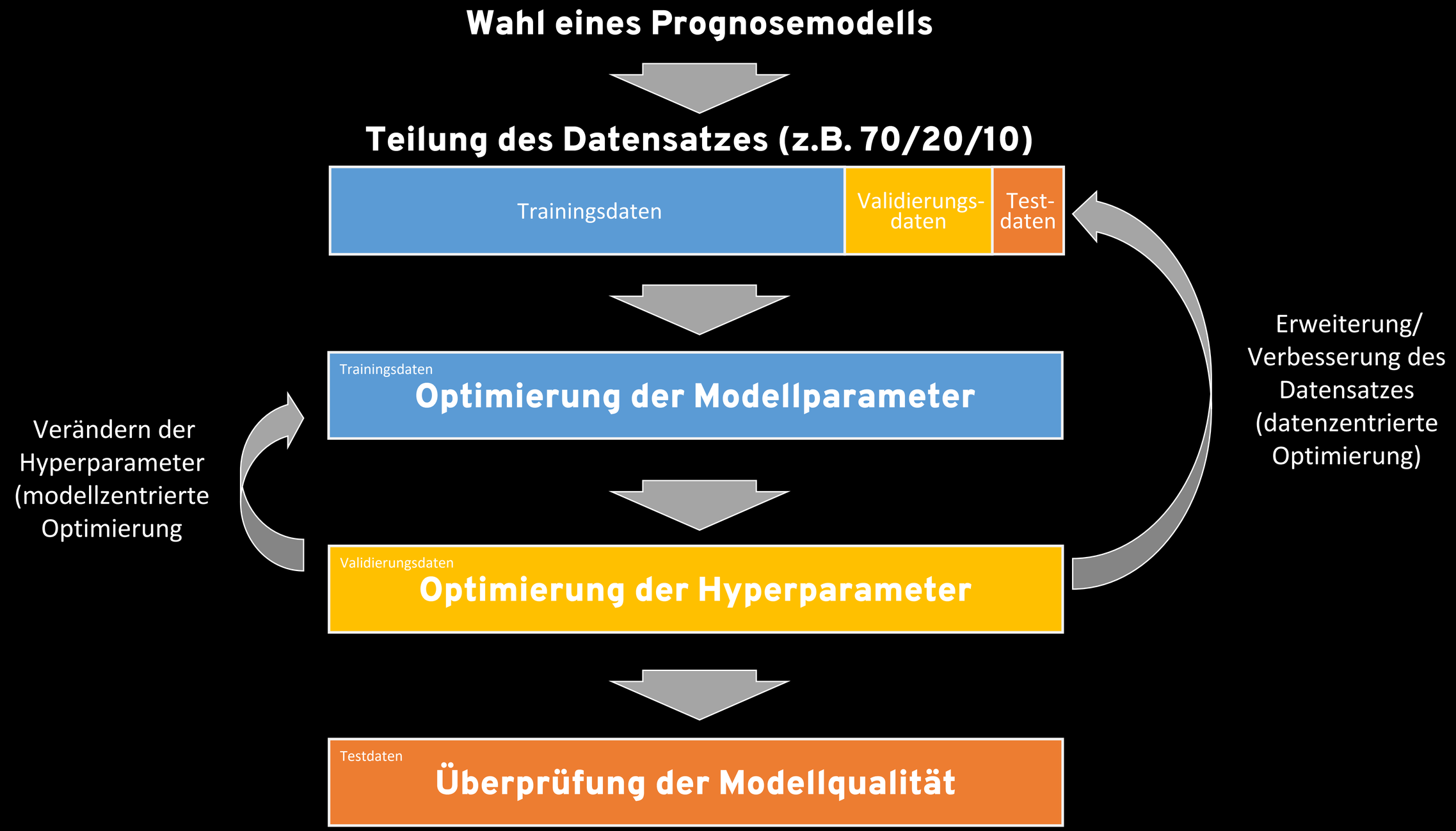
23.05.23

OPENCAMPUS.sh

Einführung in Data Science und maschinelles Lernen

OVERFITTING UND REGULARISIERUNG

- Modellgütekriterien
- Overfitting
- Regularisierung
- Einführung in neuronale Netze


