***** 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

Introduction

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

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

- 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):

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

- 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 ε to the function estimates:
  $$y = h(\boldsymbol{x}; \boldsymbol{\theta}) + \varepsilon \tag{3}$$
  2. The noise term ε is considered a Gaussian random variable with zero mean:
  $$\varepsilon \sim \mathcal{N}(0, \beta^{-1}) \tag{4}$$
- **With these assumptions y is now a random variable.** It has the following (Gaussian) probability distribution:
$$p(y|\boldsymbol{x}, \boldsymbol{\theta}, \beta) = \mathcal{N}(y|h(\boldsymbol{x}; \boldsymbol{\theta}), \beta^{-1}) \tag{5}$$

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

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

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

- Compute the log-likelihood:

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

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

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

Computation of the gradient

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

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

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

Important: Minimizing the squared error gives the maximum likelihood solution for the parameters θ assuming Gaussian noise.

Maximum Aposteriori (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 $\boldsymbol{\theta}$:

$$\overbrace{p(\boldsymbol{\theta}|\boldsymbol{X}, \boldsymbol{y}, \alpha, \beta)}^{\text{posterior}} \propto \overbrace{p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}, \beta)}^{\text{likelihood}} \cdot \overbrace{p(\boldsymbol{\theta}|\alpha)}^{\text{prior}} \tag{13}$$

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

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

The prior for the parameters

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

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

- The posterior then becomes:

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

- Compute the log-likelihood:

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

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

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

- Computation of the gradient:

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

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

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

$$\beta\boldsymbol{\Phi}^\intercal\boldsymbol{y} = (\beta\boldsymbol{\Phi}^\intercal\boldsymbol{\Phi} + \alpha\boldsymbol{I})\boldsymbol{\theta} \qquad \Rightarrow \boldsymbol{\theta}_{map} = \left(\boldsymbol{\Phi}^\intercal\boldsymbol{\Phi} + \frac{\alpha}{\beta}\boldsymbol{I}\right)^{-1} \boldsymbol{\Phi}^\intercal\boldsymbol{y} \tag{22}$$

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

$$\arg\min_{\boldsymbol{\theta}} \frac{1}{2}\|\boldsymbol{\Phi}\boldsymbol{\theta} - \boldsymbol{y}\|^2 + \frac{\lambda}{2}\|\boldsymbol{\theta}\|^2 \tag{23}$$

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

$$\boldsymbol{\theta} = (\boldsymbol{\Phi}^\intercal\boldsymbol{\Phi} + \lambda\boldsymbol{I})^{-1}\boldsymbol{\Phi}^\intercal\boldsymbol{y} \tag{24}$$

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

You assume two things when you put a regularizer λ 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 $\boldsymbol{\theta}$.

$$p(\boldsymbol{\theta}|\alpha) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}_0, \boldsymbol{\Lambda}_0) \tag{25}$$

- In ⇒ eq. (25), the mean $\boldsymbol{\mu}_0$ and the precision matrix $\boldsymbol{\Lambda}_0$ are given by $\mathbf{0}$ and $\alpha^{-1}\boldsymbol{I}$, respectively. Therefore, the prior is a zero-mean, isotropic ($\widehat{=}$ rotation-invariant) Gaussian distribution.

- The posterior distribution of the parameters $p(\boldsymbol{\theta}|\boldsymbol{X}, \boldsymbol{y}, \alpha, \beta)$ is then:

$$p(\boldsymbol{\theta}|\boldsymbol{X}, \boldsymbol{y}, \alpha, \beta) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}_n, \boldsymbol{\Lambda}_n) \tag{26}$$

  with:

$$\boldsymbol{\mu}_n = \beta \boldsymbol{\Lambda}_n^{-1} \boldsymbol{\Phi}^\intercal \boldsymbol{y} \tag{27}$$

$$\boldsymbol{\Lambda}_n = \boldsymbol{\Lambda}_0^{-1} + \beta \boldsymbol{\Phi}^\intercal \boldsymbol{\Phi} \tag{28}$$

- 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 $h(x; \boldsymbol{\theta}) = \theta_0 + \theta_1 \cdot x$.

