*** * * Artificial Intelligence and Machine Learning * * ***

Linear Regression and Maximum Likelihood Regression

Daniel Wehner, M.Sc.

SAP SE / DHBW Mannheim

Summer term 2025

Find all slides on GitHub (DaWe1992/Applied_ML_Fundamentals)

Lecture Overview

	I	Machine Learning Introduction
	II	Optimization Techniques
	III	Bayesian Decision Theory
	IV	Non-parametric Density Estimation
	V	Probabilistic Graphical Models
●	VI	Linear Regression
	VII	Logistic Regression
	VIII	Deep Learning
	IX	Evaluation
	X	Decision Trees
	XI	Support Vector Machines
	XII	Clustering
	XIII	Principal Component Analysis
	XIV	Reinforcement Learning
	XV	Advanced Regression

Agenda for this Unit

1. Introduction
2. Solutions to Regression
3. Basis Function Regression
4. Regularization Techniques
5. Wrap-Up

Section:
Introduction

What is Regression?

Least Squares Error Function

Linear Regression Overview

- Let a dataset $\mathcal{D}$ be given by

$$\mathcal{D} := \left\{ \left(\boldsymbol{x}^n, y_n \right) \right\}_{n=1}^{N}, \qquad \boldsymbol{x}^n \in \mathbb{R}^M, y_n \in \mathbb{R}$$

- Derive an (affine-)linear function *(also known as **model function** or **hypothesis**)*

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) := \theta_0 + \theta_1 x_1 + \cdots + \theta_M x_M \tag{1}$$

- $\boldsymbol{\theta} \in \mathbb{R}^{M+1}$ is the parameter vector containing the regression coefficients
- Once $\boldsymbol{\theta}$ is learned, it can be used for the prediction of unknown instances $\boldsymbol{x}'$

Linear Regression Overview (Ctd.)

- It is common to introduce the constant input $x_0 := 1$ associated to the offset-parameter θ_0 **(bias)**
- The input vector $\widetilde{\boldsymbol{x}} \in \mathbb{R}^{M+1}$ is then given by

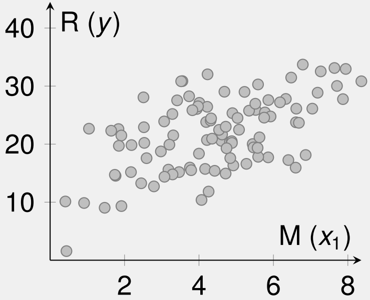
$$\widetilde{\boldsymbol{x}} := \left(1, \boldsymbol{x}\right)^\top$$

- This allows us to write equation (1) in a more compact form:

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) := \theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_M x_M = \sum_{m=0}^{M} \theta_m x_m = \boldsymbol{\theta}^\top \widetilde{\boldsymbol{x}} \tag{2}$$

Notation: In the following we shall simply write $\boldsymbol{x}$ instead of $\widetilde{\boldsymbol{x}}$

Example Dataset: Revenues



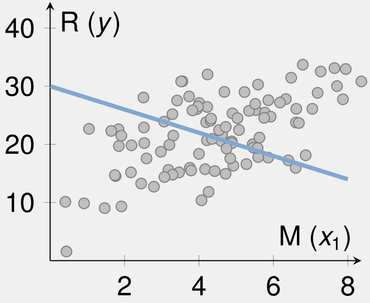
- **Assumption:** Functional dependence between revenue (R) and marketing expenses (M)
- Find an (affine-)linear function of the form:

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) := \theta_0 x_0 + \theta_1 x_1$$

- Choose $\boldsymbol{\theta}$ such that the line fits the data

Question: What is the best fitting line?

Example Dataset: Revenues



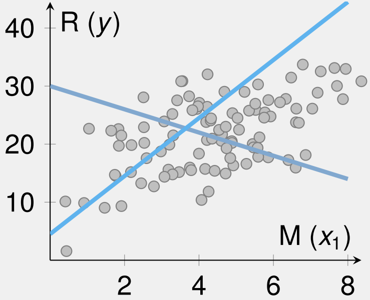
- **Assumption:** Functional dependence between revenue (R) and marketing expenses (M)
- Find an (affine-)linear function of the form:

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) := \theta_0 x_0 + \theta_1 x_1$$

- Choose $\boldsymbol{\theta}$ such that the line fits the data

Question: What is the best fitting line?

Example Dataset: Revenues



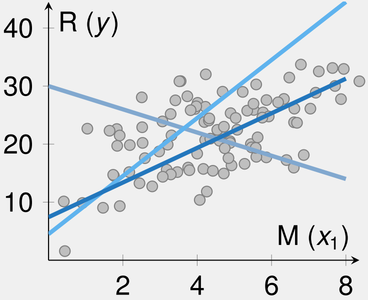
- **Assumption:** Functional dependence between revenue (R) and marketing expenses (M)
- Find an (affine-)linear function of the form:

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) := \theta_0 x_0 + \theta_1 x_1$$

- Choose $\boldsymbol{\theta}$ such that the line fits the data

Question: What is the best fitting line?

Example Dataset: Revenues



- **Assumption:** Functional dependence between revenue (R) and marketing expenses (M)
- Find an (affine-)linear function of the form:

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) := \theta_0 x_0 + \theta_1 x_1$$

- Choose $\boldsymbol{\theta}$ such that the line fits the data

Question: What is the best fitting line?

Important

Error Function for Regression

We need an error function $\mathfrak{J}(\boldsymbol{\theta})$ to know how well the regression line fits the data:

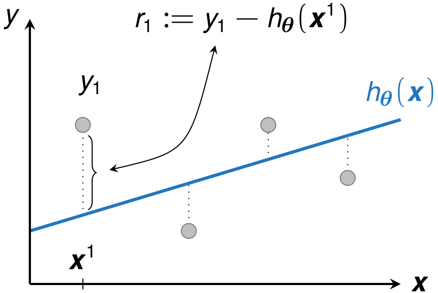
Least squares error function:

$$\mathfrak{J}(\boldsymbol{\theta}) := \frac{1}{N}\sum_{n=1}^{N} \ell^{\mathrm{LS}}\left(h_{\boldsymbol{\theta}}(\boldsymbol{x}^{n}), y_{n}\right) \quad \text{where} \quad \ell^{\mathrm{LS}}(\widehat{y}, y) := \frac{1}{2}(\widehat{y} - y)^{2} \tag{3}$$

We want to **minimize** (3) to obtain the optimal model parameters $\boldsymbol{\theta}^{\star}$, i. e.

$$\boldsymbol{\theta}^{\star} := \arg\min_{\boldsymbol{\theta}} \mathfrak{J}(\boldsymbol{\theta})$$

Error Function Intuition


y
$r_1 := y_1 - h_{\theta}(\boldsymbol{x}^1)$
y_1
$h_{\theta}(\boldsymbol{x})$
$\boldsymbol{x}^1$
$\boldsymbol{x}$

- r_n, $1 \leqslant n \leqslant N$, is called **residual**
- We want to minimize the residuals:

$$\boldsymbol{\theta}^{\star} := \arg\min_{\boldsymbol{\theta}} \frac{1}{2N} \sum_{n=1}^{N} r_n^2$$

Question:

Why do we consider the square in the error function?