Wahl eines Prognosemodells
Teilung des Datensatzes (z.B. 70/20/10)
Trainingsdaten
Validierungs-daten
Test-daten
Trainingsdaten
Optimierung der Modellparameter
Validierungsdaten
Optimierung der Hyperparameter
Testdaten
Überprüfung der Modellqualität
Verändern der Hyperparameter (modellzentrierte Optimierung
Erweiterung/ Verbesserung des Datensatzes (datenzentrierte Optimierung)

KOSTENFUNKTION

Zur Berechnung der Funktion mit den optimalen Parametern θ_0 und θ_1:

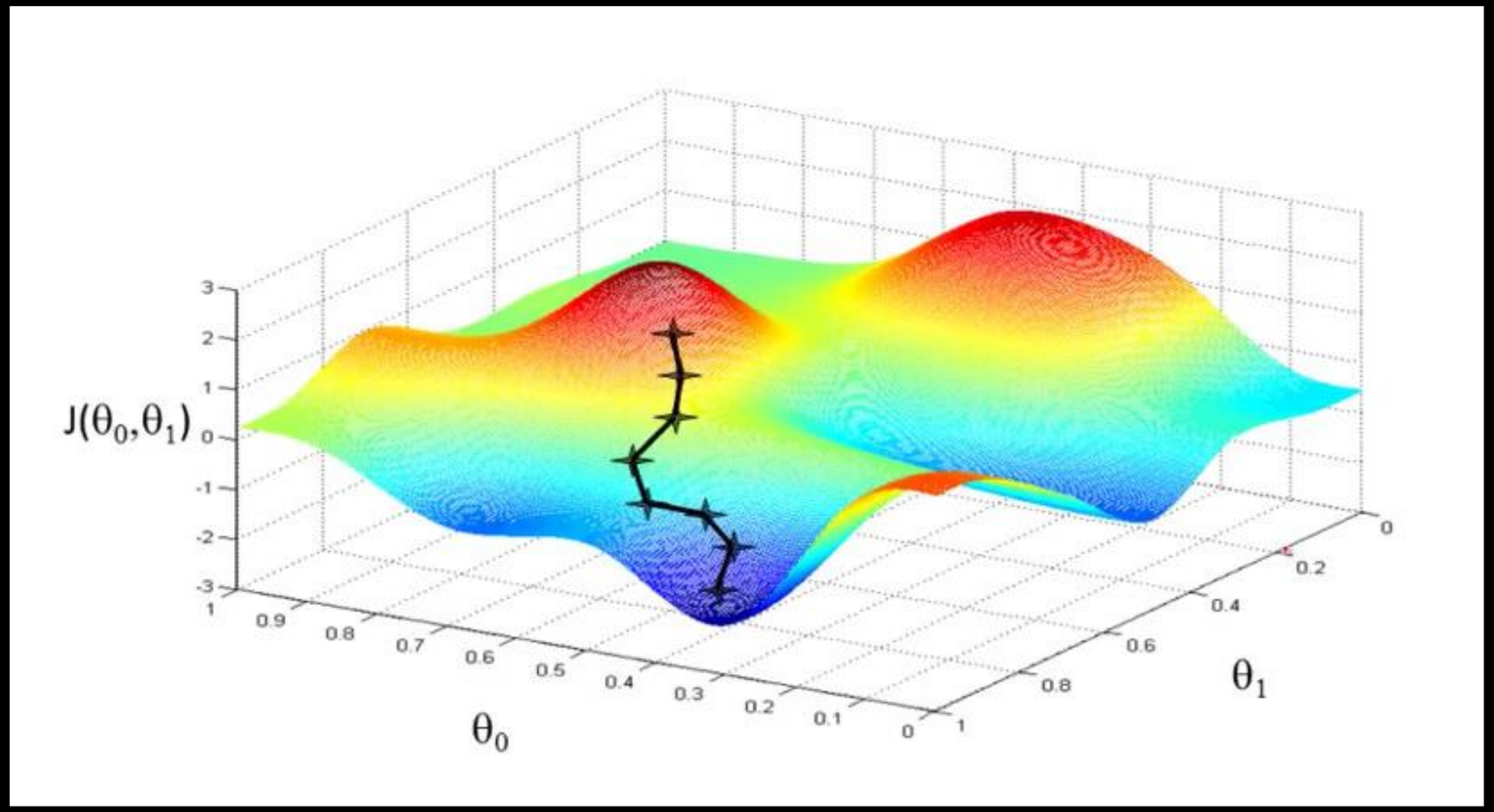
$$J_x(\theta_0, \theta_1) = \frac{1}{m} \sum |h_x(\theta_0, \theta_1) - y|$$