- 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 $f(x; \boldsymbol{a}) = a_0 + a_1 \cdot x$, 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 $f(x^{(i)}; \boldsymbol{a})$, and finally adding some Gaussian noise with precision $\beta = \frac{1}{0.2}$. α is fixed to 2.0.

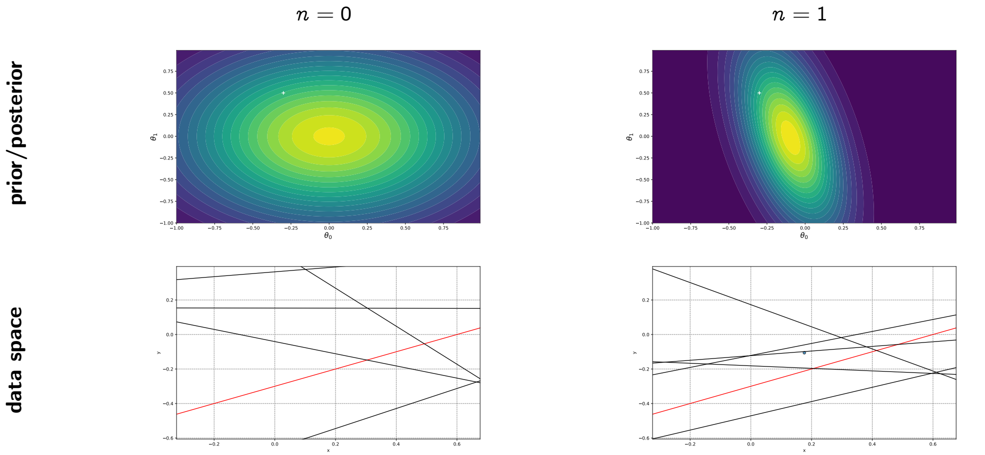

Figure 1: Example for Bayesian regression (part I)

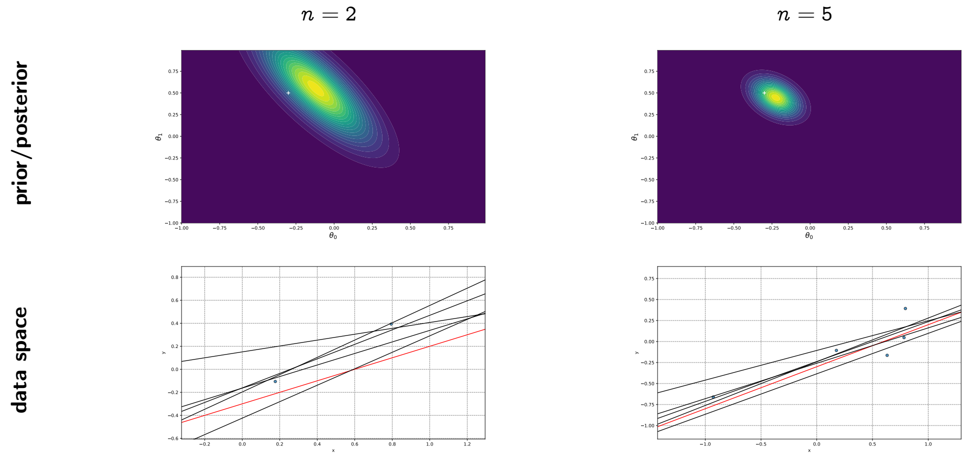

Figure 2: Example for Bayesian regression (part II)

