*** Applied Machine Learning Fundamentals ***
Regression

Daniel Wehner, M.Sc.

SAP SE / DHBW Mannheim

Winter term 2023/2024

Find all slides on GitHub (DaWe1992/Applied_ML_Fundamentals)

Lecture Overview

Unit I	Machine Learning Introduction
Unit II	Mathematical Foundations
Unit III	Bayesian Decision Theory
Unit IV	**Regression**
Unit V	Classification I
Unit VI	Evaluation
Unit VII	Classification II
Unit VIII	Clustering
Unit IX	Dimensionality Reduction

Agenda for this Unit

Section:

Introduction

What is Regression?
Least Squares Error Function

Linear Regression Overview

Type of target variable:	Continuous
Type of training information:	Supervised learning
Example availability:	Batch learning

Algorithm sketch: Given the training data $\mathcal{D}$, the algorithm derives a function (model function / hypothesis) of the type

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) := \theta_0 + \theta_1 x_1 + \cdots + \theta_m x_m \qquad (\boldsymbol{x} \in \mathbb{R}^m, \boldsymbol{\theta} \in \mathbb{R}^{m+1}) \tag{1}$$

from the data. $\boldsymbol{\theta}$ is the parameter vector containing the coefficients to be estimated by the regression algorithm. Once $\boldsymbol{\theta}$ is learned, it can be used for prediction.

Example Dataset: Revenues

- Find a linear function of the form:

$$h_{\theta}(x) := \theta_0 + \theta_1 x_1 + \cdots + \theta_m x_m$$

- Usually we set: $x_0 := 1$:

$$\widehat{x} \in \mathbb{R}^{m+1} := [1, x]^{\intercal}$$

$$h_{\theta}(\widehat{x}) = \sum_{j=0}^{m} \theta_j x_j = \theta^{\intercal} \widehat{x}$$

Error Function for Regression

- We need an error function $\mathcal{J}(\boldsymbol{\theta})$ in order to know how well the function fits:

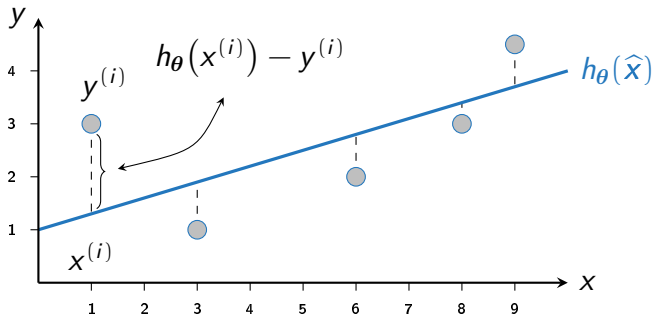
$$\mathcal{J}(\boldsymbol{\theta}) := \frac{1}{2n} \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}}\left(\widehat{\boldsymbol{x}}^{(i)}\right) - y^{(i)} \right)^2 \tag{2}$$

- We want to minimize $\mathcal{J}(\boldsymbol{\theta})$ to obtain the optimal model parameters $\boldsymbol{\theta}^\star$:

$$\boldsymbol{\theta}^\star := \arg\min_{\boldsymbol{\theta}} \frac{1}{2n} \sum_{i=1}^{n} \left(h_{\boldsymbol{\theta}}\left(\widehat{\boldsymbol{x}}^{(i)}\right) - y^{(i)} \right)^2$$

- This is **ordinary least squares (OLS)**

Error Function Intuition



Question: Why do we consider the square in the error function?

Origin of Least Squares

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.

Section:

Solutions to Regression

Closed-Form Solutions and Normal Equation
Geometric Interpretation of Least Squares
Gradient Descent
Probabilistic Interpretation of linear Regression

Closed-Form Solutions

- Usual approach (for two parameters): Calculate θ_0 and θ_1 according to (where $\overline{x} = 1/n \sum_{i=1}^{n} \boldsymbol{x}^{(i)}$ is the sample set mean)

$$\theta_0^\star := \overline{y} - \theta_1 \overline{x}, \qquad \theta_1^\star := \frac{\sum_{i=1}^{n} \left(x^{(i)} - \overline{x}\right) \cdot \left(y^{(i)} - \overline{y}\right)}{\sum_{i=1}^{n} \left(x^{(i)} - \overline{x}\right)^2} \tag{3}$$

- 'Normal equation(s)' (scales to arbitrary dimensions):

$$\boldsymbol{\theta}^\star := \left(\widehat{\boldsymbol{X}}^\intercal \widehat{\boldsymbol{X}}\right)^{-1} \widehat{\boldsymbol{X}}^\intercal \boldsymbol{y} \tag{4}$$

$\widehat{\boldsymbol{X}}$ is called 'design matrix' or 'regressor matrix'

