
# Grundlagen der Künstlichen Intelligenz

Wintersemester 25/26

## Vorlesung 2

Lineare Regression, Lineare Algebra und Wahrscheinlichkeitstheorie

T.T.-Prof. Dr. Peer Nowack
Prof. Dr. Pascal Friederich

# Ziele der heutigen Vorlesung

Künstliche Intelligenz erfordert, dass wir grundlegende mathematische Kenntnisse haben, um die meisten Methoden zu verstehen.

Die wichtigsten sind:

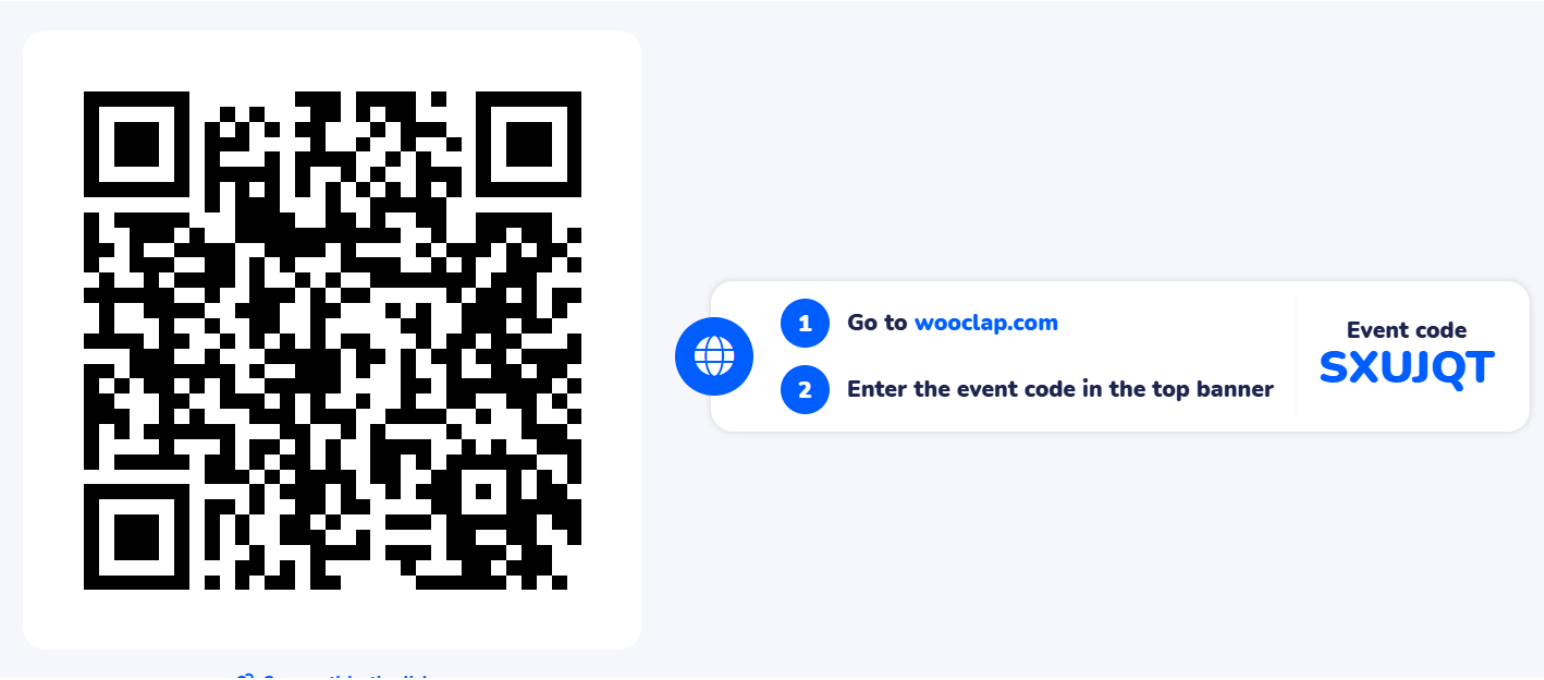
- **Lineare Algebra**
- **Wahrscheinlichkeitstheorie und Statistik**
- Optimierung (nächste Vorlesungen)

Heute werden wir (sehr kurz) die wichtigsten Grundlagen zusammenfassen, um unseren ersten einfachen Lernalgorithmus zu diskutieren: **lineare Regression**

Dazu werden wir die **Optimierungssichtweise** (optimization view) der **Wahrscheinlichkeitssichtweise** (probabilistic view) auf Lineare Regression gegenüberstellen, mit allg. Implikationen für überwachtes Lernen (supervised learning).

