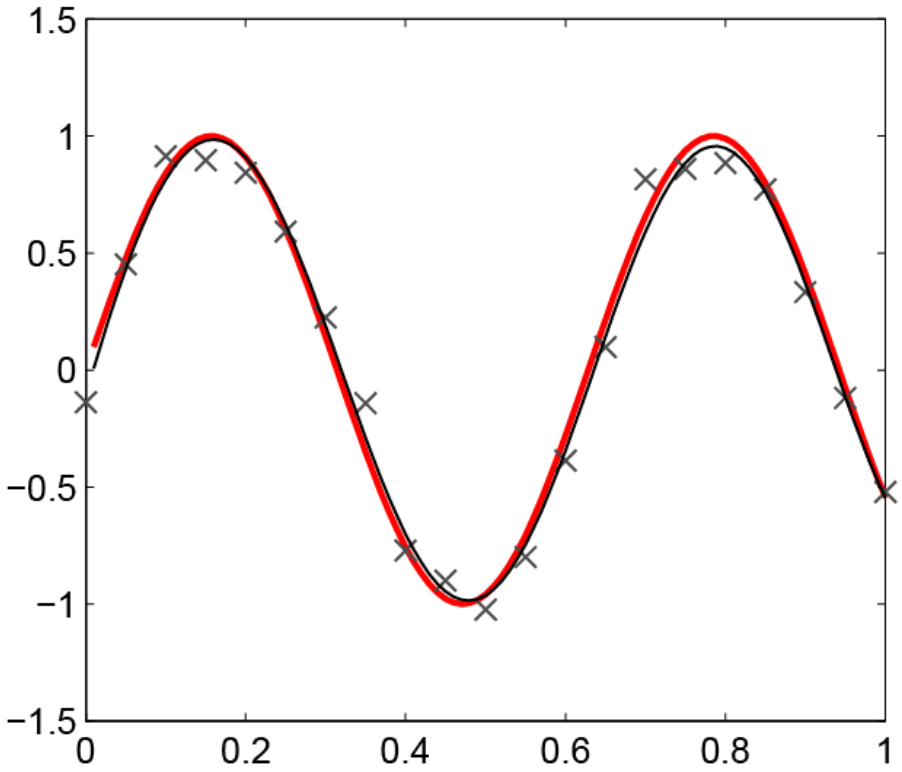
$\alpha = 0.1, m = -0.15; m = +0.15$

Gaussian radial basis functions (RBFs)



Gaussian RBF in two dimensional space.

Linear regression using RBFs



Approximation of $\sin(10x)$ by 16 Gaussian RBFs.

Black line: original function
Red line: regression

Use

$$\phi_i(\boldsymbol{x}) = \exp\left(-\frac{\left(\boldsymbol{x} - \boldsymbol{m}^{(i)}\right)^2}{2\alpha^2}\right)$$

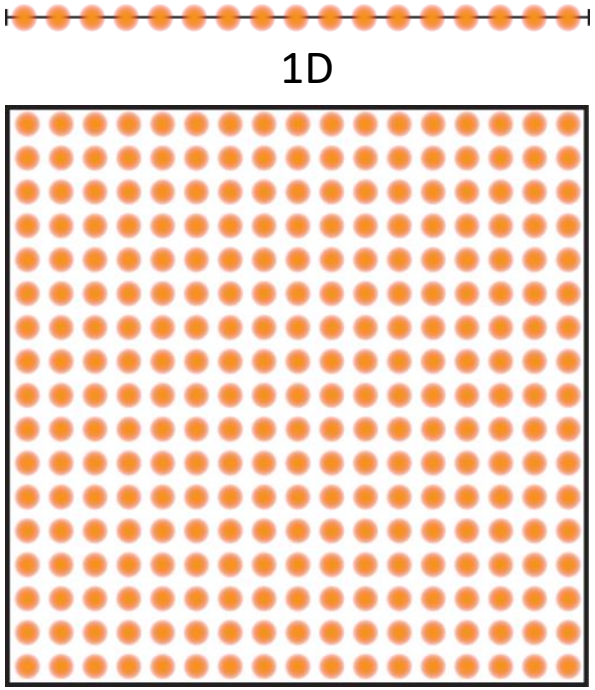
with $i = 1, \ldots, 16$ as basis functions.

Apply linear regression.

In one dimension this seems to work well, but does it scale to higher dimensions?

Curse of dimensionality

To cover the data with a constant discretization level (number of RBFs per unit volume) the number of basis functions and weights grows exponentially with the number of dimensions.



Dimensions	# of basis functions / weights
1	16
2	$16^2 = 256$
3	$16^3 = 4096$
4	$16^4 = 65\,536$
5	$16^5 = 1\,048\,576$
⋮	⋮
10	$16^{10} \approx 10^{12}$

Is there a way to reduce the number of required weights?

DUAL REPRESENTATION

Notation

$\mathcal{D}$	$\mathcal{D}$	dataset with N samples
x_n	$x^{(n)}$	n-th input sample
z_n	$y^{(n)}$	target output for n-th sample
Z	$\boldsymbol{y}$	$= \left(y^{(1)}, y^{(2)}, \dots, y^{(N)}\right)^T$
$y(x)$	$y(x)$	output computed for input x by model
$y(x_n)$	$Y^{(n)}$	output computed for n-th input sample by model
Y	$\boldsymbol{Y}$	$= \left(Y^{(1)}, Y^{(2)}, \dots, Y^{(N)}\right)^T$
W	$\boldsymbol{w}$	weights (model parameters)

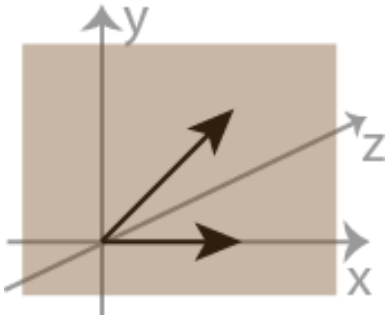
Row space and null space of a matrix

Let $A \in \mathbb{R}^{N \times M}$ be a matrix and let $\boldsymbol{A_i} = (A_{i1}, \dots, A_{iM})$ be its ith row.

