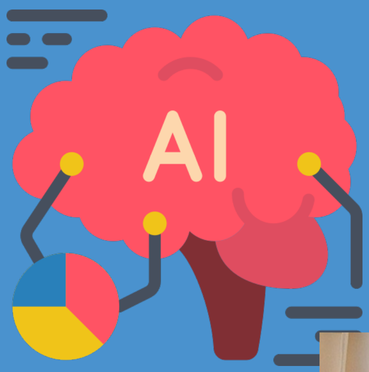
Artificial Intelligence in the Automotive Industry

Künstliche Intelligenz in der Automobilindustrie

Dr. Michael Nolting

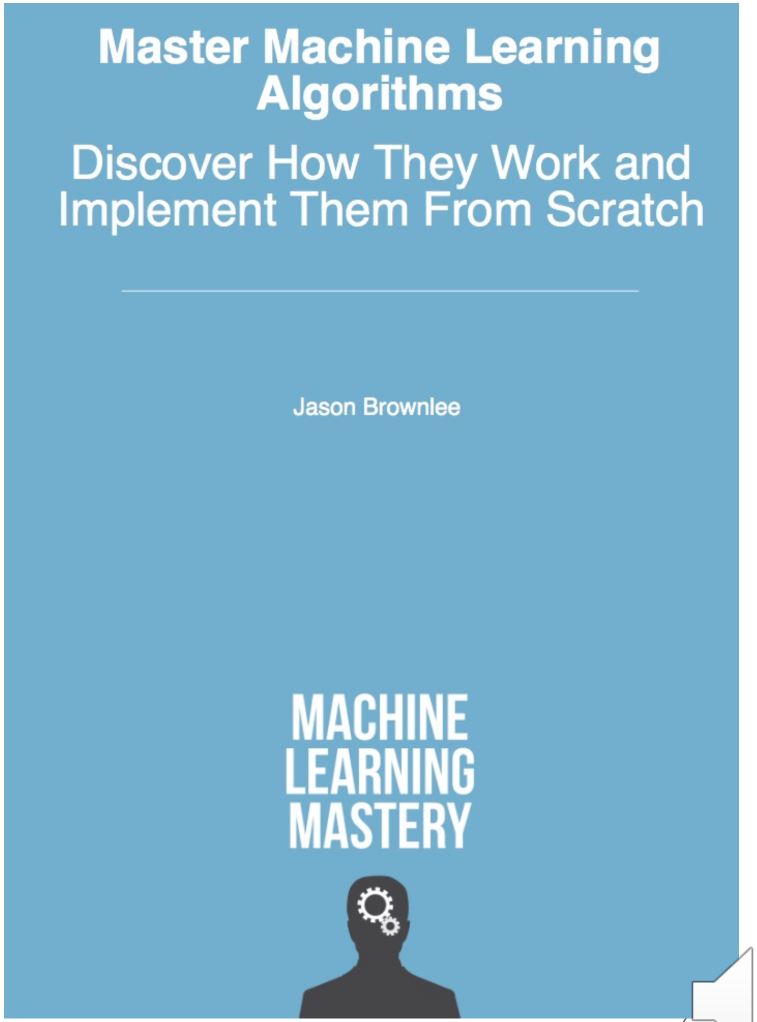
Lecture 4, 2021-05-14



Tutorials

- Tutorials are based on the book „Master Machine Learning Algorithms“ (big thanks to Jason Brownlee, www.machinelearningmastery.com)
- All tutorials are pre-implemented in Excel (Excel spreadsheet will be provided)
- All Excel spreadsheets have to be re-implemented in Python
  - You will roughly need 30 min to read the provided tutorial
  - Please try to re-implement each spreadsheet in Python (probably it will take 30 minutes per spreadsheet)
- If you do not have any Python experience, please do this free tutorial: https://www.codecademy.com/learn/learn-python

The tutorials are optional and not relevant for the oral exams



Lecture Overview

1. Introduction: The ABC of AI	**7. Supervised Learning: (Deep) Neural Networks**
Tutorial 1: Machine Learning Basics	Tutorial 7: Naïve Bayes
2. AI Methods & Automotive Value Chain	**8. Vision: One SOP per day**
Tutorial 2: Linear Regression	Tutorial 8: Gaussian Naïve Bayes
3. Autonomous Driving & Computer Vision	**9. Mission: Own your Code! Own your Data!**
Tutorial 3: Linear Regression Gradient Descent	Tutorial 9: k-Nearest Neighbors
4. Supervised Learning: Regression	**10. Organization: Building HPT**
Tutorial 4: Logistic Regression	Tutorial 10: Support Vector Machines
5. Supervised Learning: Classification	**11. Q & A – Exam**
Tutorial 5: Linear Discriminant Analysis	12. Bonus: 3 to 4 presentations from our PhD students
6. Unsupervised Learning: Clustering	
Tutorial 6: Classification and Regression Trees	

Agenda

01 Motivation

02 (Non-)Linear Models & Loss functions

03 Regularization & Validation

04 Summary

Agenda

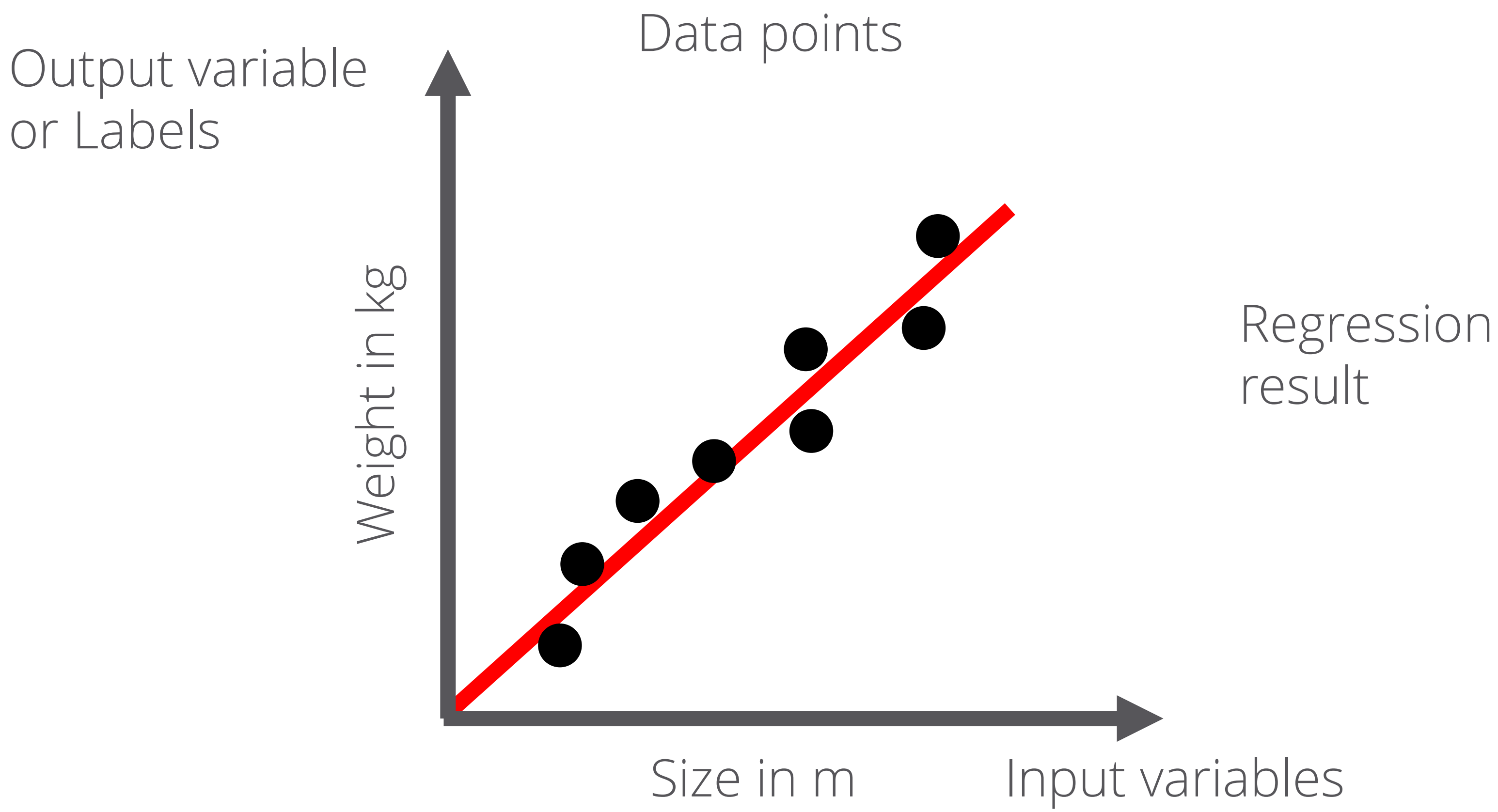
01 **Motivation**

02 (Non-)Linear Models & Loss functions

03 Regularization & Validation

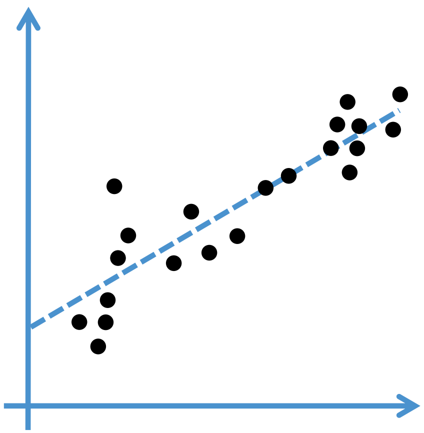
04 Summary

Motivation – Regression Example



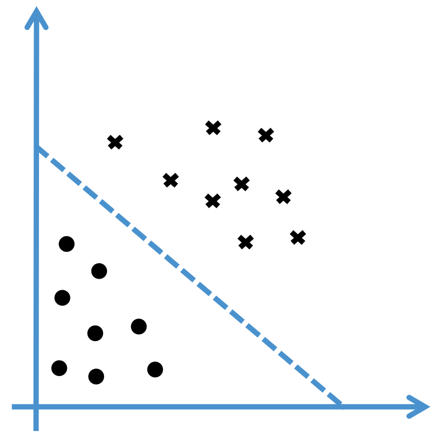
Recap – Supervised & Unsupervised Learning

Regression



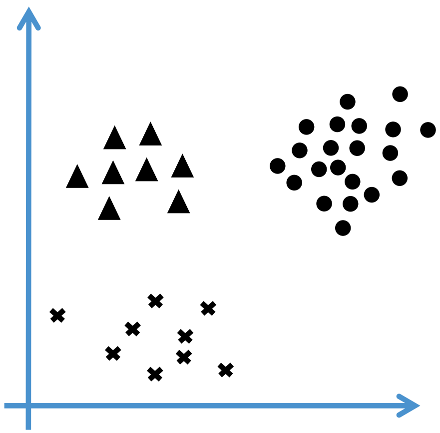
- Predicting **numerical values**
- Supervised