# Wir werden immer mal wieder Umfragen machen...

Sie können sich bereits einloggen. Nutzen Sie das KIT Wi-Fi bei schlechtem Empfang.



You cannot vote anymore

Ist lineare Regression eine Methode des maschinellen Lernens und damit auch der künstlichen Intelligenz?

Click on the projected screen to start the question

1 Ja 66% 114

2 Nein 19% 33

3 You tell me! 15% 25

wooclap

100%

172 / 280

# Überblick

## Lineare Regression

- Einführung in die Regression
- Differentialrechnung mit Matrizen und Vektoren (Matrix Calculus)
- Lösung nach der Methode der Kleinsten Quadrate (Least Squares)
- Verallgemeinerte Lineare Regression (Generalized Linear Regression)
- Ridge Regression

## Wahrscheinlichkeitstheorie

- Probabilistische Modelle
- Satz von Bayes (Bayes‘ Rule/Bayes‘ Theorem)
- Erwartungswerte und Monte-Carlo-Schätzung
- Maximum-Likelihood-Schätzung (Maximum Likelihood Estimation)

# Mathematische Notation

- Vektoren werden als fettgedruckte Symbole dargestellt

$x = 1$; ein skalarer Wert $\qquad \boldsymbol{x} = \begin{bmatrix} 1 \\ 2 \\ 4 \end{bmatrix}$; ein Vektor

- Ein Vektor $\boldsymbol{x}$ ist immer ein **Spaltenvektor** $\qquad \boldsymbol{x} = \begin{bmatrix} 1 \\ 2 \\ 4 \end{bmatrix}$

- Ein transponierter Vektor $\boldsymbol{x}^T$ ist immer ein **Zeilenvektor** $\boldsymbol{x}^T = \begin{bmatrix} 1 & 2 & 4 \end{bmatrix}$

# Matrizen im maschinellen Lernen

- In vielen Fällen kann unser Datensatz als Matrix dargestellt werden, wobei einzelne Stichproben/Samples Vektoren sind

$$\text{Joe: } \boldsymbol{x}_1 = \begin{bmatrix} 37 \\ 72 \\ 175 \end{bmatrix} \quad \text{Mary: } \boldsymbol{x}_2 = \begin{bmatrix} 10 \\ 30 \\ 61 \end{bmatrix} \quad \text{Carol: } \boldsymbol{x}_3 = \begin{bmatrix} 25 \\ 65 \\ 121 \end{bmatrix} \quad \text{Brad: } \boldsymbol{x}_4 = \begin{bmatrix} 66 \\ 67 \\ 175 \end{bmatrix}$$

- **Häufigste Darstellung:**
  - Jede Zeile repräsentiert ein Sample (z. B. Joe).
  - Jede Spalte repräsentiert einen Dateneintrag (z. B. Alter)

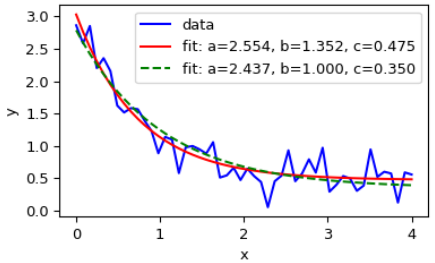
$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1^T \\ \boldsymbol{x}_2^T \\ \boldsymbol{x}_3^T \\ \boldsymbol{x}_4^T \end{bmatrix} = \begin{bmatrix} 37 & 72 & 175 \\ 10 & 30 & 61 \\ 25 & 65 & 121 \\ 66 & 67 & 175 \end{bmatrix}$$

$\boldsymbol{X}$ ist eine Matrix mit
*# Stichproben x # Einträgen*

# Regression (eine Form des überwachten Lernens)

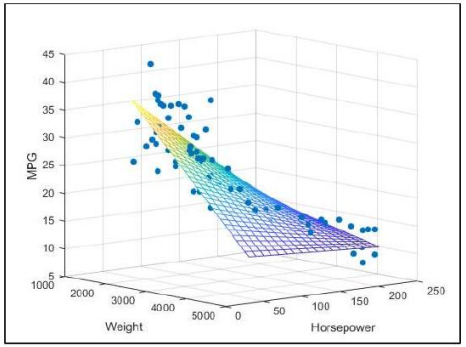
**Ziel:** Vorhersage des Wertes einer kontinuierlichen Variablen auf der Grundlage der Werte anderer (erklärender/unabhängiger) Variablen unter der Annahme eines linearen oder nichtlinearen Abhängigkeitsmodells