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \end{pmatrix}$$

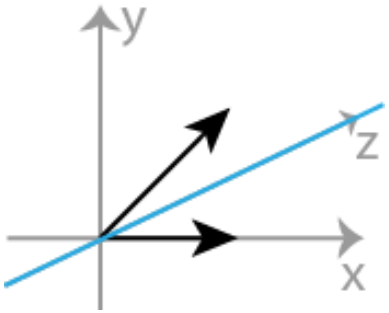
The **row space** $\mathcal{A}$ of A is the set of all possible linear combinations of its rows, $\mathcal{A} = \text{span}(\{\boldsymbol{A_1}, \dots, \boldsymbol{A_n}\})$.

$$\mathcal{A} = \{c_1(1,0,0) + c_2(1,1,0), \boldsymbol{c} \in \mathbb{R}^2\}$$



The **null space** $\ker(A)$ of A is the orthogonal complement of $\mathcal{A}$, $\ker(A) = \mathcal{A}^{\perp} = \{\boldsymbol{x} \in \mathbb{R}^M : A\boldsymbol{x} = \boldsymbol{0}\}$.

$$\ker(A) = \mathcal{A}^{\perp} = \{c(0,0,1), c \in \mathbb{R}\}$$

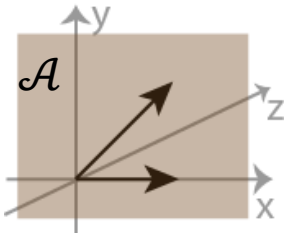


Every element $\boldsymbol{v}$ in the null space of A is orthogonal to all rows of A, $\boldsymbol{A_i} \cdot \boldsymbol{v} = \boldsymbol{0} \qquad \forall i \in \{1, \dots, N\}$.

Orthogonal Complement and Decomposition

Let $\mathcal{A}$ be a subspace of $\mathbb{R}^N$.

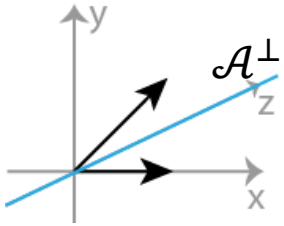
$$\mathcal{A} = \{c_1(1,0,0) + c_2(1,1,0), \boldsymbol{c} \in \mathbb{R}^2\}$$



The **orthogonal complement** $\mathcal{A}^\perp$ of $\mathcal{A}$ is the set of all vectors that are orthogonal to all elements of $\mathcal{A}$,

$$\mathcal{A}^\perp = \{\boldsymbol{z} \in \mathbb{R}^N : \boldsymbol{x}^T\boldsymbol{z} = 0 \ \ \forall \boldsymbol{x} \in \mathcal{A}\}.$$

$$\mathcal{A}^\perp = \{c(0,0,1), c \in \mathbb{R}\}$$



Orthogonal decomposition: Any $\boldsymbol{w} \in \mathbb{R}^N$ can be written uniquely in the form

$$\boldsymbol{w} = \hat{\boldsymbol{w}} + \boldsymbol{z}$$

with $\hat{\boldsymbol{w}} \in \mathcal{A}$ and $\boldsymbol{z} \in \mathcal{A}^\perp$.

$$\boldsymbol{w} = (2,3,5)$$
$$\hat{\boldsymbol{w}} = (2,3,0)$$
$$\boldsymbol{z} = (0,0,5)$$

Null space of training set

Let $\mathcal{X} = \{\boldsymbol{x}^{(n)}, n = 1, \ldots, N\}$ be the set of all training points.

Consider the predictions of the model

$$y(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x}$$

on the training set, i.e. $\boldsymbol{x} \in \mathcal{X}$.

Using orthogonal decomposition we write

$$\boldsymbol{w} = \hat{\boldsymbol{w}} + \boldsymbol{z}$$

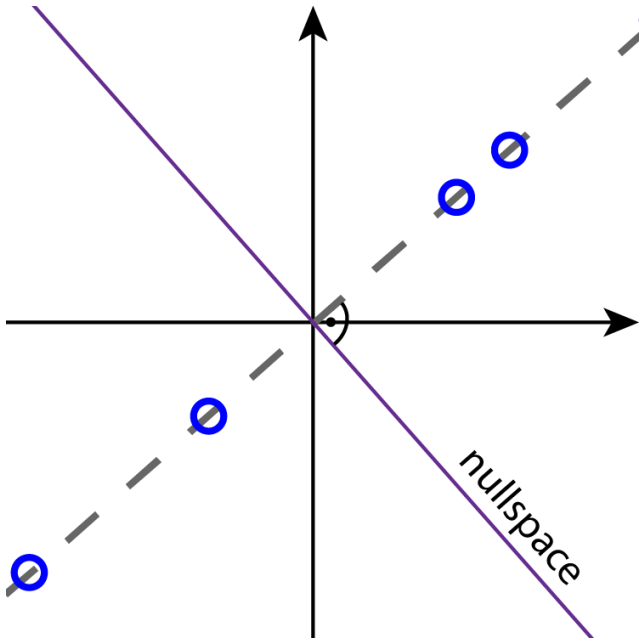
with $\hat{\boldsymbol{w}} \in \text{span}(\mathcal{X})$ and $\boldsymbol{z} \in \mathcal{X}^{\perp}$.

The prediction becomes

$$y(x) = (\hat{\boldsymbol{w}} + \boldsymbol{z})^T \boldsymbol{x} = \hat{\boldsymbol{w}}^T \boldsymbol{x} + \boldsymbol{z}^T \boldsymbol{x} = \hat{\boldsymbol{w}}^T \boldsymbol{x}.$$

Hence we can assume that

$$\boldsymbol{w} \in \text{span}(\mathcal{X}).$$



Dual representation

The weight vector $\boldsymbol{w}$ is thus a linear combination of the training samples $\mathcal{X}$,

$$\boldsymbol{w} = \sum_{n=1}^{N} a_n \, \boldsymbol{x}^{(n)} .$$

The parameters $\boldsymbol{a} = (a_1, \ldots, a_N)$ are called the *dual parameters*.

This also applies when using basis functions,

$$\boldsymbol{w} = \sum_{n=1}^{N} a_n \, \boldsymbol{\phi}\left(\boldsymbol{x}^{(n)}\right) .$$

There are **as many dual parameters as training samples**. Their number is independent of the number of basis functions.