Mean Absolute Error (MAE)

$$J_x(\theta_0, \theta_1) = \frac{1}{m} \sum (h_x(\theta_0, \theta_1) - y)^2$$

Mean Squared Error (MSE)

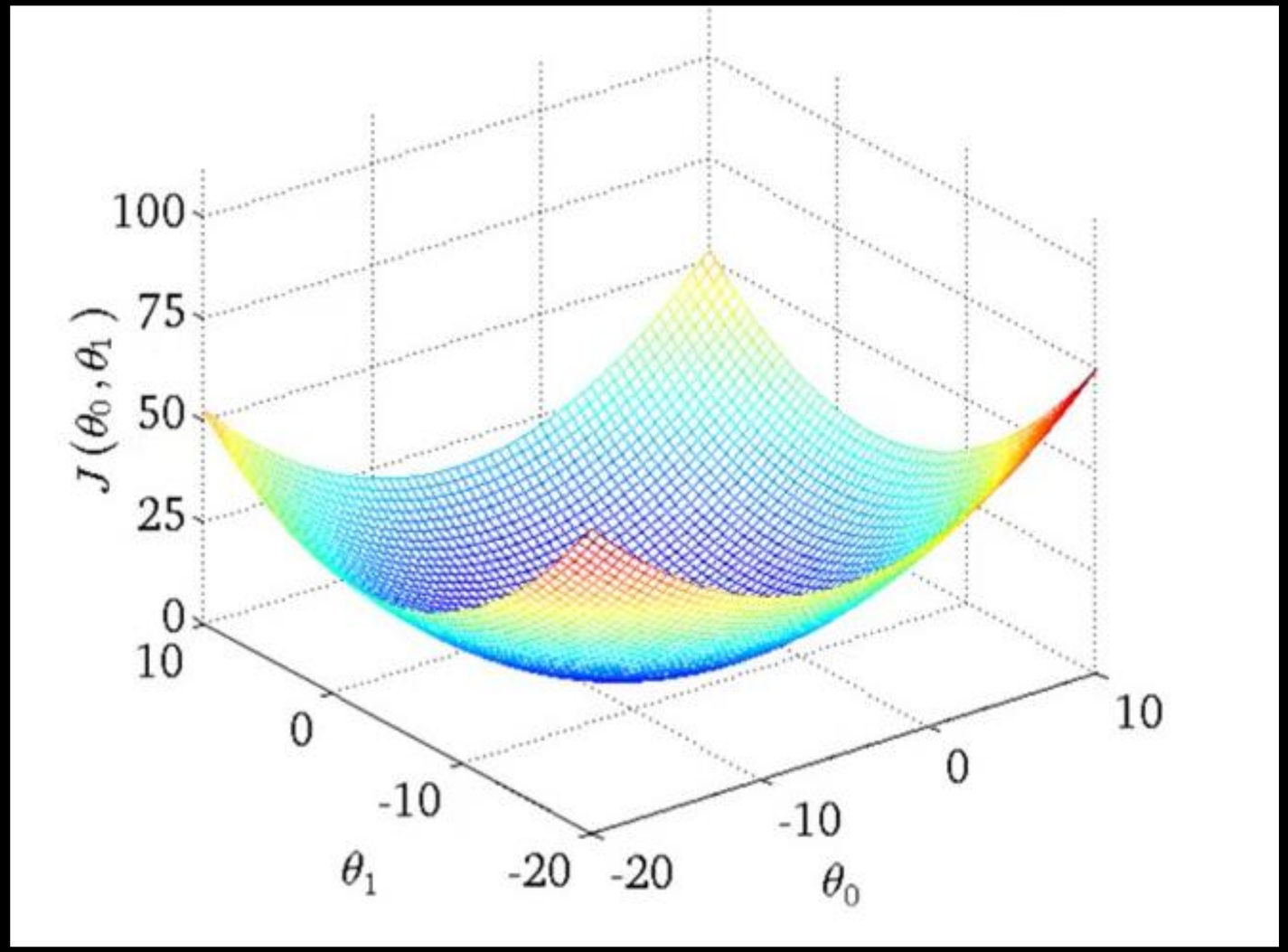
OPTIMIERUNGSFUNKTION



- **Iteratives Verfahren (Gradient Descent), um das Minimum der Kostenfunktion zu finden.**
- **Schrittgröße zur Annäherung wird durch die Lernrate („Learning Parameter“) kontrolliert**