$$y = f(x) + \epsilon$$




Wir gehen davon aus, dass die Ergebnisse (typischerweise) durch Gaußsches Rauschen beeinflusst werden:

$$\epsilon \sim \mathcal{N}(0, 1)$$

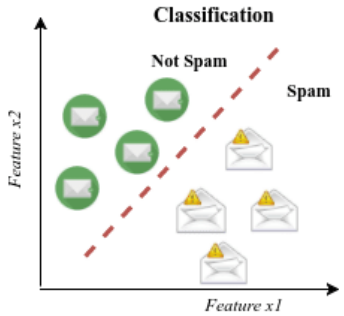


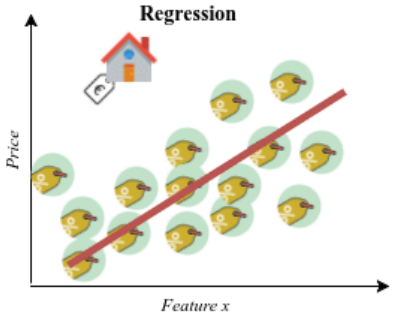

**Beispiele:** Wettervorhersage, Immobilienmarkt, Aktienmarkt

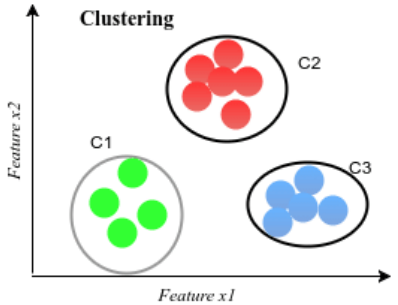
# Überwachtes und unüberwachtes Lernen (put simple)

Beispiele überwacht (mit Labels)

Beispiel unüberwacht (ohne Labels)