Predictions in dual representation

$\boldsymbol{\phi}^{(m)} = \boldsymbol{\phi}(\boldsymbol{x}^{(m)})$

Substituting $\boldsymbol{w}$ back into the linear regression $y(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x})$ yields

$$y(\boldsymbol{x}) = \sum_{n=1}^{N} a_n \left(\boldsymbol{\phi}^{(n)}\right)^T \boldsymbol{\phi}(\boldsymbol{x}) = \sum_{n=1}^{N} a_n \, K\left(\boldsymbol{x}^{(n)}, \boldsymbol{x}\right).$$

with the **kernel** function of $\boldsymbol{\phi}$**,**

$$K(\boldsymbol{x}, \boldsymbol{y}) := \boldsymbol{\phi}(\boldsymbol{x})^T \boldsymbol{\phi}(\boldsymbol{y}) \,.$$

Using the **Gram matrix**

$$\boldsymbol{K}_{mn} := K\left(\boldsymbol{x}^{(m)}, \boldsymbol{x}^{(n)}\right) = \boldsymbol{\phi}\left(\boldsymbol{x}^{(m)}\right)^T \boldsymbol{\phi}\left(\boldsymbol{x}^{(n)}\right)$$

the predictions on the training set are

$$Y^{(m)} = y\left(\boldsymbol{x}^{(m)}\right) = \sum_{n=1}^{N} a_n \, K\left(\boldsymbol{x}^{(n)}, \boldsymbol{x}^{(m)}\right) = \sum_{n=1}^{N} a_n \, \boldsymbol{K}_{mn}$$

$\boldsymbol{Y} = \boldsymbol{a}^T \boldsymbol{K}$ with $\boldsymbol{Y} = \left(Y^{(1)}, Y^{(2)}, \ldots, Y^{(N)}\right)^T$.

Solution of dual representation

Primal representation:

$E(\boldsymbol{w}) = |\boldsymbol{y}^T - \boldsymbol{w}^T \boldsymbol{\Phi}|_2^2$

where $\boldsymbol{\Phi}$ is the matrix of feature vectors.

Solution:

$\boldsymbol{w} = (\boldsymbol{\Phi}\boldsymbol{\Phi}^T)^{-1}\boldsymbol{\Phi}\, \boldsymbol{y} = \boldsymbol{\Phi}^+ \mathbf{y}$

Complexity is $O(M^3)$ where M is the *number of basis functions*.

Predictions:

$y(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x})$

Dual representation:

$E(\boldsymbol{a}) = |\boldsymbol{y}^T - \boldsymbol{a}^T \boldsymbol{K}|_2^2$

where $\mathbf{K}$ is the gram matrix (also called kernel matrix).

Solution:

$\boldsymbol{a} = (\mathbf{K}\mathbf{K}^T)^{-1}\mathbf{K}\, \boldsymbol{y} = \boldsymbol{K}^+ \boldsymbol{y}$

Complexity is $O(N^3)$ where N is the *number of training samples*.

Predictions:

$$y(\boldsymbol{x}) = \sum_{n=1}^{N} a_n \, K(\boldsymbol{x}^{(n)}, \boldsymbol{x})$$

"Component a_n weighted with similarity to training sample $\boldsymbol{x}^{(n)}$."

Why is the dual representation useful?

$$y(\boldsymbol{x}) = \sum_{n=1}^{N} a_n \, K(\boldsymbol{x}^{(n)}, \boldsymbol{x})$$

N is the *number of training samples.*

Lots of basis functions and moderate number of training samples:

⇒ dual representation saves lots of parameters.

Example: Kernel of RBF basis (*not* RBF kernel)

Gaussian RBF basis functions:

$$\phi_i(\boldsymbol{x}) = \exp\left(-\frac{\left(\boldsymbol{x} - \boldsymbol{m}^{(i)}\right)^2}{2\alpha^2}\right), \qquad i = 1, \ldots, M$$

Calculate kernel:

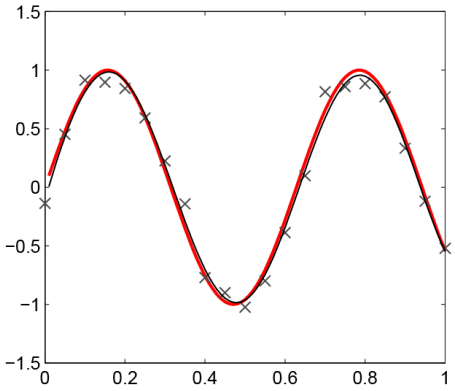
$$K(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{\phi}(\boldsymbol{x})^T \boldsymbol{\phi}(\boldsymbol{y}) = \sum_{i=1}^{M} \phi_i(\boldsymbol{x})\phi_i(\boldsymbol{y})$$

$$= \exp\left(-\frac{\boldsymbol{x}^2 + \boldsymbol{y}^2}{2\alpha^2}\right) \sum_{i=1}^{M} \exp\left(\frac{\boldsymbol{m}^{(i)}(\boldsymbol{x} + \boldsymbol{y}) - \boldsymbol{m}^{(i)^2}}{\alpha^2}\right)$$

Linear regression:

$$y(\boldsymbol{x}) = \sum_{n=1}^{N} a_n \, K\left(\boldsymbol{x}^{(n)}, \boldsymbol{x}\right)$$

where N is number of training samples.



Example: Kernel of polynomial basis

The polynomial basis of order D

$$\phi_i(x) = x^{i-1}, \qquad i = 1, \ldots, D$$

induces the kernel

$$K(x, y) = \boldsymbol{\phi}(x)^T \boldsymbol{\phi}(y) = \sum_{i=1}^{D} x^{i-1} y^{i-1} = \sum_{i=0}^{D-1} (xy)^i = \frac{1 - (xy)^D}{1 - xy}$$

and the Gram matrix

$$K_{nm} = K\left(x^{(n)}, x^{(m)}\right) = \frac{1 - \left(x^{(n)} x^{(m)}\right)^D}{1 - x^{(n)} x^{(m)}}.$$

Evaluating the kernel is much cheaper than calculating the mapping $\boldsymbol{\phi}$ into feature space and the scalar product explicitly.

