

# \*\*\*\*\* Advanced Machine Learning \*\*\*\*\*

## Advanced Regression Techniques

M. Sc. Daniel Wehner

SAP SE / DHBW Mannheim

Summer term 2020

# Agenda for this Unit

<b>Bayesian Regression</b>	<b>4</b>
Introduction	4
Maximum Likelihood Regression	5
Maximum A posteriori (MAP) Regression	8
Full Bayesian Regression	12
 <b>Kernel Ridge Regression</b>	 <b>19</b>
Introduction	19
Woodbury Matrix Identity	20

<b>Example</b>	<b>23</b>
<b>Gaussian Process Regression</b>	<b>24</b>
Introduction	24
Learning a Gaussian Process Model	26
Example	29
Learning the Hyper-Parameters	34
<b>Support Vector Regression</b>	<b>35</b>
Introduction	35
Optimization	39
Karush-Kuhn-Tucker Conditions	43
Example	45

# Bayesian Regression

## Introduction

- You already know what linear regression is and how you can solve it:

$$\theta = (\Phi^T \Phi)^{-1} \Phi^T y \quad (1)$$

- It is possible to treat regression in a more probabilistic fashion.
- With this probabilistic perspective we can see where regularization comes from and we can derive the least squares error function.
- **Bayes theorem** will play an important role (you should keep it in mind):

$$p(A|B) = \frac{p(B|A) \cdot p(A)}{p(B)} \quad (2)$$

- It gives rise to what it referred to as **Bayesian learning**.

## Maximum Likelihood Regression

- In probabilistic regression we make two general assumptions:

1. The data is noisy. Therefore, we add an additive noise term  $\varepsilon$  to the function estimates:

$$y = h(\mathbf{x}; \boldsymbol{\theta}) + \varepsilon \quad (3)$$

2. The noise term  $\varepsilon$  is considered a Gaussian random variable with zero mean:

$$\varepsilon \sim \mathcal{N}(0, \beta^{-1}) \quad (4)$$

- **With these assumptions  $y$  is now a random variable.** It has the following (Gaussian) probability distribution:

$$p(y|\mathbf{x}, \boldsymbol{\theta}, \beta) = \mathcal{N}(y|h(\mathbf{x}; \boldsymbol{\theta}), \beta^{-1}) \quad (5)$$

- We are given a labeled data set  $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^n$ .
- Assuming the data is i. i. d. (independent and identically distributed), the conditional likelihood is computed as follows:

$$p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}, \beta) = \prod_{i=1}^n \mathcal{N}(y^{(i)} | h(\mathbf{x}^{(i)}; \boldsymbol{\theta}), \beta^{-1}) \quad (6)$$

$$= \prod_{i=1}^n \mathcal{N}(y^{(i)} | \boldsymbol{\theta}^\top \mathbf{x}^{(i)}, \beta^{-1}) \quad (7)$$

- Compute the log-likelihood:

$$\log p(\mathbf{y}|\mathbf{X}, \boldsymbol{\theta}, \beta) = \sum_{i=1}^n \log \mathcal{N}(y^{(i)} | \boldsymbol{\theta}^\top \mathbf{x}^{(i)}, \beta^{-1}) \quad (8)$$

$$= \sum_{i=1}^n \left[ \log \left( \frac{\sqrt{\beta}}{\sqrt{2\pi}} \right) - \frac{\beta}{2} (y^{(i)} - \boldsymbol{\theta}^\top \mathbf{x}^{(i)})^2 \right] \quad (9)$$

$$= \frac{n}{2} \log \beta - \frac{n}{2} \log 2\pi - \frac{\beta}{2} \sum_{i=1}^n (y^{(i)} - \boldsymbol{\theta}^\top \mathbf{x}^{(i)})^2 \quad (10)$$

**Computation of the gradient**

$$\nabla_{\theta} \log p(\mathbf{y} | \mathbf{X}, \theta, \beta) \stackrel{!}{=} \mathbf{0} \quad (11)$$

$$-\beta \sum_{i=1}^n (y^{(i)} - \theta^{\top} \mathbf{x}^{(i)}) \mathbf{x}^{(i)} \stackrel{!}{=} \mathbf{0} \quad (12)$$

- Solving for  $\theta$  gives the normal equation:  $\theta_{ml} = (\mathbf{X}^{\top} \mathbf{X})^{-1} \mathbf{X}^{\top} \mathbf{y}$ .
- This is the same result as in least squares regression.
- Additionally, we can get a global estimate of the uncertainty:  $\beta_{ml} = \left( \frac{1}{n} \sum_{i=1}^n (y^{(i)} - \theta^{\top} \mathbf{x}^{(i)})^2 \right)^{-1}$



**Important: Minimizing the squared error gives the maximum likelihood solution for the parameters  $\theta$  assuming Gaussian noise.**

## Maximum A posteriori (MAP) Regression

- The problem with maximum likelihood regression is that it might lead to **overfitting**.
- *What can we do to tackle this kind of problem?*
- We can use a more Bayesian approach and put a **prior** on the parameters  $\theta$ :

$$p(\theta|X, y, \alpha, \beta) \propto p(y|X, \theta, \beta) \cdot p(\theta|\alpha) \quad (13)$$

- The prior probability distribution  $p(\theta|\alpha)$  encodes our **prior belief** about the parameters  $\theta$ .