Portrait: Carl Friedrich Gauss

Carl Friedrich Gauss (1777 – 1855) was a German mathematician, geodesist, and physicist who made significant contributions to many fields in mathematics and science. Gauss ranks among history's most influential mathematicians. Gauss published the second and third complete proofs of the **fundamental theorem of algebra** and made contributions to **number theory**.

He was instrumental in the discovery of the dwarf planet Ceres. His work on the motion of planetoids disturbed by large planets led to the introduction of the Gaussian gravitational constant and the method of **least squares**, which is still used in all sciences to minimize measurement error.

(Wikipedia)

Section:

Solutions to Regression

Closed-Form Solutions and the Normal Equations

Geometric Interpretation of Least Squares

What if Matrix Inversion fails?

Numerical Methods: Gradient Descent

Probabilistic Interpretation of linear Regression

Closed-Form Solutions

Usual approach: Let $x \in \mathbb{R}$ (1-dimensional). Calculate θ_0 and θ_1 according to

$$\theta_0^\star := \overline{y} - \theta_1^\star \overline{x} \qquad \text{and} \qquad \theta_1^\star := \frac{\sum_{n=1}^{N} \left(x_n - \overline{x}\right) \cdot \left(y_n - \overline{y}\right)}{\sum_{n=1}^{N} \left(x_n - \overline{x}\right)^2}, \tag{4}$$

where $\overline{x} := \frac{1}{N} \sum_{n=1}^{N} x_n$ and $\overline{y} := \frac{1}{N} \sum_{n=1}^{N} y_n$

Normal equations: *(scale to arbitrary dimensions)*

$$\boldsymbol{\theta}^\star := \left(\boldsymbol{X}^\top \boldsymbol{X}\right)^{-1} \boldsymbol{X}^\top \boldsymbol{y} \tag{5}$$

Design Matrix / Regressor Matrix $\boldsymbol{X}$

- $\boldsymbol{X}$ is called **design matrix** or **regressor matrix**
- The design matrix $\boldsymbol{X} \in \mathbb{R}^{N \times (M+1)}$ has the form *(first column consists of 1s)*

$$\boldsymbol{X} := \begin{pmatrix} — & \boldsymbol{x}^1 & — \\ — & \boldsymbol{x}^2 & — \\ \vdots & \vdots & \vdots \\ — & \boldsymbol{x}^N & — \end{pmatrix} = \begin{pmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \cdots & x_M^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & \cdots & x_M^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(N)} & x_2^{(N)} & \cdots & x_M^{(N)} \end{pmatrix} \tag{6}$$

- The label vector $\boldsymbol{y} \in \mathbb{R}^N$ is given by:

$$\boldsymbol{y} := \begin{pmatrix} y_1 & y_2 & y_3 & \ldots & y_N \end{pmatrix}^\top \tag{7}$$

Derivation: Useful Rules

In the derivation of equation (5) we will need the following rules:

Matrix transposition rules:

$$(\boldsymbol{A}^\top)^\top = \boldsymbol{A} \tag{8}$$

$$(\boldsymbol{A} + \boldsymbol{B})^\top = \boldsymbol{A}^\top + \boldsymbol{B}^\top \tag{9}$$

$$(\boldsymbol{AB})^\top = \boldsymbol{B}^\top \boldsymbol{A}^\top \tag{10}$$

Vector derivatives:

$$\frac{\partial}{\partial \boldsymbol{x}} \boldsymbol{x}^\top \boldsymbol{A} \boldsymbol{x} = 2\boldsymbol{A}\boldsymbol{x} \tag{11}$$

[Equation (11) only holds if $\boldsymbol{A}$ is symmetric]

$$\frac{\partial}{\partial \boldsymbol{x}} \boldsymbol{a}^\top \boldsymbol{x} = \boldsymbol{a} \tag{12}$$

Derivation: Rewrite the Error Function

Step ❶

- We rewrite the least squares error function (3) in matrix/vector notation:

$$\mathfrak{J}(\boldsymbol{\theta}) = \frac{1}{2N}\|\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y}\|^2 \overset{(\triangle)}{=} \frac{1}{2N}\left(\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y}\right)^\top\left(\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y}\right) \tag{13}$$

- **Remarks:**
  - $\boldsymbol{X}\boldsymbol{\theta} \in \mathbb{R}^N$ is the vector containing the model predictions
  - $\|\cdot\|$ denotes the **Euclidean norm**: $\|\boldsymbol{x}\| := \sqrt{\sum_{m=1}^{M} x_m^2}$
  - In step $(\triangle)$ we have used $\|\boldsymbol{x}\|^2 = \boldsymbol{x}^\top\boldsymbol{x}$

Derivation: Rewrite the Error Function (Ctd.)

Step ❷ We use the matrix transposition rules to further rewrite the error function:

$$\begin{aligned}
\mathfrak{J}(\boldsymbol{\theta}) &= \frac{1}{2N}(\boldsymbol{X\theta}-\boldsymbol{y})^\top(\boldsymbol{X\theta}-\boldsymbol{y}) \overset{(9)}{=} \frac{1}{2N}\left((\boldsymbol{X\theta})^\top-\boldsymbol{y}^\top\right)(\boldsymbol{X\theta}-\boldsymbol{y}) \\
&= \frac{1}{2N}\left((\boldsymbol{X\theta})^\top\boldsymbol{X\theta}-(\boldsymbol{X\theta})^\top\boldsymbol{y}-\boldsymbol{y}^\top(\boldsymbol{X\theta})+\boldsymbol{y}^\top\boldsymbol{y}\right) \\
&\overset{(10)}{=} \frac{1}{2N}\left(\boldsymbol{\theta}^\top\boldsymbol{X}^\top\boldsymbol{X\theta}-2(\boldsymbol{X\theta})^\top\boldsymbol{y}+\boldsymbol{y}^\top\boldsymbol{y}\right) \\
&\overset{(10)}{=} \frac{1}{2N}\left(\boldsymbol{\theta}^\top\boldsymbol{X}^\top\boldsymbol{X\theta}-2\boldsymbol{\theta}^\top\boldsymbol{X}^\top\boldsymbol{y}+\boldsymbol{y}^\top\boldsymbol{y}\right) \qquad (14)
\end{aligned}$$

Derivation: Compute the Derivative

Step ❸ We compute the derivative of (14):

$$\frac{\partial}{\partial \boldsymbol{\theta}} \boldsymbol{\theta}^\top \boldsymbol{X}^\top \boldsymbol{X} \boldsymbol{\theta} \overset{(11)}{=} 2 \boldsymbol{X}^\top \boldsymbol{X} \boldsymbol{\theta} \tag{15}$$

$$-\frac{\partial}{\partial \boldsymbol{\theta}} 2 \boldsymbol{\theta}^\top \boldsymbol{X}^\top \boldsymbol{y} \overset{(12)}{=} -2 \boldsymbol{X}^\top \boldsymbol{y} \tag{16}$$

$$\frac{\partial}{\partial \boldsymbol{\theta}} \boldsymbol{y}^\top \boldsymbol{y} = 0 \tag{17}$$

