*** Applied Machine Learning Fundamentals *** Regression

Daniel Wehner

SAP SE

August 19, 2019

DHBW
Duale Hochschule
Baden-Württemberg

Agenda August 19, 2019

Section:

Introduction to Regression

Regression

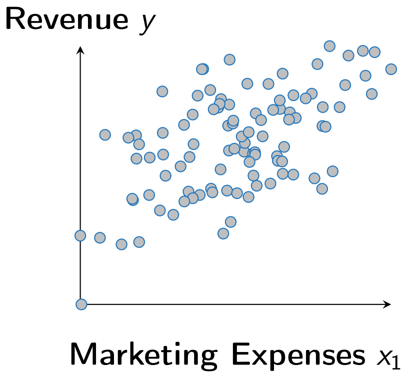
Type of target variable	Continuous
Type of training information	Supervised
Example Availability	Batch learning

Algorithm sketch: Given the training data $\mathcal{D}$ the algorithm derives a function of the type

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

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 Data Set: Revenues



- Find a linear function:

$$h_{\boldsymbol{\theta}}(\boldsymbol{x}) = \theta_0 + \theta_1 x_1 + \cdots + \theta_{m+1} x_m$$

- Usually: $x_0 = 1$:

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

$$h_{\boldsymbol{\theta}}(\widehat{\boldsymbol{x}}) = \sum_{j=0}^{m+1} \theta_j x_j = \boldsymbol{\theta}^\intercal \widehat{\boldsymbol{x}}$$

Error Function for Regression

- In order to know how good the function fits we need an error function $\mathcal{J}(\boldsymbol{\theta})$:

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

- We want to minimize $\mathcal{J}(\boldsymbol{\theta})$:

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

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

Error Function Intuition

Section:

Solutions to Regression

Closed-Form Solutions

- Usual approach (for two unknowns): Calculate θ_0 and θ_1 according to

sample mean $\overline{x}$

$$\theta_0 = \overline{y} - \theta_2 \overline{x} \qquad \theta_1 = \frac{\sum_{i=1}^{n} (x^{(i)} - \overline{x}) \cdot (y^{(i)} - \overline{y})}{\sum_{i=1}^{n} (x^{(i)} - \overline{x})^2} \tag{3}$$

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

$$\boldsymbol{\theta} = \underbrace{(\widehat{\boldsymbol{X}}^\intercal \widehat{\boldsymbol{X}})^{-1} \widehat{\boldsymbol{X}}^\intercal}_{\text{Moore-Penrose pseudo-inverse}} \boldsymbol{y} \tag{4}$$

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

Design Matrix / Regressor Matrix

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

In the following
$\hat{\boldsymbol{X}} \equiv \boldsymbol{X}$

$$\hat{\boldsymbol{X}} = \begin{pmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \cdots & x_m^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & \cdots & x_m^{(2)} \\ 1 & x_1^{(3)} & x_2^{(3)} & \cdots & x_m^{(3)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(n)} & x_2^{(n)} & \cdots & x_m^{(n)} \end{pmatrix} \tag{5}$$

- And the $n \times 1$ label vector:

$$\boldsymbol{y} = \left(y^{(1)}, y^{(2)}, y^{(3)}, \ldots, y^{(n)}\right)^\intercal$$

Derivation of the Normal Equation

- The derivation involves a bit of linear algebra
- Step ❶: Rewrite $\mathcal{J}(\boldsymbol{\theta})$ in matrix-vector notation:

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

- To be continued...

Derivation of the Normal Equation (Ctd.)