Classification



- Predicting **discrete** valued output
- Supervised

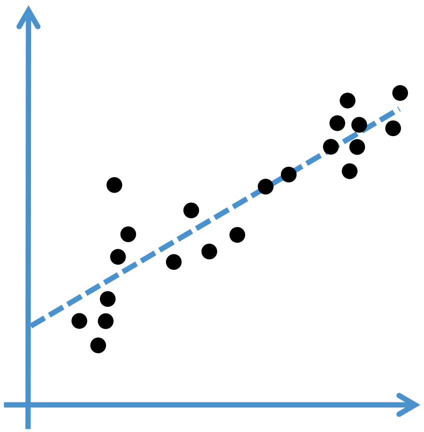
Clustering



- Predicting **discrete** valued output
- Unsupervised

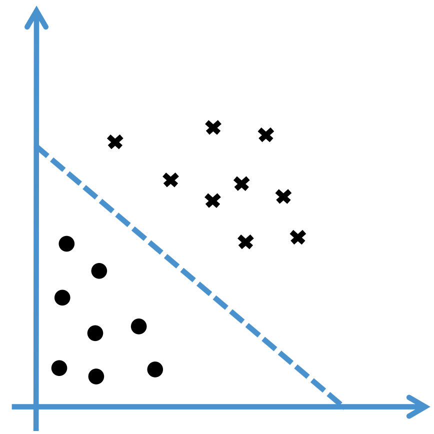
Applications

Regression



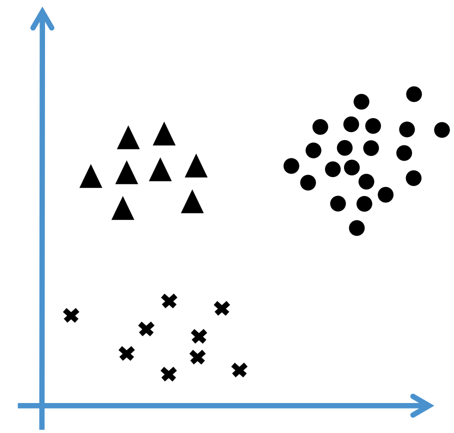
- House prices
- Sales planning
- Child's weight gain

Classification



- Object detection
- Spam detection
- Cancer detection

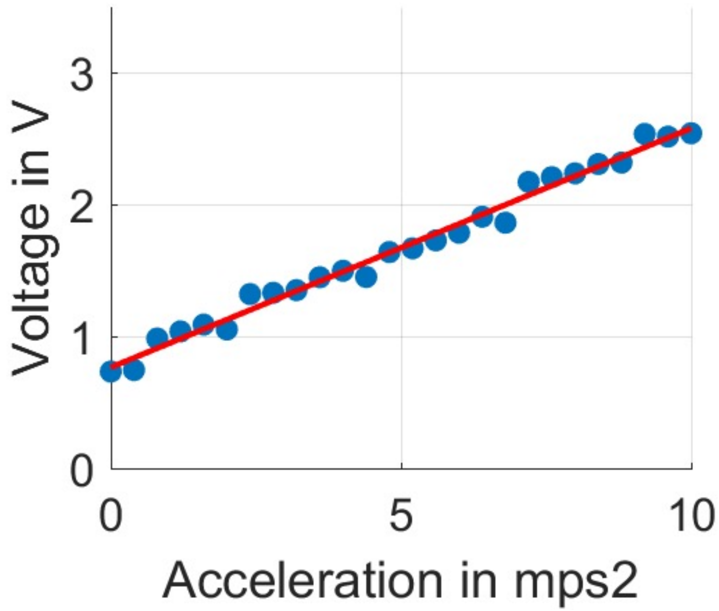
Clustering

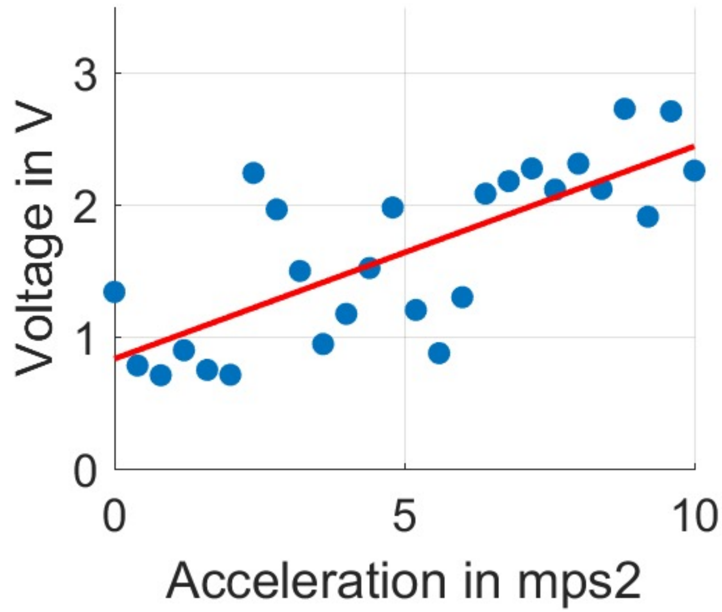


- Genome patterns
- Personalized news
- Point Cloud (LIDAR) processing

Motivation – Regression in Automotive Technology

Application: Sensor calibration

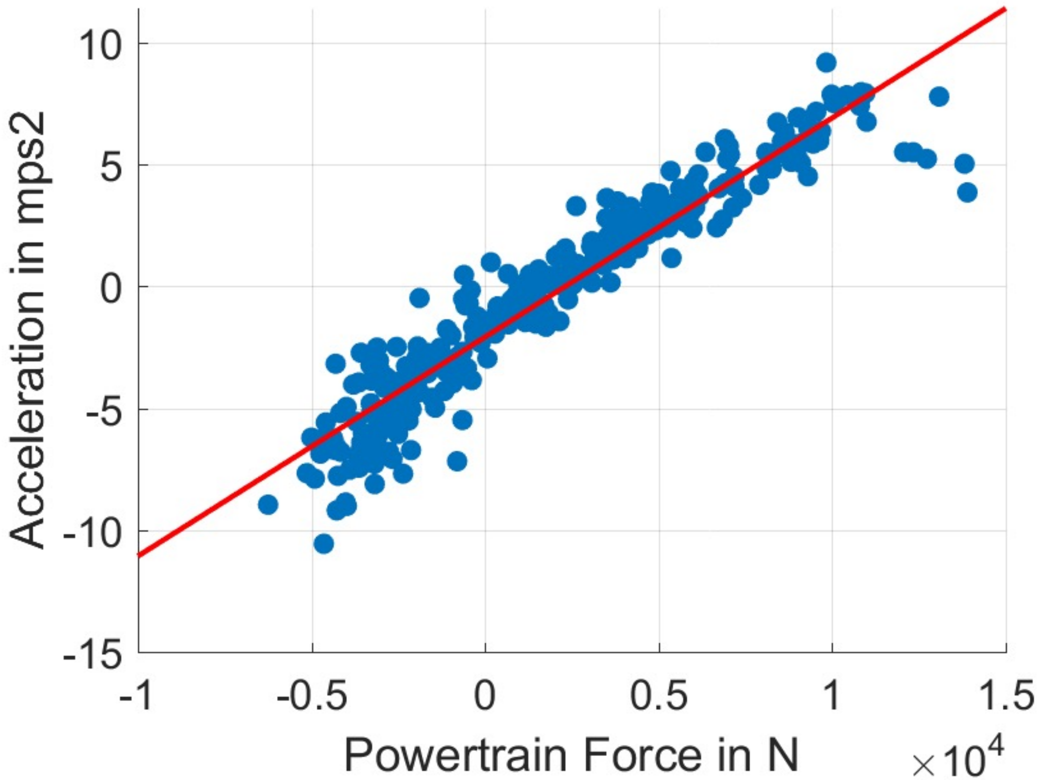




- Electrical measurands like voltage, current and power are given
- Physical quantities are deduced

- Examples:
  - Accelerometers
  - Gyroscopes
  - Displacement sensors

Application: Parameter estimation



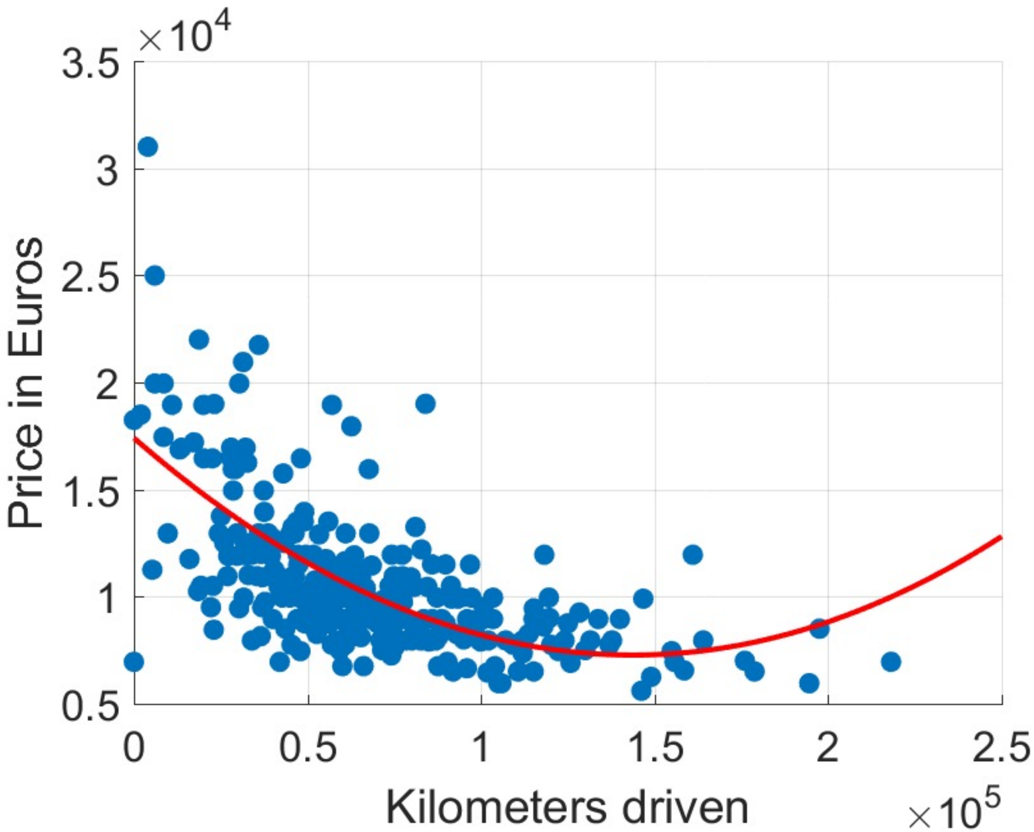
- Vehicle parameters are not always given and known
- Estimating unknown parameters by using linear regression

$$a_x = \frac{1}{m} F_{\mathrm{PT}}$$

e.g. estimating the freight weight

Motivation – Regression in Automotive Technology

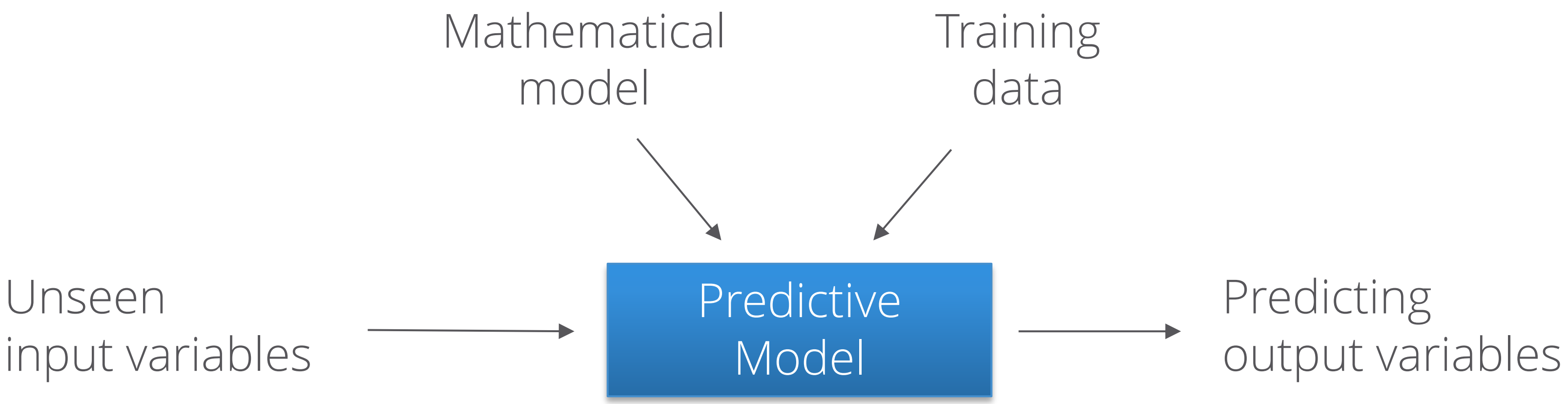
Application: Estimating the residual value of a vehicle