PROPERTIES OF KERNELS

Notation

$\mathcal{D}$	$\mathcal{D}$	dataset with N samples
x_n	$x^{(n)}$	n-th input sample
z_n	$y^{(n)}$	target output for n-th sample
Z	$\boldsymbol{y}$	$= \left(y^{(1)}, y^{(2)}, \dots, y^{(N)}\right)^T$
$y(x)$	$y(x)$	output computed for input x by model
$y(x_n)$	$Y^{(n)}$	output computed for n-th input sample by model
Y	$\boldsymbol{Y}$	$= \left(Y^{(1)}, Y^{(2)}, \dots, Y^{(N)}\right)^T$
W	$\boldsymbol{w}$	weights (model parameters)

Properties of a kernel

$$\boldsymbol{\Phi}_{ij} = \boldsymbol{\phi}_i\left(\boldsymbol{x}^{(j)}\right)$$

Calculate $\boldsymbol{\Phi}^T\boldsymbol{\Phi}$ component-wise,

$$(\boldsymbol{\Phi}^T\boldsymbol{\Phi})_{\mathrm{mn}} = \sum_k (\boldsymbol{\Phi}^T)_{mk}\boldsymbol{\Phi}_{kn} = \sum_k \boldsymbol{\Phi}_{km}\boldsymbol{\Phi}_{kn}$$

$$= \boldsymbol{\phi}\left(\boldsymbol{x}^{(m)}\right)^T\boldsymbol{\phi}\left(\boldsymbol{x}^{(n)}\right) = K_{mn}\,.$$

Thus we have

$$\boldsymbol{K} = \boldsymbol{\Phi}^T\boldsymbol{\Phi}$$

with **K symmetric**.

$\boldsymbol{K}$ is **positive semi-definite**, that is $\boldsymbol{b}^T\boldsymbol{K}\boldsymbol{b} \geq \boldsymbol{0}$ for any $\boldsymbol{b}$, since

$$0 \leq |\boldsymbol{\Phi}\boldsymbol{b}|_2^2 = \boldsymbol{b}^T\boldsymbol{\Phi}^T\boldsymbol{\Phi}\boldsymbol{b} = \boldsymbol{b}^T\boldsymbol{K}\boldsymbol{b} \qquad \text{for any } \boldsymbol{b}.$$

When is a function a kernel?

Mercer's theorem (for finite input spaces): Consider a *finite* input space $\mathcal{X} = \{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(N)}\}$ with $K(\boldsymbol{x}^{(n)}, \boldsymbol{x}^{(m)})$ a function on $\mathcal{X}$.

Then $K(\boldsymbol{x}^{(n)}, \boldsymbol{x}^{(m)})$ is a *kernel function*, that is a scalar product in a feature space, if and only if K is symmetric and the matrix $\boldsymbol{K}_{nm} = K(\boldsymbol{x}^{(n)}, \boldsymbol{x}^{(m)})$ is positive semi-definite.

Proof.

$\boldsymbol{K}$ symmetric $\Longrightarrow$ The eigenvalue decomposition has the form $\boldsymbol{K} = \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T$, where $\boldsymbol{\Lambda}$ is a diagonal matrix containing the eigenvalues $\lambda_t = \boldsymbol{\Lambda}_{tt}$ of $\boldsymbol{K}$, and $\boldsymbol{V}$ is an orthogonal matrix containing the eigenvectors $\boldsymbol{V}_{\cdot t}$.

$\boldsymbol{K}$ is positive semi-definite $\Longrightarrow$ All eigenvalues are non-negative, $\lambda_t \geq 0$.

Define the feature map

$$\boldsymbol{\phi}_t(\boldsymbol{x}^{(n)}) = \sqrt{\lambda_t}\, \boldsymbol{V}_{nt}$$

and see that it corresponds to the kernel by calculating the scalar product

$$\boldsymbol{\phi}(\boldsymbol{x}^{(n)})^T \boldsymbol{\phi}(\boldsymbol{x}^{(m)}) = \sum_{t=1}^{N} \boldsymbol{V}_{nt}\lambda_t \boldsymbol{V}_{mt} = (\boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T)_{nm} = \boldsymbol{K}_{nm}\,.$$

■

Making kernels from kernels

Use the following rules to construct more complex kernels from simple ones.

Let K_1 and K_2 be kernels on $\mathcal{X} \subseteq \mathbb{R}^n$, then the following functions are kernels:

1. $K(\boldsymbol{x}, \boldsymbol{y}) = K_1(\boldsymbol{x}, \boldsymbol{y}) + K_2(\boldsymbol{x}, \boldsymbol{y})$,
2. $K(\boldsymbol{x}, \boldsymbol{y}) = a\, K_1(\boldsymbol{x}, \boldsymbol{y})$ for $a > 0$,
3. $K(\boldsymbol{x}, \boldsymbol{y}) = K_1(\boldsymbol{x}, \boldsymbol{y}) K_2(\boldsymbol{x}, \boldsymbol{y})$,
4. $K(\boldsymbol{x}, \boldsymbol{y}) = K_3\big(\boldsymbol{\phi}(\boldsymbol{x}), \boldsymbol{\phi}(\boldsymbol{y})\big)$ for K_3 kernel on $\mathbb{R}^m$ and $\boldsymbol{\phi}: \mathcal{X} \to \mathbb{R}^m$,
5. $K(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}^T \boldsymbol{B} \boldsymbol{y}$ for $\boldsymbol{B}$ symmetric and positive semi-definite $n \times n$ matrix.

Proofs.

Fix a finite set of points $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_l\}$ and let $\mathbf{K}, \boldsymbol{K}_1, \boldsymbol{K}_2$ be the corresponding Gram matrices of the kernel functions K, K_1, K_2.

Obviously the resulting $\boldsymbol{K}$ is symmetric for every case.

It remains to be shown that $\boldsymbol{K}$ is positive semi-definite, that is

$$\boldsymbol{z}^T \boldsymbol{K} \boldsymbol{z} \geq 0 \quad \text{for all } \boldsymbol{z}.$$

Making kernels from kernels

1. $K(\boldsymbol{x}, \boldsymbol{y}) = K_1(\boldsymbol{x}, \boldsymbol{y}) + K_2(\boldsymbol{x}, \boldsymbol{y})$

We have

$$\boldsymbol{z}^T(\boldsymbol{K}_1 + \boldsymbol{K}_2)\boldsymbol{z} = \boldsymbol{z}^T\boldsymbol{K}_1\boldsymbol{z} + \boldsymbol{z}^T\boldsymbol{K}_2\boldsymbol{z} \geq 0$$

since K_1 and K_2 are positive semi-definite.