- Step ❷: Calculate the derivative of $\mathcal{J}(\boldsymbol{\theta})$ and set it to zero:

$$\nabla_{\boldsymbol{\theta}}\mathcal{J}(\boldsymbol{\theta}) = 2\boldsymbol{X}^\intercal\boldsymbol{X}\boldsymbol{\theta} - 2\boldsymbol{X}^\intercal\boldsymbol{y} \stackrel{!}{=} \boldsymbol{0}$$
$$\Leftrightarrow \boldsymbol{X}^\intercal\boldsymbol{X}\boldsymbol{\theta} = \boldsymbol{X}^\intercal\boldsymbol{y}$$

- If $\boldsymbol{X}^\intercal\boldsymbol{X}$ is invertible, we can multiply both sides by $(\boldsymbol{X}^\intercal\boldsymbol{X})^{-1}$:

Normal equation:

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

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 smooth 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:

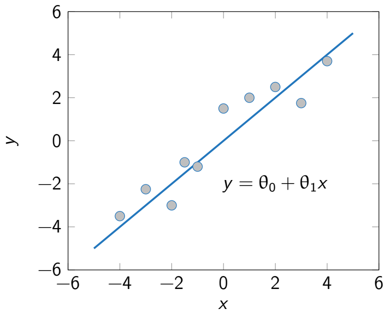
$$\boldsymbol{\theta}^{(t+1)} \longleftarrow \boldsymbol{\theta}^{(t)} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}^{(t)}) \tag{6}$$

- where $\alpha > 0$ (learning rate) and $\nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta})$ is the gradient of $\mathcal{J}(\boldsymbol{\theta})$ w. r. t. $\boldsymbol{\theta}$:

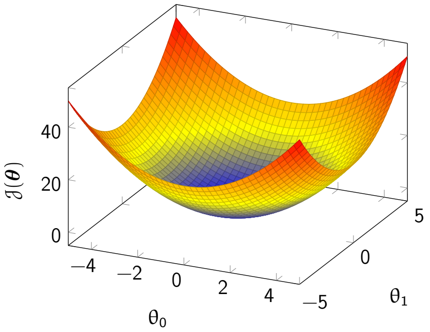
$$\nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}) = \left(\frac{\partial \mathcal{J}(\boldsymbol{\theta})}{\partial \theta_0}, \frac{\partial \mathcal{J}(\boldsymbol{\theta})}{\partial \theta_1}, \dots, \frac{\partial \mathcal{J}(\boldsymbol{\theta})}{\partial \theta_{m+1}} \right)^{\mathsf{T}}$$

Data Input Space vs. Hypothesis Space

Data input space



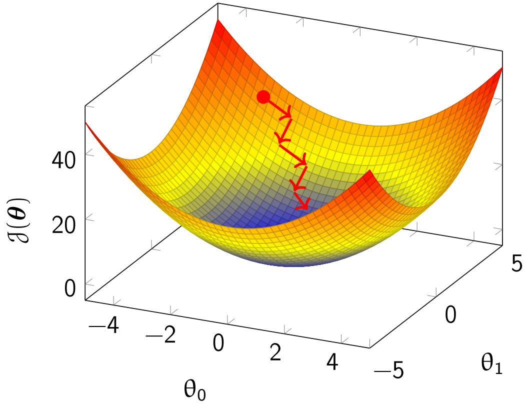
Hypothesis space $\mathcal{H}$



Data Input Space vs. Hypothesis Space (Ctd.)

- **Data input space**
  - Determined by the m **attributes** of the data set $x_1, x_2, \ldots, x_m$
  - 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** $\mathcal{H}$

Visualization of Gradient Descent in 3 Dimensions



Versions of Gradient Descent

- Assume some training data $\mathcal{D}$: $\{\boldsymbol{x}^{(i)}, y^{(i)}\}_{i=1}^{n}$
- Squared error for a **single** example: $\ell(y_{pred}, y_{true}) = (y_{pred} - y_{true})^2$
- Our objective is to minimize the **total** error:

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

- Three versions of gradient descent:
  1. Batch gradient descent
  2. Stochastic gradient descent
  3. Mini-batch gradient descent

Versions of Gradient Descent (Ctd.)