Design Matrix / Regressor Matrix

- The design matrix $\widehat{\boldsymbol{X}} \in \mathbb{R}^{n \times (m+1)}$ looks as follows:

$$\widehat{\boldsymbol{X}} := \begin{pmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \cdots & x_m^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & \cdots & x_m^{(2)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(n)} & x_2^{(n)} & \cdots & x_m^{(n)} \end{pmatrix} \tag{5}$$

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

$$\boldsymbol{y} := \left(y^{(1)}, y^{(2)}, y^{(3)}, \ldots, y^{(n)}\right)^\mathsf{T} \tag{6}$$

Notation: In the following we use the symbol $\boldsymbol{X}$ to denote $\widehat{\boldsymbol{X}}$ defined in (5)

Derivation of the Normal Equation

In the derivation of equation (4) we will need the following rules:

Matrix transposition rules

$$(\boldsymbol{A}^\intercal)^\intercal = \boldsymbol{A} \tag{7}$$

$$(\boldsymbol{A} + \boldsymbol{B})^\intercal = \boldsymbol{A}^\intercal + \boldsymbol{B}^\intercal \tag{8}$$

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

Vector derivatives

$$\frac{\partial}{\partial \boldsymbol{x}} \boldsymbol{x}^\intercal \boldsymbol{A} \boldsymbol{x} = 2\boldsymbol{A}\boldsymbol{x} \tag{10}$$

$$\frac{\partial}{\partial \boldsymbol{x}} \boldsymbol{a}^\intercal \boldsymbol{x} = \boldsymbol{a} \tag{11}$$

Equation (10) only holds true if $\boldsymbol{A}$ is a symmetric matrix!

Derivation of the Normal Equation (Ctd.)

Step ❶:

- We rewrite the least squares error function (2) in matrix/vector notation:

$$\mathcal{J}(\boldsymbol{\theta}) = \frac{1}{2n}\|\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y}\|^2 \overset{(\triangle)}{=} \frac{1}{2n}(\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y})^\intercal(\boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{y}) \tag{12}$$

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

Derivation of the Normal Equation (Ctd.)

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

$$\begin{aligned}\mathcal{J}(\boldsymbol{\theta}) &= \frac{1}{2n}(\boldsymbol{X}\boldsymbol{\theta}-\boldsymbol{y})^\intercal(\boldsymbol{X}\boldsymbol{\theta}-\boldsymbol{y}) \overset{(8)}{=} \frac{1}{2n}((\boldsymbol{X}\boldsymbol{\theta})^\intercal-\boldsymbol{y}^\intercal)(\boldsymbol{X}\boldsymbol{\theta}-\boldsymbol{y}) \\ &= \frac{1}{2n}((\boldsymbol{X}\boldsymbol{\theta})^\intercal\boldsymbol{X}\boldsymbol{\theta}-(\boldsymbol{X}\boldsymbol{\theta})^\intercal\boldsymbol{y}-\boldsymbol{y}^\intercal(\boldsymbol{X}\boldsymbol{\theta})+\boldsymbol{y}^\intercal\boldsymbol{y}) \\ &\overset{(9)}{=} \frac{1}{2n}(\boldsymbol{\theta}^\intercal\boldsymbol{X}^\intercal\boldsymbol{X}\boldsymbol{\theta}-2(\boldsymbol{X}\boldsymbol{\theta})^\intercal\boldsymbol{y}+\boldsymbol{y}^\intercal\boldsymbol{y}) \\ &\overset{(9)}{=} \frac{1}{2n}(\boldsymbol{\theta}^\intercal\boldsymbol{X}^\intercal\boldsymbol{X}\boldsymbol{\theta}-2\boldsymbol{\theta}^\intercal\boldsymbol{X}^\intercal\boldsymbol{y}+\boldsymbol{y}^\intercal\boldsymbol{y}) \qquad (13)\end{aligned}$$

Derivation of the Normal Equation (Ctd.)

Step ❸: We compute the derivative of (13) piece-wise:
(This is allowed due to the sum rule)

$$\frac{\partial}{\partial \boldsymbol{\theta}} \boldsymbol{\theta}^\mathsf{T} \boldsymbol{X}^\mathsf{T} \boldsymbol{X} \boldsymbol{\theta} \overset{(10)}{=} 2 \boldsymbol{X}^\mathsf{T} \boldsymbol{X} \boldsymbol{\theta} \qquad [\boldsymbol{X}^\mathsf{T}\boldsymbol{X} \text{ is symmetric: } (\boldsymbol{X}^\mathsf{T}\boldsymbol{X})^\mathsf{T} \overset{(9)}{=} \boldsymbol{X}^\mathsf{T}(\boldsymbol{X}^\mathsf{T})^\mathsf{T} \overset{(7)}{=} \boldsymbol{X}^\mathsf{T}\boldsymbol{X}]$$

$$\frac{\partial}{\partial \boldsymbol{\theta}} - 2\boldsymbol{\theta}^\mathsf{T} \boldsymbol{X}^\mathsf{T} y \overset{(11)}{=} -2 \boldsymbol{X}^\mathsf{T} y$$

$$\frac{\partial}{\partial \boldsymbol{\theta}} y^\mathsf{T} y = 0$$

Derivation of the Normal Equation (Ctd.)