Quelle: https://www.coursera.org/learn/machine-learning

MINIMIERUNG BEI LINEAREN MODELLEN



Quelle: https://www.coursera.org/learn/machine-learning

- **Für lineare Modelle ist die Kostenfunktion konvex und besitzt keine lokalen Minima.**
- **Hier werden häufig auch andere statistische Verfahren als Gradient Descent genutzt.**

  (Insbesondere wenn das Modell nur wenige Variablen umfasst.)

MODELLGÜTEKRITERIEN

errors:	forecast – actual	(auch: residuals)
mae:	mean(abs(errors))	
mape:	mean(abs(errors/actual))	
mse:	mean(errors^2)	
rmse:	sqrt(mean(errors^2))	
rse:	sum(errors^2) / sum((actual-mean(actual))^2)	
r^2 =	1 – rse	

Video (3 Minuten) mit Erklärung und Darstellung der Kriterien:
https://www.coursera.org/lecture/machine-learning-with-python/evaluation-metrics-in-regression-models-5SxtZ

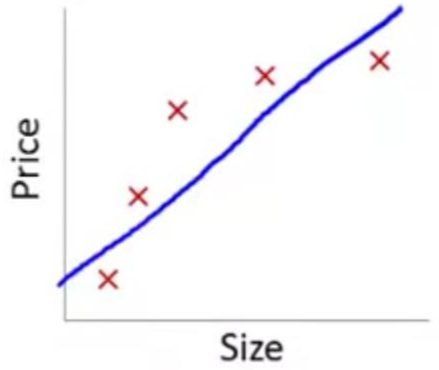
BREAKOUT

- Nutzt Euer lineares Modell, um eine Vorhersage für den Validierungsdatensatz zu erstellen.

- Vergleicht die Ergebnisse von Trainings- und Validierungsdatensatz.

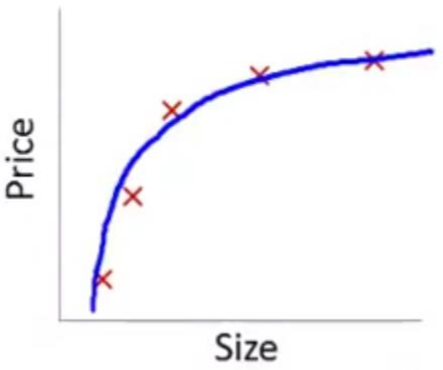
OVERFITTING

Example: Linear regression (housing prices)



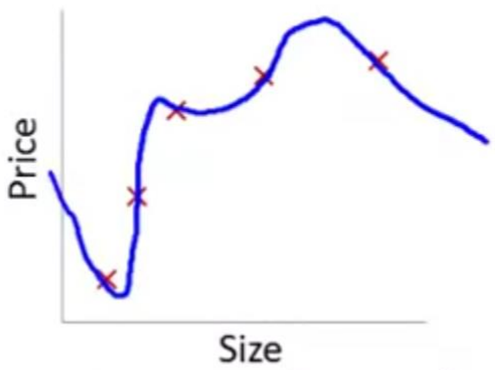
$\rightarrow \theta_0 + \theta_1 x$

"Underfit" "High bias"



$\rightarrow \theta_0 + \theta_1 x + \theta_2 x^2$

"Just right"



$\rightarrow \theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$

"Overfit" "High variance"

Overfitting: If we have too many features, the learned hypothesis may fit the training set very well ($J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 \approx 0$), but fail to generalize to new examples (predict prices on new examples).