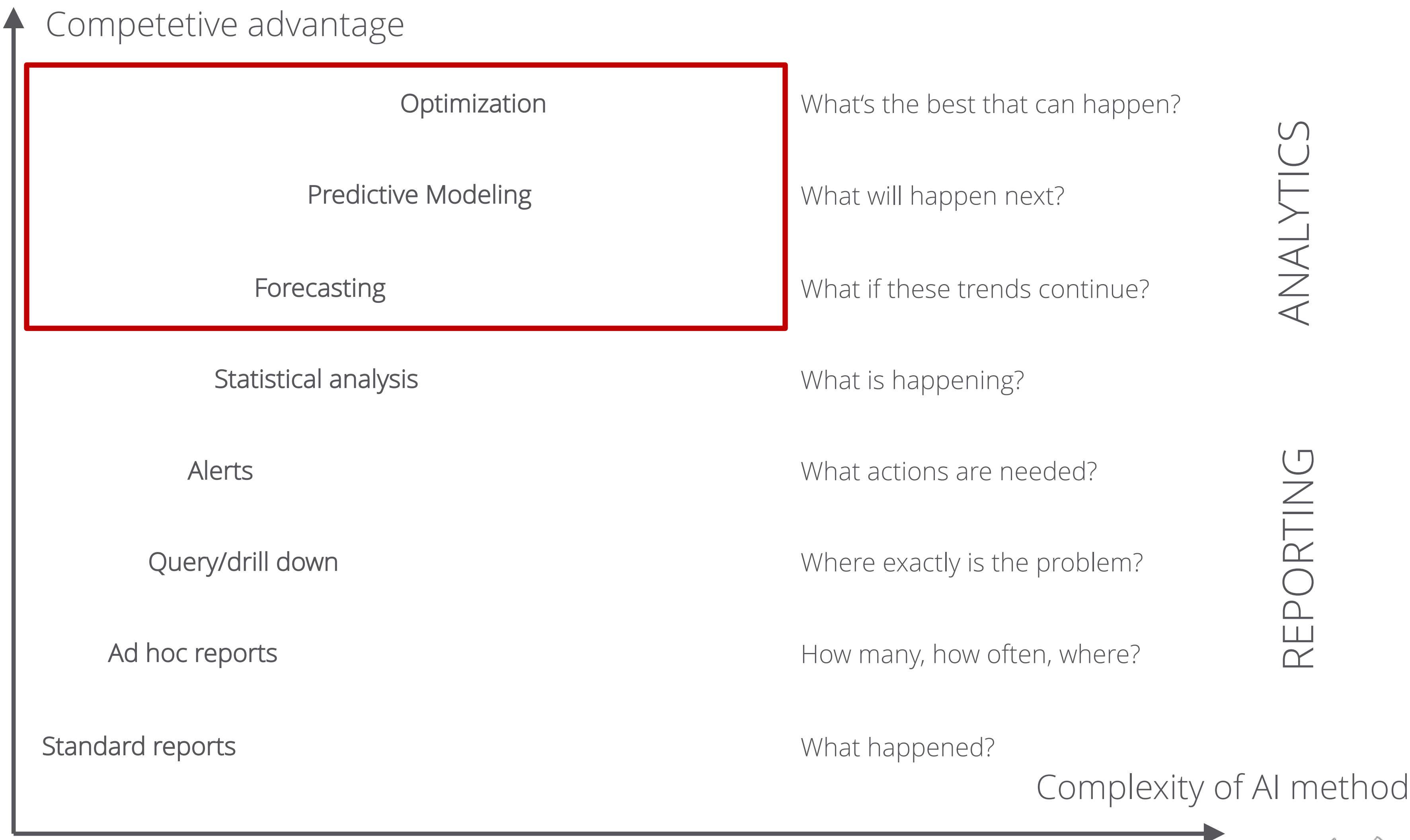
- Regression is widely used in the financial sector
- Enables compressing data into a simple and compact model and do analytics on it

Motivation – Regression Paves the Way for Prediction ☺

- By feeding data into a mathematical model, it is possible to **predict any outcome** for a process or system
- Training data is sparse and noisy
- Can be applied to simulation, optimization, etc.



Statistics is a Good Friend of Regression



Agenda

01 Motivation

02 **(Non-)Linear Models & Loss functions**

03 Regularization & Validation

04 Summary

Linear Basis Function Model

Bias term

Input variables

$$y = b + \sum w_i \phi_i(x)$$

Output variables

Weight parameters

Basis functions

$$y = \begin{bmatrix} 1 & \phi_1(x) & \dots & \phi_k \end{bmatrix} \begin{bmatrix} b \\ w_1 \\ \vdots \\ w_k \end{bmatrix}$$

Weight parameters

Basis functions

Representing the Dataset as a Matrix

$$\begin{bmatrix} y_1 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} 1 & \phi_1(x_1) & \dots & \phi_k(x_1) \\ \vdots & \ddots & \ddots & \vdots \\ 1 & \phi_1(x_N) & \dots & \phi_k(x_N) \end{bmatrix} \begin{bmatrix} b \\ w_1 \\ \vdots \\ w_k \end{bmatrix}$$

$$\vec{y} = X\vec{w}$$

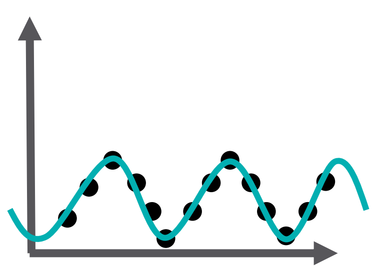
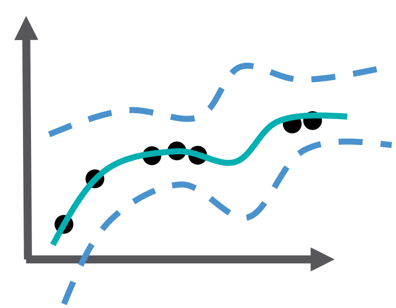
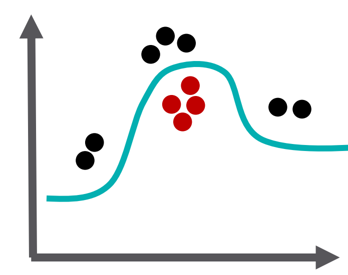
Output vector

Design matrix

Weight vector

Nonlinear regression

Nonlinear optimization	Gaussian Processes	Support Vector Machines
- Specify set of parameters	- Leveraging kernel methods	- Leveraging kernel methods
- Solve nonlinear optimization problems	- Parameter-free	- Parameter-free
	- Bayesian interpretation	- Commonly applied to classification problems

Workflow of Tuning a Model

01 – Select model

- Select a basis function model type
- Define the number of basis functions

02 – Compute parameters

- Select a <u>loss function</u> which measures „how good the model fits the data"
- Define constraints on the parameters
- Apply math or iterative algorithms to compute the parameters

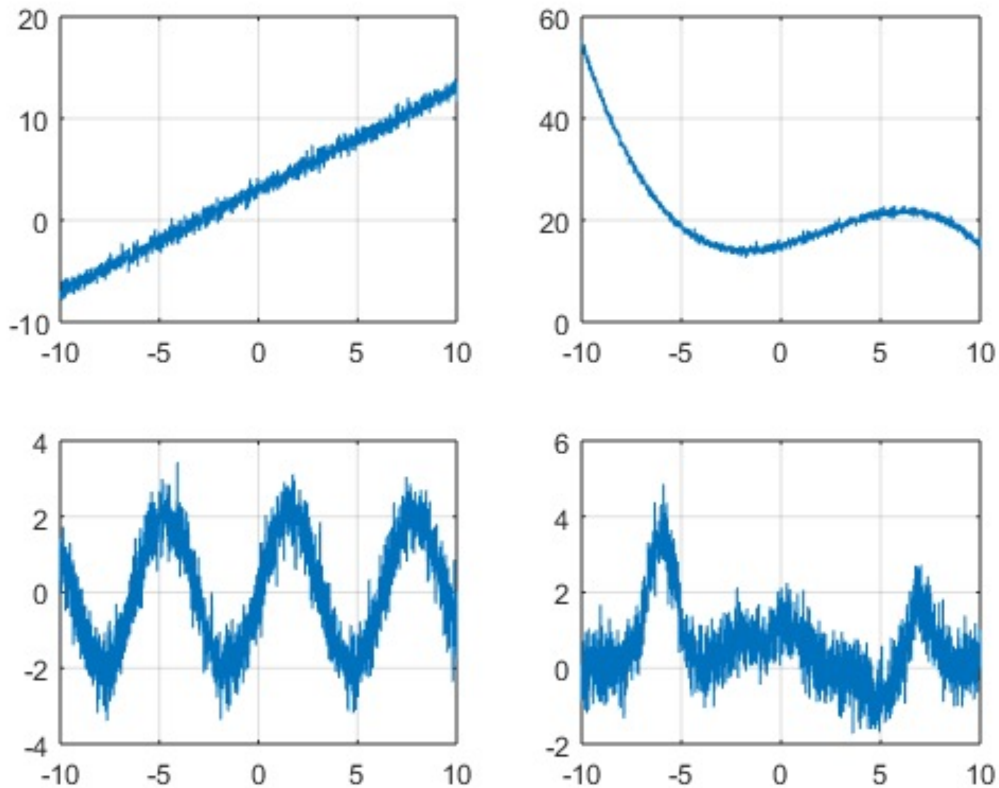
03 – Validate model

- Use a second dataset which was not used for training to evaluate the performance of your model
- Validated? Generalize to unseen data

Basis Function Models – Examples

Linear functions (e.g. ax + b)

Polynomial functions (e.g. x^2, x^3, …)



Sinusoidal functions (e.g. sin x)

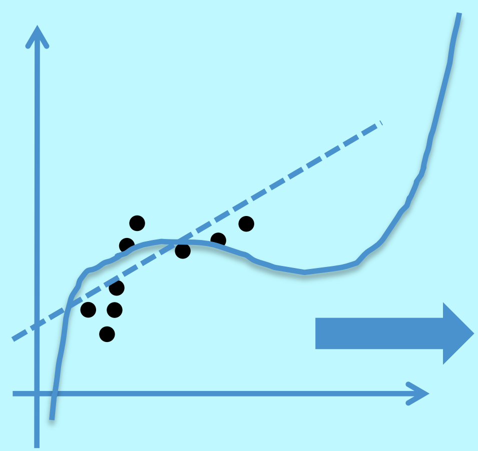
Gaussian basis functions (e.g. bell curve)