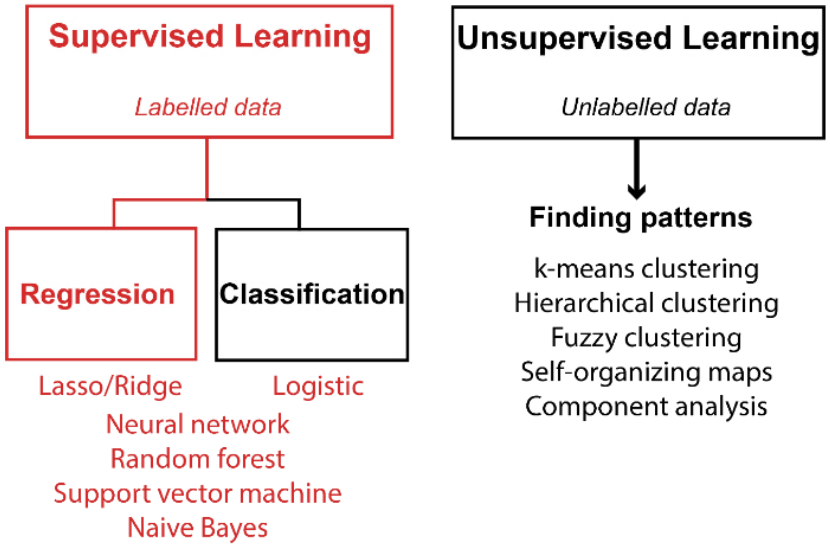


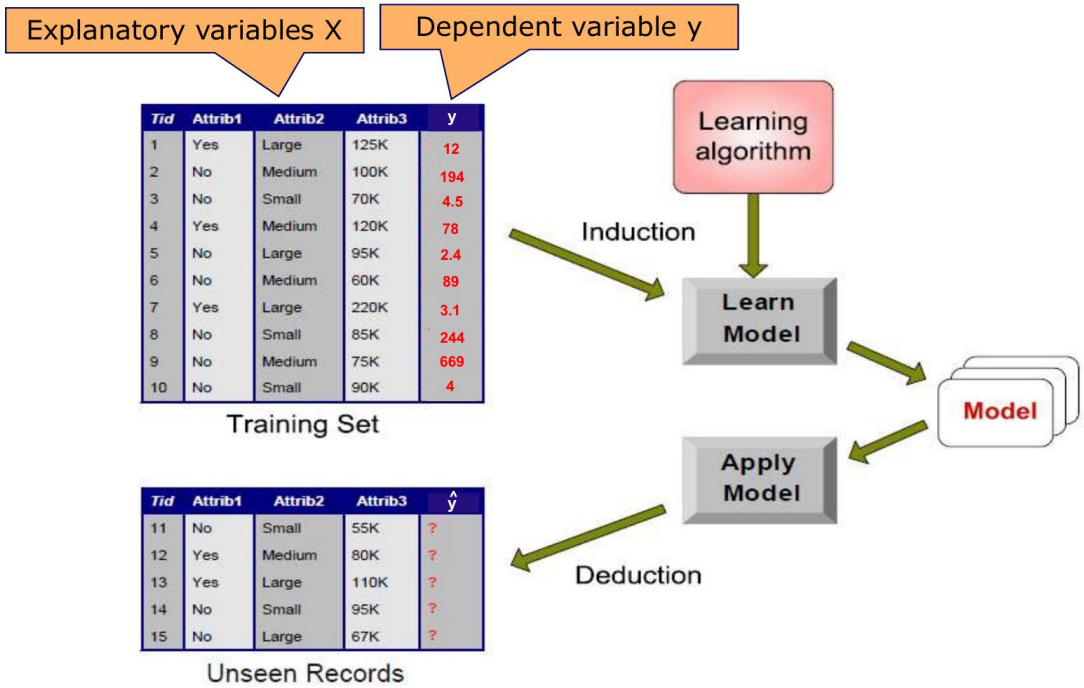




S. Kaplan

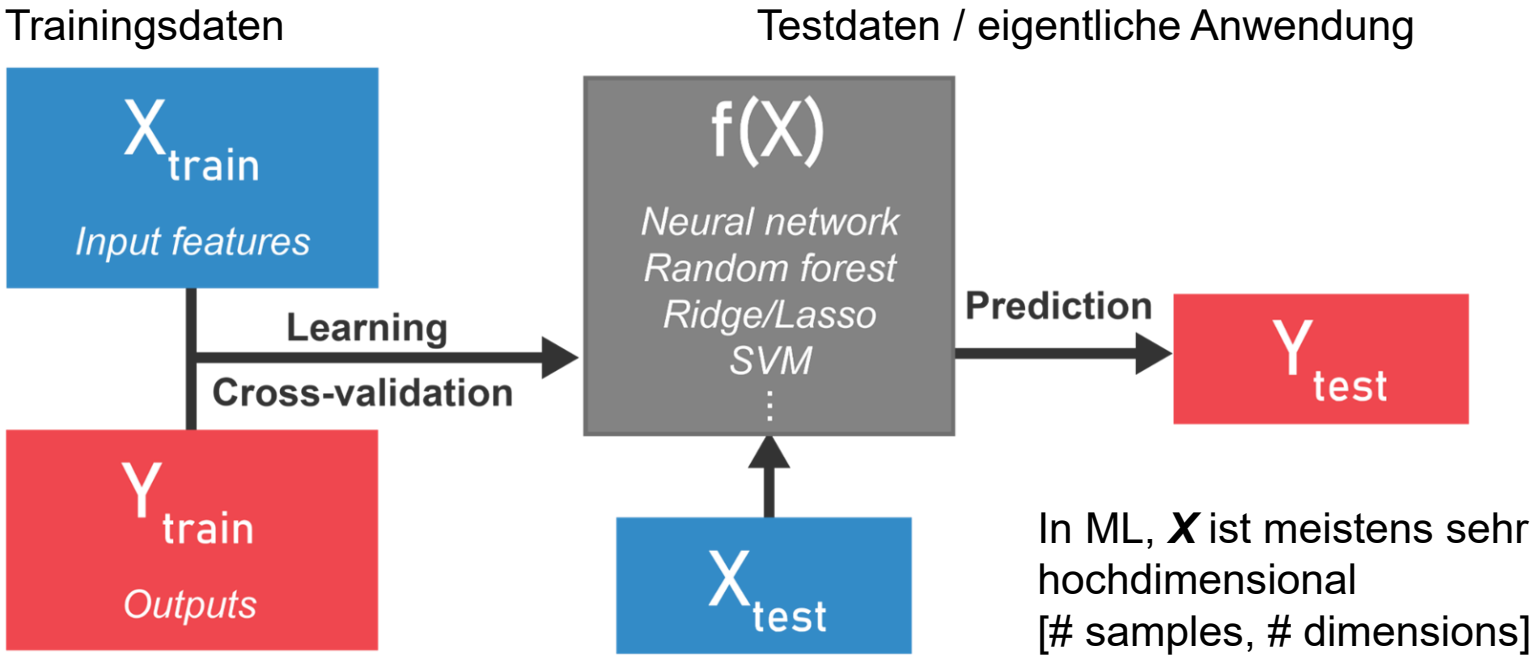
# Überwachtes und unüberwachtes Lernen



# Der „Lernprozess“ in Regressionsmodellen



# Der „Lernprozess“ in Regressionsmodellen



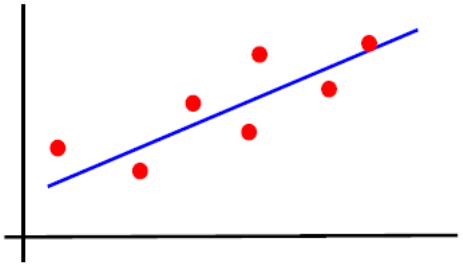
# Einige Regressionsalgorithmen

- Lineare Regression, Polynomiale Regression, Ridge Regression
- K-Nearest-Neighbours Regression
- Kernel-Regression und Gauß-Prozesse (Gaussian Processes)
- (Tiefe) neuronale Netze (Deep neural networks)

