***** Advanced Machine Learning ***** Advanced Regression Techniques

M. Sc. Daniel Wehner

SAP SE / DHBW Mannheim

Summer term 2020

DHBW
Duale Hochschule
Baden-Württemberg

Agenda for this Unit

Bayesian Regression **4**

Introduction 4

Kernel Ridge Regression **5**

Introduction 5

Woodbury Matrix Identity 6

Gaussian Process Regression **10**

Introduction 10

Learning a Gaussian Process Model 12

Example 15

Learning the Hyper-Parameters

20

Bayesian Regression

Introduction

-

Kernel Ridge Regression

Introduction

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

$$\boldsymbol{\theta} = (\boldsymbol{\Phi}^\intercal \boldsymbol{\Phi} + \lambda \boldsymbol{I})^{-1} \boldsymbol{\Phi}^\intercal \boldsymbol{y} \tag{1}$$

- In the above formula, $\boldsymbol{\Phi}$ denotes the design matrix (regressor matrix), $\boldsymbol{y}$ is the label vector and λ 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 $\boldsymbol{x}_q$, we have to calculate:

$$y_q = \varphi(\boldsymbol{x}_q)^\intercal \boldsymbol{\theta} \tag{2}$$

Step ❶: Insert normal equation ⇒ eq. (1):

$$= \varphi(\boldsymbol{x}_q)^\intercal (\boldsymbol{\Phi}^\intercal \boldsymbol{\Phi} + \lambda \boldsymbol{I})^{-1} \boldsymbol{\Phi}^\intercal \boldsymbol{y} \tag{3}$$

Step ❷: Apply Woodbury matrix identity:

$$= \varphi(\boldsymbol{x}_q)^\intercal \boldsymbol{\Phi}^\intercal (\boldsymbol{\Phi} \boldsymbol{\Phi}^\intercal + \lambda \boldsymbol{I})^{-1} \boldsymbol{y} \tag{4}$$

- The formula given in ⇒ eq. (4) exclusively uses dot products of input features and is therefore susceptible to kernels.

- Replace the dot products by kernel functions:

Rewrite of $\varphi(\boldsymbol{x}_q)^\intercal \boldsymbol{\Phi}^\intercal$:

$$\varphi(\boldsymbol{x}_q)^\intercal \boldsymbol{\Phi}^\intercal = \varphi(\boldsymbol{x}_q)^\intercal \begin{bmatrix} \varphi(\boldsymbol{x}^{(1)})^\intercal \\ \vdots \\ \varphi(\boldsymbol{x}^{(n)})^\intercal \end{bmatrix}^\intercal = \begin{bmatrix} \mathcal{K}(\boldsymbol{x}_q, \boldsymbol{x}^{(1)}) \\ \vdots \\ \mathcal{K}(\boldsymbol{x}_q, \boldsymbol{x}^{(n)}) \end{bmatrix} = \boldsymbol{K}_*(\boldsymbol{x}_q) \tag{5}$$

Rewrite of $\boldsymbol{\Phi}\boldsymbol{\Phi}^\intercal$:

$$\boldsymbol{\Phi}\boldsymbol{\Phi}^\intercal = \begin{bmatrix} \varphi(\boldsymbol{x}^{(1)})^\intercal \\ \vdots \\ \varphi(\boldsymbol{x}^{(n)})^\intercal \end{bmatrix} \begin{bmatrix} \varphi(\boldsymbol{x}^{(1)})^\intercal \\ \vdots \\ \varphi(\boldsymbol{x}^{(n)})^\intercal \end{bmatrix}^\intercal = \begin{bmatrix} \mathcal{K}(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(1)}) & \dots & \mathcal{K}(\boldsymbol{x}^{(n)}, \boldsymbol{x}^{(1)}) \\ \vdots & \ddots & \vdots \\ \mathcal{K}(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(n)}) & \dots & \mathcal{K}(\boldsymbol{x}^{(n)}, \boldsymbol{x}^{(n)}) \end{bmatrix} = \boldsymbol{K} \tag{6}$$

- The kernel matrices $\boldsymbol{K}$ and $\boldsymbol{K}_*$ 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:

$$y_q = \boldsymbol{K}_*(\boldsymbol{x_q})(\boldsymbol{K} + \lambda \boldsymbol{I})^{-1}\boldsymbol{y} \tag{7}$$

- 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!