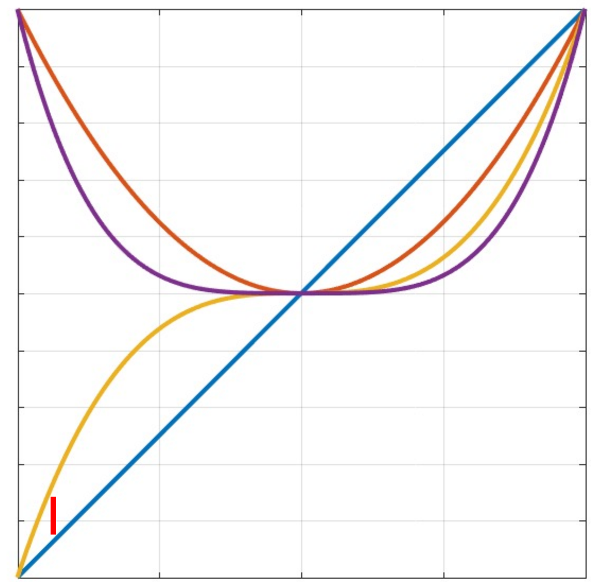
Polynomial Basis Function Models

- <u>Imbalanced data set challenge:</u>
  Globally defined on a probably "imbalanced" data set
- <u>Numerical challenge:</u>
  Design matrix may get ill-conditioned for large input domain parameters (multicolineartiy)

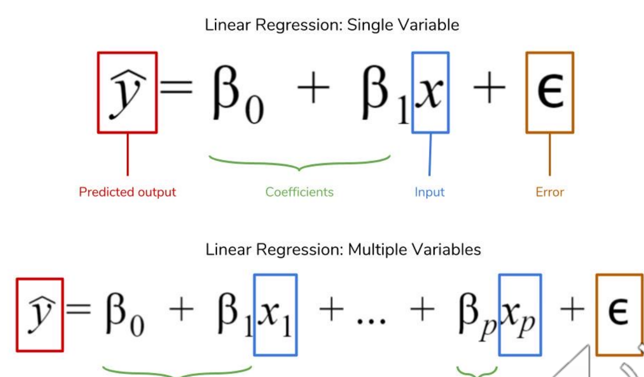
- Hyper parameters:
  - Polynomial degree

What does imbalanced mean?



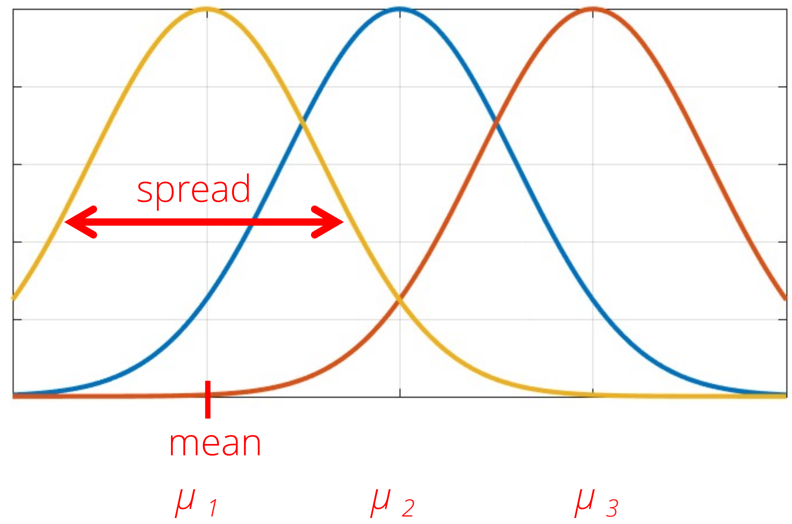


$x, x^2, x^3, \ldots, x^i$



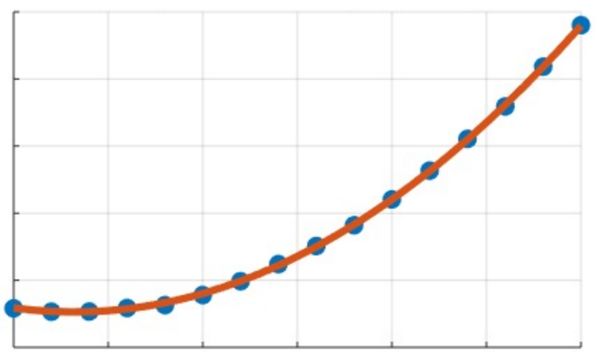
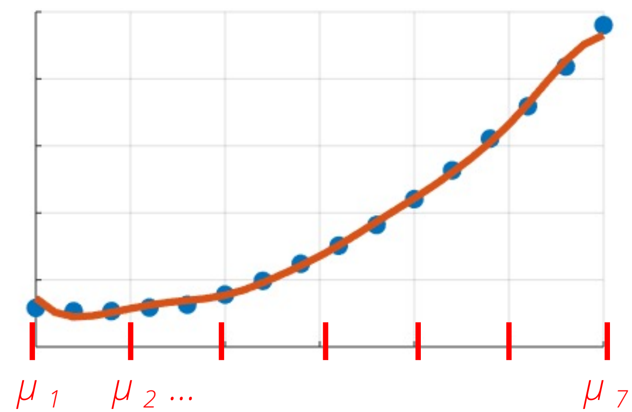
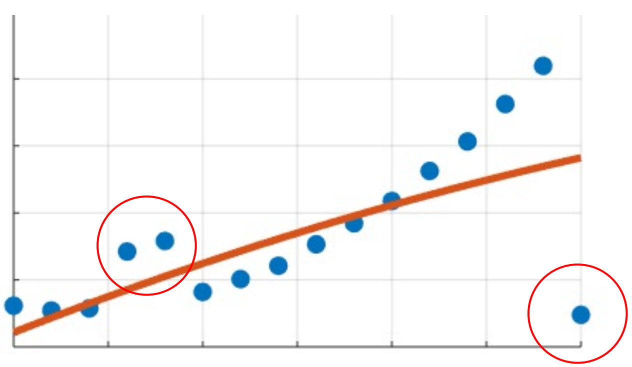
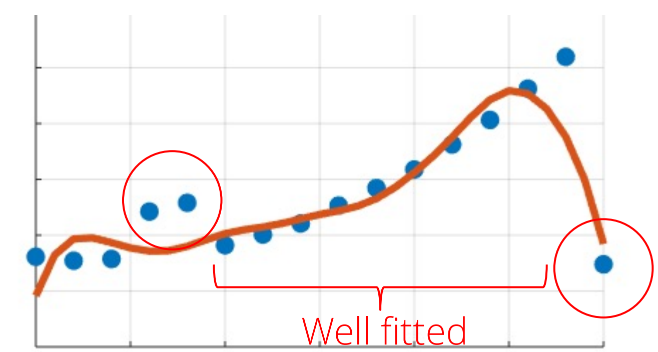
Gaussian Basis Function Models

- Locally defined on the independent variable domain
- Sparse design matrix
- Smoothness: Infinitely differentiable
- Hyper parameters:
  - Number of Gaussian functions
  - Spread *s* per basis function
  - Mean *μ* per basis function



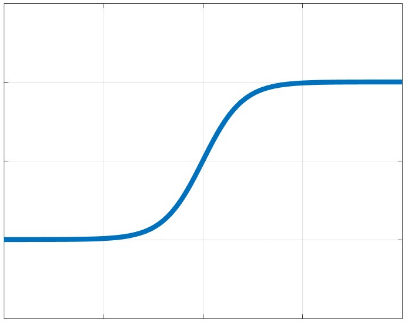
$$\phi_j(x) = \exp\left\{-\frac{(x-\mu_j)^2}{2s^2}\right\}$$

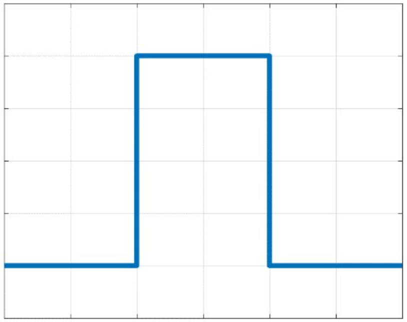
Comparison of both

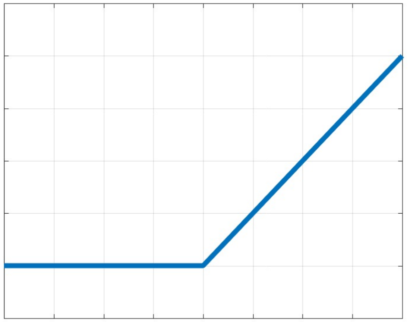
	Polynomial quadratic	Gaussian Spread parameter: 0.3
Data set x^2		
Data set with outliers		

Further Basis Function Models

Sigmoid Functions	Bin functions	Partly linear
- Bounded	- Bounded	- Not bounded
- Globally defined	- Locally defined	- Globally defined
	- Not differentiable	







Loss functions

- The loss functions measures the accuracy (or the fit) of the model in relation to the training dataset

- The best achievable model is the model with minimal loss ☺
  - A loss function is an essential ingredient for the regression problem

$$\underset{\vec{w}}{\text{minimize}}\ L(\vec{x}, \vec{y}, \vec{w})$$

  - Minimize the loss function *L* for the training dataset (taking all independent variables and target variables into consideration; adapting the basis function model weights) .

Loss function #1: Mean Square Error (MSE or L2)

Pros:

- De-facto standard
- Solution can be simply computed analytically

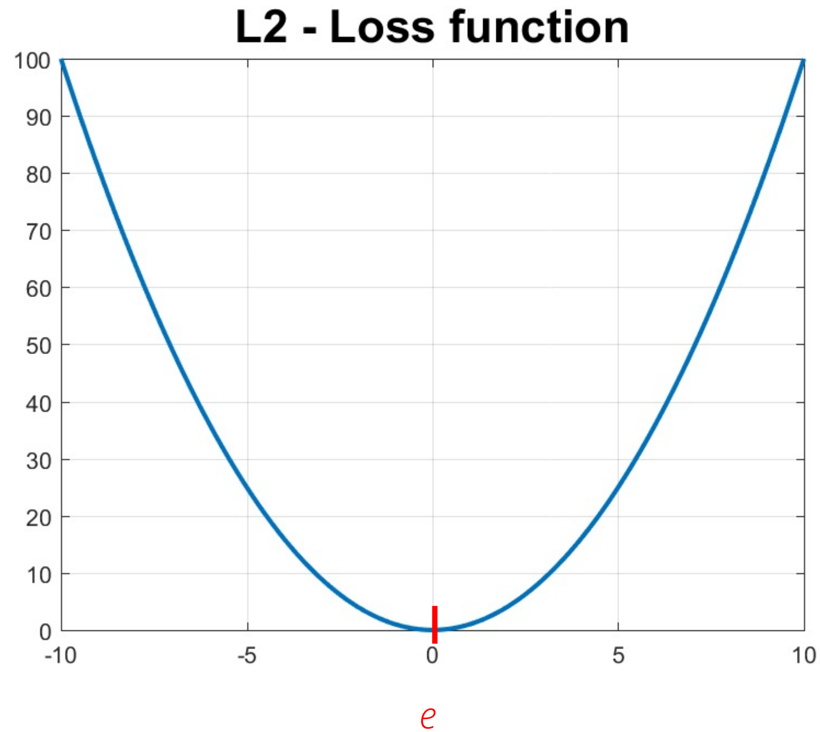
Cons:

- No robustness for outliers

Example applications:

- Should be initial start point for regression
- Starting point for training neural networks in many ML suites like Tensorflow etc

$$MSE = \frac{\sum_{i=1}^{n} (y_i - y_i^p)^2}{n}$$



https://heartbeat.fritz.ai/5-regression-loss-functions-all-machine-learners-should-know-4fb140e9d4b0

Loss function #2: Mean Absolute Error (MAE or L1)

$$MAE = \frac{\sum_{i=1}^{n} |y_i - y_i^p|}{n}$$

Pros:

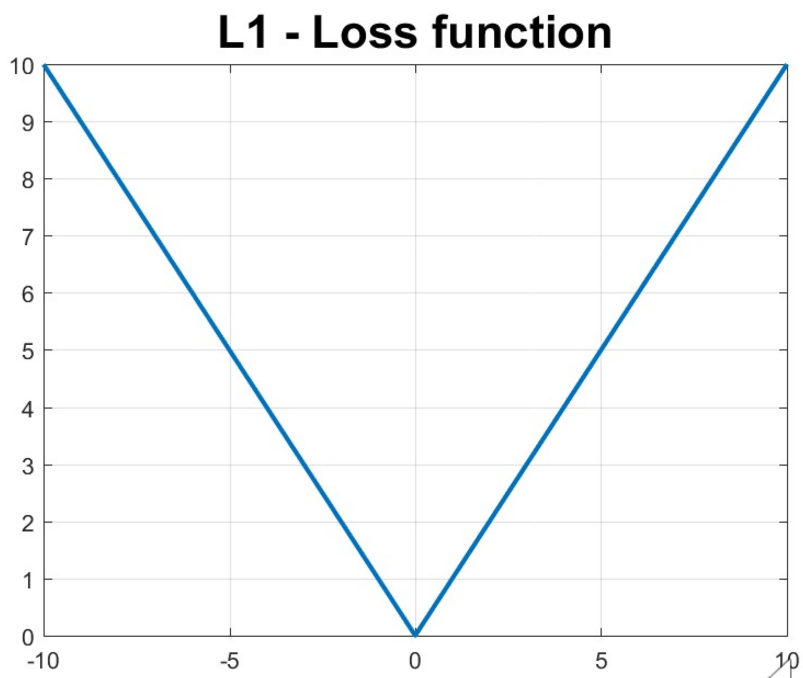
- Robustness for outliers

Cons:

- No smoothness → no analytical solution
- Non-differentiable in (0, 0)

Example applications:

- Financial applications



Loss function #3: Huber Loss

$$L = \sum \begin{cases} 0.5\,(y-\hat{y})^2\,, & \text{for } |\, y-\hat{y} \,| \leq \delta \\ \delta \,|\, y-\hat{y} \,|\, -0.5\delta^2, & \text{otherwise} \end{cases}$$

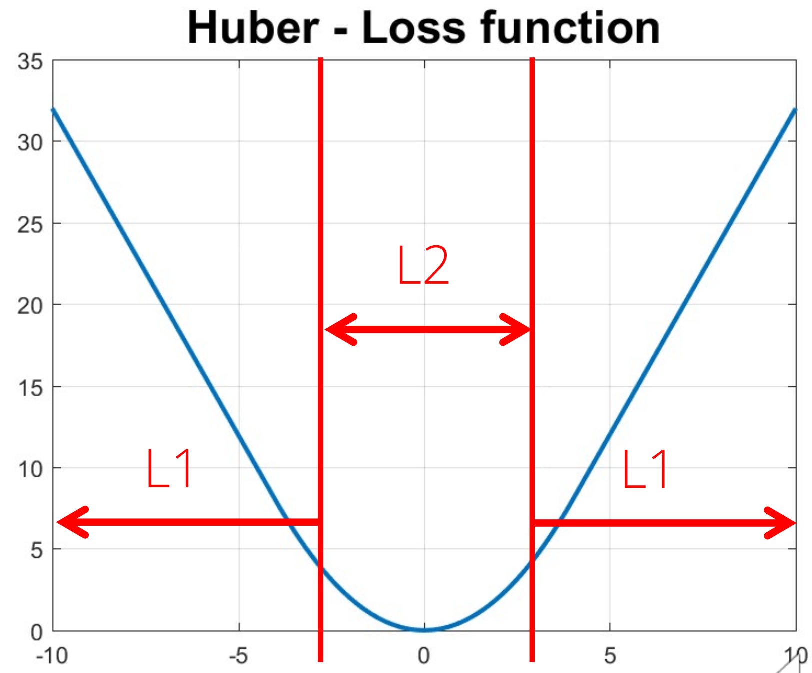
Pros:

- Combination of the L1 and L2 loss functions
- Robust + differentiable

Cons:

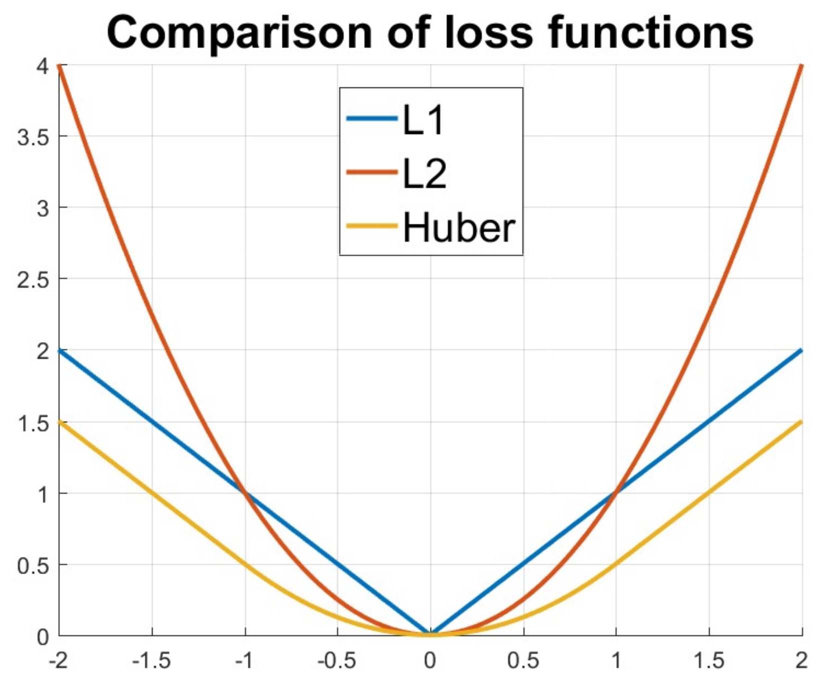
- More hyper parameters
- No analytical solution

e



Summary of all Loss functions

- L2 loss is differentiable
- L1 loss is more intuitive
- Huber Loss combines the best of both



<u>Practical hints:</u>

- Start with L2 loss as de-facto standard whenever possible
- Use domain knowledge! Any physical insights available? What is your objective?

Getting our Hands dirty: Low Dimensional Example

Solving the following optimization problem

$$\min_{\vec{w}} \frac{1}{2} \sum (y - \hat{y})^2$$

with the model $\hat{y} = w_1 x + b$

A zero derivative means that the function has some special behavior at the given point. It may have a local maximum, a local minimum, (or in some cases, as we will see later, a "turning" point)

Applying the model and data points $\{x_1, y_1\}, \{x_2, y_2\}, \{x_3, y_3\}$

$$\min_{w_1, b} \frac{1}{2} \left[(y_1 - w_1 x_1 - b)^2 + (y_2 - w_1 x_2 - b)^2 + (y_3 - w_1 x_3 - b)^2 \right]$$

In general, optimal solutions are found at the points where a zero derivative is found.

Getting our Hands dirty: Low Dimensional Example

$$\nabla = \left(\frac{\partial}{\partial x_1}, \ldots, \frac{\partial}{\partial x_n}\right).$$

Equivalently, the Laplacian of f is the sum of all the *unmixed* second partial derivatives in the Cartesian coordinates x_i:

$$\Delta f = \sum_{i=1}^{n} \frac{\partial^2 f}{\partial x_i^2}$$

Calculate the partial derivatives

$$\nabla L(x, y, w) = \begin{bmatrix} \frac{\partial L(x,y,w)}{\partial w_1} \\ \frac{\partial L(x,y,w)}{\partial b} \end{bmatrix}$$

$$= \begin{bmatrix} -(y_1 - w_1 x_1 - b)\, x_1 - (y_2 - w_1 x_2 - b)\, x_2 - (y_3 - w_1 x_3 - b)\, x_3 \\ -(y_1 - w_1 x_1 - b) - (y_2 - w_1 x_2 - b) - (y_3 - w_1 x_3 - b) \end{bmatrix}$$

$$= \begin{bmatrix} x_1^2 + x_2^2 + x_3^2 & x_1 + x_2 + x_3 \\ x_1 + x_2 + x_3 & 3 \end{bmatrix} \begin{bmatrix} w_1 \\ b \end{bmatrix} - \begin{bmatrix} y_1 x_1 + y_2 x_2 + y_3 x_3 \\ y_1 + y_2 + y_3 \end{bmatrix}$$

and set it equal to zero

$$\begin{bmatrix} x_1^2 + x_2^2 + x_3^2 & x_1 + x_2 + x_3 \\ x_1 + x_2 + x_3 & 3 \end{bmatrix} \begin{bmatrix} w_1 \\ b \end{bmatrix} - \begin{bmatrix} y_1 x_1 + y_2 x_2 + y_3 x_3 \\ y_1 + y_2 + y_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Getting our Hands dirty: Low Dimensional Example

Solving the resulting equation (also named normal equation):

$$\begin{bmatrix} x_1^2 + x_2^2 + x_3^2 & x_1 + x_2 + x_3 \\ x_1 + x_2 + x_3 & 3 \end{bmatrix} \begin{bmatrix} w_1 \\ b \end{bmatrix} - \begin{bmatrix} y_1 x_1 + y_2 x_2 + y_3 x_3 \\ y_1 + y_2 + y_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} w_1 \\ b \end{bmatrix} = \begin{bmatrix} x_1^2 + x_2^2 + x_3^2 & x_1 + x_2 + x_3 \\ x_1 + x_2 + x_3 & 3 \end{bmatrix}^{-1} \begin{bmatrix} y_1 x_1 + y_2 x_2 + y_3 x_3 \\ y_1 + y_2 + y_3 \end{bmatrix}$$

Generalizing the Analytic Solution

- Minimizing MSE loss function can be rewritten in matrix form

$$\min_{\vec{w}} \frac{1}{2} \sum (y - \hat{y})^2$$

$$\min_{\vec{w}} \frac{1}{2} (\vec{y} - X\vec{w})^T (\vec{y} - X\vec{w})$$

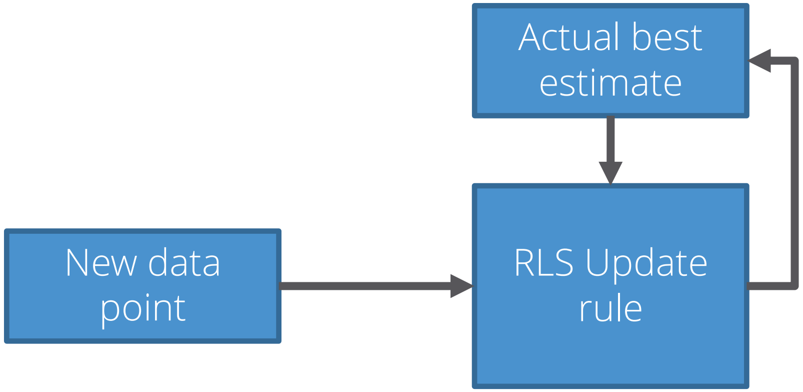
- Optimum value for is equal to setting the gradient to zero and solve for

$$\vec{w} = (X^T X)^{-1} X^T \vec{y}$$

- The importance of this loss function is tightly related to the fact that the analytical solution is available and can be calculated explicitly for low- to medium sized datasets!