**Please note the very important difference: In this setting you do not get a single parameter vector  $\theta$ , rather a probability distribution over the parameters given the data  $p(\theta|X, y, \alpha, \beta)$ !**



### The prior for the parameters

- We decided to put a prior on the parameters  $\theta$ .
- One obvious choice is to use a Gaussian distribution for the prior (with zero mean and spherical covariance):

$$\theta \sim p(\theta|\alpha) = \mathcal{N}(\theta|\mathbf{0}, \alpha^{-1}\mathbf{I}) \quad (14)$$

- The posterior then becomes:

$$\begin{aligned} p(\theta|\mathbf{X}, \mathbf{y}, \alpha, \beta) &\propto p(\mathbf{y}|\mathbf{X}, \theta, \beta) \cdot p(\theta|\alpha) \\ &\propto p(\mathbf{y}|\mathbf{X}, \theta, \beta) \cdot \mathcal{N}(\theta|\mathbf{0}, \alpha^{-1}\mathbf{I}) \end{aligned} \quad (15)$$

- Compute the log-likelihood:

$$\log p(\boldsymbol{\theta} | \mathbf{X}, \mathbf{y}, \alpha, \beta) = \log p(\mathbf{y} | \mathbf{X}, \boldsymbol{\theta}, \beta) + \log \mathcal{N}(\boldsymbol{\theta} | \mathbf{0}, \alpha^{-1} \mathbf{I}) + \text{const} \quad (16)$$

$$= \sum_{i=1}^n \log \mathcal{N}(y^{(i)} | \boldsymbol{\theta}^\top \boldsymbol{\varphi}(\mathbf{x}^{(i)}), \beta^{-1}) + \log \mathcal{N}(\boldsymbol{\theta} | \mathbf{0}, \alpha^{-1} \mathbf{I}) + \text{const} \quad (17)$$

$$= -\frac{\beta}{2} \sum_{i=1}^n (y^{(i)} - \boldsymbol{\theta}^\top \boldsymbol{\varphi}(\mathbf{x}^{(i)}))^2 - \frac{\alpha}{2} \boldsymbol{\theta}^\top \boldsymbol{\theta} + \text{const} \quad (18)$$

- Computation of the gradient:

$$\nabla_{\boldsymbol{\theta}} \log p(\boldsymbol{\theta} | \mathbf{X}, \mathbf{y}, \alpha, \beta) = \beta \sum_{i=1}^n (y^{(i)} - \boldsymbol{\theta}^\top \boldsymbol{\varphi}(\mathbf{x}^{(i)})) \boldsymbol{\varphi}(\mathbf{x}^{(i)}) - \alpha \boldsymbol{\theta} \stackrel{!}{=} \mathbf{0} \quad (19)$$

$$\beta \sum_{i=1}^n y^{(i)} \boldsymbol{\varphi}(\mathbf{x}^{(i)}) = \beta \left[ \sum_{i=1}^n \boldsymbol{\varphi}(\mathbf{x}^{(i)})^\top \boldsymbol{\varphi}(\mathbf{x}^{(i)}) \right] \boldsymbol{\theta} + \alpha \boldsymbol{\theta} \quad (20)$$

$$\beta \sum_{i=1}^n y^{(i)} \boldsymbol{\varphi}(\mathbf{x}^{(i)}) = \left[ \beta \sum_{i=1}^n \boldsymbol{\varphi}(\mathbf{x}^{(i)})^\top \boldsymbol{\varphi}(\mathbf{x}^{(i)}) + \alpha \right] \boldsymbol{\theta} \quad (21)$$

$$\beta \boldsymbol{\Phi}^\top \mathbf{y} = (\beta \boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \alpha \mathbf{I}) \boldsymbol{\theta} \quad \Rightarrow \boldsymbol{\theta}_{\text{map}} = \left( \boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \frac{\alpha}{\beta} \mathbf{I} \right)^{-1} \boldsymbol{\Phi}^\top \mathbf{y} \quad (22)$$

- The prior **regularizes** the parameters  $\theta$ .
- This approach is referred to as **ridge regression**.
- You already know this result from regularized least squares regression:

$$\arg \min_{\theta} \frac{1}{2} \|\Phi\theta - y\|^2 + \frac{\lambda}{2} \|\theta\|^2 \quad (23)$$

- Solving for  $\theta$ , we get the estimate:

$$\theta = (\Phi^T \Phi + \lambda I)^{-1} \Phi^T y \quad (24)$$

- Here:  $\lambda = \frac{\alpha}{\beta}$



**You assume two things when you put a regularizer  $\lambda$  in least-squares regression:**

- ❶ The targets are noisy, where the noise is distributed according to a Gaussian distribution.
- ❷ The parameters are Gaussian distributed as well.

## Full Bayesian Regression

- Again, we put a prior on the parameters  $\theta$ .

$$p(\theta|\alpha) = \mathcal{N}(\theta|\mu_0, \Lambda_0) \quad (25)$$

- In  $\Rightarrow$  eq. (25), the mean  $\mu_0$  and the precision matrix  $\Lambda_0$  are given by  $\mathbf{0}$  and  $\alpha^{-1}I$ , respectively. Therefore, the prior is a zero-mean, isotropic ( $\hat{=}$  rotation-invariant) Gaussian distribution.
- The posterior distribution of the parameters  $p(\theta|X, y, \alpha, \beta)$  is then:

$$p(\theta|X, y, \alpha, \beta) = \mathcal{N}(\theta|\mu_n, \Lambda_n) \quad (26)$$

with:

$$\mu_n = \beta \Lambda_n^{-1} \Phi^T y \quad (27)$$

$$\Lambda_n = \Lambda_0^{-1} + \beta \Phi^T \Phi \quad (28)$$

- This can be phrased as a sequential update rule. The prior must be **conjugate** in order for this to work.

- In Bayesian probability theory, if the posterior distribution is in the **same probability distribution family** as the prior probability distribution, the prior and posterior are then called **conjugate distributions**, and the prior is called **conjugate prior**.
- Here: If we multiply two Gaussian distributions, the result is again Gaussian.