2. $K(\boldsymbol{x}, \boldsymbol{y}) = a\, K_1(\boldsymbol{x}, \boldsymbol{y})$ for $a > 0$

Similarly, we have

$$\boldsymbol{z}^T a\boldsymbol{K}_1\boldsymbol{z} = a\, \boldsymbol{z}^T\boldsymbol{K}_1\boldsymbol{z} \geq 0\,.$$

Making kernels from kernels

3. $K(\boldsymbol{x}, \boldsymbol{y}) = K_1(\boldsymbol{x}, \boldsymbol{y})K_2(\boldsymbol{x}, \boldsymbol{y})$

We explicitly construct a feature space corresponding to $K(\boldsymbol{x}, \boldsymbol{y})$,

$$K(\boldsymbol{x}, \boldsymbol{y}) = K_1(\boldsymbol{x}, \boldsymbol{y})K_2(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{\phi_1}(\boldsymbol{x})^T \boldsymbol{\phi_1}(\boldsymbol{y})\, \boldsymbol{\phi_2}(\boldsymbol{x})^T \boldsymbol{\phi_2}(\boldsymbol{y})$$

$$= \sum_i \phi_1(\boldsymbol{x})_i\, \phi_1(\boldsymbol{y})_i \sum_j \phi_2(\boldsymbol{x})_j\, \phi_2(\boldsymbol{y})_j$$

$$= \sum_{i,j} \phi_1(\boldsymbol{x})_i \phi_2(\boldsymbol{x})_j\ \ \phi_1(\boldsymbol{y})_i \phi_2(\boldsymbol{y})_j\ = \boldsymbol{\phi}(\boldsymbol{x})^T \boldsymbol{\phi}(\boldsymbol{y})$$

with $\boldsymbol{\phi}(\boldsymbol{x}) = \boldsymbol{\phi_1}(\boldsymbol{x}) \otimes \boldsymbol{\phi_2}(\boldsymbol{x})$ (Kronecker product).

Making kernels from kernels

4. $K(\boldsymbol{x}, \boldsymbol{y}) = K_3\big(\boldsymbol{\phi}(\boldsymbol{x}), \boldsymbol{\phi}(\boldsymbol{y})\big)$ for K_3 kernel on $\mathbb{R}^m$ and $\boldsymbol{\phi}$: $\mathcal{X} \to \mathbb{R}^m$

Since K_3 is a kernel for all input values there is nothing to prove.

5. $K(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}^T \boldsymbol{B} \boldsymbol{y}$ for $\boldsymbol{B}$ symmetric and positive semi-definite $n \times n$ matrix

A positive-definite matrix has exactly one square root $\boldsymbol{B}^{1/2}$ with the property $\boldsymbol{B}^{1/2}\boldsymbol{B}^{1/2} = \boldsymbol{B}$.

Define the feature map $\boldsymbol{\phi}(\boldsymbol{x}) = \boldsymbol{B}^{1/2}\boldsymbol{x}$ and see that $\boldsymbol{\phi}(x)^T \boldsymbol{\phi}(y) = \big(\boldsymbol{B}^{1/2}\boldsymbol{x}\big)^T \boldsymbol{B}^{1/2}\boldsymbol{y} = \boldsymbol{x}^T \boldsymbol{B}^{T/2}\boldsymbol{B}^{1/2}\boldsymbol{x} = \boldsymbol{x}^T \boldsymbol{B} \boldsymbol{y} = K(\boldsymbol{x}, \boldsymbol{y})$.

■

EXAMPLE AND SUMMARY

Notation

$\mathcal{D}$	$\mathcal{D}$	dataset with N samples
x_n	$x^{(n)}$	n-th input sample
z_n	$y^{(n)}$	target output for n-th sample
Z	$\boldsymbol{y}$	$= \left(y^{(1)}, y^{(2)}, \dots, y^{(N)}\right)^T$
$y(x)$	$y(x)$	output computed for input x by model
$y(x_n)$	$Y^{(n)}$	output computed for n-th input sample by model
Y	$\boldsymbol{Y}$	$= \left(Y^{(1)}, Y^{(2)}, \dots, Y^{(N)}\right)^T$
W	$\boldsymbol{w}$	weights (model parameters)

Example: Gaussian kernel

Linear regression:

$$y(\boldsymbol{x}) = \sum_{n=1}^{N} a_n \, K\left(\boldsymbol{x}^{(n)}, \boldsymbol{x}\right)$$

„Similarity“ between data points $\boldsymbol{x}$ and $\boldsymbol{y}$ is calculated by the **Gaussian kernel**,

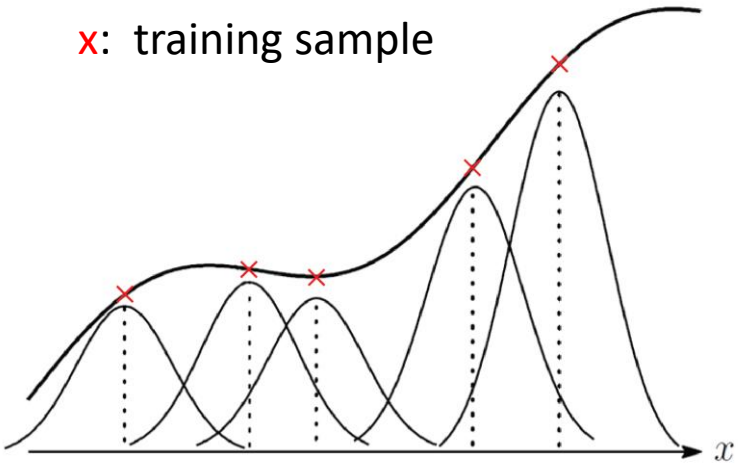
$$K(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(-\frac{|\boldsymbol{x} - \boldsymbol{y}|^2}{2\sigma^2}\right),$$

where σ is a scale parameter.