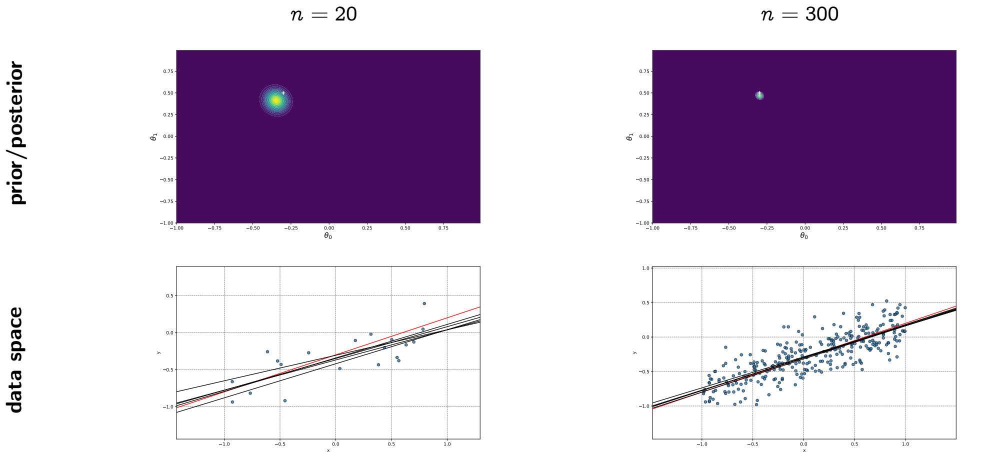

Figure 3: Example for Bayesian regression (part III)

Predictive distribution

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

$$p(y_q|\boldsymbol{x}_q, \boldsymbol{X}, \boldsymbol{y}, \alpha, \beta) = \int_{\boldsymbol{\theta}} \underbrace{p(y_q|\boldsymbol{x}_q, \boldsymbol{X}, \boldsymbol{\theta}, \beta)}_{\text{regression model}} \cdot \underbrace{p(\boldsymbol{\theta}|\boldsymbol{X}, \boldsymbol{y}, \alpha, \beta)}_{\text{parameter posterior}} \mathrm{d}\boldsymbol{\theta} \tag{29}$$

- 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:

$$p(y_q|\boldsymbol{x}_q, \boldsymbol{X}, \boldsymbol{y}, \alpha, \beta) = \mathcal{N}(y_q|\boldsymbol{\mu}_n^{\intercal}\varphi(\boldsymbol{x}_q), \sigma_n^2(\boldsymbol{x}_q)) \tag{30}$$

with:

$$\sigma_n^2(\boldsymbol{x}_q) = \frac{1}{\beta} + \varphi(\boldsymbol{x}_q)^{\intercal}\boldsymbol{\Lambda}_n^{-1}\varphi(\boldsymbol{x}_q) \tag{31}$$

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

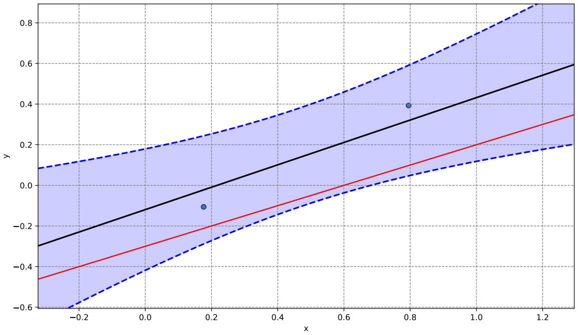

Figure 4: Bayesian regression and uncertainty estimate

Kernel Ridge Regression

Introduction

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

$$\boldsymbol{\theta} = (\boldsymbol{\Phi}^\intercal \boldsymbol{\Phi} + \lambda \boldsymbol{I})^{-1} \boldsymbol{\Phi}^\intercal \boldsymbol{y} \tag{32}$$

- 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:

$$y_q = \varphi(\boldsymbol{x}_q)^\intercal \boldsymbol{\theta} \tag{33}$$

Step ❶: Insert normal equation ⇒ eq. (32):

$$= \varphi(\boldsymbol{x}_q)^\intercal (\boldsymbol{\Phi}^\intercal \boldsymbol{\Phi} + \lambda \boldsymbol{I})^{-1} \boldsymbol{\Phi}^\intercal \boldsymbol{y} \tag{34}$$

Step ❷: Apply Woodbury matrix identity:

$$= \varphi(\boldsymbol{x}_q)^\intercal \boldsymbol{\Phi}^\intercal (\boldsymbol{\Phi} \boldsymbol{\Phi}^\intercal + \lambda \boldsymbol{I})^{-1} \boldsymbol{y} \tag{35}$$

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

- Replace the dot products by kernel functions:

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

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

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

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

- 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:

$$y_q = \boldsymbol{K}_*(\boldsymbol{x_q})(\boldsymbol{K} + \lambda \boldsymbol{I})^{-1}\boldsymbol{y} \tag{38}$$

- 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

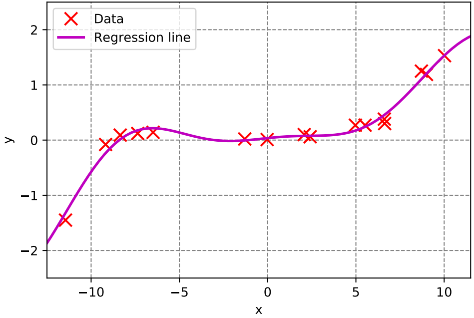

Figure 5: Result of kernel ridge regression

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:

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

- In ⇒ eq. (39), $m(\boldsymbol{x})$ 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:

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

- 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:

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

$$\boldsymbol{K}_* = \begin{bmatrix} \mathcal{K}(\boldsymbol{x}_q, \boldsymbol{x}^{(1)}) & \mathcal{K}(\boldsymbol{x}_q, \boldsymbol{x}^{(2)}) & \dots & \mathcal{K}(\boldsymbol{x}_q, \boldsymbol{x}^{(n)}) \end{bmatrix}^{\mathsf{T}} \tag{42}$$

$$\boldsymbol{K}_{**} = \mathcal{K}(\boldsymbol{x}_q, \boldsymbol{x}_q) \tag{43}$$

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 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:

$$\begin{bmatrix} \boldsymbol{y} \\ y_q \end{bmatrix} \sim \mathcal{N}\left(\boldsymbol{0}, \begin{bmatrix} \boldsymbol{K} & \boldsymbol{K}_*^\intercal \\ \boldsymbol{K}_* & \boldsymbol{K}_{**} \end{bmatrix}\right) \tag{44}$$

- 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:

$$y_q|\boldsymbol{y} \sim \mathcal{N}(\underbrace{\boldsymbol{K}_* \boldsymbol{K}^{-1} \boldsymbol{y}}_{\text{Matrix of regr. coeff.}}, \underbrace{\boldsymbol{K}_{**} - \boldsymbol{K}_* \boldsymbol{K}^{-1} \boldsymbol{K}_*^\intercal}_{\text{Schur complement}}) \tag{45}$$

- 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 %):

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

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

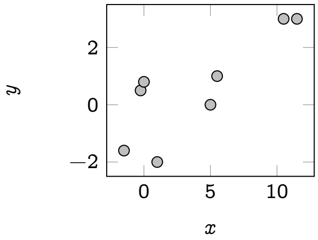

Figure 6: Example data set for a Gaussian process

- 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

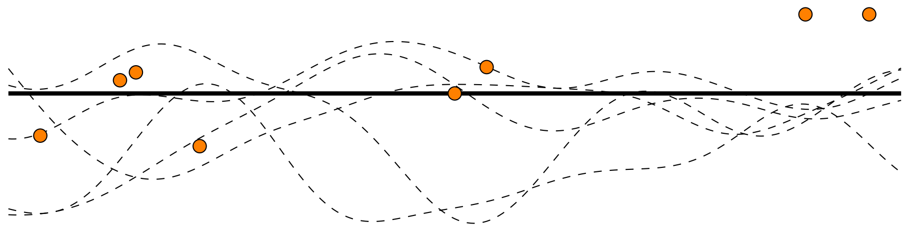

Figure 7: Prior distribution for the Gaussian process

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

Posterior distribution

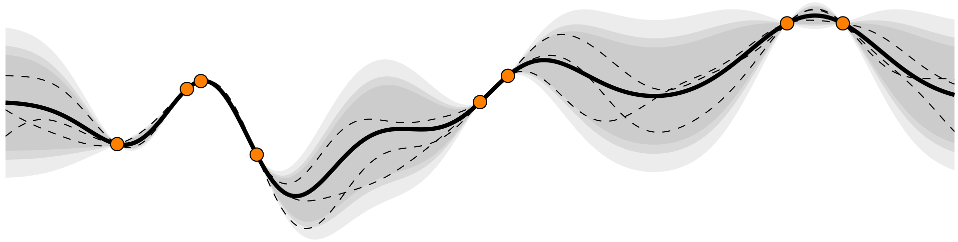

Figure 8: Posterior distribution for the Gaussian process

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 (σ_n is the variance of the noise):

$$\boldsymbol{K}_{\sigma_n} \longleftarrow \boldsymbol{K} + \sigma_n \boldsymbol{I} \tag{47}$$