### Example for Bayesian regression

- We can illustrate Bayesian learning in a linear basis function model.
- Consider a single input variable  $x$  (scalar) and a linear model of the form  $h(x; \theta) = \theta_0 + \theta_1 \cdot x$ .
- Because this model only has two adaptive parameters, we can plot the prior and the posterior distributions directly in parameter space.
- We generate synthetic data from the function  $f(x; \mathbf{a}) = a_0 + a_1 \cdot x$ , with  $a_0 = -0.3$  and  $a_1 = 0.5$  by first choosing values of  $x^{(i)}$  from the uniform distribution  $\mathcal{U}(x | -1, 1)$ , then evaluating  $f(x^{(i)}; \mathbf{a})$ , and finally adding some Gaussian noise with precision  $\beta = \frac{1}{0.2}$ .  $\alpha$  is fixed to 2.0.

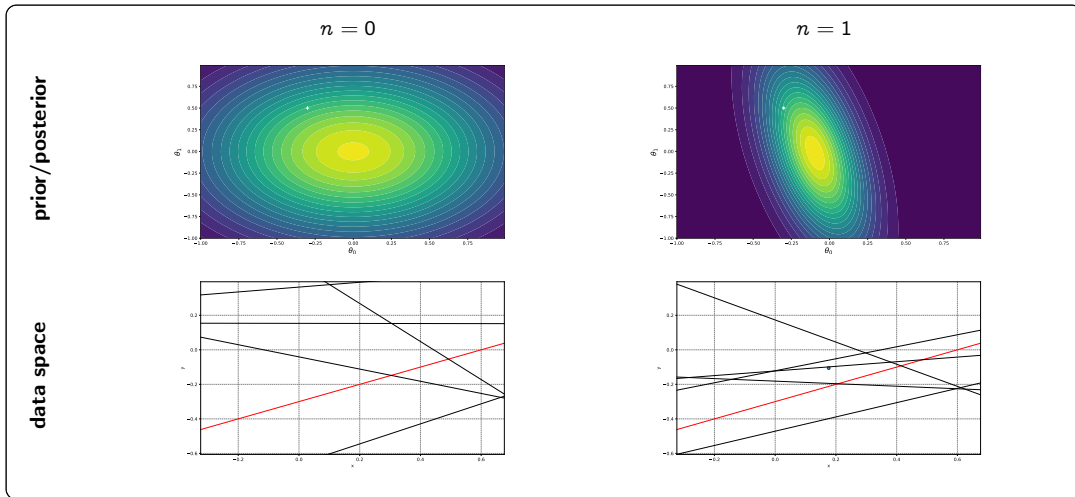


Figure 1:

Example for Bayesian regression (part I)

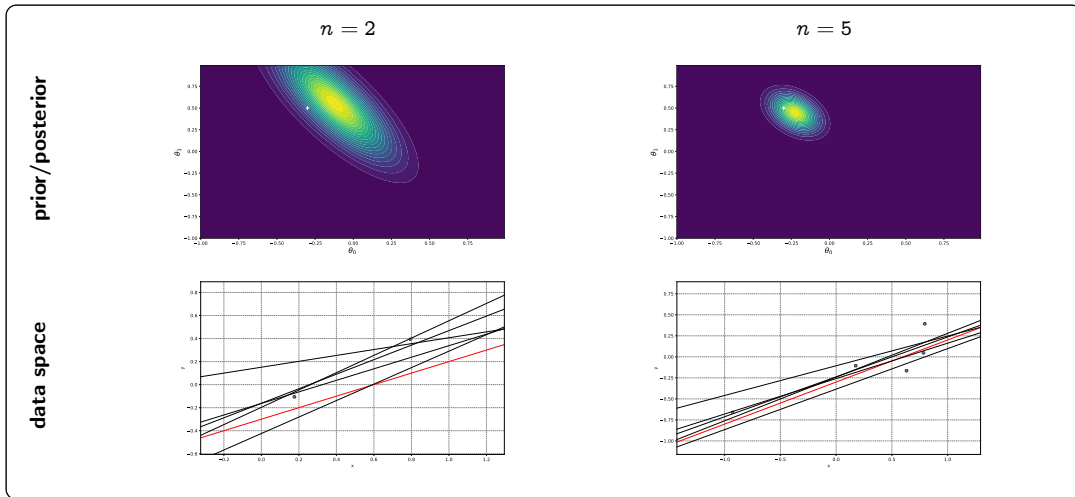


Figure 2:

Example for Bayesian regression (part II)

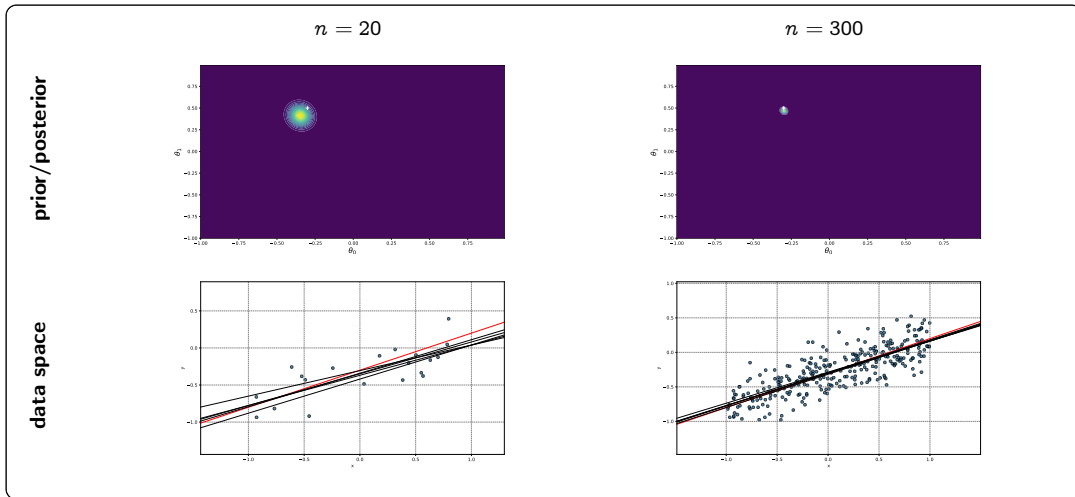


Figure 3:

Example for Bayesian regression (part III)



## Predictive distribution

- Usually, we are not interested in  $\theta$  itself, but rather in making a prediction  $y_q$  for a new instance  $x_q$ .
- This requires that we evaluate the **predictive distribution**:

$$p(y_q | x_q, \mathbf{X}, \mathbf{y}, \alpha, \beta) = \int_{\theta} \underbrace{p(y_q | x_q, \mathbf{X}, \theta, \beta)}_{\text{regression model}} \cdot \underbrace{p(\theta | \mathbf{X}, \mathbf{y}, \alpha, \beta)}_{\text{parameter posterior}} d\theta \quad (29)$$

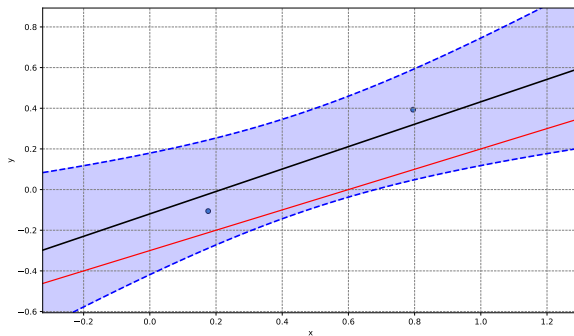
- We integrate over all possible models and give each model a weight corresponding to how probable it is.
- Think of it as a weighted average.
- The predictive distribution takes the form:

$$p(y_q | x_q, \mathbf{X}, \mathbf{y}, \alpha, \beta) = \mathcal{N}(y_q | \boldsymbol{\mu}_n^T \boldsymbol{\varphi}(x_q), \sigma_n^2(x_q)) \quad (30)$$

with:

$$\sigma_n^2(x_q) = \frac{1}{\beta} + \boldsymbol{\varphi}(x_q)^T \boldsymbol{\Lambda}_n^{-1} \boldsymbol{\varphi}(x_q) \quad (31)$$

- The first term in  $\Rightarrow$  eq. (31) reflects the noise in the data, the second one the uncertainty associated with the model parameters  $\theta$ .

**Figure 4:**

Bayesian regression and uncertainty estimate

# Kernel Ridge Regression

## Introduction

- In ridge regression, the optimal parameters  $\theta$  can be found using the **normal equation**:

$$\theta = (\Phi^T \Phi + \lambda I)^{-1} \Phi^T y \quad (32)$$

- In the above formula,  $\Phi$  denotes the design matrix (regressor matrix),  $y$  is the label vector and  $\lambda$  is the regularization parameter.
- In order to apply kernels, we have to rephrase this equation in terms of dot products of the input features. Replacing these dot products by kernels avoids operating in feature space.
- This can be achieved by using the **Woodbury matrix identity**.

## Woodbury Matrix Identity

- For the prediction  $y_q$  of a new query data point  $x_q$ , we have to calculate:

$$y_q = \varphi(x_q)^\top \theta \quad (33)$$

Step ①: Insert normal equation  $\Rightarrow$  eq. (32):

$$= \varphi(x_q)^\top (\Phi^\top \Phi + \lambda I)^{-1} \Phi^\top y \quad (34)$$

Step ②: Apply Woodbury matrix identity:

$$= \varphi(x_q)^\top \Phi^\top (\Phi \Phi^\top + \lambda I)^{-1} y \quad (35)$$

- The formula given in  $\Rightarrow$  eq. (35) exclusively uses dot products of input features and is therefore susceptible to kernels.

- Replace the dot products by kernel functions:

Rewrite of  $\varphi(\mathbf{x}_q)^\top \Phi^\top$ :

$$\varphi(\mathbf{x}_q)^\top \Phi^\top = \varphi(\mathbf{x}_q)^\top \begin{bmatrix} \varphi(\mathbf{x}^{(1)})^\top \\ \vdots \\ \varphi(\mathbf{x}^{(n)})^\top \end{bmatrix}^\top = \begin{bmatrix} \mathcal{K}(\mathbf{x}_q, \mathbf{x}^{(1)}) \\ \vdots \\ \mathcal{K}(\mathbf{x}_q, \mathbf{x}^{(n)}) \end{bmatrix} = \mathbf{K}_*(\mathbf{x}_q) \quad (36)$$

Rewrite of  $\Phi \Phi^\top$ :

$$\Phi \Phi^\top = \begin{bmatrix} \varphi(\mathbf{x}^{(1)})^\top \\ \vdots \\ \varphi(\mathbf{x}^{(n)})^\top \end{bmatrix} \begin{bmatrix} \varphi(\mathbf{x}^{(1)})^\top \\ \vdots \\ \varphi(\mathbf{x}^{(n)})^\top \end{bmatrix}^\top = \begin{bmatrix} \mathcal{K}(\mathbf{x}^{(1)}, \mathbf{x}^{(1)}) & \dots & \mathcal{K}(\mathbf{x}^{(n)}, \mathbf{x}^{(1)}) \\ \vdots & \ddots & \vdots \\ \mathcal{K}(\mathbf{x}^{(1)}, \mathbf{x}^{(n)}) & \dots & \mathcal{K}(\mathbf{x}^{(n)}, \mathbf{x}^{(n)}) \end{bmatrix} = \mathbf{K} \quad (37)$$

- The kernel matrices  $\mathbf{K}$  and  $\mathbf{K}_*$  must fulfill **Mercer's condition** and therefore have to be **positive-semi definite (psd)**. Famous choices: Polynomial kernel or radial basis function (RBF) kernel.

- The final kernel ridge regression formula is given by:

$$y_q = K_*(x_q)(K + \lambda I)^{-1}y \quad (38)$$

- Like all kernel methods, it is a **non-parametric** approach.