Remark: We are allowed to apply rule (11) in equation (15) because $\boldsymbol{X}^\top \boldsymbol{X}$ is a symmetric matrix:

$$(\boldsymbol{X}^\top \boldsymbol{X})^\top \overset{(10)}{=} \boldsymbol{X}^\top (\boldsymbol{X}^\top)^\top$$

$$\overset{(8)}{=} \boldsymbol{X}^\top \boldsymbol{X}$$

Derivation: Solve for $\boldsymbol{\theta}$

Step ❹ We plug the results together to obtain the derivative:

$$\frac{\partial}{\partial \boldsymbol{\theta}}\mathfrak{J}(\boldsymbol{\theta}) = \frac{1}{2N}\left(2\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{\theta} - 2\boldsymbol{X}^\top\boldsymbol{y}\right) = \frac{1}{N}\left(\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{X}^\top\boldsymbol{y}\right) \tag{18}$$

Step ❺ We set the derivative to zero and solve for $\boldsymbol{\theta}$:

$$\begin{aligned}
\frac{\partial}{\partial \boldsymbol{\theta}}\mathfrak{J}(\boldsymbol{\theta}) \overset{!}{=} \mathbf{0} \quad &\Longleftrightarrow \quad \boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{X}^\top\boldsymbol{y} = \mathbf{0} \\
&\Longleftrightarrow \quad \boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{\theta} = \boldsymbol{X}^\top\boldsymbol{y} \\
&\Longleftrightarrow \quad \boldsymbol{\theta} = \left(\boldsymbol{X}^\top\boldsymbol{X}\right)^{-1}\boldsymbol{X}^\top\boldsymbol{y}
\end{aligned}$$

⇒ **Does $\left(\boldsymbol{X}^\top\boldsymbol{X}\right)^{-1}$ exist?**

Derivation: Check for Minimum

Step ❻ Check for a minimum:

- We have found a candidate solution
- We have not yet shown that (5) is indeed a **minimum of the error function**
- For this we consider the second-order derivative:

$$\frac{\partial^2}{\partial\boldsymbol{\theta}\partial\boldsymbol{\theta}}\mathfrak{J}(\boldsymbol{\theta}) = \frac{1}{N}\boldsymbol{X}^\top\boldsymbol{X} \tag{19}$$

- We have to show that $\boldsymbol{X}^\top\boldsymbol{X} \succ 0$ (positive definite)
- The cost function then fulfills the **second-order convexity condition** and therefore only has one global minimum

Some useful Lemmas

Lemma: $\boldsymbol{A}^\top \boldsymbol{A}$ is positive semi-definite (i. e. $\boldsymbol{A}^\top \boldsymbol{A} \succeq 0$) for all $\boldsymbol{A} \in \mathbb{R}^{D \times L}$.

Proof: Let $\boldsymbol{z} \in \mathbb{R}^L$ be an arbitrary vector. We have

$$\boldsymbol{z}^\top (\boldsymbol{A}^\top \boldsymbol{A})\boldsymbol{z} \overset{(10)}{=} (\boldsymbol{A}\boldsymbol{z})^\top (\boldsymbol{A}\boldsymbol{z}) = \|\boldsymbol{A}\boldsymbol{z}\|^2 \geqslant 0 \tag{20}$$

due to the properties of the (Euclidean) norm. ■

Some useful Lemmas (Ctd.)

Lemma ($\star$): If $\boldsymbol{A} \in \mathbb{R}^{D \times L}$ has full column-rank, then $\boldsymbol{A}^\top \boldsymbol{A}$ is positive definite, (i. e. $\boldsymbol{A}^\top \boldsymbol{A} \succ 0$).

Proof: Let $\boldsymbol{A} \in \mathbb{R}^{D \times L}$ be a matrix with full column-rank (i. e. the columns of $\boldsymbol{A}$ are linearly independent), and $\boldsymbol{z} \in \mathbb{R}^L \backslash \{\boldsymbol{0}\}$ be a non-zero vector. The product $\boldsymbol{Az}$ is a non-trivial linear combination of the columns of $\boldsymbol{A}$, because $\boldsymbol{A}$ has full column-rank ($\operatorname{rank}(\boldsymbol{A}) = L$). Therefore, $\boldsymbol{Az} \neq \boldsymbol{0}$. We obtain

$$\boldsymbol{z}^\top \boldsymbol{A}^\top \boldsymbol{Az} \overset{(10)}{=} (\boldsymbol{Az})^\top (\boldsymbol{Az}) = \|\boldsymbol{Az}\|^2 > 0 \tag{21}$$

due to the properties of the (EUCLIDean) norm. We conclude that $\boldsymbol{A}^\top \boldsymbol{A}$ is positive definite. ■

Some useful Lemmas (Ctd.)

Lemma ($\star\star$): A positive definite matrix $\boldsymbol{A} \in \mathbb{R}^{L \times L}$ is invertible.

Proof: Suppose $\boldsymbol{A}$ was not invertible. This implies the existence of a non-zero vector $\boldsymbol{z} \in \mathbb{R}^{L} \backslash \{\boldsymbol{0}\}$ for which we have $\boldsymbol{A}\boldsymbol{z} = \boldsymbol{0}$. Then

$$\boldsymbol{z}^{\top}\boldsymbol{A}\boldsymbol{z} = \boldsymbol{z}^{\top}\boldsymbol{0} = 0. \tag{22}$$

This contradicts the positive definiteness of $\boldsymbol{A}$. Therefore, $\boldsymbol{A}$ must be invertible. ■

Summary: Normal Equations

Corollary:

If the design matrix $\boldsymbol{X} \in \mathbb{R}^{N \times (M+1)}$ has **full column-rank** (i. e. the features are linearly independent), then $\boldsymbol{X}^\top \boldsymbol{X} \in \mathbb{R}^{(M+1) \times (M+1)}$ is **positive definite** due to lemma ($\star$). As a positive definite matrix, the inverse of $\boldsymbol{X}^\top \boldsymbol{X}$ is **guaranteed to exist** due to lemma ($\star\star$). Hence

$$\boldsymbol{\theta}^\star := \left(\boldsymbol{X}^\top \boldsymbol{X}\right)^{-1} \boldsymbol{X}^\top \boldsymbol{y}$$

is well-defined, and $\boldsymbol{\theta}^\star$ is indeed the **global minimum of the least squares error function** given in equation (3).