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

$$\boldsymbol{K}_* \boldsymbol{K}_{\sigma_n}^{-1} \boldsymbol{y} \tag{48}$$

  - Schur complement:

$$\boldsymbol{K}_{**} - \boldsymbol{K}_* \boldsymbol{K}_{\sigma_n}^{-1} \boldsymbol{K}_*^{\intercal} \tag{49}$$

Prior distribution (with noise)

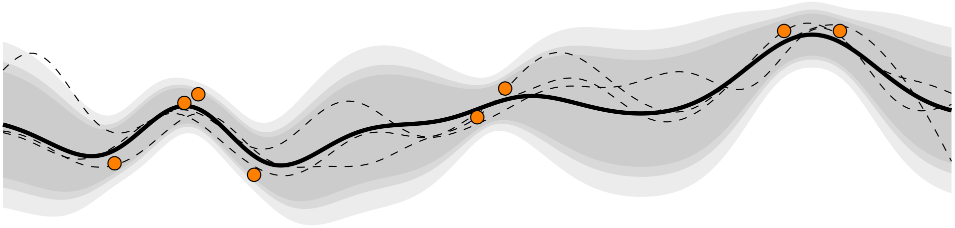

Figure 9: Posterior distribution for the Gaussian process with noise

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 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.

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:

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

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

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

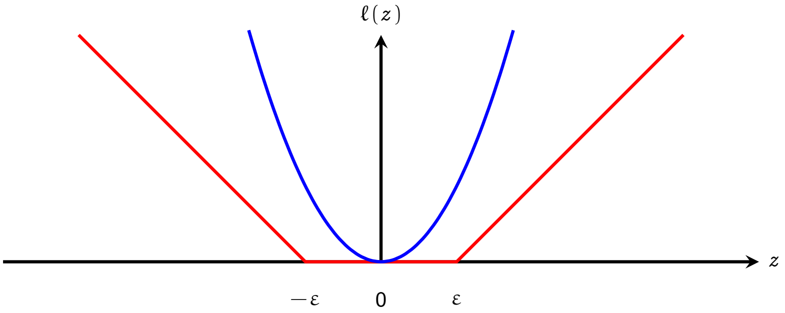

Figure 10: An ε-insensitive error function (red) compared to the quadratic error function (blue)

- We therefore minimize a regularized error function given by:

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

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

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

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

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

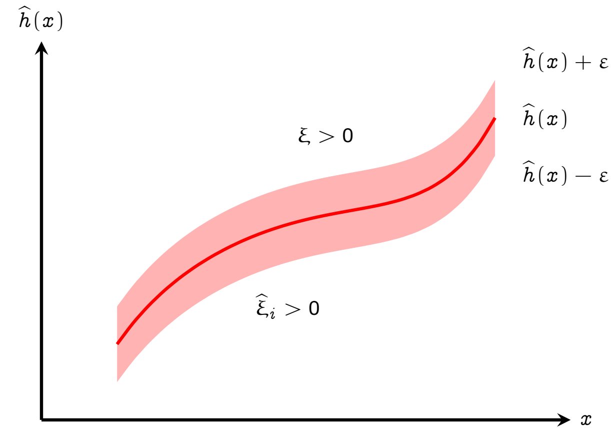

Figure 11: Illustration of support vector regression

Optimization

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

$$\xi_i \geqslant 0 \tag{54}$$

$$\widehat{\xi}_i \geqslant 0 \tag{55}$$

$$y^{(i)} \leqslant h(\boldsymbol{x}^{(i)}) + \varepsilon + \xi_i \tag{56}$$

$$y^{(i)} \geqslant h(\boldsymbol{x}^{(i)}) - \varepsilon - \widehat{\xi}_i \tag{57}$$