**Kernel methods do not work well for very large data sets ( $> 10,000$  data points), since we have to calculate all pairwise similarities!**

## Example

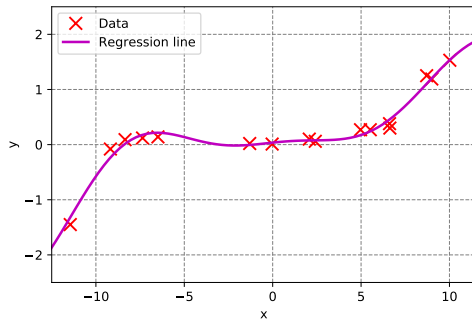


Figure 5:

Result of kernel ridge regression

# Gaussian Process Regression

## Introduction

- Similarly to kernel ridge regression, Gaussian processes do not make any assumptions about the type of regression function (e. g. linear, quadratic, ...)
- It is non-parametric and a form of supervised learning:

$$h(\mathbf{x}) = \mathcal{GP}(m(\mathbf{x}), \mathcal{K}(\mathbf{x}, \mathbf{x}')) \quad (39)$$

- In  $\Rightarrow$  eq. (39),  $m(\mathbf{x})$  denotes the mean function, whereas  $\mathcal{K}(\mathbf{x}, \mathbf{x}')$  denotes the kernel function, which – in the context of Gaussian processes – is referred to as the covariance function.
- Definition of a Gaussian process:  
*Formally, a Gaussian process is a collection of random variables, any finite number of which has a **joint Gaussian distribution**.*



- Instead of modeling a distribution over parameters (cf. Bayesian regression), we model a **distribution over possible regression functions**.
- Thus, Gaussian processes extend multivariate Gaussian distributions to **infinite dimensions**.
  - E. g. a function  $f : \mathbb{R} \mapsto \mathbb{R}$  can be thought of as a sample from some infinite Gaussian distribution.
  - Pick the function which maximizes the posterior distribution over functions.
- The mean of the prior  $m(\mathbf{x})$  distribution is usually set to 0 everywhere.
- In practice, the squared exponential function ( $\hat{=}$  RBF-kernel) is frequently used:

$$\mathcal{K}(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \cdot \exp \left\{ \frac{-\|\mathbf{x} - \mathbf{x}'\|^2}{2 \cdot l^2} \right\} \quad (40)$$

- Hyper-Parameters:
  - $\sigma_f^2$  denotes the maximum allowable covariance. It should be high for functions covering a broad range of the  $y$ -axis. If  $\mathbf{x} \approx \mathbf{x}'$ ,  $\mathcal{K}(\mathbf{x}, \mathbf{x}')$  approaches this maximum.
  - $l$  (landmark) controls how much the data points influence each other.

## Learning a Gaussian Process Model

- We are given a training data set  $\mathcal{D}$  comprising  $n$  observations:

$$\mathcal{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \dots, (\mathbf{x}^{(n)}, y^{(n)})\} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^n$$

- Also, we have a query data point  $\mathbf{x}_q$ , for which  $y_q$  has to be predicted.
- To do so, we compute the covariance between all example pairs.
- This results in three matrices  $\mathbf{K}$  (matrix),  $\mathbf{K}_*$  (vector) and  $\mathbf{K}_{**}$  (scalar).

The matrices have the following form:

$$K = \begin{bmatrix} \mathcal{K}(\mathbf{x}^{(1)}, \mathbf{x}^{(1)}) & \mathcal{K}(\mathbf{x}^{(2)}, \mathbf{x}^{(1)}) & \dots & \mathcal{K}(\mathbf{x}^{(n)}, \mathbf{x}^{(1)}) \\ \mathcal{K}(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}) & \mathcal{K}(\mathbf{x}^{(2)}, \mathbf{x}^{(2)}) & \dots & \mathcal{K}(\mathbf{x}^{(n)}, \mathbf{x}^{(2)}) \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{K}(\mathbf{x}^{(1)}, \mathbf{x}^{(n)}) & \mathcal{K}(\mathbf{x}^{(2)}, \mathbf{x}^{(n)}) & \dots & \mathcal{K}(\mathbf{x}^{(n)}, \mathbf{x}^{(n)}) \end{bmatrix} \quad (41)$$

$$K_* = [\mathcal{K}(\mathbf{x}_q, \mathbf{x}^{(1)}) \quad \mathcal{K}(\mathbf{x}_q, \mathbf{x}^{(2)}) \quad \dots \quad \mathcal{K}(\mathbf{x}_q, \mathbf{x}^{(n)})]^\top \quad (42)$$

$$K_{**} = \mathcal{K}(\mathbf{x}_q, \mathbf{x}_q) \quad (43)$$



**$K$  is a matrix (contains the similarities of training data pairs),  $K_*$  is a vector (contains similarities of the query data point with the training data), while  $K_{**}$  is actually a scalar (comparison of data point  $\mathbf{x}_q$  to itself)!**

- Since we assume that the data can be modeled as a sample from a multivariate Gaussian distribution, we can model the Gaussian process prior as follows:

$$\begin{bmatrix} \mathbf{y} \\ y_q \end{bmatrix} \sim \mathcal{N} \left( \mathbf{0}, \begin{bmatrix} \mathbf{K} & \mathbf{K}_*^\top \\ \mathbf{K}_* & K_{**} \end{bmatrix} \right) \quad (44)$$

- What we actually want is the **posterior distribution**  $p(y_q | \mathbf{y})$ : 'Given the data, what is  $y_q$ ?'
  - For Gaussian distributions, the posterior distribution can be computed analytically:

$$y_q | \mathbf{y} \sim \mathcal{N} \left( \underbrace{\mathbf{K}_* \mathbf{K}^{-1} \mathbf{y}}_{\text{Matrix of regr. coeff.}}, \underbrace{K_{**} - \mathbf{K}_* \mathbf{K}^{-1} \mathbf{K}_*^\top}_{\text{Schur complement}} \right) \quad (45)$$