Geometric Interpretation of Least Squares

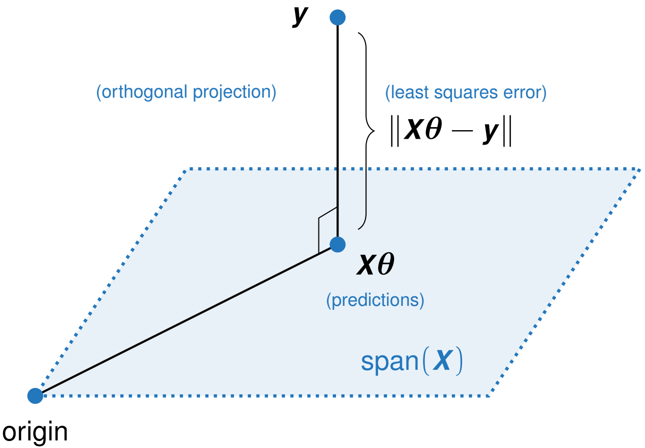
- We want to find a solution to the system of equations

$$\theta_0 \widetilde{\boldsymbol{x}}^0 + \theta_1 \widetilde{\boldsymbol{x}}^1 + \ldots + \theta_M \widetilde{\boldsymbol{x}}^M = \boldsymbol{y} \tag{23}$$

- $\boldsymbol{y}$ is a **linear combination** of the columns $\widetilde{\boldsymbol{x}}^m$ $(0 \leqslant m \leqslant M)$ of $\boldsymbol{X}$
- However, we will **not find a solution** unless the data is perfectly linear

Goal: Find the point in the span of the matrix $\boldsymbol{X}$ (space spanned by the column vectors of $\boldsymbol{X}$ / column space) that is closest to $\boldsymbol{y}$!

Geometric Interpretation of Least Squares (Ctd.)


$\boldsymbol{y}$
(orthogonal projection)
(least squares error)
$\|\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y}\|$
$\boldsymbol{X}\boldsymbol{\theta}$
(predictions)
$\mathrm{span}(\boldsymbol{X})$
origin

Geometric Interpretation of Least Squares (Ctd.)

- The point closest to $\boldsymbol{y}$ is the **orthogonal projection** of $\boldsymbol{y}$ onto the span of $\boldsymbol{X}$
- Any other point in the span of $\boldsymbol{X}$ has a larger EUCLIDean distance to $\boldsymbol{y}$ and therefore cannot be the opitmal solution

The geometry of least squares also explains the term ‘normal equations’:

- A normal line is a line perpendicular to another line or subspace
- The normal equations given by (5) compute $\boldsymbol{\theta}$ such that the residuals are orthogonal to the span of $\boldsymbol{X}$

Important

Problems with Matrix Inversion?

Problem 1: The design matrix X does not have full column-rank due to linearly dependent features, e. g. size in m^2 and size in feet^2 (constant conversion factor)

Solution 1: Delete correlated features from the dataset

Remark: Earlier we have seen that the design matrix should have full column-rank to guarantee that the inverse exists

Problems with Matrix Inversion? (Ctd.)

Problem 2:

The dataset contains too many features, especially problematic is the case when there are more features than data points, i. e. $M > N$

Solution 2:

Delete features (e. g. using PCA), or add more training examples

Important

Problems with Matrix Inversion? (Ctd.)

Problem 3: Correct matrix inversion could be prevented by numerical instabilities

Solution 3: Add a regularization term (we shall introduce this concept later)

Remark: Even if the inverse exists mathematically, computing it on a computer might be difficult because the set of floating point numbers available to the computer is finite *(although working with singular matrices is numerically much more challenging)*

Important

Problems with Matrix Inversion? (Ctd.)

Problem 4: The computation of the inverse is too expensive

Solution 4: Use numerical methods, e. g. gradient descent which does not have to compute the inverse

Remark: Matrix inversion has a time complexity of $\mathcal{O}(M^3)$, i. e. the time needed to invert an $(M \times M)$-matrix grows cubically with M (here: number of features)

Important

Gradient Descent

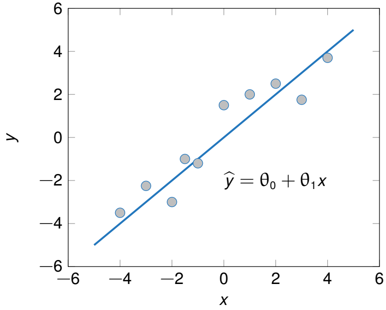
- We want to minimize a continuous function $\mathfrak{J} : \mathbb{R}^{M+1} \to \mathbb{R}$: $\min_{\boldsymbol{\theta}} \mathfrak{J}(\boldsymbol{\theta})$
- **Gradient Descent:** Update the parameters iteratively:

$$\boldsymbol{\theta}^{t+1} \longleftarrow \boldsymbol{\theta}^{t} - \alpha \nabla_{\boldsymbol{\theta}} \mathfrak{J}(\boldsymbol{\theta}) \tag{24}$$

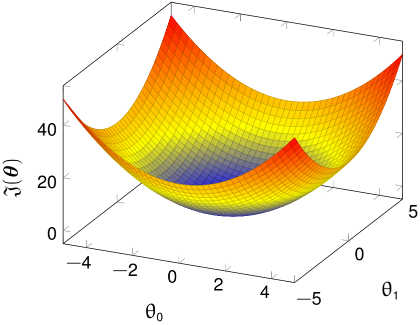
- **Legend:**
  - $\boldsymbol{\theta}^t$ denotes the vector of model parameters at timestep t
  - $\alpha \in (0, 1)$ denotes the **learning rate**
  - $\nabla_{\boldsymbol{\theta}} \mathfrak{J}(\boldsymbol{\theta}) := \left(\frac{\partial \mathfrak{J}(\boldsymbol{\theta})}{\partial \theta_0}, \frac{\partial \mathfrak{J}(\boldsymbol{\theta})}{\partial \theta_1}, \dots, \frac{\partial \mathfrak{J}(\boldsymbol{\theta})}{\partial \theta_M} \right)^\top$ is the gradient of $\mathfrak{J}(\boldsymbol{\theta})$

Data Input Space vs. Hypothesis Space

Data input space



Hypothesis space $\mathcal{H}$

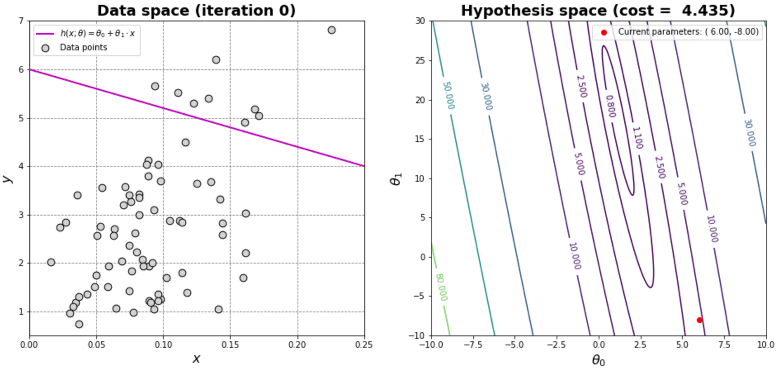


Data Input Space vs. Hypothesis Space (Ctd.)