Computer Science Engineering. (2017). *3.4.1 The Problem of Overfitting by Andrew Ng*. Abgerufen von https://www.youtube.com/watch?v=OSd30QGMl88

BEISPIEL ZU OVERFITTING

````
---
title: "Linear Regression"
output: html_notebook
---

```{r}
# Importing Function Packages
library(dplyr)
library(readr)
library(lubridate)
library(broom)
library(Metrics)
```


```{r}
# Importing Training and Test Data
house_pricing_train <- read_csv("./house_pricing_data/house_pricing_train.csv")
house_pricing_test <- read_csv("./house_pricing_data/house_pricing_test.csv")
```


```{r}
# Estimating (Training) Models
mod1 <- lm(price ~ bathrooms, house_pricing_train)
mod2 <- lm(price ~ as.factor(bathrooms), house_pricing_train)
mod3 <- lm(price ~ as.factor(bathrooms) + as.factor(zipcode), house_pricing_train)
mod4 <- lm(price ~ as.factor(bathrooms) + as.factor(zipcode) + condition, house_pricing_train)
mod5 <- lm(price ~ as.factor(bathrooms) + as.factor(zipcode) + as.factor(condition), house_pricing_train)
mod6 <- lm(price ~ as.factor(bathrooms) + as.factor(zipcode) + as.factor(condition) + sqft_living15, house_pricing_train)
mod7 <- lm(price ~ as.factor(bathrooms) + as.factor(zipcode) + as.factor(condition) + sqft_living15 + floors + view + grade +
as.factor(zipcode)*as.factor(bathrooms), house_pricing_train)
```


```{r}
summary(mod1)
````
````

STRATEGIEN ZUR VERMEIDUNG VON OVERFITTING

Options:

1. Reduce number of features.
   - — Manually select which features to keep.
   - — Model selection algorithm

2. Regularization.
   - — Keep all the features, but reduce magnitude/values of parameters θ_j.
   - — Works well when we have a lot of features, each of which contributes a bit to predicting y.

Computer Science Engineering. (2017). *3.4.1 The Problem of Overfitting by Andrew Ng*. Abgerufen von https://www.youtube.com/watch?v=OSd30QGMl88

REGULARISIERUNG

„Bestrafen“ der Verwendung von Variableninformation im Rahmen der Kostenfunktion

Lineares Modell mit mehreren Variablen x_1, x_2 und vielen möglichen weiteren:

$$h_x(\theta_0, \theta_1, \theta_2, \dots) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots$$

(mit $\theta_0, \theta_1, \theta_2, \dots$ als den zu schätzenden Modellparametern)

Kostenfunktion mit Regularisierung:

$$J_x(\theta_0, \theta_1, \theta_2, \dots) = \frac{1}{m}\left[\sum_m (h_x(\theta_0, \theta_1, \theta_2, \dots) - y)^2 \right]$$

GENERALIZED LINEAR MODEL

```r
library(glmnet)

# Reformat features and label from tibble to matrices as expected by glmnet
train_features_matrix <- as.matrix(train_features)
train_label_matrix <- as.matrix(train_label)

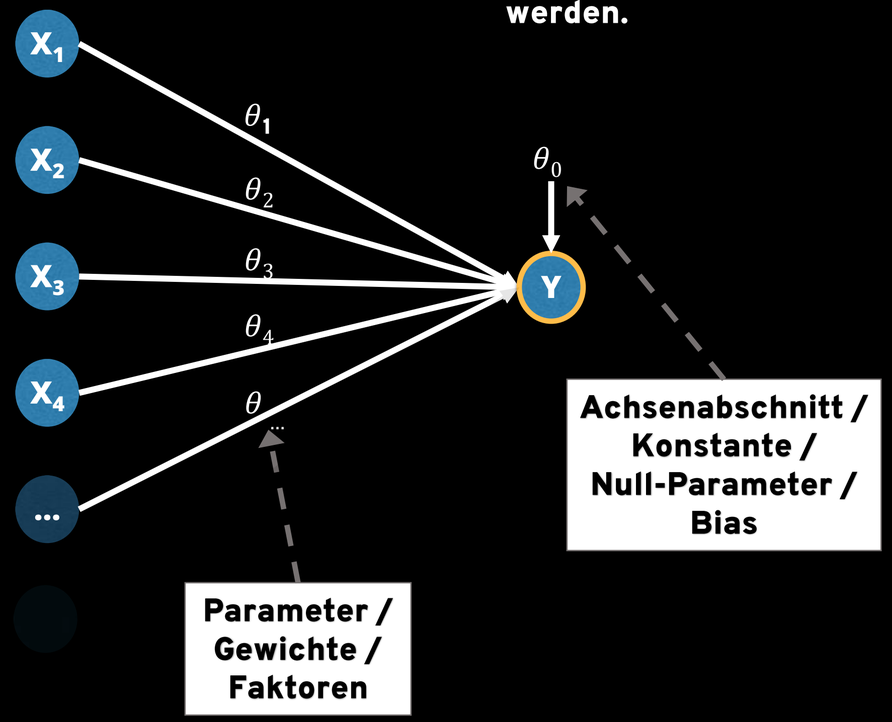
# Calibration of linear regressions with regularisation
mod1 <- glmnet(x=train_features_matrix, y=train_label_matrix, lambda=.001)
mod1

# Calibration of linear regressions with regularisation
mod2 <- glmnet(x=train_features_matrix, y=train_label_matrix, lambda=1000)
mod2
```