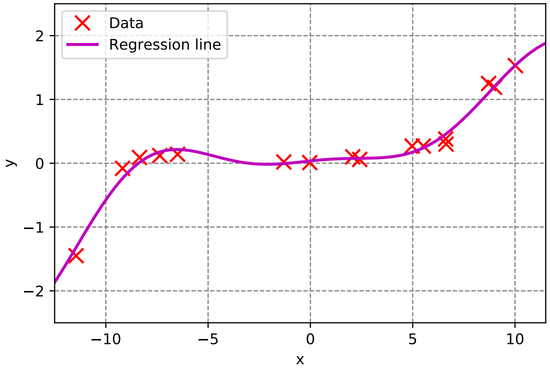

Figure 1: Result of kernel ridge regression

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:

$$h(\boldsymbol{x}) = \mathfrak{GP}(m(\boldsymbol{x}), \mathcal{K}(\boldsymbol{x}, \boldsymbol{x}')) \tag{8}$$

- In ⇒ eq. (8), $m(\boldsymbol{x})$ denotes the mean function, whereas $\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}')$ 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 $m(\boldsymbol{x})$ distribution is usually set to 0 everywhere.
- In practice, the squared exponential function ($\widehat{=}$ RBF-kernel) is frequently used:

$$\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}') = \sigma_f^2 \cdot \exp\left\{ \frac{-\|\boldsymbol{x} - \boldsymbol{x}'\|^2}{2 \cdot l^2} \right\} \tag{9}$$

- Hyper-Parameters:
  - σ_f^2 denotes the maximum allowable covariance. It should be high for functions covering a broad range of the y-axis. If $\boldsymbol{x} \approx \boldsymbol{x}'$, $\mathcal{K}(\boldsymbol{x}, \boldsymbol{x}')$ 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:

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

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

The matrices have the following form:

$$\boldsymbol{K} = \begin{bmatrix} \mathcal{K}(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(1)}) & \mathcal{K}(\boldsymbol{x}^{(2)}, \boldsymbol{x}^{(1)}) & \dots & \mathcal{K}(\boldsymbol{x}^{(n)}, \boldsymbol{x}^{(1)}) \\ \mathcal{K}(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}) & \mathcal{K}(\boldsymbol{x}^{(2)}, \boldsymbol{x}^{(2)}) & \dots & \mathcal{K}(\boldsymbol{x}^{(n)}, \boldsymbol{x}^{(2)}) \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{K}(\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(n)}) & \mathcal{K}(\boldsymbol{x}^{(2)}, \boldsymbol{x}^{(n)}) & \dots & \mathcal{K}(\boldsymbol{x}^{(n)}, \boldsymbol{x}^{(n)}) \end{bmatrix} \tag{10}$$

$$\boldsymbol{K}_* = \begin{bmatrix} \mathcal{K}(\boldsymbol{x}_q, \boldsymbol{x}^{(1)}) & \mathcal{K}(\boldsymbol{x}_q, \boldsymbol{x}^{(2)}) & \dots & \mathcal{K}(\boldsymbol{x}_q, \boldsymbol{x}^{(n)}) \end{bmatrix}^\intercal \tag{11}$$

$$\boldsymbol{K}_{**} = \mathcal{K}(\boldsymbol{x}_q, \boldsymbol{x}_q) \tag{12}$$

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 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:

$$\begin{bmatrix} \boldsymbol{y} \\ y_q \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \boldsymbol{K} & \boldsymbol{K}_*^\intercal \\ \boldsymbol{K}_* & \boldsymbol{K}_{**} \end{bmatrix}\right) \tag{13}$$

- What we actually want is the **posterior distribution** $p(y_q|\boldsymbol{y})$: *'Given the data, what is y_q?'*

- For Gaussian distributions, the posterior distribution can be computed analytically:

$$y_q|\boldsymbol{y} \sim \mathcal{N}(\underbrace{\boldsymbol{K}_*\boldsymbol{K}^{-1}\boldsymbol{y}}_{\text{Matrix of regr. coeff.}}, \underbrace{\boldsymbol{K}_{**} - \boldsymbol{K}_*\boldsymbol{K}^{-1}\boldsymbol{K}_*^\intercal}_{\text{Schur complement}}) \tag{14}$$