Step ❹: Plug everything together, set the derivative to zero and solve:

$$\frac{\partial}{\partial \boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}) = \frac{1}{2n}\left(2\boldsymbol{X}^\intercal \boldsymbol{X}\boldsymbol{\theta} - 2\boldsymbol{X}^\intercal y\right) = \frac{1}{n}\left(\boldsymbol{X}^\intercal \boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{X}^\intercal y\right) \stackrel{!}{=} \mathbf{0}$$

$$\iff \quad \boldsymbol{X}^\intercal \boldsymbol{X}\boldsymbol{\theta} - \boldsymbol{X}^\intercal y = \mathbf{0}$$

$$\iff \quad \boldsymbol{X}^\intercal \boldsymbol{X}\boldsymbol{\theta} = \boldsymbol{X}^\intercal y$$

[If $\boldsymbol{X}^\intercal \boldsymbol{X}$ is invertible]

$$\iff \quad \boxed{\left(\boldsymbol{X}^\intercal \boldsymbol{X}\right)^{-1} \boldsymbol{X}^\intercal y} = \boldsymbol{\theta}$$

Derivation of the Normal Equation (Ctd.)

- We have not yet shown that (4) is a minimum of the error function
- For this we consider the second-order derivative:

$$\frac{\partial^2}{(\partial \boldsymbol{\theta})^2} \mathcal{J}(\boldsymbol{\theta}) = \frac{1}{n} \boldsymbol{X}^\intercal \boldsymbol{X} \tag{14}$$

- We notice that $\boldsymbol{X}^\intercal \boldsymbol{X} \succeq 0$ (positive semi-definite): Let $z \in R^m \backslash \{\mathbf{0}\}$, then

$$z^\intercal (\boldsymbol{X}^\intercal \boldsymbol{X}) z \overset{(9)}{=} (\boldsymbol{X}z)^\intercal (\boldsymbol{X}z) = \|\boldsymbol{X}z\|^2 \geqslant 0 \tag{15}$$

- The cost function fulfills the second-order convexity condition

Geometric Interpretation of Least Squares

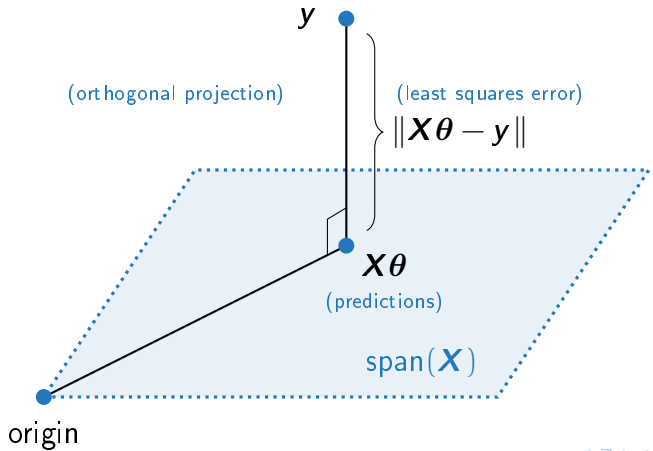
- We want to find a solution to the system of equations

$$\theta_0 x_0 + \theta_1 x_1 + \ldots + \theta_m x_m = y \tag{16}$$

- y is a **linear combination** of the column vectors x_j $(0 \leqslant j \leqslant m)$ of $\boldsymbol{X}$
- However, we will **not find a solution** unless the data is perfectly linear

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

Geometric Interpretation of Least Squares (Ctd.)



Geometric Interpretation of Least Squares (Ctd.)

- We see that the point closest to $\boldsymbol{y}$ is given by the **orthogonal projection** of $\boldsymbol{y}$ onto the span of $\boldsymbol{X}$
- Any other point on the span of $\boldsymbol{X}$ has a larger Euclidean distance to $\boldsymbol{y}$ and therefore cannot be opitmal
- The geometry of least squares also explains the term 'normal equation':
  - A normal line is a line perpendicular to another line or subspace
  - The normal equation given by (4) computes $\boldsymbol{\theta}$ such that the residuals are orthogonal to the span of $\boldsymbol{X}$

Problems with Matrix Inversion?