This puts a bump shaped function on every training sample,

$$y(\boldsymbol{x}) = \sum_{n=1}^{N} a_n \exp\left(-\frac{\left|\boldsymbol{x} - \boldsymbol{x}^{(n)}\right|^2}{2\sigma^2}\right).$$

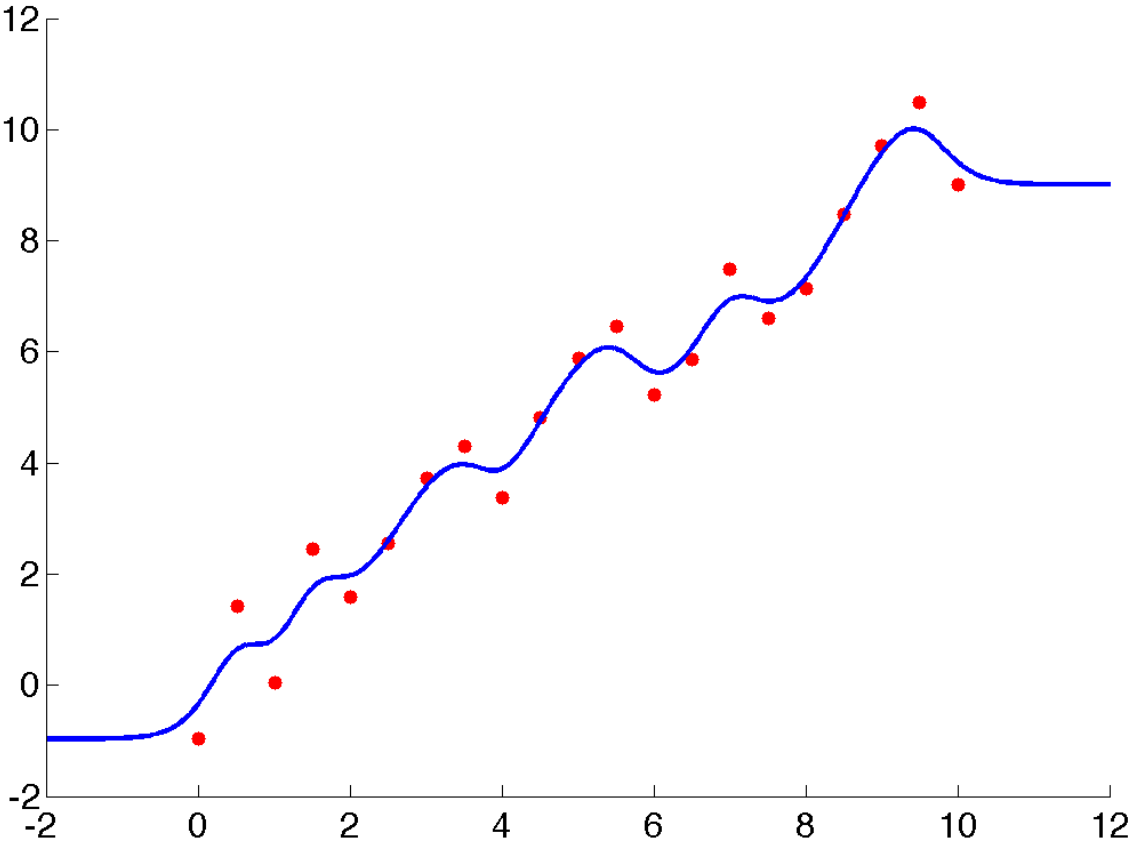
Due to the fast decline of the exponential function, data points far away do not influence the result.



Is this really a kernel?
Yes, use rules from „Making kernels from kernels“ to prove it.

Corresponding feature space?
Infinite dimensional, will be shown in an exercise.

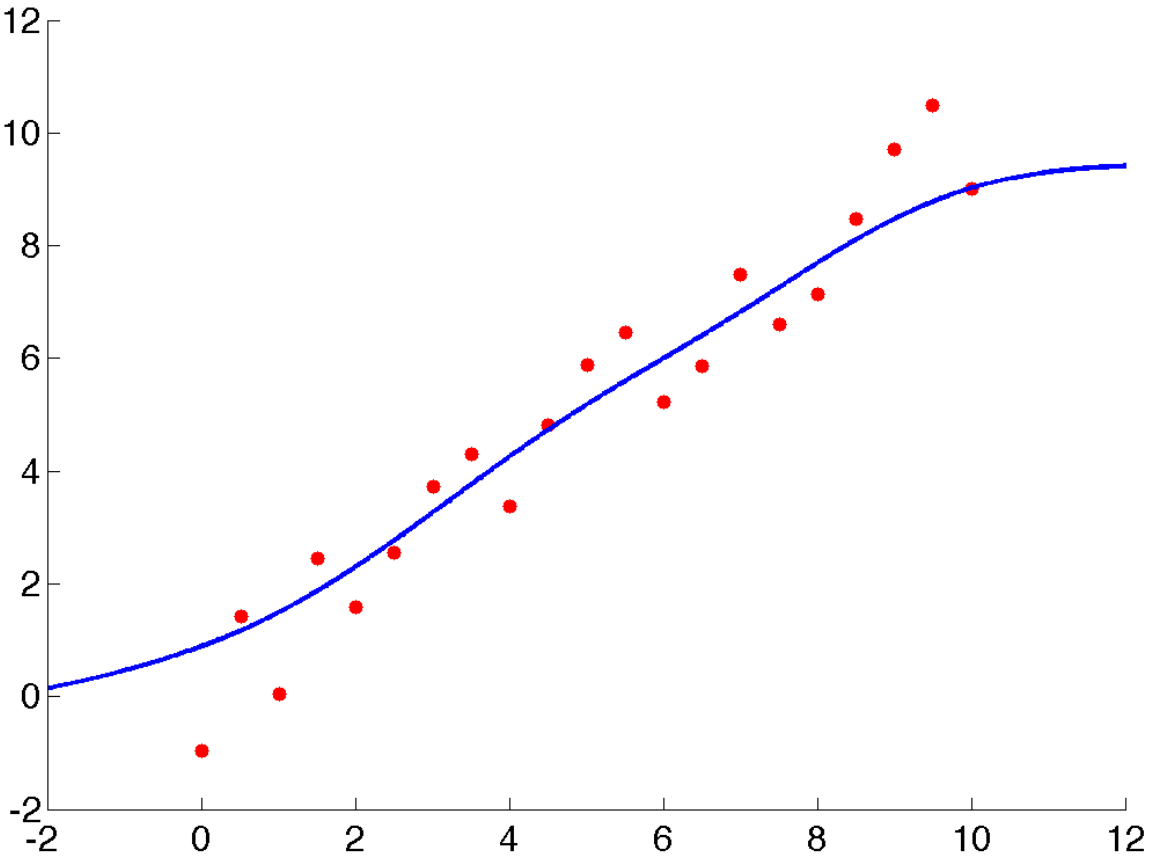
Example: Gaussian kernel



https://alliance.seas.upenn.edu/~cis520/wiki/index.php?n=Lectures.LocalLearning

Linear regression using a Gaussian kernel with $\sigma = 0.5$.
Generating function was linear.