ZUSAMMENFASSUNG LINEARE REGRESSION

Input Layer
Elemente sind die Input-Variablen; auch genannt: Input-Features oder Input-Dimensionen.

Output Layer
Nutzt eine *„Aktivierungsfunktion“* (hier lineare Funktion) mit der die Parameter θ der eingehenden Schicht zusammengefasst werden.



- Ziel ist, anhand des Trainingsdatensatzes die Parameter θ der Aktivierungsfunktion für eine bestmögliche Vorhersage des Testdatensatzes zu optimieren.

- Die Optimierung mit Regularisierung erlaubt, viele Variablen in das Modell eingehen zu lassen und über einen Regularisierungs- (oder Shrinkage-) Parameter den Umfang des Einsatzes der Variablen zu kontrollieren, um so Over-/ Underfitting zu kontrollieren.

WICHTIGE HYPERPARAMETER

- Wahl des Modells bzw. der Modellarchitektur
- Wahl der Aktivierungsfunktionen („Vorhersagefunktionen“)
- Wahl der Kostenfunktion
  - mit oder ohne Regularisierung
  - Höhe des Regularisierungsparameters
- Wahl der Optimierungsfunktion (zur Minimierung der Kostenfunktion)
  - Höhe der Lernrate
- Je nach Modellarchitektur zahlreiche weitere…