Sequential Analytic Solution - Motivation

- Consider the following streaming use case:
  - Apply regression online, i.e. during operation of the product (e.g. for predicting the load time of a website according to the visits)
  - It's a big website with heavy load: There is not enough memory to store all data points ☺
- A possible solution is given by Recursive Least Squares (RLS)



Sequential Analytic Solution – The algorithm

- New data point triggers an update of the parameters

$$\vec{w}(k+1) = \vec{w}(k) + \underbrace{P(k)\bar{x}^T \left(I + \bar{x}^T P(k)\bar{x}\right)^{-1}}_{\text{Correction gain}} \underbrace{\left(\bar{y} - \bar{x}^T \vec{w}(k)\right)}_{\text{Residual}}$$

Old parameter estimate: $\vec{w}(k)$

Prediction based on old parameters: $\bar{x}^T \vec{w}(k)$

based on the memory matrix

$$P(k+1) = \left(I - P(k)\bar{x}^T \left(I + \bar{x}^T P(k)\bar{x}\right)^{-1} \bar{x}\right) P(k)$$

with I being the identity matrix of appropriate dimension

Sequential Analytic Solution – Forgetting factor

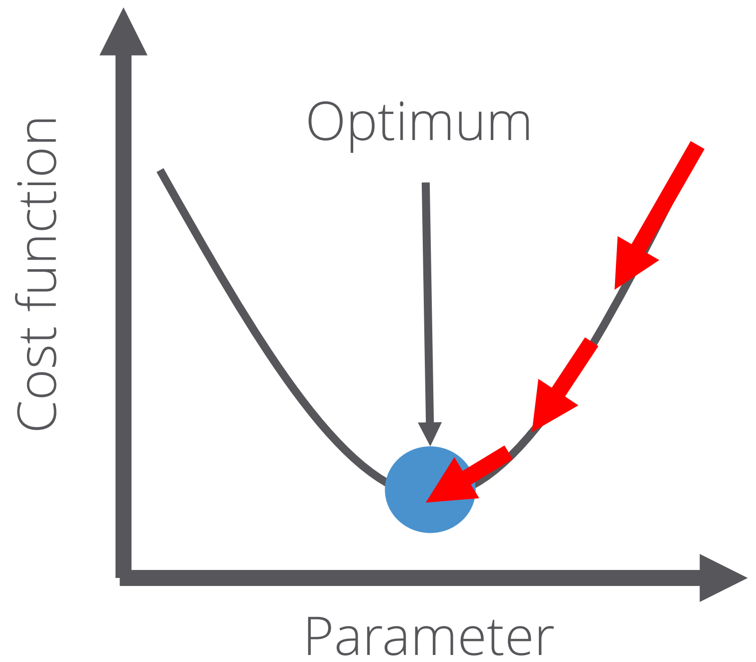
- Some applications show slowly varying conditions in the long term, but can be considered stationary on short to medium time periods
  - Aging of products leads to slight parameter changes
  - Vehicle mass is usually constant over a significant period of time
- The RLS algorithm can deal with this by introduction of a forgetting factor . This leads to a reduction of weight for old samples.

$$\vec{w}(k+1) = \vec{w}(k) + P(k)\bar{x}^T \left(\gamma I + \bar{x}^T P(k)\bar{x}\right)^{-1} \left(\bar{y} - \bar{x}^T \vec{w}(k)\right)$$

$$P(k+1) = \gamma^{-1} \left(I - P(k)\bar{x}^T \left(\gamma I + \bar{x}^T P(k)\bar{x}\right)^{-1} \bar{x}\right) P(k)$$

Numerical Iterative Solutions – e.g. Gradient Descent

- Regression can be solved numerically
- Important for large-scale problems and for non-quadratic loss functions
- Popular methods:
  - Gradient descent
  - Gauss-Newton
  - Levenberg-Marquardt



Pros:

- Very generic

Cons:

- Knowledge about numeric optimization necessary

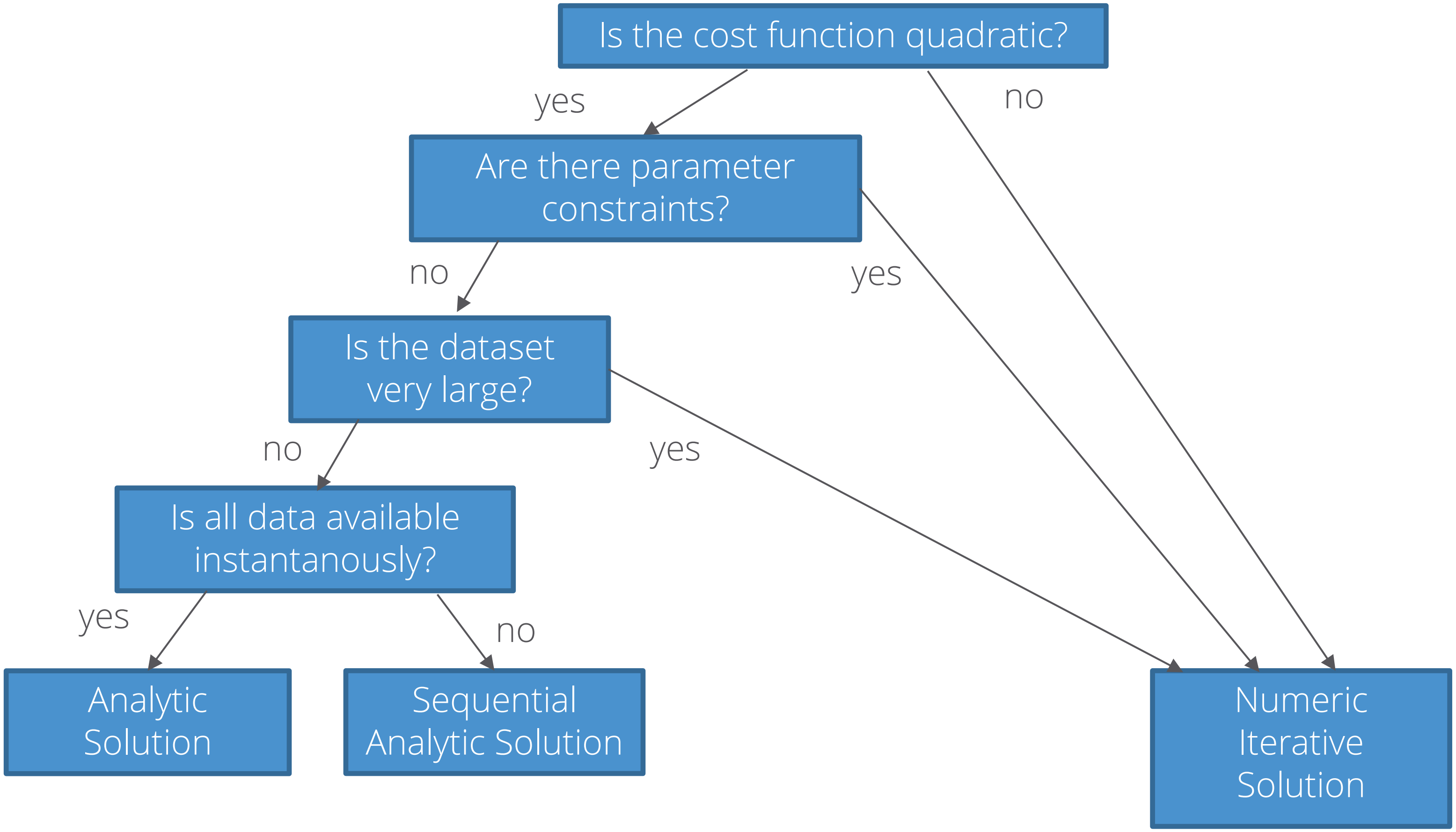
Constraining the Weights

- Weights can be interpreted as physical quantities
  - Temperature (non-negative)
  - Spring constants (non-negative)
  - Mass (non-negative)
- A valid range is known for the weights
  - Tire and other friction models
  - Efficiency (0 – 100 %)

$$\underset{\vec{w}}{\text{minimize}} \; L(\vec{x}, \vec{y}, \vec{w})$$
$$\text{subject to } \vec{c}_1 \leq \vec{w} \leq \vec{c}_2$$

- Improves robustness
- More difficult to solve

A Decision Tree for Solving the Regression Problem



Agenda

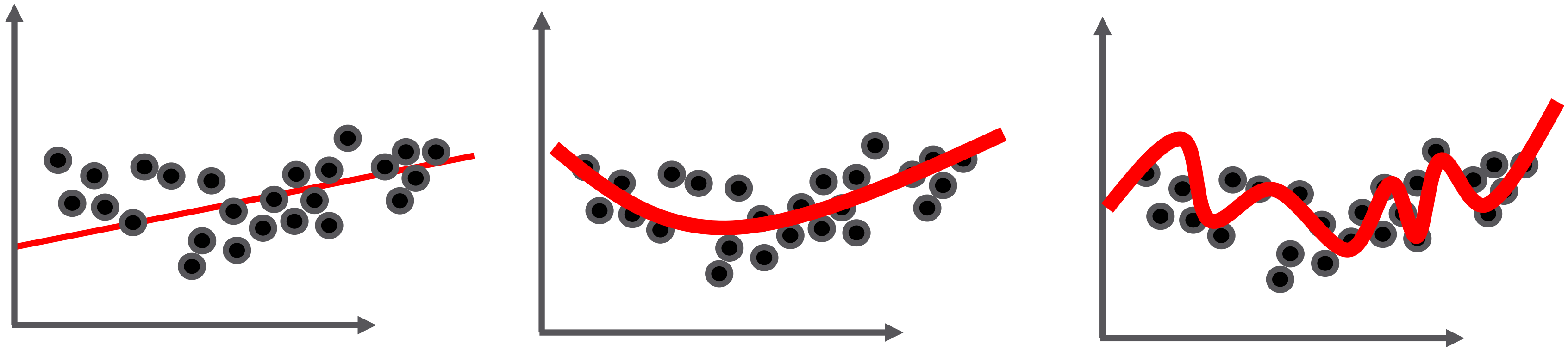
01 Motivation

02 (Non-)Linear Models & Loss functions

03 **Regularization & Validation**

04 Summary

How much Fitting is good for the Model?