- The mean of the posterior distribution is given by the **matrix of regression coefficients**, its variance can be computed using the **Schur complement**.
- We can compute confidence intervals (e. g. 90 % | 95 % | 99 %):

$$(1.65 \mid 1.96 \mid 2.58) \cdot \sqrt{\text{var}(y_q)} \quad (46)$$

## Example

x	y
-1.50	-1.60
-0.25	0.50
0.00	0.80
1.00	-2.00
5.00	0.00
5.50	1.00
10.50	3.00
11.50	3.00

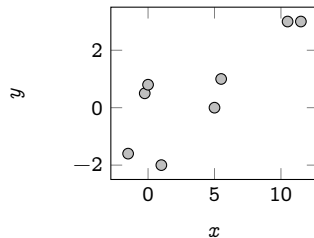
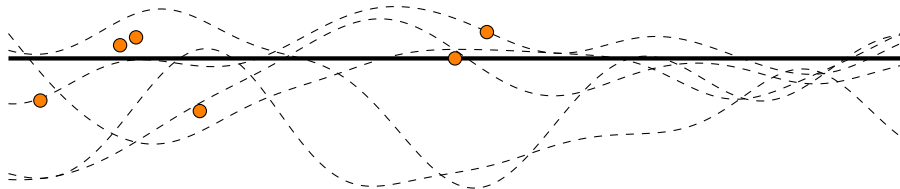


Figure 6:

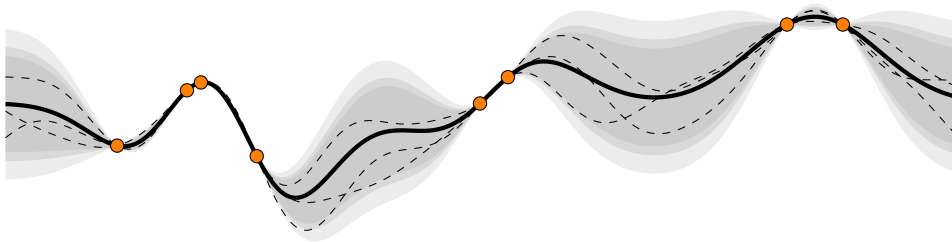
Example data set for a Gaussian process

- Suppose  $\sigma_f = 1.27, l = 1.00$ . What is  $y_q$  for  $x_q = 8$ ?
- Let's plot the prior distribution first.

**Prior distribution****Figure 7:**

Prior distribution for the Gaussian process

- Naturally, the prior does not fit the data well (we have not fitted the model yet).
- We have zero mean everywhere.

**Posterior distribution****Figure 8:**

Posterior distribution for the Gaussian process

**Wait a minute: Isn't this model overfitting the training data?**

- The model clearly overfits the data as can be seen from the previous slide (the regression line goes through each training data point perfectly).
- This is because the model assumes the data to be **noise-free**.
- It is possible to add a little bit of noise, in order to deal with this easily ( $\sigma_n$  is the variance of the noise):

$$K_{\sigma_n} \longleftarrow K + \sigma_n I \quad (47)$$

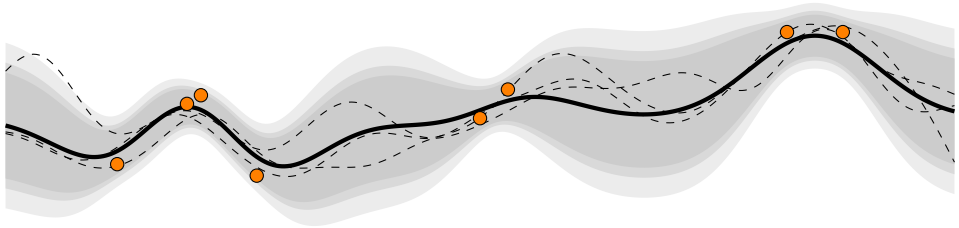
- The updated formulas look like this:
  - Matrix of regression coefficients (**same result as in kernel ridge regression**):

$$K_* K_{\sigma_n}^{-1} y \quad (48)$$

- Schur complement:

$$K_{**} - K_* K_{\sigma_n}^{-1} K_*^\top \quad (49)$$



**Prior distribution (with noise)****Figure 9:**

Posterior distribution for the Gaussian process with noise

## Learning the Hyper-Parameters

- The results of Gaussian process regression depend heavily on the parameters  $\{\sigma_f, l\}$ , which is why these parameters should be optimized for the task at hand.
- This can be done by maximizing the **marginal likelihood** (e. g. by using gradient ascent).



The exact procedure is very involved and out of scope for this lecture.

# Support Vector Regression

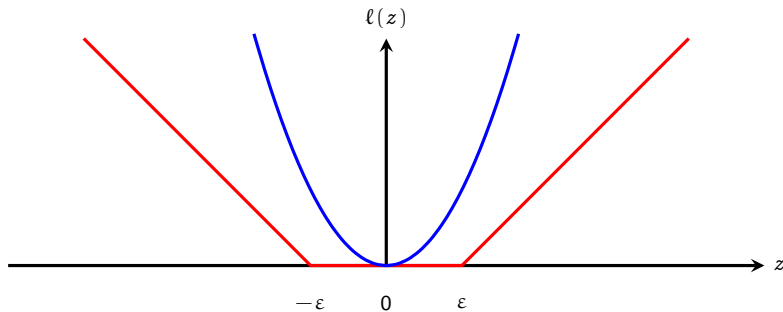
## Introduction

- Support vector machines can be extended to regression problems, while preserving the property of sparseness.
- In ordinary least squares, we minimize a regularized error function given by:

$$\mathcal{J}(\boldsymbol{\theta}) = \frac{1}{2} \sum_{i=1}^n (\hat{h}(\mathbf{x}^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2} \|\mathbf{w}\|^2 \quad (50)$$