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

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

Derivatives of $\mathcal{L}$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}} \overset{!}{=} 0 \qquad \Rightarrow \qquad \boldsymbol{w} = \sum_{i=1}^{n} (\alpha_i - \widehat{\alpha}_i)\, \varphi(\boldsymbol{x}^{(i)}) \tag{59}$$

$$\frac{\partial \mathcal{L}}{\partial b} \overset{!}{=} 0 \qquad \Rightarrow \qquad \sum_{i=1}^{n} (\alpha_i - \widehat{\alpha}_i) = 0 \tag{60}$$

$$\frac{\partial \mathcal{L}}{\partial \xi_i} \overset{!}{=} 0 \qquad \Rightarrow \qquad \alpha_i + \mu_i = C \tag{61}$$

$$\frac{\partial \mathcal{L}}{\partial \widehat{\xi}_i} \overset{!}{=} 0 \qquad \Rightarrow \qquad \widehat{\alpha}_i + \widehat{\mu}_i = C \tag{62}$$

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

Dual formulation

- The dual formulation is given by:

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

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

$$0 \leqslant \alpha_i \leqslant C \tag{64}$$

$$0 \leqslant \widehat{\alpha}_i \leqslant C \tag{65}$$

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

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

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

- The support vectors are those data points for which $\alpha_i \neq 0$ or $\widehat{\alpha}_i \neq 0$. Such points either lie on the boundary of the ε-tube or outside the tube. All points within the tube have $\alpha_i = \widehat{\alpha}_i = 0$.

- 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:

$$\alpha_i \left(\varepsilon + \xi_i + \widehat{h}(\boldsymbol{x}^{(i)}) - y^{(i)} \right) = 0 \tag{67}$$

$$\widehat{\alpha}_i \left(\varepsilon + \widehat{\xi}_i - \widehat{h}(\boldsymbol{x}^{(i)}) + y^{(i)} \right) = 0 \tag{68}$$

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

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

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

- First of all, we note that α_i can only be **non-zero**, if $\varepsilon + \xi_i + \widehat{h}(\boldsymbol{x}^{(i)}) - y^{(i)} = 0$. This implies that the data point either lies on the upper boundary of the ε-tube ($\xi_i = 0$) or above it ($\xi_i > 0$).

- Analogous: $\widehat{\alpha}_i$

- The two constraints $\varepsilon + \xi_i + \widehat{h}(\boldsymbol{x}^{(i)}) - y^{(i)}$ and $\varepsilon + \widehat{\xi}_i - \widehat{h}(\boldsymbol{x}^{(i)}) + y^{(i)}$ are incompatible. This can be seen by adding them together and noting that $\xi_i, \widehat{\xi}_i$ are non-negative and ε is strictly positive. So for every data point $\boldsymbol{x}^{(i)}$, either α_i or $\widehat{\alpha}_i$ (or both) must be zero.

- Parameter b in ⇒ eq. (66) can be found by considering a data point for which $0 < \alpha_i < C$ ($\widehat{=}$ support vector). From ⇒ eq. (69) it must have $\xi_i = 0$. Therefore, according to ⇒ eq. (67) it must satisfy $\varepsilon + \widehat{h}(\boldsymbol{x}^{(i)}) - y^{(i)} = 0$.

- For b we obtain:

$$b = y^{(i)} - \varepsilon - \boldsymbol{w}^{\intercal} \varphi(\boldsymbol{x}^{(i)}) \tag{71}$$

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

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

Example

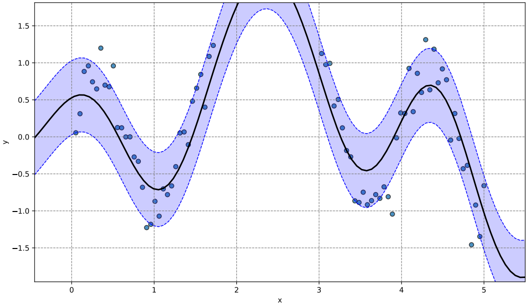

Figure 12: Example of support vector regression using `scikit-learn`

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

Do you have any questions?