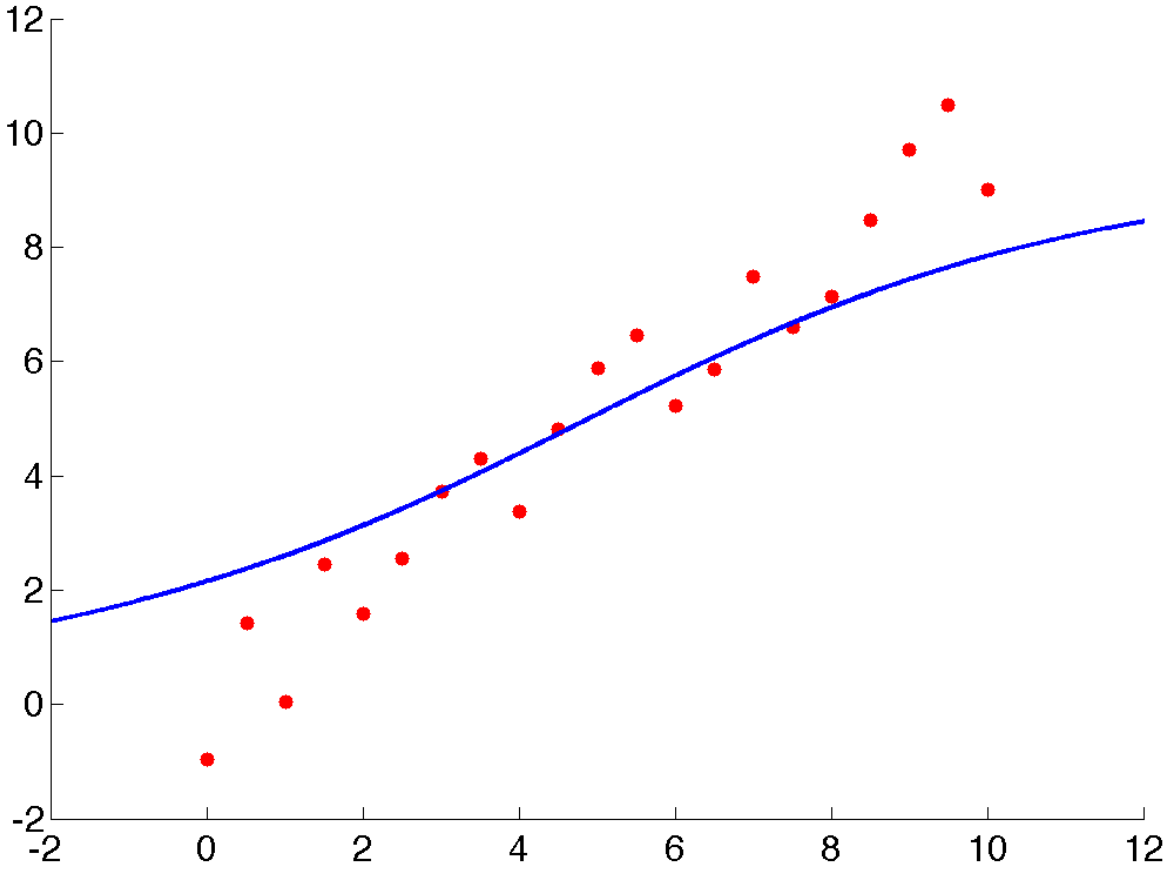
Example: Gaussian kernel



https://alliance.seas.upenn.edu/~cis520/wiki/index.php?n=Lectures.LocalLearning

Linear regression using a Gaussian kernel with $\sigma = 2.0$.
Generating function was linear.