- In the following,  $\boldsymbol{\theta} = \{\mathbf{w}, b\}$  and  $\hat{h}(\mathbf{x}) = \mathbf{w}^\top \boldsymbol{\varphi}(\mathbf{x}) + b$ .
- To obtain sparse solutions, the quadratic error is replaced by an  **$\varepsilon$ -insensitive error function**, which gives zero error if the absolute difference between the prediction and the target is less than  $\varepsilon$ :

$$\ell_{\varepsilon}(\hat{h}(\mathbf{x}) - y) = \begin{cases} 0 & \text{if } |\hat{h}(\mathbf{x}) - y| < \varepsilon \\ |\hat{h}(\mathbf{x}) - y| - \varepsilon & \text{otherwise} \end{cases} \quad (51)$$

**Figure 10:**

An  $\varepsilon$ -insensitive error function (red) compared to the quadratic error function (blue)

- We therefore minimize a regularized error function given by:

$$\mathcal{J}(\boldsymbol{\theta}) = C \sum_{i=1}^n \ell_{\varepsilon}(\hat{h}(\mathbf{x}^{(i)}) - y^{(i)}) + \frac{1}{2} \|\mathbf{w}\|^2 \quad (52)$$

- Analogously to support vector machines for classification,  $C$  denotes the (inverse) regularization parameter.
- Again, we introduce slack variables:
  - We now need two slack variables  $\xi_i \geq 0$  and  $\hat{\xi}_i \geq 0$  for each data point  $\mathbf{x}^{(i)}$ .
  - $\xi_i > 0$  corresponds to a point for which  $y^{(i)} > \hat{h}(\mathbf{x}^{(i)}) + \varepsilon$ .
  - $\hat{\xi}_i \geq 0$  corresponds to a point for which  $y^{(i)} < h(\mathbf{x}^{(i)}) - \varepsilon$ .
- The error function for support vector regression can then be rewritten as:

$$\mathcal{J}(\boldsymbol{\theta}) = C \sum_{i=1}^n (\xi_i + \hat{\xi}_i) + \frac{1}{2} \|\mathbf{w}\|^2 \quad (53)$$

Illustration of SVM regression, showing the regression curve together with the  $\varepsilon$ -insensitive 'tube'. Also shown are examples of the slack variables  $\xi$  and  $\hat{\xi}$ .

Points above the  $\varepsilon$ -tube have  $\xi > 0$  and  $\hat{\xi} = 0$ , points below the tube have  $\xi = 0$  and  $\hat{\xi} > 0$ . Points inside the tube are characterized by  $\xi = \hat{\xi} = 0$ .

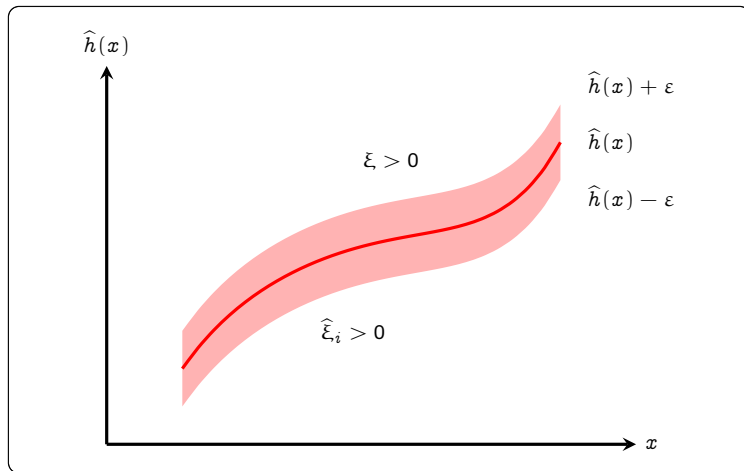
**Figure 11:**

Illustration of support vector regression

## Optimization

- The cost function given by  $\Rightarrow$  eq. (53) must be minimized subject to the constraints:

$$\xi_i \geq 0 \quad (54)$$

$$\hat{\xi}_i \geq 0 \quad (55)$$

$$y^{(i)} \leq h(\mathbf{x}^{(i)}) + \varepsilon + \xi_i \quad (56)$$

$$y^{(i)} \geq h(\mathbf{x}^{(i)}) - \varepsilon - \hat{\xi}_i \quad (57)$$

- This can be achieved by introducing Lagrange multipliers  $\alpha_i \geq 0$ ,  $\hat{\alpha}_i \geq 0$ ,  $\mu_i \geq 0$  and  $\hat{\mu}_i \geq 0$ :

$$\begin{aligned} \mathcal{L} = & C \sum_{i=1}^n (\xi_i + \hat{\xi}_i) + \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^n (\mu_i \xi_i + \hat{\mu}_i \hat{\xi}_i) \\ & - \sum_{i=1}^n \alpha_i (\varepsilon + \xi_i + \hat{h}(\mathbf{x}^{(i)}) - y^{(i)}) - \sum_{i=1}^n \hat{\alpha}_i (\varepsilon + \hat{\xi}_i - \hat{h}(\mathbf{x}^{(i)}) + y^{(i)}) \end{aligned} \quad (58)$$

Derivatives of  $\mathcal{L}$ 

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} \stackrel{!}{=} 0 \quad \Rightarrow \quad \mathbf{w} = \sum_{i=1}^n (\alpha_i - \hat{\alpha}_i) \varphi(\mathbf{x}^{(i)}) \quad (59)$$

$$\frac{\partial \mathcal{L}}{\partial b} \stackrel{!}{=} 0 \quad \Rightarrow \quad \sum_{i=1}^n (\alpha_i - \hat{\alpha}_i) = 0 \quad (60)$$

$$\frac{\partial \mathcal{L}}{\partial \xi_i} \stackrel{!}{=} 0 \quad \Rightarrow \quad \alpha_i + \mu_i = C \quad (61)$$

$$\frac{\partial \mathcal{L}}{\partial \hat{\xi}_i} \stackrel{!}{=} 0 \quad \Rightarrow \quad \hat{\alpha}_i + \hat{\mu}_i = C \quad (62)$$



We can use these results to obtain the dual formulation which has to be maximized.