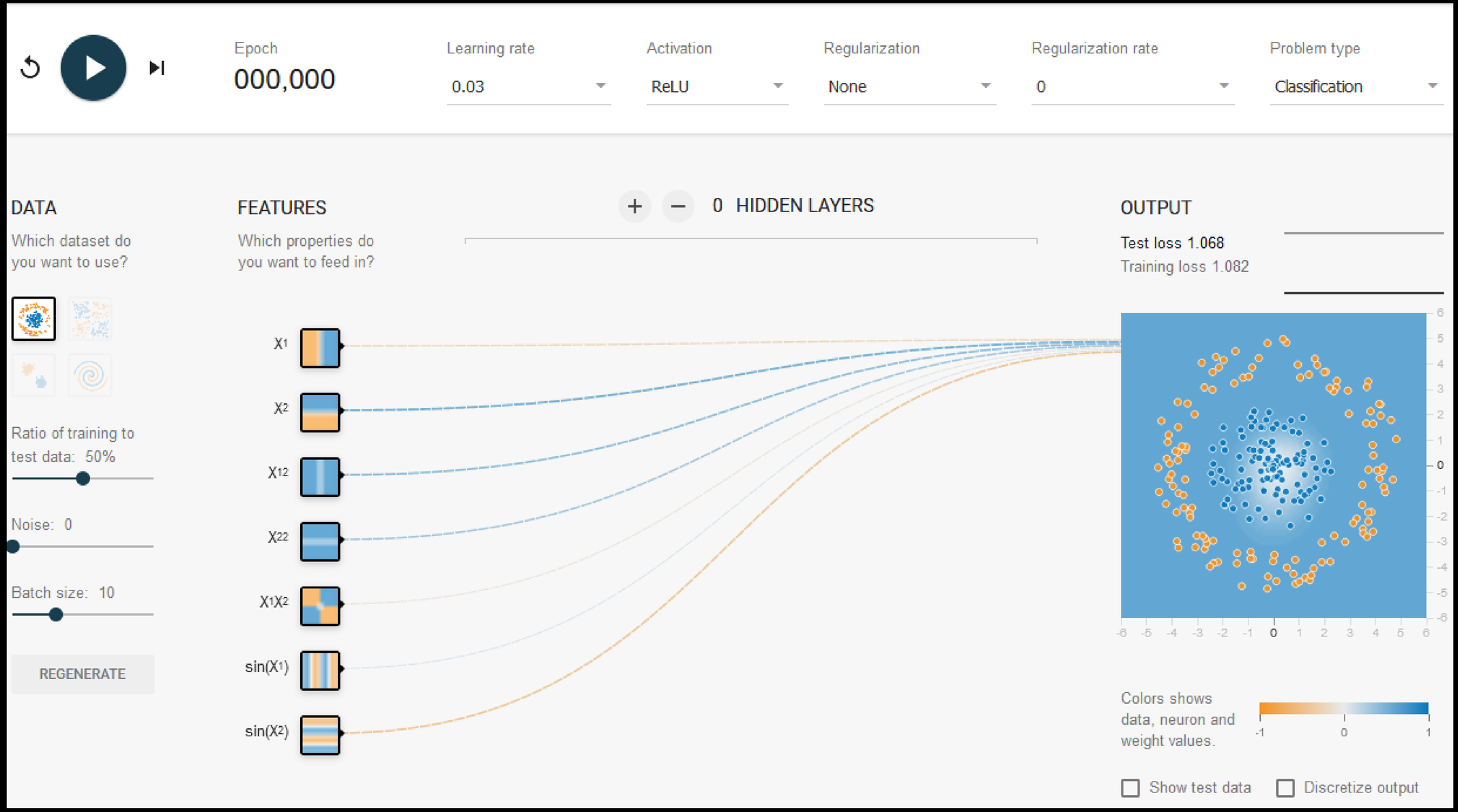
EIGENSCHAFTEN DES LINEAREN MODELLS

- Die Funktion lm() liefert optimierte Parameter für das lineare Modell ohne Regularisierung (Angabe eines Lernparameters ist hier nicht nötig)

  → Für einfache Modelle ist es einfach optimierte Parameter zu erhalten.

- Einfacher zu schätzende Modelle haben stärkere Annahmen über den Zusammenhang der Variablen (hier linearer Zusammenhang)

  → Die optimale Kodierung/Kategorisierung der Variablen entsprechend der Annahmen ist umso wichtiger.


Epoch
000,000
Learning rate
0.03
Activation
ReLU
Regularization
None
Regularization rate
0
Problem type
Classification
DATA
Which dataset do you want to use?
Ratio of training to test data: 50%
Noise: 0
Batch size: 10
REGENERATE
FEATURES
Which properties do you want to feed in?
0 HIDDEN LAYERS
X1
X2
X12
X22
X1X2
sin(X1)
sin(X2)
OUTPUT
Test loss 1.068
Training loss 1.082
Colors shows data, neuron and weight values.
Show test data
Discretize output

BREAKOUT

Ruft folgendes Tool auf: https://playground.tensorflow.org/

1) **Definiert ein lineares Modell (keine Hidden Layer)**
   - **Welche der 4 Datensätze könnt Ihr mit dem linearen Modell erfolgreich klassifizieren?**
   - **Inwieweit könnt Ihr die Ergebnisse hinsichtlich der verwendeten Features (Variablen) interpretieren?**

2) **Definiert zwei Hidden Layer und probiert die Anzahlen der Neuronen so zu ändern, dass Ihr den spiralförmigen Datensatz vorhersagen könnt.**
   - **Welche Verteilungen könnt Ihr erfolgreich vorhersagen?**
   - **Inwieweit könnt Ihr die Ergebnisse hinsichtlich der verwendeten Features interpretieren?**

ABRUFEN DES MAPE FÜR DEN TESTDATENSATZ

```r
library(dplyr)
library(readr)
library(httr)

# Dataframe for request must include columns `Datum`, `Warengruppe` und `Umsatz`
predictions <- read_csv("prediction_template.csv")

# name must not be provided; however, each team must upload at least one not
# anonymous prediction
name <- "Gruppe X"

# Execution of the request
r <- POST("https://bakery-sales-mape-tolicqztoq-ey.a.run.app/",
          body = list(name=name, predictions=predictions),
          encode = "json")
# Output of MAPE in Percent
content(r, "parsed", "application/json")
```

AUFGABEN

- Datensatz weiter um zusätzliche Variablen ergänzen, die für die Schätzung des Umsatzes relevant sein könnten.
- Einmal mit Hilfe dieser Vorlage zum Aufruf der Evaluationsfunktion die Vorhersagegüte Eures linearen Modells für den Zeitraum vom 09.06. bis zum 30.07.2019 überprüfen.
- Dieses Video (12 Minuten) zur Einführung in Neuronale Netze an anschauen.