What if $(\boldsymbol{X}^\intercal \boldsymbol{X})^{-1}$ does not exist? Problems and solutions:

1. Linearly dependent (redundant) features or design matrix does not have full rank? (E. g. size in m^2 and size in feet^2)
   $\Rightarrow$ **Delete correlated features**
2. Too many features ($m > n$)?
   $\Rightarrow$ **Delete features (e. g. using PCA) / add training examples**
3. Other numerical instabilities?
   $\Rightarrow$ **Add a regularization term** (later)
4. Computationally too expensive?
   $\Rightarrow$ **Use gradient descent**

Gradient Descent

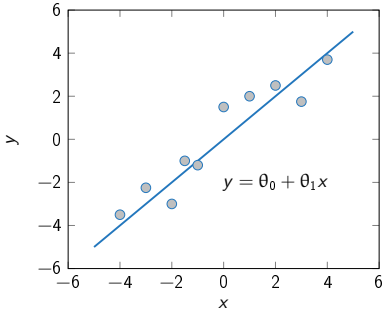
- We want to minimize a continuous function $\mathcal{J} : \mathbb{R}^{m+1} \to \mathbb{R}$: $\min_{\boldsymbol{\theta} \in \mathbb{R}^{m+1}} \mathcal{J}(\boldsymbol{\theta})$
- Update the parameters iteratively:

$$\boldsymbol{\theta}_{t+1} \longleftarrow \boldsymbol{\theta}_t - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}_t) \tag{17}$$

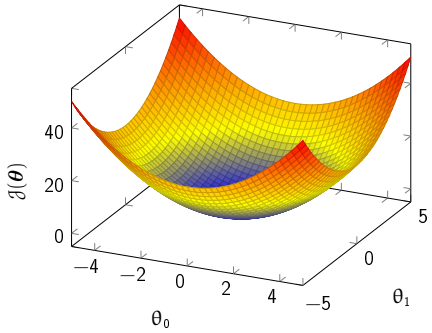
- **Legend**:
  - $\boldsymbol{\theta}_t$ denotes the set of model parameters at timestamp t
  - $\alpha \in (0, 1)$ denotes the learning rate
  - $\nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}) := \left(\frac{\partial}{\partial \theta_0} \mathcal{J}(\boldsymbol{\theta}), \frac{\partial}{\partial \theta_1} \mathcal{J}(\boldsymbol{\theta}), \ldots, \frac{\partial}{\partial \theta_m} \mathcal{J}(\boldsymbol{\theta}) \right)^{\mathsf{T}}$ is the gradient of $\mathcal{J}(\boldsymbol{\theta})$

Data Input Space vs. Hypothesis Space

Data input space



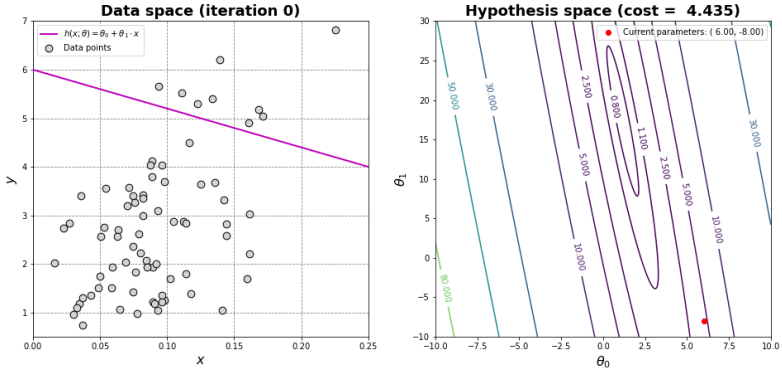
Hypothesis space $\mathcal{H}$



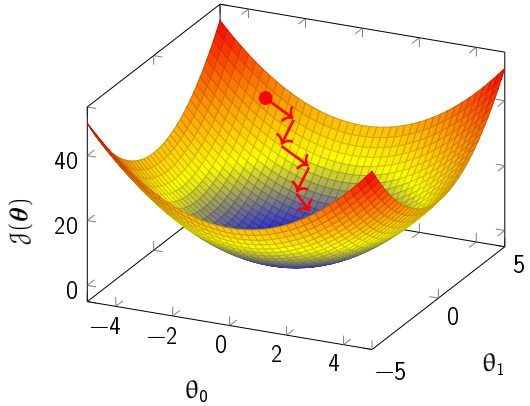
Data Input Space vs. Hypothesis Space (Ctd.)

- **Data input space**
  - Determined by the $m+1$ **attributes** of the dataset $x_0, x_1, x_2, \ldots, x_m$
  - Often high-dimensional
- **Hypothesis space** $\mathcal{H}$
  - Determined by the **number of parameters** of the model
  - 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** $\mathcal{H}$

Data Input Space vs. Hypothesis Space (Ctd.)