**Dual formulation**

- The dual formulation is given by:

$$\mathcal{L}(\alpha, \hat{\alpha}) = -\frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n (\alpha_i - \hat{\alpha}_i)(\alpha_j - \hat{\alpha}_j) \mathcal{K}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) - \varepsilon \sum_{i=1}^n (\alpha_i + \hat{\alpha}_i) + \sum_{i=1}^n (\alpha_i - \hat{\alpha}_i) y^{(i)} \quad (63)$$

- The dual is expressed in terms of a kernel function  $\mathcal{K}(\mathbf{x}, \mathbf{x}')$ .
- Maximize the dual function:  $\max_{\alpha, \hat{\alpha}} \mathcal{L}(\alpha, \hat{\alpha})$
- Again, this is a constraint optimization problem which is optimized subject to:

$$0 \leq \alpha_i \leq C \quad (64)$$

$$0 \leq \hat{\alpha}_i \leq C \quad (65)$$

- We again have the **box constraints** which directly follow from the fact that the Lagrange multipliers have to be  $\geq 0$  together with  $\Rightarrow$  eq. (61) and  $\Rightarrow$  eq. (62).

- Substituting  $\Rightarrow$  eq. (59) into  $\hat{h}(\mathbf{x})$ , we see that predictions for new inputs can be made using:

$$\hat{h}(\mathbf{x}) = \sum_{i=1}^n (\alpha_i - \hat{\alpha}_i) \mathcal{K}(\mathbf{x}, \mathbf{x}^{(i)}) + b \quad (66)$$

- The support vectors are those data points for which  $\alpha_i \neq 0$  or  $\hat{\alpha}_i \neq 0$ . Such points either lie on the boundary of the  $\varepsilon$ -tube or outside the tube. All points within the tube have  $\alpha_i = \hat{\alpha}_i = 0$ .
- It is again a **sparse solution**, since we only need the support vectors for the prediction.

## Karush-Kuhn-Tucker Conditions

- The **Karush-Kuhn-Tucker (KKT) conditions** state that at the solution, the product of dual variables and constraints must vanish.
- The KKT conditions for support vector regression are given by:

$$\alpha_i (\varepsilon + \xi_i + \widehat{h}(\mathbf{x}^{(i)}) - y^{(i)}) = 0 \quad (67)$$

$$\widehat{\alpha}_i (\varepsilon + \widehat{\xi}_i - \widehat{h}(\mathbf{x}^{(i)}) + y^{(i)}) = 0 \quad (68)$$

$$\overbrace{(C - \alpha_i)}^{\mu_i} \xi_i = 0 \quad (69)$$

$$\underbrace{(C - \widehat{\alpha}_i)}_{\widehat{\mu}_i} \widehat{\xi}_i = 0 \quad (70)$$



We can derive useful results from the KKT conditions (cf. next slide).

- First of all, we note that  $\alpha_i$  can only be **non-zero**, if  $\varepsilon + \xi_i + \widehat{h}(\mathbf{x}^{(i)}) - y^{(i)} = 0$ . This implies that the data point either lies on the upper boundary of the  $\varepsilon$ -tube ( $\xi_i = 0$ ) or above it ( $\xi_i > 0$ ).
- Analogous:  $\widehat{\alpha}_i$
- The two constraints  $\varepsilon + \xi_i + \widehat{h}(\mathbf{x}^{(i)}) - y^{(i)}$  and  $\varepsilon + \widehat{\xi}_i - \widehat{h}(\mathbf{x}^{(i)}) + y^{(i)}$  are incompatible. This can be seen by adding them together and noting that  $\xi_i, \widehat{\xi}_i$  are non-negative and  $\varepsilon$  is strictly positive. So for every data point  $\mathbf{x}^{(i)}$ , either  $\alpha_i$  or  $\widehat{\alpha}_i$  (or both) must be zero.
- Parameter  $b$  in  $\Rightarrow$  eq. (66) can be found by considering a data point for which  $0 < \alpha_i < C$  ( $\hat{=}$  support vector). From  $\Rightarrow$  eq. (69) it must have  $\xi_i = 0$ . Therefore, according to  $\Rightarrow$  eq. (67) it must satisfy  $\varepsilon + \widehat{h}(\mathbf{x}^{(i)}) - y^{(i)} = 0$ .
- For  $b$  we obtain:

$$b = y^{(i)} - \varepsilon - \mathbf{w}^\top \varphi(\mathbf{x}^{(i)}) \quad (71)$$

$$= y^{(i)} - \varepsilon - \sum_{j=1}^n (\alpha_j - \widehat{\alpha}_j) \mathcal{K}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) \quad (72)$$

- In practice, it is better to consider all support vectors to find  $b$  (average).

## Example

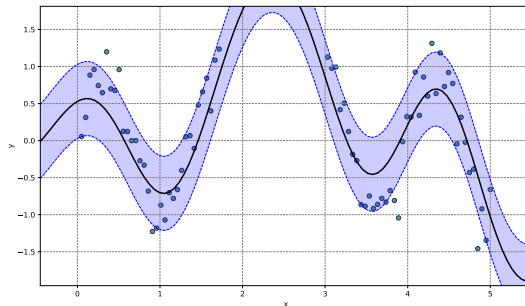


Figure 12:

Example of support vector regression using `scikit-learn`

**Thank you very much for the attention!**

**Topic:**       \*\*\*\*\* Advanced Machine Learning \*\*\*\*\* Advanced Regression Techniques  
**Term:**       Summer term 2020

**Contact:**  
M. Sc. Daniel Wehner  
SAP SE / DHBW Mannheim  
[daniel.wehner@sap.com](mailto:daniel.wehner@sap.com)

Do you have any questions?