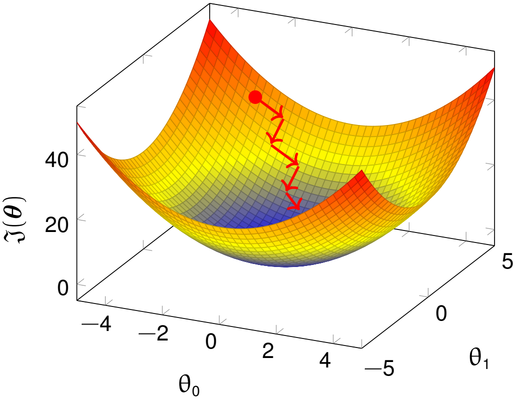
- **Data input space**
  - Is determined by the *M* **attributes** of the dataset $x_1, x_2, \dots, x_M$
  - It is often high-dimensional
- **Hypothesis space** $\mathcal{H}$
  - Is determined by the $M + 1$ **parameters** of the model $\theta_0, \theta_1, \dots, \theta_M$
  - Each point in the hypothesis space corresponds to a **specific assignment of model parameters**
  - The error function gives information about how good this assignment is
  - **Gradient descent is applied in the hypothesis space** $\mathcal{H}$

Important

Data Input Space vs. Hypothesis Space (Ctd.)



Visualization of Gradient Descent in 3 Dimensions



Important

Versions of Gradient Descent

- Assume some training data $\mathcal{D} := \left\{ \left(\boldsymbol{x}^n, y_n \right) \right\}_{n=1}^{N}$
- The squared error of a **single** example is given by:

$$\ell^{\text{LS}}\left(\widehat{y}, y\right) := \frac{1}{2}\left(\widehat{y} - y\right)^2$$

- Our objective is to minimize the **total error**:

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^{M+1}} \mathfrak{J}(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta} \in \mathbb{R}^{M+1}} \sum_{n=1}^{N} \ell^{\text{LS}}\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^n\right), y_n\right)$$

Important

Batch Gradient Descent

Three versions of gradient descent:

1. **Batch gradient descent**
   - Compute the gradient based on **ALL** N training data points

$$\boldsymbol{\theta}^{t+1} \longleftarrow \boldsymbol{\theta}^{t} - \alpha \sum_{n=1}^{N} \nabla \ell^{\mathrm{LS}}\big(h(\boldsymbol{x}^{n}; \boldsymbol{\theta}^{t}), y_n\big) \tag{25}$$

   - Most accurate, but may be costly depending on the size of the dataset!
2. Stochastic gradient descent
3. Mini-batch gradient descent

Important

Stochastic Gradient Descent

Three versions of gradient descent:

1. Batch gradient descent
2. **Stochastic gradient descent**
   - Compute the gradient based on a **SINGLE** data point $(\boldsymbol{x}, y)$

$$\boldsymbol{\theta}^{t+1} \longleftarrow \boldsymbol{\theta}^{t} - \alpha \nabla \ell^{\text{LS}}\left(h(\boldsymbol{x}; \boldsymbol{\theta}^{t}), y\right) \tag{26}$$

   - **Pick training examples randomly, and not sequentially!**
   - Efficient, but inaccurate!
3. Mini-batch gradient descent

Important

Mini-Batch Gradient Descent

Three versions of gradient descent:

1. Batch gradient descent
2. Stochastic gradient descent
3. **Mini-batch gradient descent**
   - Compute the gradient based on a mini-batch comprising N_b examples

$$\boldsymbol{\theta}^{t+1} \longleftarrow \boldsymbol{\theta}^{t} - \alpha \sum_{n=1}^{N_b} \nabla \ell^{\mathrm{LS}}\big(h(\boldsymbol{x}^{n}; \boldsymbol{\theta}^{t}), y_n\big) \tag{27}$$

   - Good trade-off between batch and stochastic gradient descent

Stochastic Gradient of the Least Squares Error Function

- The **partial derivatives** of $\mathfrak{J}$ are:

$$\frac{\partial}{\partial \theta_m}\mathfrak{J}(\boldsymbol{\theta}) = \frac{\partial}{\partial \theta_m}\frac{1}{2}\big(h_{\boldsymbol{\theta}}(\boldsymbol{x}) - y\big)^2 = 2 \cdot \frac{1}{2}\big(h_{\boldsymbol{\theta}}(\boldsymbol{x}) - y\big) \cdot \frac{\partial}{\partial \theta_m}\big(h_{\boldsymbol{\theta}}(\boldsymbol{x}) - y\big)$$

$$= \big(h_{\boldsymbol{\theta}}(\boldsymbol{x}) - y\big) \cdot \frac{\partial}{\partial \theta_m}\big(\theta_0 x_0 + \cdots + \theta_M x_M - y\big) = \boxed{\big(h_{\boldsymbol{\theta}}(\boldsymbol{x}) - y\big) x_m}$$

- The **vectorized version** is given by:

$$\frac{\partial}{\partial \boldsymbol{\theta}}\mathfrak{J}(\boldsymbol{\theta}) = \big(h_{\boldsymbol{\theta}}(\boldsymbol{x}) - y\big)\boldsymbol{x} \tag{28}$$

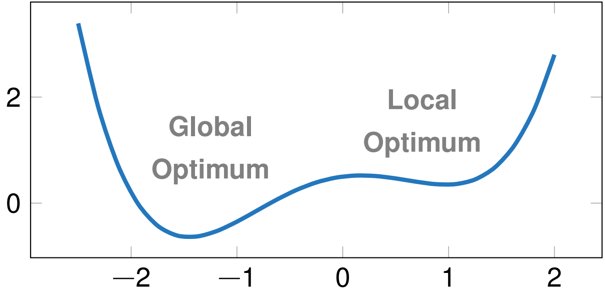
Batch Gradient of the Least Squares Error Function

We have computed the batch gradient already in equation (18) when deriving the normal equations: *(we ignore the factor $^1/_N$ here)*

$$\frac{\partial}{\partial \boldsymbol{\theta}} \mathfrak{J}(\boldsymbol{\theta}) = \boldsymbol{X}^\top \boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{X}^\top \boldsymbol{y} = \boldsymbol{X}^\top \left(\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y} \right) \tag{29}$$

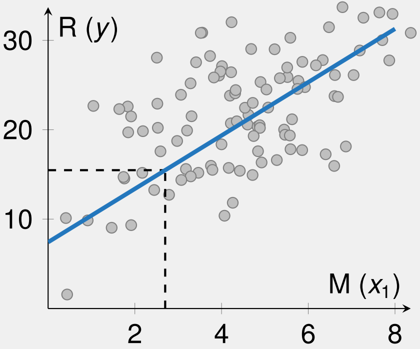
Remark: We can use the same formula for mini-batch gradient descent by replacing $\boldsymbol{X}$ with $\boldsymbol{X}_b$ – the design matrix comprising only the mini-batch examples, and $\boldsymbol{y}$ with $\boldsymbol{y}_b$ – the respective labels

Disadvantage of Gradient Descent



Question: Why is this not a problem here?