Visualization of Gradient Descent in 3 Dimensions



Versions of Gradient Descent

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

$$\ell(y_{pred}, y_{true}) := \frac{1}{2}(y_{pred} - y_{true})^2$$

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

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^{m+1}} \mathcal{J}(\boldsymbol{\theta}) = \min_{\boldsymbol{\theta} \in \mathbb{R}^{m+1}} \sum_{i=1}^{n} \ell\left(h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right), y^{(i)}\right)$$

Batch Gradient Descent

Three versions of gradient descent:

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

$$\boldsymbol{\theta}_{t+1} \longleftarrow \boldsymbol{\theta}_t - \alpha \sum_{i=1}^{n} \nabla \ell\left(h_{\boldsymbol{\theta}_t}\left(\boldsymbol{x}^{(i)}\right), y^{(i)}\right) \tag{18}$$

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

Stochastic Gradient Descent

Three versions of gradient descent:

1. Batch gradient descent
2. **Stochastic gradient descent**
   - Compute gradient based on a **SINGLE** data point

$$\boldsymbol{\theta}_{t+1} \longleftarrow \boldsymbol{\theta}_t - \alpha \nabla \ell\left(h_{\boldsymbol{\theta}_t}\left(\boldsymbol{x}^{(i)}\right), y^{(i)}\right) \tag{19}$$

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

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

$$\boldsymbol{\theta}_{t+1} \longleftarrow \boldsymbol{\theta}_t - \alpha \sum_{i=1}^{n_b} \nabla \ell\left(h_{\boldsymbol{\theta}_t}\left(\boldsymbol{x}^{(i)}\right), y^{(i)}\right) \tag{20}$$

   - Good trade-off between batch and stochastic gradient descent

Stochastic Gradient of the Least Squares Error Function

$$\begin{aligned}\frac{\partial}{\partial \theta_j}\mathcal{J}(\boldsymbol{\theta}) &= \frac{\partial}{\partial \theta_j}\frac{1}{2}\big(h_{\boldsymbol{\theta}}(\boldsymbol{x}) - y\big)^2 \\ &= 2 \cdot \frac{1}{2}\big(h_{\boldsymbol{\theta}}(\boldsymbol{x}) - y\big) \cdot \frac{\partial}{\partial \theta_j}\big(h_{\boldsymbol{\theta}}(\boldsymbol{x}) - y\big) \\ &= \big(h_{\boldsymbol{\theta}}(\boldsymbol{x}) - y\big) \cdot \frac{\partial}{\partial \theta_j}\big(\theta_0 x_0 + \cdots + \theta_m x_m - y\big) \\ &= \boxed{\big(h_{\boldsymbol{\theta}}(\boldsymbol{x}) - y\big)x_j} \qquad (21)\end{aligned}$$

The vectorized version is given by $\frac{\partial}{\partial \boldsymbol{\theta}}\mathcal{J}(\boldsymbol{\theta}) = \big(h_{\boldsymbol{\theta}}(\boldsymbol{x}) - y\big)\boldsymbol{x}$

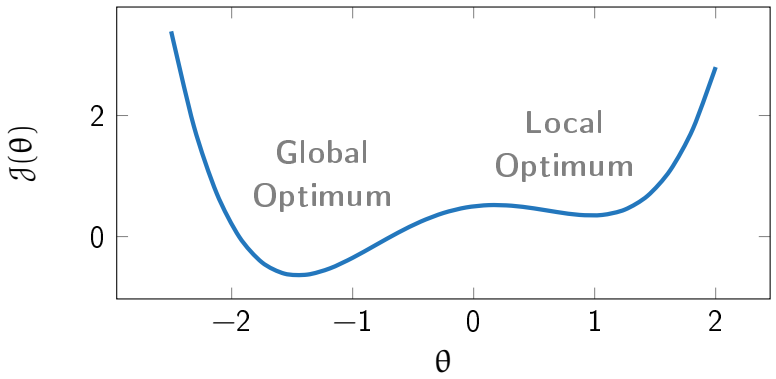
Batch Gradient of the Least Squares Error Function

- We have computed the batch gradient already when deriving the normal equation (we ignore the factor $1/n$ here):

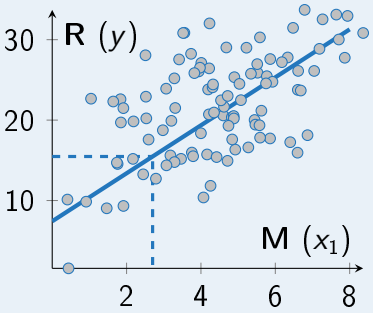
$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}) &= \boldsymbol{X}^\intercal \boldsymbol{X} \boldsymbol{\theta} - \boldsymbol{X}^\intercal \boldsymbol{y} \\ &= \boldsymbol{X}^\intercal (\boldsymbol{X} \boldsymbol{\theta} - \boldsymbol{y}) \end{aligned} \tag{22}$$

- We can use the same formula for the mini-batch gradient 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



Solving the introductory Example



- $\theta_0 \approx 7.4218$
- $\theta_1 \approx 2.9827$
- $\mathcal{J}(\boldsymbol{\theta}) \approx 446.9584$
- $h_{\boldsymbol{\theta}}(\boldsymbol{x}) = 7.4218 + 2.9827 \cdot x_1$
- $R = h_{\boldsymbol{\theta}}(2.7) = \underline{\underline{15.4750}}$