- **Batch gradient descent**: Compute gradient based on **ALL** data points

$$\boldsymbol{\theta}^{(t+1)} \longleftarrow \boldsymbol{\theta}^{(t)} - \alpha \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \nabla \ell(h_{\boldsymbol{\theta}^{(t)}}(\boldsymbol{x}^{(i)}), y^{(i)}) \tag{7}$$

- **Stochastic gradient descent**: Compute gradient based on a **SINGLE** data point **(pick training example randomly and not sequentially!)**
- For $i \in \{1, \ldots, n\}$ do:

$$\boldsymbol{\theta}^{(t+1)} \longleftarrow \boldsymbol{\theta}^{(t)} - \alpha \nabla \ell(h_{\boldsymbol{\theta}^{(t)}}(\boldsymbol{x}^{(i)}), y^{(i)}) \tag{8}$$

Solving linear Regression using Gradient Descent

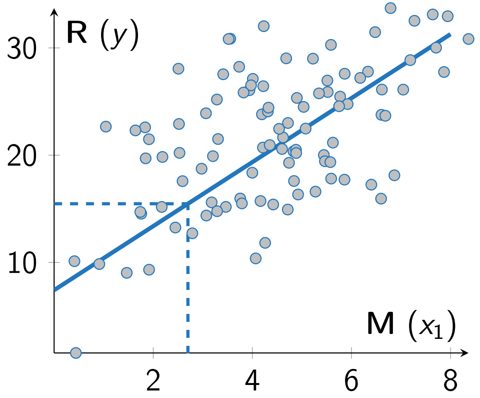
- Randomly initialize $\boldsymbol{\theta}$
- To minimize the error, keep changing $\boldsymbol{\theta}$ according to:

$$\boldsymbol{\theta}^{(t+1)} \longleftarrow \boldsymbol{\theta}^{(t)} - \alpha \nabla_{\boldsymbol{\theta}} \mathcal{J}(\boldsymbol{\theta}^{(t)}) \tag{9}$$

- We need to calculate $\nabla_{\theta_j} \mathcal{J}(\boldsymbol{\theta}^{(t)})$: (based on a single example)

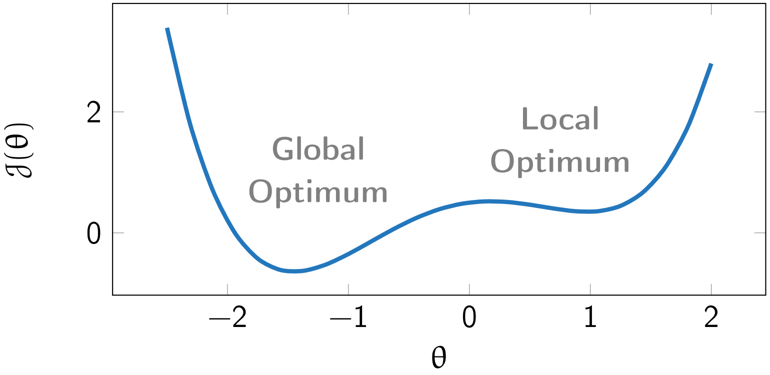
$$\begin{aligned}\nabla_{\theta_j} \mathcal{J}(\boldsymbol{\theta}^{(t)}) &= \frac{\partial}{\partial \theta_j} \frac{1}{2} (h_{\boldsymbol{\theta}}(\boldsymbol{x}) - y)^2 = 2 \cdot \frac{1}{2} (h_{\boldsymbol{\theta}}(\boldsymbol{x}) - y) \cdot \frac{\partial}{\partial \theta_j} (h_{\boldsymbol{\theta}}(\boldsymbol{x}) - y) \\ &= (h_{\boldsymbol{\theta}}(\boldsymbol{x}) - y) \cdot \frac{\partial}{\partial \theta_j} (\theta_0 x_0 + \cdots + \theta_{m+1} x_{m+1} - y) = \boxed{(h_{\boldsymbol{\theta}}(\boldsymbol{x}) - y) x_j}\end{aligned}$$

Solving the introductory Example



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

Disadvantage of Gradient Descent



Section:

Probabilistic Regression

Probabilistic Regression

- **Assumption 1:** The target function values are generated by adding noise the true function's estimate:

$$y = f(\mathbf{x}, \boldsymbol{\theta}) + \epsilon \tag{10}$$

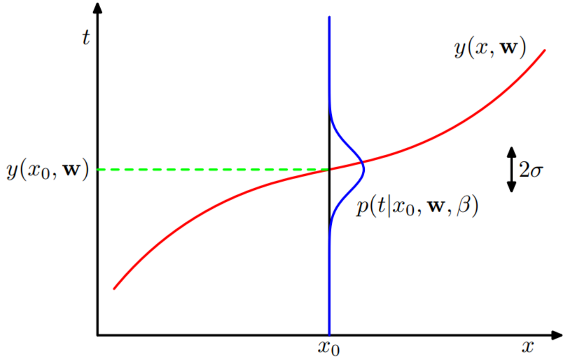
- **Assumption 2:** The noise is a Gaussian random variable:

$$\epsilon \sim \mathcal{N}(0, \beta^{-1}) \tag{11}$$

$$p(y|\mathbf{x}, \boldsymbol{\theta}, \beta) = \mathcal{N}(y|f(\mathbf{x}, \boldsymbol{\theta}), \beta^{-1}) \tag{12}$$

- **y is now a random variable!**

Probabilistic Regression (Ctd.)



Maximum Likelihood Regression

- **Given:** A labeled set of training data points $\{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{n}$
- **Conditional likelihood** (assuming the data is i. i. d.):

$$p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}, \beta) = \prod_{i=1}^{n} \mathcal{N}(y^{(i)}|f(\boldsymbol{x}^{(i)}), \beta^{-1}) \tag{13}$$

$$= \prod_{i=1}^{n} \mathcal{N}(y^{(i)}|\boldsymbol{\theta}^{\intercal}\boldsymbol{x}^{(i)}, \beta^{-1}) \tag{14}$$

- Maximize the likelihood w. r. t. $\boldsymbol{\theta}$ and β

Maximum Likelihood Regression (Ctd.)

Simplify using the **log**-likelihood:

$$\log p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}, \beta) = \sum_{i=1}^{n} \log \mathcal{N}(y^{(i)}|\boldsymbol{\theta}^{\intercal}\boldsymbol{x}^{(i)}, \beta^{-1}) \tag{15}$$

$$= \sum_{i=1}^{n} \left[\log\left(\frac{\sqrt{\beta}}{\sqrt{2\pi}}\right) - \frac{\beta}{2}(y^{(i)} - \boldsymbol{\theta}^{\intercal}\boldsymbol{x}^{(i)})^2\right] \tag{16}$$

$$= \frac{n}{2}\log\beta - \frac{n}{2}\log(2\pi) - \frac{\beta}{2}\sum_{i=1}^{n}(y^{(i)} - \boldsymbol{\theta}^{\intercal}\boldsymbol{x}^{(i)})^2 \tag{17}$$

Remember log-rules?

Maximum Likelihood Regression (Ctd.)

- Compute the gradient w.r.t. $\boldsymbol{\theta}$:

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

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

$$\dots$$

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

- **Same result as in least squares regression**

Important

We have derived the squared Error!

Minimizing the squared error gives the maximum likelihood solution for the parameters $\boldsymbol{\theta}$ assuming Gaussian noise.

- The maximum likelihood approach gives rise to the squared error
- But it is much more powerful than regular least squares $\Rightarrow$ **We can estimate the uncertainty β**

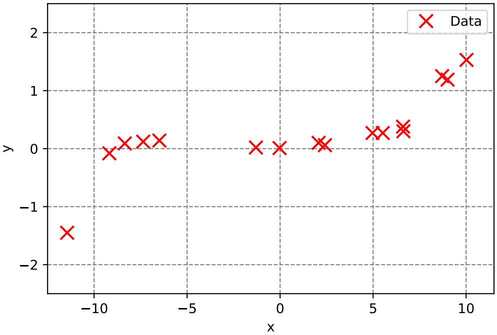
$$\beta_{ML} = \left(\frac{1}{n} \sum_{i=1}^{n} (y^{(i)} - \boldsymbol{\theta}_{ML}^{\intercal} \boldsymbol{x}^{(i)})^2 \right)^{-1} \tag{18}$$

Section:

Basis Function Regression

What if...?

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



This best-fitting function is obviously **not a straight line**!

What would you do?