Solving the introductory Example



$$h_\theta(\boldsymbol{x}) = 7.4218 + 2.9827 \cdot x_1$$

- **Parameters:**
  - $\theta_0 \approx 7.4218$
  - $\theta_1 \approx 2.9827$
- **Associated error:** $\mathfrak{J}(\boldsymbol{\theta}) \approx 446.9584$
- **Prediction:** $h_\theta(2.7) = \mathbf{15.4750}$

A probabilistic View

- **Assumption 1:** The target values y and the inputs $\boldsymbol{x}$ are related via the equation

$$y = h_{\boldsymbol{\theta}}(\boldsymbol{x}) + \varepsilon = \boldsymbol{\theta}^{\top}\boldsymbol{x} + \varepsilon, \tag{30}$$

  where ε is an error term which captures unmodeled effects or noise in the data

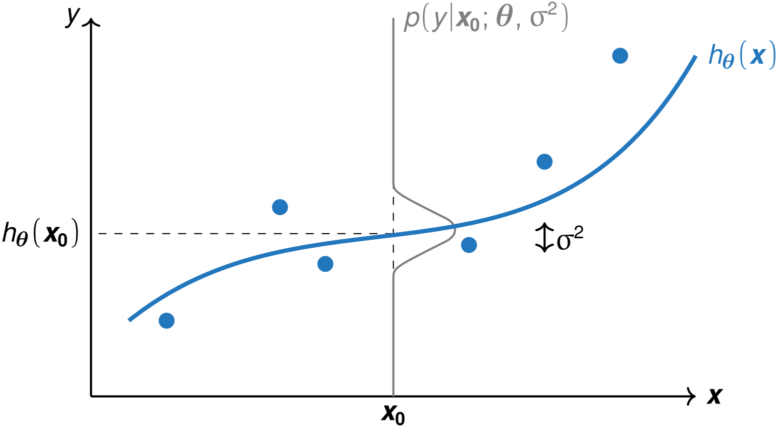
- **Assumption 2:** The noise ε is a zero mean Gaussian random variable

$$\varepsilon \sim \mathcal{N}(0, \sigma^2) \tag{31}$$

- We consider y a random variable which is distributed according to

$$p(y|\boldsymbol{x}; \boldsymbol{\theta}, \sigma^2) = \mathcal{N}(y; h_{\boldsymbol{\theta}}(\boldsymbol{x}), \sigma^2) \tag{32}$$

A probabilistic View (Ctd.)


y
$p(y|\boldsymbol{x}_0; \boldsymbol{\theta}, \sigma^2)$
$h_{\boldsymbol{\theta}}(\boldsymbol{x})$
$h_{\boldsymbol{\theta}}(\boldsymbol{x}_0)$
$\updownarrow \sigma^2$
$\boldsymbol{x}$
$\boldsymbol{x}_0$

Likelihood Function for Regression

- We are given a dataset $\mathcal{D} := \{(\boldsymbol{x}^n, y_n)\}_{n=1}^N$
- The (conditional) **likelihood** is given by:

$$\begin{aligned} p(\boldsymbol{y}|\boldsymbol{X}; \boldsymbol{\theta}, \sigma^2) &= \prod_{n=1}^{N} \mathcal{N}(y_n; h_{\boldsymbol{\theta}}(\boldsymbol{x}^n), \sigma^2) \\ &= \prod_{n=1}^{N} \mathcal{N}(y_n; \boldsymbol{\theta}^\top \boldsymbol{x}^n, \sigma^2) \end{aligned} \tag{33}$$

Important

Likelihood Function for Regression (Ctd.)

- The **log-likelihood** is then given by *(we have computed this earlier already)*:

$$\mathcal{L}(\boldsymbol{\theta}) := \log p(\boldsymbol{y}|\boldsymbol{X}; \boldsymbol{\theta}, \sigma^2) = -\frac{N}{2}\log(2\pi\sigma^2) - \frac{1}{\sigma^2} \cdot \underbrace{\frac{1}{2}\sum_{n=1}^{N}\left(y_n - \boldsymbol{\theta}^\top \boldsymbol{x}^n\right)^2}_{\text{least squares error}} \tag{34}$$

- We have to minimize the least squares error to maximize the likelihood!

When minimizing the squared error we implicitly assume GAUSSian noise!

Important

The Maximum Likelihood Solution to Regression

- Optimization of the log-likelihood function gives:

$$\boldsymbol{\theta}^{\mathrm{ML}} := (\boldsymbol{X}^{\top}\boldsymbol{X})^{-1}\boldsymbol{X}^{\top}\boldsymbol{y} \tag{35}$$

$$\left(\sigma^{2}\right)^{\mathrm{ML}} := \frac{1}{N}\sum_{n=1}^{N}\left(y_n - \left(\boldsymbol{\theta}^{\mathrm{ML}}\right)^{\top}\boldsymbol{x}^{n}\right)^{2} \tag{36}$$

- The probabilistic interpretation of linear regression allows us **to quantify the (global) uncertainty** of the model

Section:

Basis Function Regression

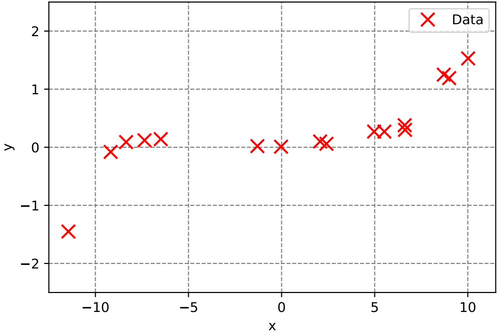
General Idea

Polynomial Basis Functions

Radial Basis Functions

What if the Data is non-linear?

- So far we have fitted straight lines
- **What if the data is not linear...?**



Important

Basis Functions

- Remember: **‘When stuck switch to a different perspective’**
- We add **higher-order** features by using **basis functions** φ:

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) := \sum_{p=0}^{P} \theta_p \varphi_p(\boldsymbol{x}) \tag{37}$$

- Commonly used basis functions:
    - **Linear:** $\varphi_0(\boldsymbol{x}) = 1$ and $\varphi_1(\boldsymbol{x}) = x_1, \dots, \varphi_P(\boldsymbol{x}) = x_M$ *(not very interesting...)*
    - **Polynomial** (see below)
    - **Radial basis functions** (see below)

New Design Matrix

By applying the basis functions to $\boldsymbol{X}$, we get a new design matrix $\boldsymbol{\Phi}$:

$$\boldsymbol{\Phi} := \begin{pmatrix} \varphi_0(\boldsymbol{x}^1) & \varphi_1(\boldsymbol{x}^1) & \varphi_2(\boldsymbol{x}^1) & \dots & \varphi_P(\boldsymbol{x}^1) \\ \varphi_0(\boldsymbol{x}^2) & \varphi_1(\boldsymbol{x}^2) & \varphi_2(\boldsymbol{x}^2) & \dots & \varphi_P(\boldsymbol{x}^2) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \varphi_0(\boldsymbol{x}^N) & \varphi_1(\boldsymbol{x}^N) & \varphi_2(\boldsymbol{x}^N) & \dots & \varphi_P(\boldsymbol{x}^N) \end{pmatrix} \tag{38}$$

The model is still linear in the parameters, so we can still use the same algorithm as before. This is still linear regression (!!!)