A probabilistic View

- **Assumption 1:** The target values y are generated by an unknown function $f(\boldsymbol{x})$ with additive noise ε:

$$y = f(\boldsymbol{x}) + \varepsilon \tag{23}$$

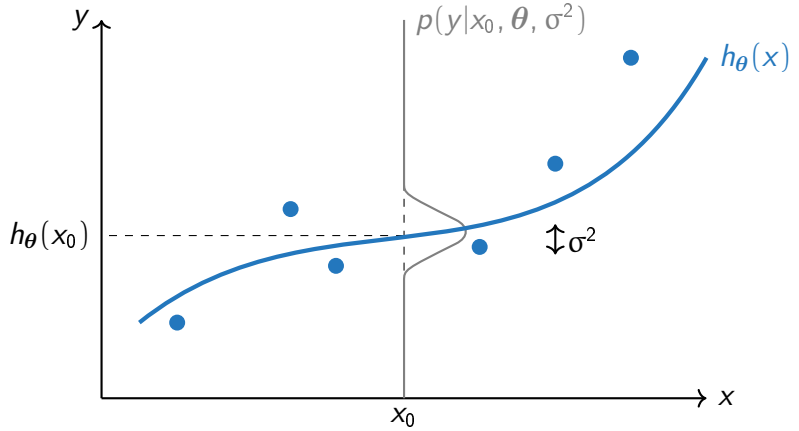
- **Assumption 2:** The noise ε is a zero mean Gaussian random variable

$$\varepsilon \sim \mathcal{N}(0, \sigma^2) \tag{24}$$

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

$$p(y|\boldsymbol{x}, \boldsymbol{\theta}, \sigma^2) = \mathcal{N}(y|h_{\boldsymbol{\theta}}(\boldsymbol{x}), \sigma^2) \tag{25}$$

A probabilistic View (Ctd.)



Likelihood Function for Regression

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

$$p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}, \sigma^2) = \prod_{i=1}^{n} \mathcal{N}\left(y^{(i)}|h_{\boldsymbol{\theta}}\left(\boldsymbol{x}^{(i)}\right), \sigma^2\right)$$

$$= \prod_{i=1}^{n} \mathcal{N}\left(y^{(i)}|\boldsymbol{\theta}^{\intercal}\boldsymbol{x}^{(i)}, \sigma^2\right) \tag{26}$$

Likelihood Function for Regression (Ctd.)

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

$$\log p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}, \sigma^2) = -\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{\sigma^2} \cdot \underbrace{\frac{1}{2}\sum_{i=1}^{n}\left(y^{(i)} - \boldsymbol{\theta}^\intercal \boldsymbol{x}^{(i)}\right)^2}_{\text{least squares error}} \tag{27}$$

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

When minimizing the squared error we implicitly assume Gaussian noise!

The Maximum Likelihood Solution to Regression

- Optimization of the log-likelihood function gives:

$$\boldsymbol{\theta}_{\mathrm{ML}} = (\boldsymbol{X}^{\intercal}\boldsymbol{X})^{-1}\boldsymbol{X}^{\intercal}\boldsymbol{y} \tag{28}$$

$$\sigma^2_{\mathrm{ML}} = \frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - \boldsymbol{\theta}^{\intercal}_{\mathrm{ML}}\boldsymbol{x}^{(i)}\right)^2 \tag{29}$$

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

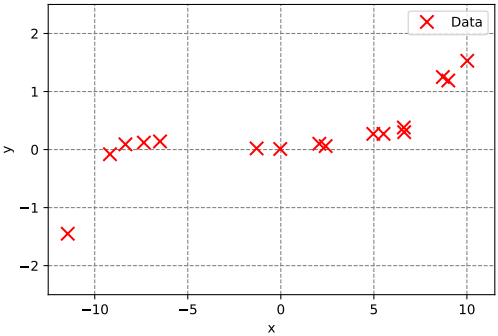
Section:

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...?**



Basis Functions

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

$$h_{\theta}(\boldsymbol{x}) := \sum_{j=0}^{p} \theta_j \varphi_j(\boldsymbol{x}) \tag{30}$$

- Commonly used basis functions:
  - **Linear:** $\varphi_0(\boldsymbol{x}) = 1$ and $\varphi_1(\boldsymbol{x}) = x_1, \ldots, \varphi_m(\boldsymbol{x}) = x_m$
  - **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 $\boldsymbol{\Phi}$:

$$\boldsymbol{\Phi} := \begin{pmatrix} \varphi_0\left(\boldsymbol{x}^{(1)}\right) & \varphi_1\left(\boldsymbol{x}^{(1)}\right) & \varphi_2\left(\boldsymbol{x}^{(1)}\right) & \dots & \varphi_p\left(\boldsymbol{x}^{(1)}\right) \\ \varphi_0\left(\boldsymbol{x}^{(2)}\right) & \varphi_1\left(\boldsymbol{x}^{(2)}\right) & \varphi_2\left(\boldsymbol{x}^{(2)}\right) & \dots & \varphi_p\left(\boldsymbol{x}^{(2)}\right) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \varphi_0\left(\boldsymbol{x}^{(n)}\right) & \varphi_1\left(\boldsymbol{x}^{(n)}\right) & \varphi_2\left(\boldsymbol{x}^{(n)}\right) & \dots & \varphi_p\left(\boldsymbol{x}^{(n)}\right) \end{pmatrix} \tag{31}$$