Basis Functions

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

We assume 1-D data

$$h_{\boldsymbol{\theta}}(x) = \sum_{j=0}^{P} \theta_j \varphi_j(x) \tag{19}$$

- There exist several types of basis functions:
  - linear: $\varphi_0(x) = 1$ and $\varphi_1(x) = x$
  - polynomial $\Rightarrow$ see below
  - radial basis functions (RBFs) $\Rightarrow$ see below
  - Fourier basis

New Design Matrix

Applying the basis functions to $\boldsymbol{X}$ we get the new design matrix $\boldsymbol{\Phi}$:

$$\boldsymbol{\Phi} = \begin{pmatrix} \varphi_0(x^{(1)}) & \varphi_1(x^{(1)}) & \varphi_2(x^{(1)}) & \dots & \varphi_P(x^{(1)}) \\ \varphi_0(x^{(2)}) & \varphi_1(x^{(2)}) & \varphi_2(x^{(2)}) & \dots & \varphi_P(x^{(2)}) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \varphi_0(x^{(n)}) & \varphi_1(x^{(n)}) & \varphi_2(x^{(n)}) & \dots & \varphi_P(x^{(n)}) \end{pmatrix} \tag{20}$$

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

Polynomial Basis Functions

- A quite frequently used basis function: The **polynomial basis**

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

For N-D data we would also include cross-terms!

$$h_{\theta}(x) = \sum_{j=0}^{P} \theta_j \varphi_j(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \cdots + \theta_P x^P$$

- Here, P is the degree of the polynomial
- Here: $\varphi(x) = [1, x, x^2, x^3, \ldots, x^P]$

Basis Functions: Radial Basis Functions

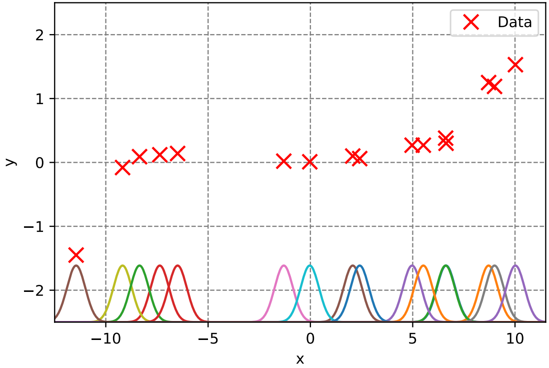
- Yet another possible choice of basis function: **Radial basis functions**

$$\varphi_0(x) = 1 \tag{21}$$

$$\varphi_j(x) = \exp\left\{-{}^1\!/_2 \|x - z_j\|^2 / 2\sigma^2\right\} \tag{22}$$

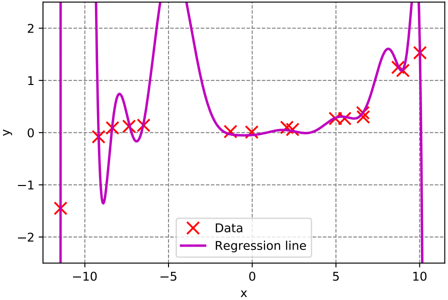
- $\{z_j\}$ are the centers of the radial basis functions
- P denotes the number of centers / number of radial basis functions
- Often we take each data point as a center, so $P = N$

Radial Basis Functions (Ctd.)

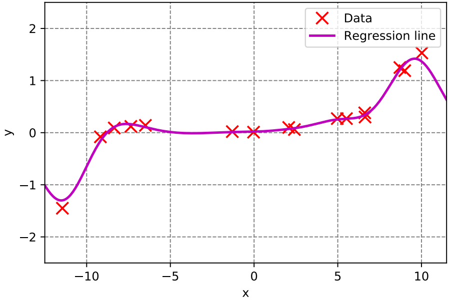


The Danger of too expressive Models...

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



RBF with $\sigma = 1.00$, $P = N$
(About right)



Summary

Lecture Overview

Unit I: Machine Learning Introduction

Self-Test Questions

Recommended Literature and further Reading

Thank you very much for the attention!

Topic: *** Applied Machine Learning Fundamentals *** Regression
Date: August 19, 2019

Contact:
Daniel Wehner (D062271)
SAP SE
daniel.wehner@sap.com

Do you have any questions?