## Lineare Regression:

- Wir fitten „einfach” eine gerade Linie

$$y = f(x) + \epsilon = w_0 + w_1 x + \epsilon$$

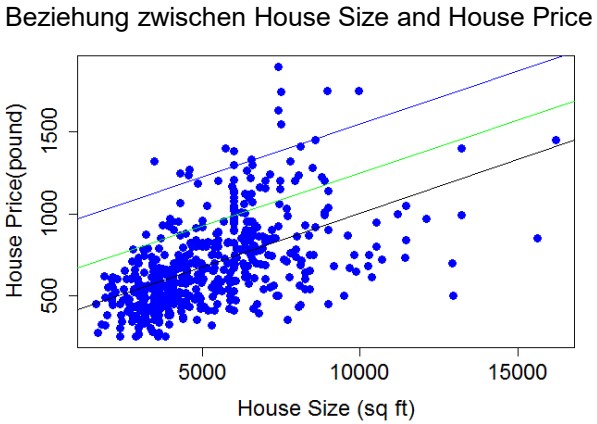


# Lineare Regressionsmodelle

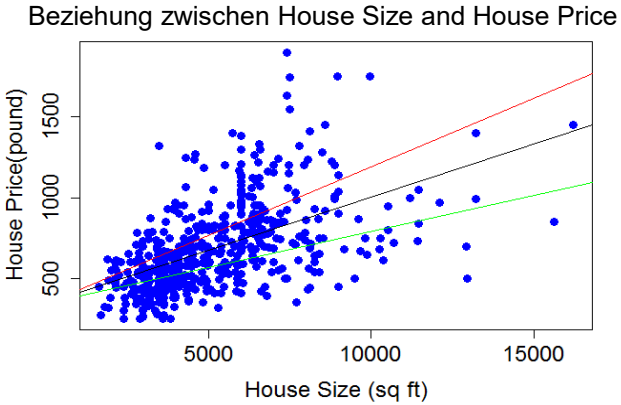
Es wird die Ausgabe $y$ als lineare Funktion der Eingabe $\boldsymbol{x}_i$ modelliert

$$y = f(x) + \epsilon = w_0 + w_1 x + \epsilon$$

## Effect of $w_0$



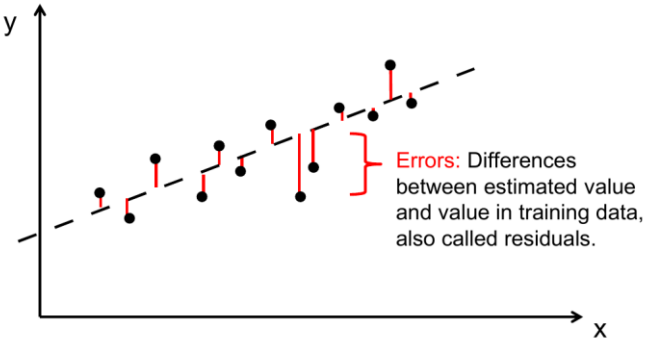

## Effect of $w_1$



# Ziel der Regression (wie die Gewichte schätzen?)

Minimiere die Summe (oder mittleren Wert) der quadratischen Fehler (Summed Squared Error)