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

Basis Functions: Polynomial Basis Functions

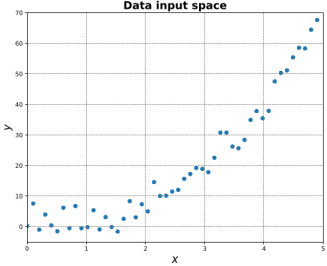
- Let us assume a one-dimensional dataset ($m = 1$) for now
- A quite frequently used basis function is the **polynomial basis**

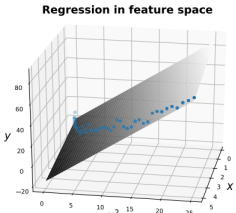
$$\varphi_0(x) := 1, \varphi_j(x) := x^j$$

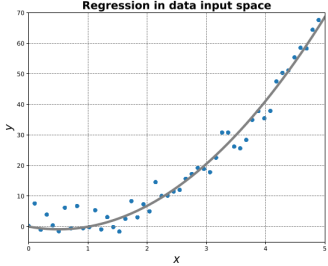
$$h_{\boldsymbol{\theta}}(x) := \sum_{j=0}^{p} \theta_j \varphi_j(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \cdots + \theta_p x^p$$

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

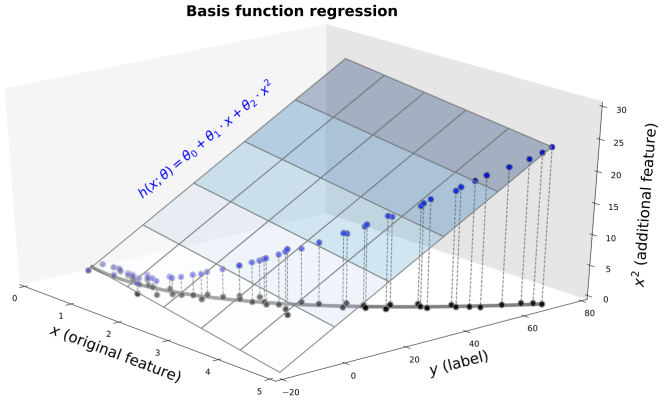
It is still linear!







It is still linear! (Ctd.)



Basis Functions: Radial Basis Functions (RBFs)

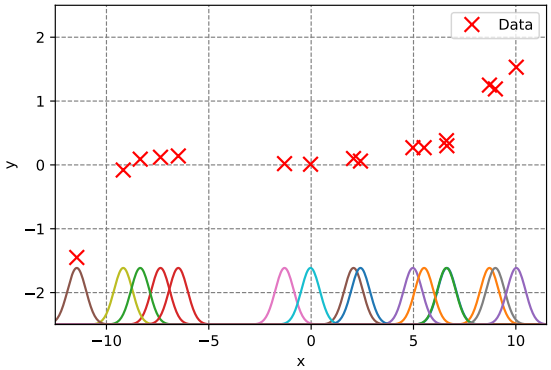
- Another possible choice of basis function: **Radial basis functions**

$$\varphi_0(x) := 1 \tag{32}$$

$$\varphi_j(x) := \exp\left(-1/2\|x - z_j\|^2/2\sigma^2\right) \tag{33}$$

- $\{z_j\}$ 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.)



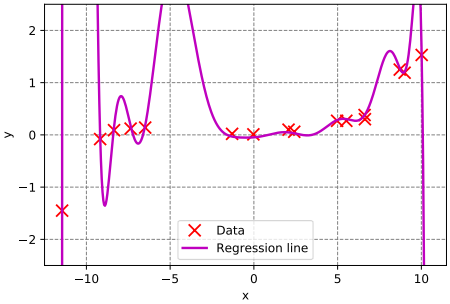
Section:

Regularization Techniques

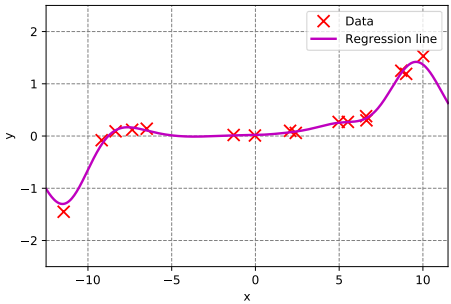
Underfitting and Overfitting
L1 and L2 Regularization

The Danger of too expressive Models...