- 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 %):

$$(1.65 \mid 1.96 \mid 2.58) \cdot \sqrt{var(y_q)} \tag{15}$$

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

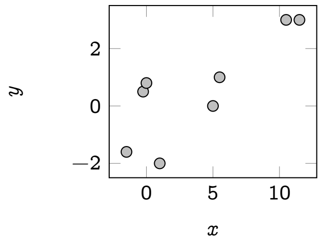

Figure 2: Example data set for a Gaussian process

- 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

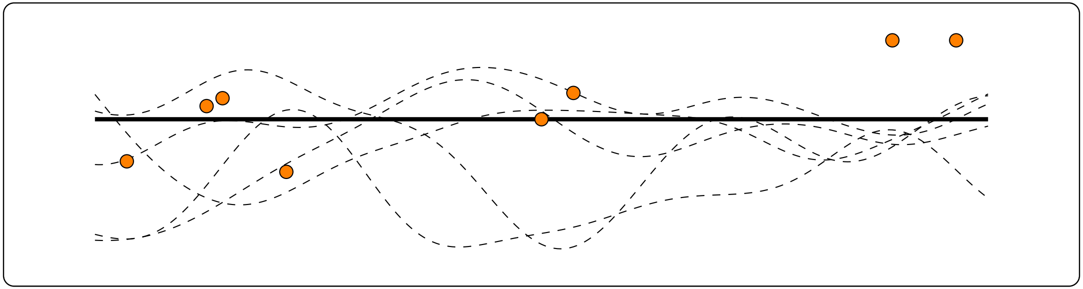

Figure 3: Prior distribution for the Gaussian process

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

Posterior distribution

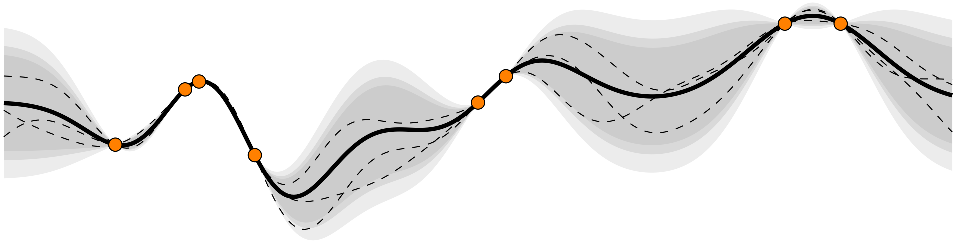

Figure 4: Posterior distribution for the Gaussian process

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 (σ_n is the variance of the noise):

$$\boldsymbol{K}_{\sigma_n} \longleftarrow \boldsymbol{K} + \sigma_n \boldsymbol{I} \tag{16}$$

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

$$\boldsymbol{K}_* \boldsymbol{K}_{\sigma_n}^{-1} \boldsymbol{y} \tag{17}$$

  - Schur complement:

$$\boldsymbol{K}_{**} - \boldsymbol{K}_* \boldsymbol{K}_{\sigma_n}^{-1} \boldsymbol{K}_*^{\intercal} \tag{18}$$

Prior distribution (with noise)

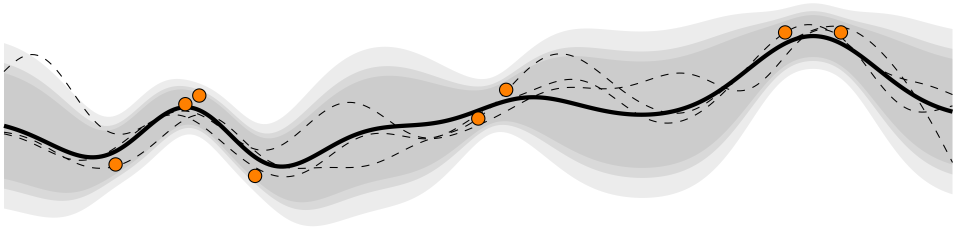

Figure 5: Posterior distribution for the Gaussian process with noise

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 down by maximizing the **marginal likelihood** (e. g. by using gradient ascent).
- The exact procedure is very involved and out of scope for this lecture.