Important

Basis Functions: Polynomial Basis Functions

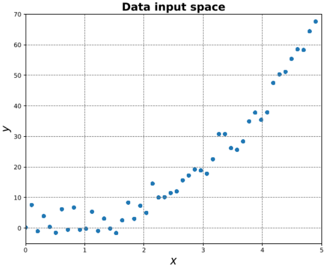
- Let us assume a 1-dimensional dataset ($M = 1$) for now
- A quite frequently used basis function is the **polynomial basis**

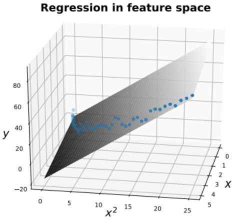
$$\varphi_0(x) := 1, \; \varphi_p(x) := x^p$$

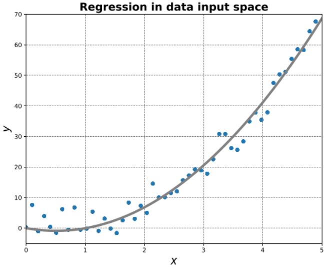
$$h_{\boldsymbol{\theta}}(x) := \sum_{p=0}^{P} \theta_p \varphi_p(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \ldots + \theta_P x^P$$

- P is the degree of the polynomial
- For higher-dimensional datasets we can also include cross-terms, e. g. $\varphi_p(\boldsymbol{x}) = x_\ell^2 x_k$

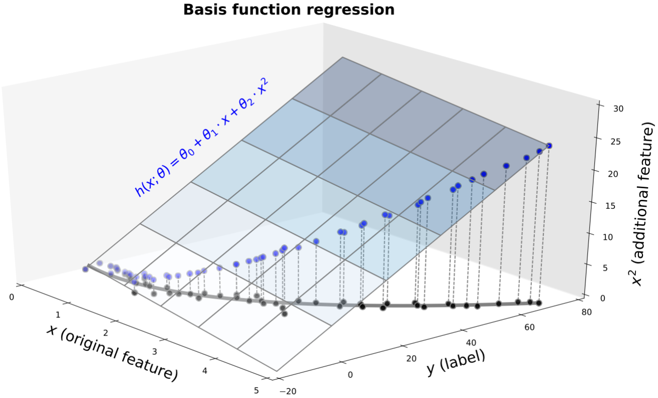
It is still linear!







It is still linear! (Ctd.)



Basis Functions: Radial Basis Functions (RBFs)

- Another possible choice of basis function: **Radial basis functions**

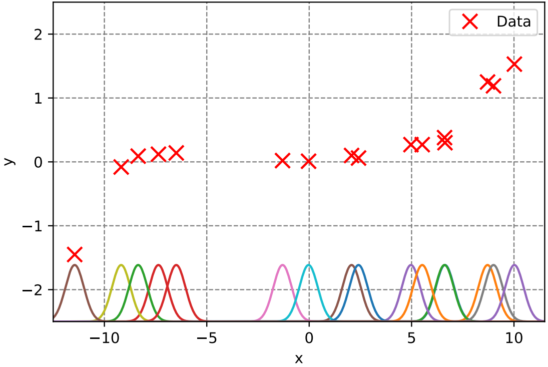
$$\varphi_0(x) := 1 \tag{39}$$

$$\varphi_p(x) := \exp\left(-{}^1\!/\!{}_2 \|x - z_p\|^2 / 2\sigma^2\right) \tag{40}$$

- $\{z_p\}$ are the centers of the radial basis functions, σ^2 is the scale
- P denotes the number of centers / number of radial basis functions
- Often we take each data point as a center, so $P = N$

  (but in general we are free to put the centers wherever we want)

Radial Basis Functions (Ctd.)



Section:

Regularization Techniques

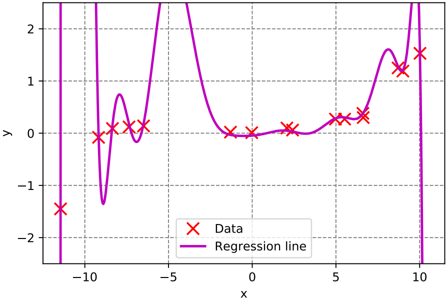
Underfitting and Overfitting
L1 and L2 Regularization

Important

The Danger of too expressive Models...

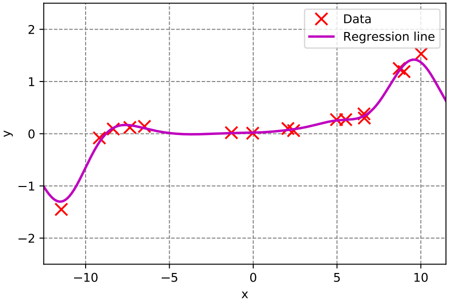
Polynomial of degree $P = 16$

(☠ severe overfitting ☠)



RBF with $\sigma^2 = 1.00$, $P = N$

(about right)

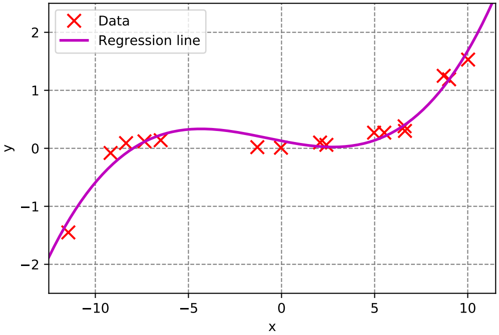


Overfitting vs. Underfitting

- **Underfitting**
  - The model is not complex enough to fit the data well ⇒ **High bias**
  - Make the model more complex; adding new examples **does not help**
- **Overfitting**
  - The model predicts the training data perfectly
  - But it **fails to generalize** to unseen instances ⇒ **High variance**
  - Decrease the degree of freedom, or add more training examples
  - Also: Try **regularization**
- **Bias-Variance trade-off**

First Solution: Smaller Degree

One solution: Use a **smaller degree** (here: $P = 3$)



Second Solution: Regularization

- Enrich the cost function $\mathfrak{J}(\boldsymbol{\theta})$ with a **regularization term**
- This helps **prevent overfitting** and results in a smoother regression curve