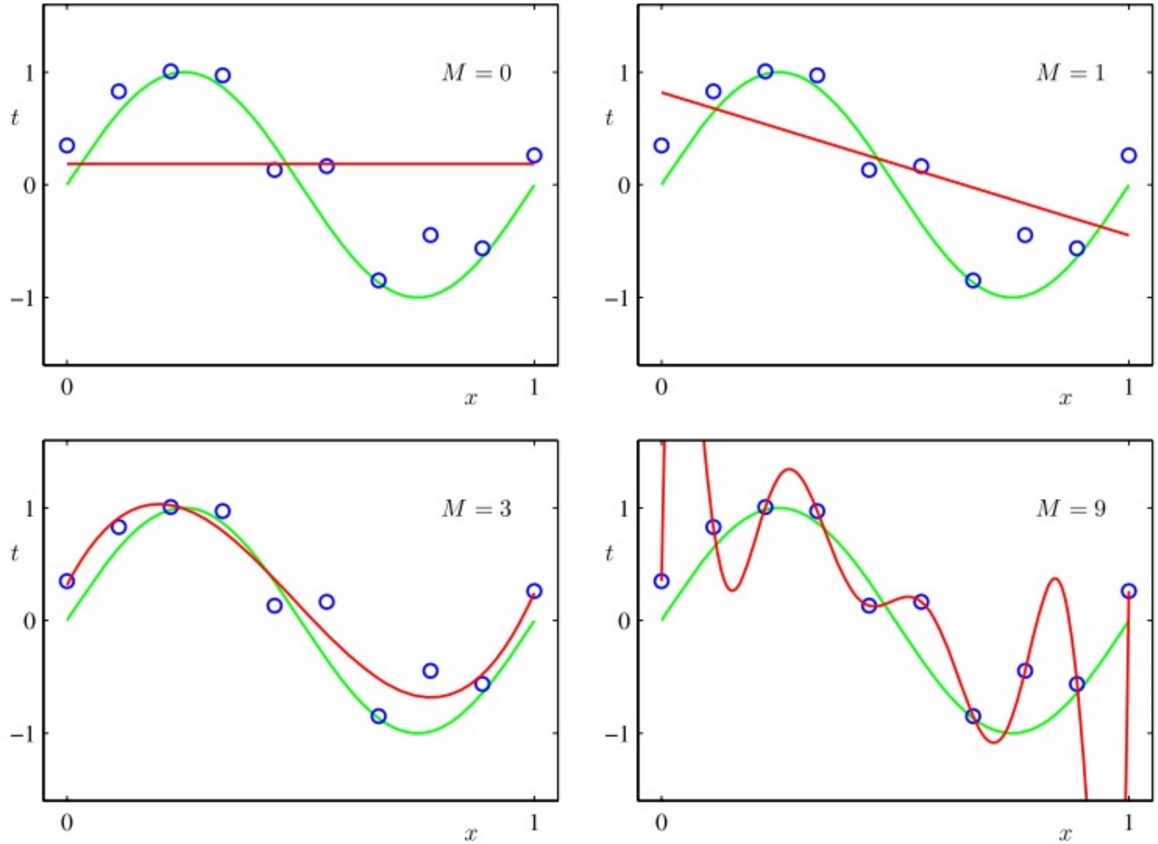
Underfitting | Well fitted | Overfitting

- Insufficient number of features
- False reconstruction

- Too many features considered
- Irrelevant features

Overfitting due to Model Complextiy

- Overfitting means: Model is not able to properly generalize; sticks too much to the given data points
- Increasing the model complexity diminishes the value of the cost function
- Outliers and noise are taken too much into consideration

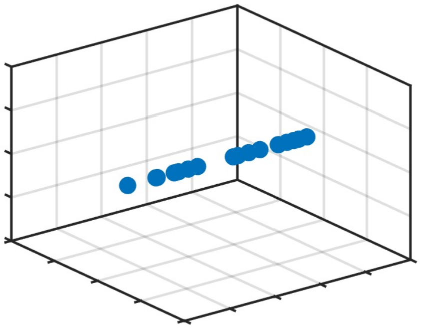


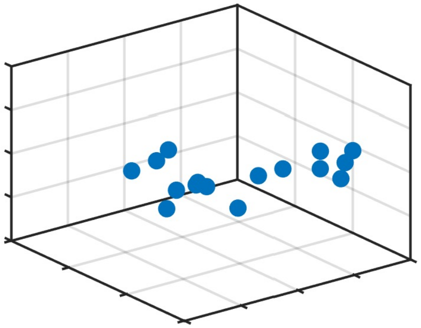
Source:: Bishop – Pattern Recognition and Machine Learning

Overfitting due to Curse of Dimensionality

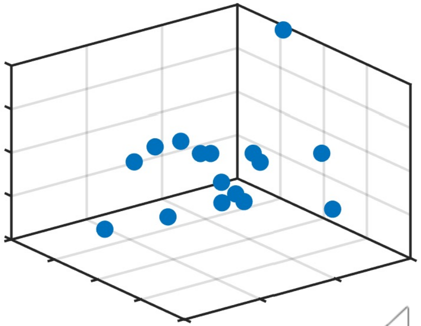
- To sum it up: Overfitting occurs if
  1. data points are sparse or
  2. model complexity is high

- Data get sparse, if many input dimensions are taken into consideration



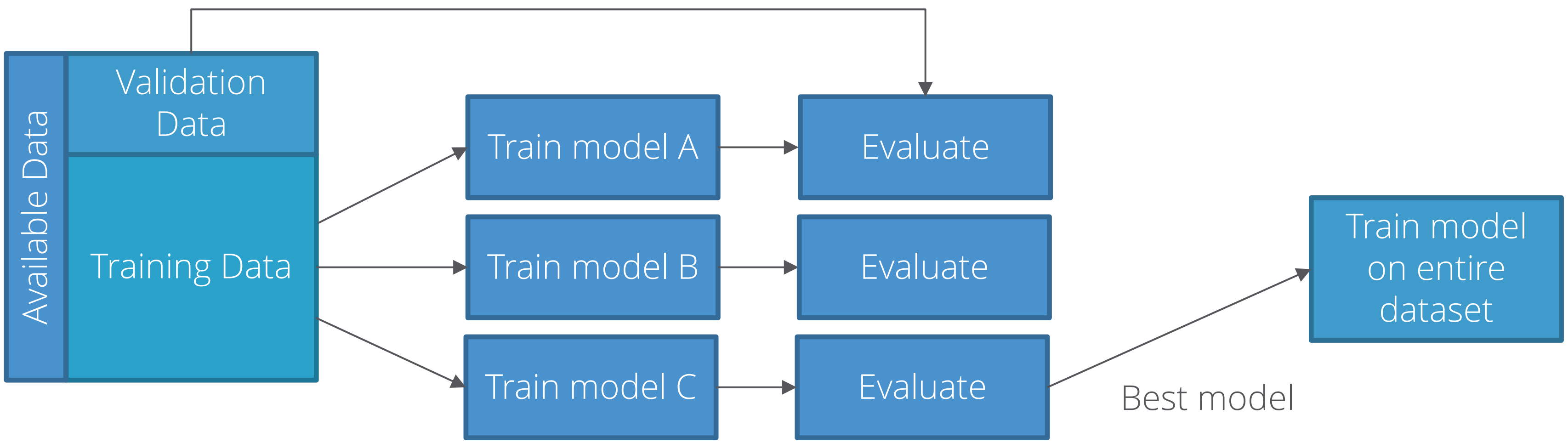


16 samples
In a 1-, 2 and
3-dimensional
space



Avoiding Overfitting through Validation Datasets

- Hard to easily detect overfitting in high-dimensional domains and complex systems
- A off-the-shelf technique is to split the data into training and validation data sets



Avoiding Overfitting through Validation Datasets

- You have to detect the turning point when the optimization of your hyper parameters leads to <u>poor generalization</u>

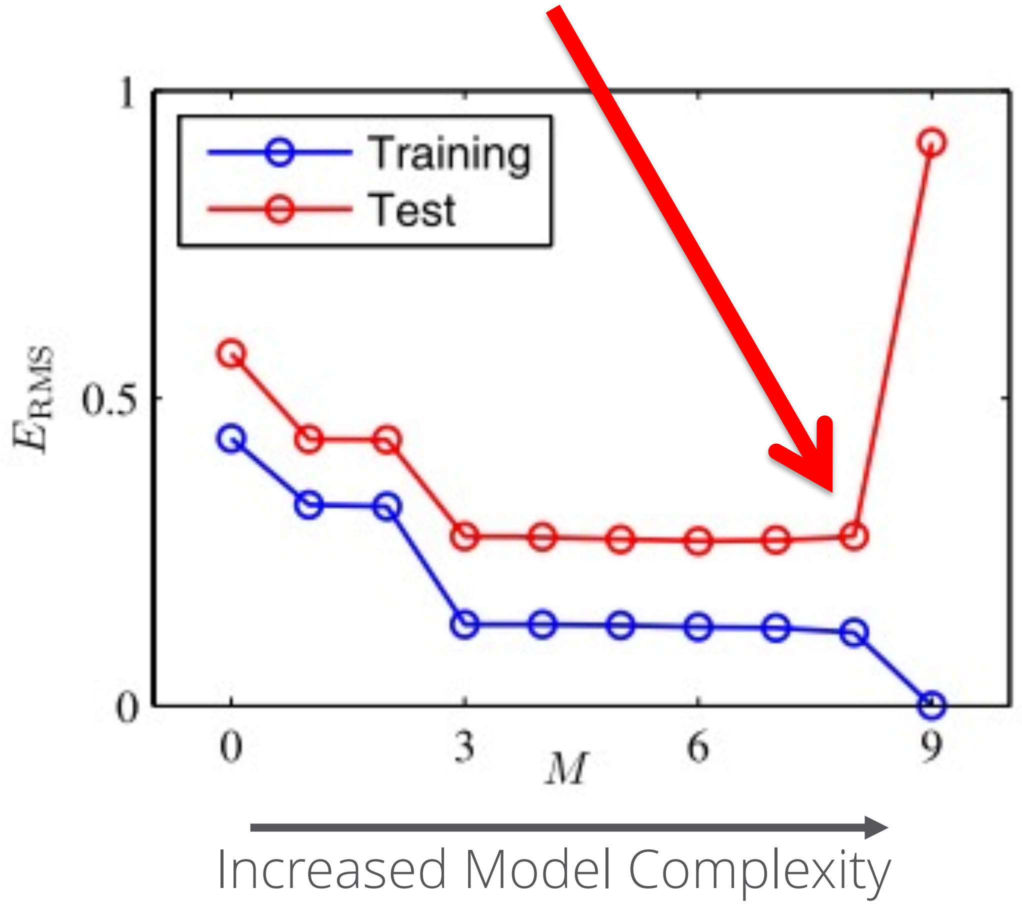


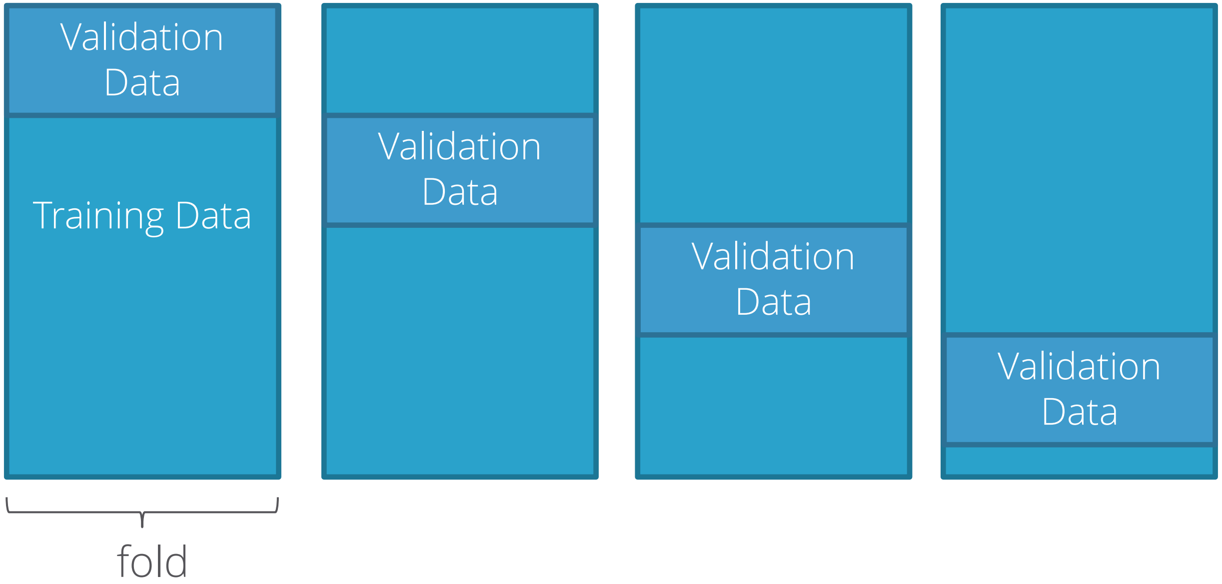
Figure source: Bishop – Pattern Recognition and Machine Learning