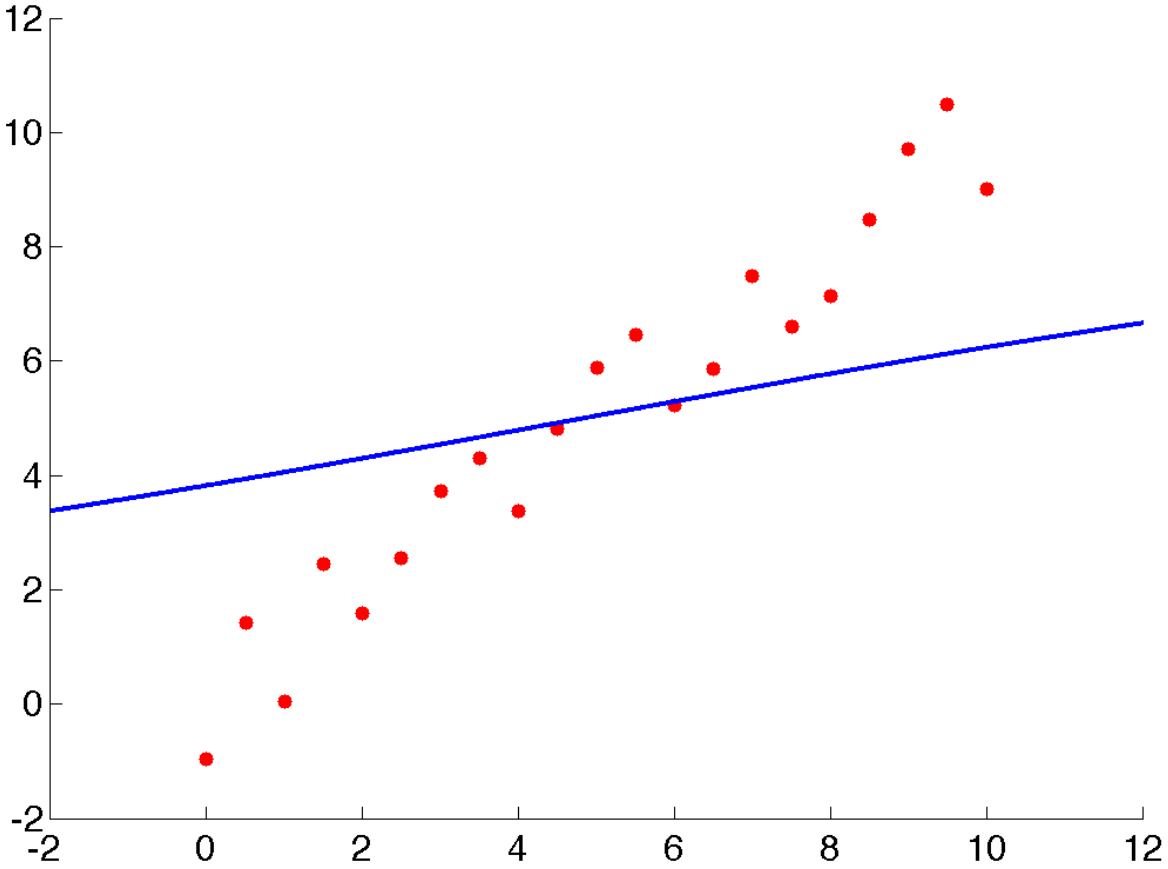
Example: Gaussian kernel



https://alliance.seas.upenn.edu/~cis520/wiki/index.php?n=Lectures.LocalLearning

Linear regression using a Gaussian kernel with $\sigma = 4.0$.
Generating function was linear.

Example: Gaussian kernel



https://alliance.seas.upenn.edu/~cis520/wiki/index.php?n=Lectures.LocalLearning

Linear regression using a Gaussian kernel with $\sigma = 8.0$.
Generating function was linear.

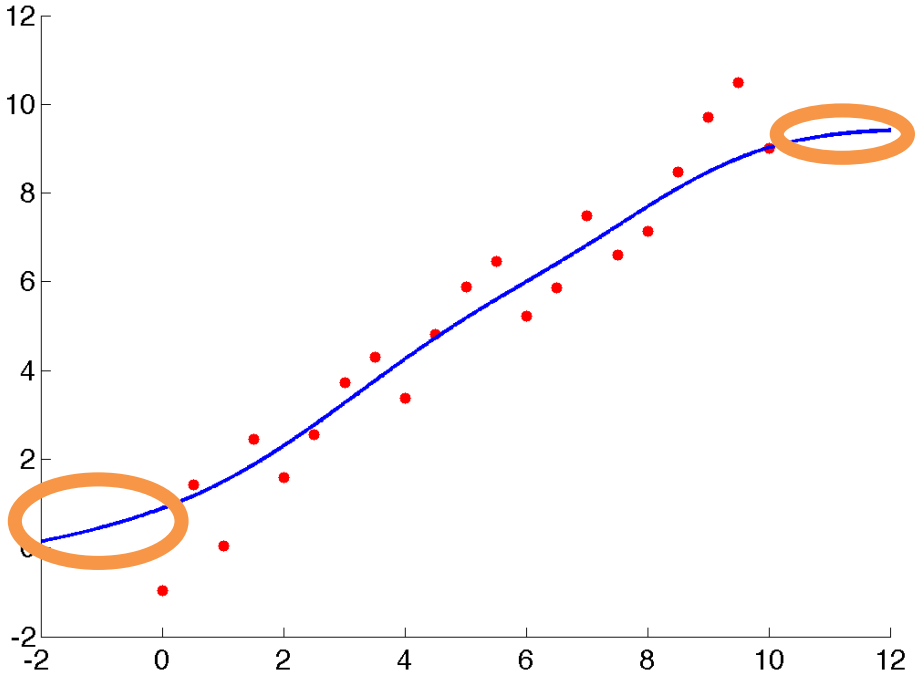
Example: Gaussian kernel

$$K(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(-\frac{|\boldsymbol{x} - \boldsymbol{y}|^2}{2\sigma^2}\right)$$

The quality of the result is very sensitive to the choice of the variance σ.

Use cross-validation to choose the right value.

Combining model-based and model-free approaches



Problem: Gaussian kernel cannot extrapolate, since

$$\sum_n a_n \exp\left(-\frac{|\boldsymbol{x}-\boldsymbol{x}^{(n)}|^2}{2\sigma^2}\right) \to 0$$

further away from the training set.

If there is a feature space $\boldsymbol{\phi}(\boldsymbol{x})$ that partly explains the data, we can add the kernel induced by that feature space, $\boldsymbol{\phi}(\boldsymbol{x})^T\boldsymbol{\phi}(\boldsymbol{y})$, to a generic kernel $K_2(\boldsymbol{x},\boldsymbol{y})$ to obtain a better fit.

$$K(\boldsymbol{x},\boldsymbol{y}) = \boldsymbol{\phi}(\boldsymbol{x})^T\boldsymbol{\phi}(\boldsymbol{y}) + K_2(\boldsymbol{x},\boldsymbol{y})$$

Here, we could combine a linear model, $\boldsymbol{\phi}(\boldsymbol{x}) = \boldsymbol{x}$, with the Gaussian kernel $K_2(\boldsymbol{x},\boldsymbol{y})$.

Linear model: computes general trend of data.

Gaussian kernel: captures local variations.

Commonly used kernels

- Linear: $K(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}^T \boldsymbol{y}$
- Polynomial: $K(\boldsymbol{x}, \boldsymbol{y}) = \left((\boldsymbol{x}^T \boldsymbol{y}) + c\right)^d$
- Gaussian: $K(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(-\frac{|\boldsymbol{x}-\boldsymbol{y}|^2}{2\sigma^2}\right)$
- Sometimes functions are used as kernels that are not positive semi-definite. Usually this works but it can lead to strange results.

Applications of kernels

- Regression
- Gaussian Processes (GPs)
- Classification (especially Support Vector Machines)
- Principal Component Analysis (PCA)
- *Every algorithm with a scalar product*.

- Kernels can also be defined on domains other than $\mathbb{R}^N$, for example strings. This allows the above methods to be applied on new domains.

Summary

- Kernels compute the value of the scalar product in a high dimensional feature space without explicitly computing the features.
- They can be used in any model that depends (or that can be rewritten so that it depends) on the scalar product of the training samples.
- Not every function is a kernel.
- Use kernel construction rules to prove that a function is a valid kernel.

References

C. Bishop
Pattern Recognition and Machine Learning
Springer-Verlag 2006

N. Cristianini, J. Shawe-Taylor
An Introduction to Support Vector Machines and other kernel-based learning methods
Cambridge University Press 2000