Polynomial of degree $p = 16$
(☠ severe overfitting ☠)



RBF with $\sigma^2 = 1.00, p = n$
(about right)

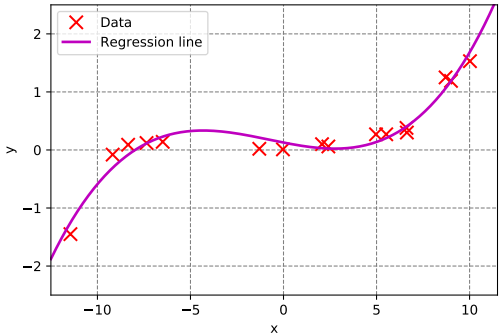


Overfitting vs. Underfitting

- **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 $\mathcal{J}(\boldsymbol{\theta})$ with a **regularization term**
- This helps **prevent overfitting** and results in a smoother function

L1-Regularization:

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

$$|\boldsymbol{\theta}| = \sum_{j=1}^{m} |\theta_j|$$

L2-Regularization:

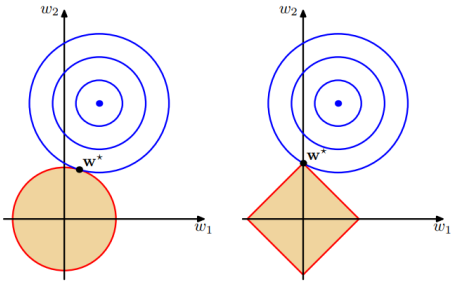
$$\widetilde{\mathcal{J}}(\boldsymbol{\theta}) := \mathcal{J}(\boldsymbol{\theta}) + \lambda\|\boldsymbol{\theta}\|^2$$

$$\|\boldsymbol{\theta}\|^2 = \sum_{j=1}^{m} \theta_j^2$$

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

Regularization visualized

- Here: $\boldsymbol{w} \equiv \boldsymbol{\theta}$
- L1-Regularization
  $\Rightarrow$ **Lasso regression**
  (least abs. shrinkage and select. operator)
- L2-Regularization
  $\Rightarrow$ **Ridge regression**
  (Tikhonov regularization)
- The combination of both is called **elastic net**



cf. Bishop.2006, page 146, left: L2, right: L1

Incorporating Regularization

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

$$\boldsymbol{\theta}^{\star} := (\boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{\Phi} + \lambda \boldsymbol{I})^{-1}\boldsymbol{\Phi}^{\mathsf{T}}\boldsymbol{y} \tag{34}$$

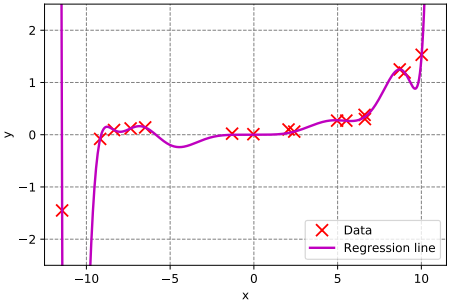
- Regularization also helps overcome numerical issues (matrix inversion!)
- Regularized least squares error gradient:

$$\frac{\partial}{\partial \theta_j}\mathcal{J}(\boldsymbol{\theta}) = \big(h_{\boldsymbol{\theta}}(\boldsymbol{\varphi}(\boldsymbol{x})) - y\big)\varphi_j(\boldsymbol{x}) + \lambda\theta_j \tag{35}$$

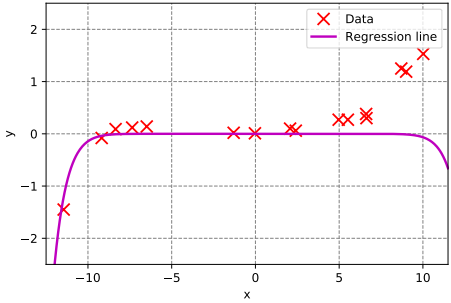
where $\boldsymbol{\varphi}(\boldsymbol{x}) := \big(\varphi_0(\boldsymbol{x}), \ldots, \varphi_m(\boldsymbol{x})\big)^{\mathsf{T}}$

Polynomial Regression with Regularization

At least better



Way too much regularization



Section:

Wrap-Up

Summary
Self-Test Questions
Lecture Outlook

Summary

- In regression we predict **continuous target variables**
- The algorithm minimizes the **(mean) squared error**
- **Two approaches:**
  1. Normal equation
  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)

Self-Test Questions

1. What is the goal of regression?
2. What can you do if matrix inversion fails for the normal equation?
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 should you apply it?

What's next...?

Unit I	Machine Learning Introduction
Unit II	Mathematical Foundations
Unit III	Bayesian Decision Theory
Unit IV	Regression
Unit V	**Classification I**
Unit VI	Evaluation
Unit VII	Classification II
Unit VIII	Clustering
Unit IX	Dimensionality Reduction

Thank you very much for the attention!

Topic: *** Applied Machine Learning Fundamentals *** Regression
Term: Winter term 2023/2024

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

Do you have any questions?