Common Recommendation when Working with Validation Datasets

1. Validation and training datasets need to have a similar distribution
2. Balance the risk by not reusing validation datasets
3. Split the data before fitting the model and take 2/3 of the data as training data set.

k-Fold Cross-Validation

- In case of limited data size sets, one is seduced to use a big portion of the data for training and only a small portion for validation
  → don't do that
- Instead: Use smaller validations sets to estimate the true prediction error by splitting the data into multiple ‚folds'
- The Variance of the estimation error over all folds is an indicator for model stability

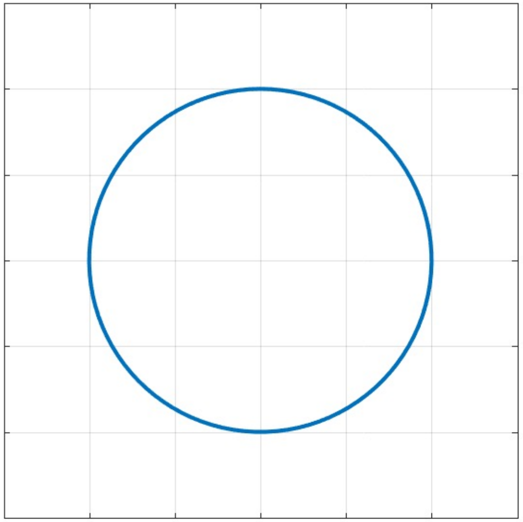


Regularization

- From a design perspective, we want to select the model structure based on underlying physical principles and not on the characteristics of the dataset
  - Polynomial basis functions tend to have large coefficients for sparse data sets
  - Gaussian basis functions have the tendency to locally overfit, which leads to single, large coefficients
- A method to circumvent this is <u>regularization:</u>
  1. Penalize high coefficients in the optimization prevents these effects
  2. Weighting of penalty term gives an intuitive hyper parameter to manage model complexity

Regularization #1: Ridge Regression

- Also known as: L2 regularization, Thikonov regularization
- Can prevent overfitting pretty well
- Analytic solution is available as it is ar extension of the MSE problem
- Hard to apply and fine-tune in high-dimensional feature spaces



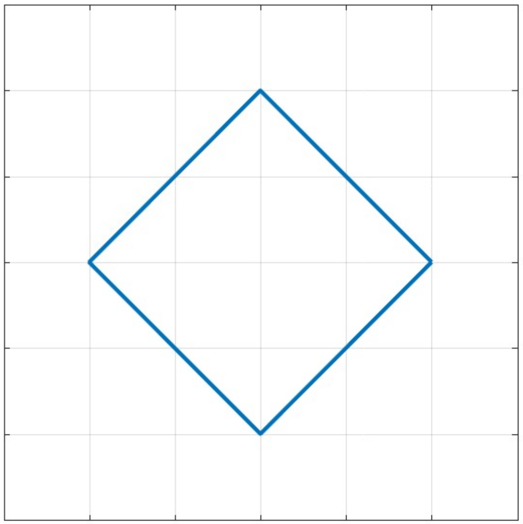
$$\min_{\vec{w}} L(\vec{x}, \vec{y}, \vec{w}) + \lambda \vec{w}^T \vec{w}$$

Regularization Term

Regularization #2: Lasso Regression

- Also known as: L1 regularization
- Tends to produce sparse solutions and can therefore be applied for feature selection (PCA low)
- Sparse solution means, that several coefficients go to zero:

$$\vec{w} = \begin{bmatrix} 0 & w_1 & 0 & 0 & 0 & w_2 \end{bmatrix}$$



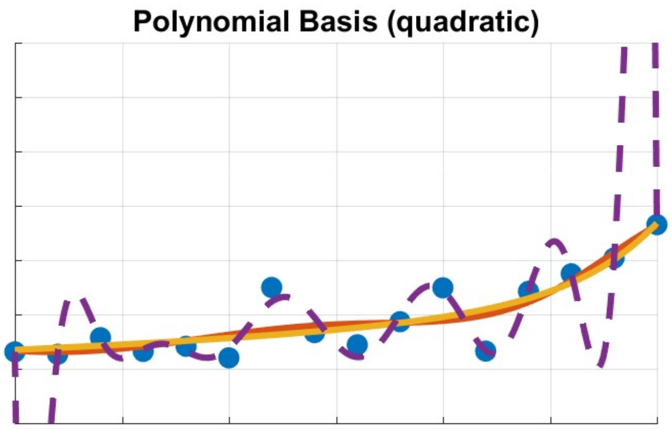
$$\min_{\vec{w}} L(\vec{x}, \vec{y}, \vec{w}) + \lambda \sum_i | w_i |$$

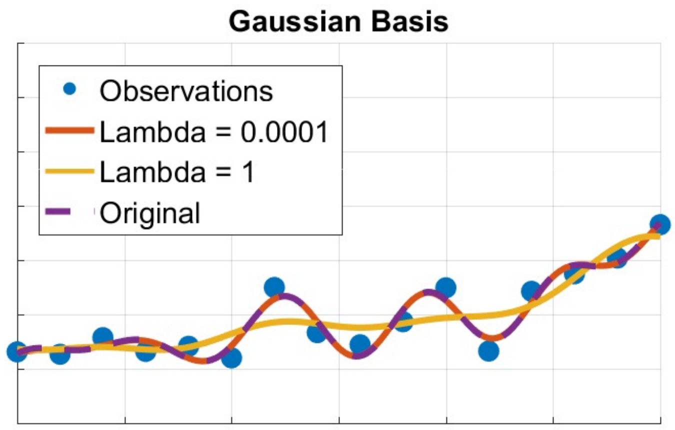
Regularization Term

Application of Regularization

- Regularized solutions perform far better at interpolation

- <u>Keep in mind:</u> You must evaluate at points **between** your sample points





Application of Regularization – Understand when you're done

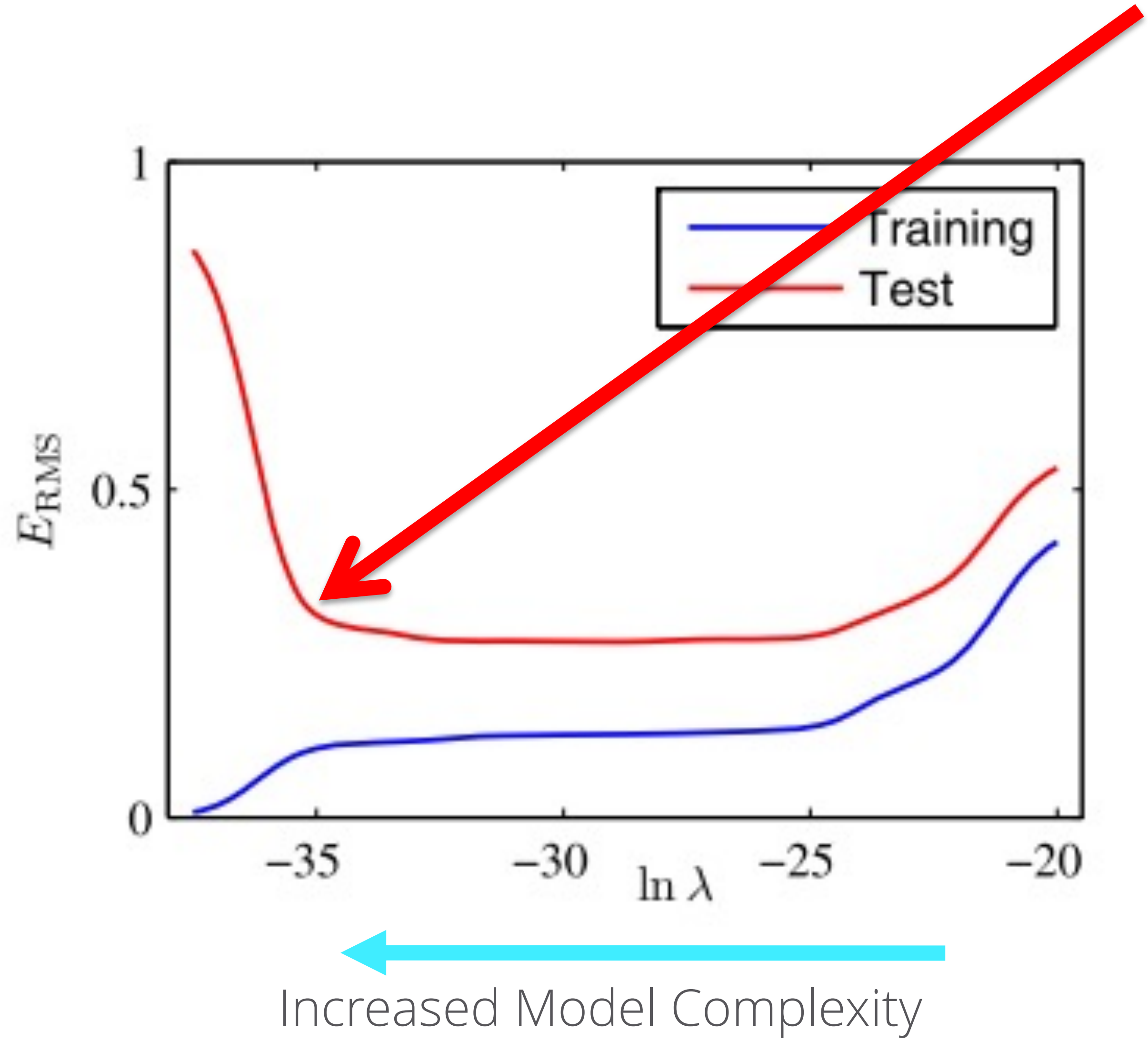


Figure source: Bishop – Pattern Recognition and Machine Learning

Agenda

01 Motivation

02 (Non-)Linear Models & Loss functions

03 Regularization & Validation

04 **Summary**

Summary

1. Why regression is a great method and what's the difference to clustering and classification
2. What's the difference between linear and nonlinear regression
3. What complex use cases and applications in the Automotive domain can be solved by applying regression (e.g. estimating the freight weight for commercial vehicles)
4. What's the difference between local and global basis functions
5. What kind of loss functions are available
6. When to use numerical iterative and analytical solution methods for model training
7. A rough understanding of regularization and validation technique