L1 regularization:

$$\widetilde{\mathfrak{J}}(\boldsymbol{\theta}) := \mathfrak{J}(\boldsymbol{\theta}) + \lambda|\boldsymbol{\theta}|$$

$$|\boldsymbol{\theta}| := \sum_{p=1}^{P} |\theta_p|$$

L2 regularization:

$$\widetilde{\mathfrak{J}}(\boldsymbol{\theta}) := \mathfrak{J}(\boldsymbol{\theta}) + \lambda||\boldsymbol{\theta}||^2$$

$$||\boldsymbol{\theta}||^2 := \sum_{p=1}^{P} \theta_p^2$$

($\lambda \geqslant 0$ controls the **degree of regularization**)

Regularization visualized

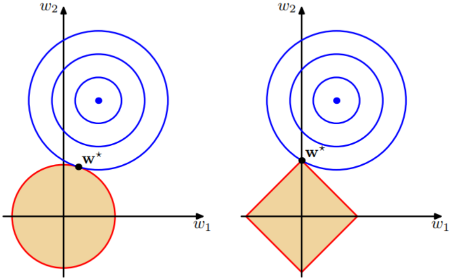
- L1-Regularization

  $\Rightarrow$ **Lasso regression**

  (**l**east **a**bsolute **s**hrinkage and **s**election **o**perator)
- L2-Regularization

  $\Rightarrow$ **Ridge regression**

  (TIKHONOV regularization)
- The combination of both is called **elastic net**
- Image on the right: $\boldsymbol{w} \equiv \boldsymbol{\theta}$



cf. [BISHOP.2006], page 146, left: L2, right: L1

Incorporating Regularization

- Normal equations with regularization: **Ridge regression**

$$\boldsymbol{\theta}^\star := \left(\boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \lambda \boldsymbol{I}\right)^{-1} \boldsymbol{\Phi}^\top \boldsymbol{y} \tag{41}$$

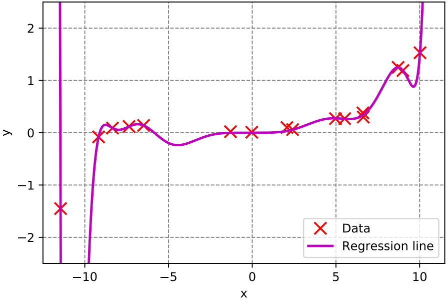
- Regularization also helps **overcome numerical issues** (matrix inversion!)
- Regularized least squares error gradient:

$$\frac{\partial}{\partial \theta_p} \mathfrak{J}(\boldsymbol{\theta}) = \left(h_{\boldsymbol{\theta}}(\boldsymbol{\varphi}(\boldsymbol{x})) - y\right) \varphi_p(\boldsymbol{x}) + \lambda \theta_p \tag{42}$$

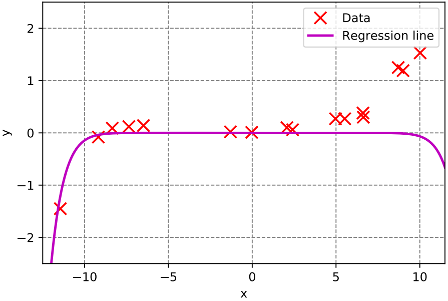
where $\boldsymbol{\varphi}(\boldsymbol{x}) := \left(\varphi_0(\boldsymbol{x}), \ldots, \varphi_P(\boldsymbol{x})\right)^\top$

Polynomial Regression with Regularization

At least better



Way too much regularization



Section:

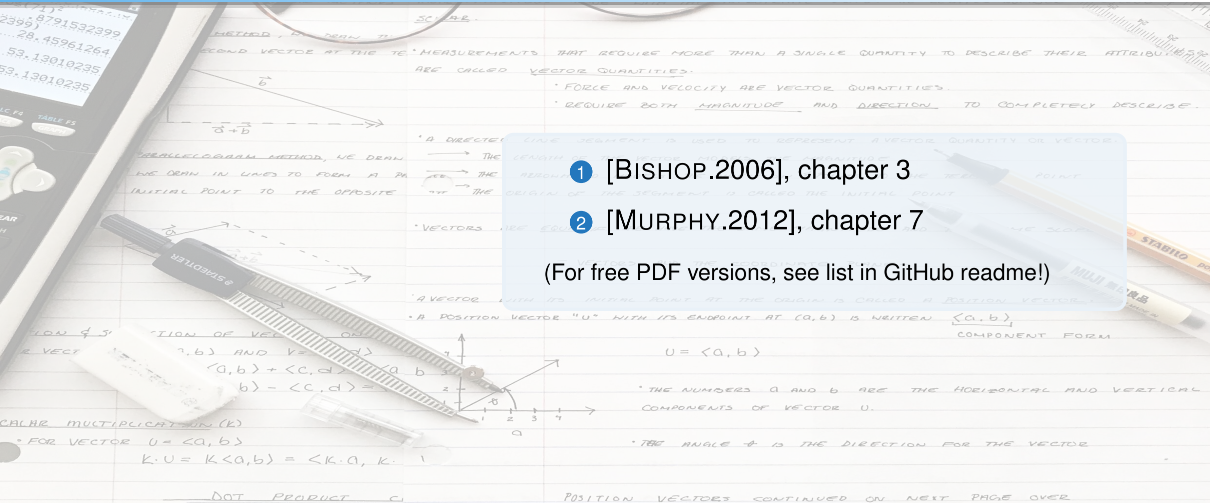
Wrap-Up

Summary
Recommended Literature
Self-Test Questions
Lecture Outlook

Summary

- In regression we predict **continuous target variables**
- The algorithm minimizes the **(mean) squared error**
- **Two approaches:**
  1. Normal equations
  2. (Batch / stochastic / mini-batch) gradient descent
- Geometric interpretation: Predictions are the orthogonal projection of the label vector onto the span of the design matrix
- Use **basis functions** to fit non-linear regression lines
- **Regularization** is important (especially when using basis functions)

Recommended Literature



1. [Bishop.2006], chapter 3
2. [Murphy.2012], chapter 7

(For free PDF versions, see list in GitHub readme!)

Self-Test Questions

1. What is the goal of regression?
2. What can you do if matrix inversion fails for the normal equations?
3. What is a suitable cost function for regression? Where does it come from?
4. Does gradient descent give the exact solution?
5. Which versions of gradient descent do you know?
6. What are basis functions? Why use them? State some examples.
7. What is overfitting and underfitting?
8. What is regularization? Why and when should you apply it?

What’s next...?

I Machine Learning Introduction

II Optimization Techniques

III Bayesian Decision Theory

IV Non-parametric Density Estimation

V Probabilistic Graphical Models

VI Linear Regression

● **VII** Logistic Regression

VIII Deep Learning

IX Evaluation

X Decision Trees

XI Support Vector Machines

XII Clustering

XIII Principal Component Analysis

XIV Reinforcement Learning

XV Advanced Regression

Thank you very much for the attention!

* * * **Artificial Intelligence and Machine Learning** * * *

Topic: Linear Regression and Maximum Likelihood Regression

Term: Summer term 2025

Contact:
Daniel Wehner, M.Sc.
SAP SE / DHBW Mannheim
daniel.wehner@sap.com

Do you have any questions?