$$\mathrm{SSE} = \sum_{i=1}^{N} \left(y_i - f(\boldsymbol{x}_i)\right)^2$$

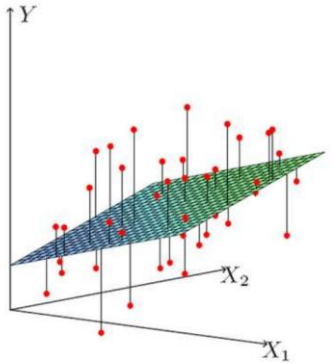



wobei die Eingabe $\boldsymbol{x}_i$ ein d-dimensionaler Vektor ist

**Warum verwenden wir den quadratischen Fehler?**

- Er ist vollständig differenzierbar und leicht zu optimieren
- Außerdem lässt sich leicht zeigen lässt, dass:

$$f^*(\boldsymbol{x}) = \mathrm{argmin}_{f(\boldsymbol{x})} \mathrm{SSE} \Rightarrow f^*(\boldsymbol{x}) = \mathbb{E}[y|\boldsymbol{x}]$$




Daher schätzen wir immer den Mittelwert der Zielfunktion!

# Das Ziel der (multiplen) linearen Regression

Wir betrachten hier eine lineare Funktion mit mehreren Eingaben

$$f(\boldsymbol{x}_i) = w_0 + \sum_j w_j x_{i,j}$$

Das “Lernziel” wird dann

$$\mathrm{SSE} = \sum_{i=1}^{N} \left( y_i - \left( w_0 + \sum_j w_j x_{i,j} \right) \right)^2$$

Können wir diesen Ausdruck mit Hilfe von Matrizen vereinfachen?

# Lineare Regressionsmodelle in Matrixform

Erweitere $\boldsymbol{x}$ um einen Bias-Eintrag um $w_0$ in das Skalarprodukt zu absorbieren

**Gleichung für die i-te Stichprobe:**

$$\hat{y}_i = w_0 + \sum_{j=1}^{D} w_j x_{i,j} = \tilde{\boldsymbol{x}}_i^T \boldsymbol{w}, \quad \text{with } \tilde{\boldsymbol{x}}_i = \begin{bmatrix} 1 \\ \boldsymbol{x}_i \end{bmatrix} \text{ and } \boldsymbol{w} = \begin{bmatrix} w_0 \\ \vdots \\ w_D \end{bmatrix}$$

**Gleichung für den vollständigen Datensatz:**

$$\hat{\boldsymbol{y}} = \begin{bmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_n \end{bmatrix} = \begin{bmatrix} \tilde{\boldsymbol{x}}_1^T \boldsymbol{w} \\ \vdots \\ \tilde{\boldsymbol{x}}_n^T \boldsymbol{w} \end{bmatrix} = \boldsymbol{X}\boldsymbol{w}$$

$\hat{\boldsymbol{y}}$ ist ein Vektor und enthält die Ausgabe für jede Stichprobe

mit Datenmatrix $\boldsymbol{X}$.
Die erste Spalte mit Einsen gibt den bias/intercept.

$$\boldsymbol{X} = \begin{bmatrix} \tilde{\boldsymbol{x}}_1^T \\ \vdots \\ \tilde{\boldsymbol{x}}_n^T \end{bmatrix} = \begin{bmatrix} 1 & \boldsymbol{x}_1^T \\ \vdots & \vdots \\ 1 & \boldsymbol{x}_n^T \end{bmatrix}$$

# Lineare Regressionsmodelle in Matrixform

- Fehler(error)vektor: $\boldsymbol{e} = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} - \begin{bmatrix} \hat{y}_1 \\ \vdots \\ \hat{y}_n \end{bmatrix} = \boldsymbol{y} - \hat{\boldsymbol{y}} = \boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}$

- Summe der quadratischen Fehler / Sum of squared errors (SSE)

$$\text{SSE} = \sum_i (y_i - \hat{y}_i)^2 = \sum_i e_i^2 = \boldsymbol{e}^T \boldsymbol{e} = (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})^T (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})$$

- Wir haben nun den **SSE vollständig in Matrixform** geschrieben!
  Bringt verschiedene Vorteile, insbesondere für Ableitungen des Fehlers in der Optimierung von Modellen im maschinellen Lernen.

# Herleitung der linearen Regression

Wie erhalten wir die optimalen Parameter/Gewichte $\boldsymbol{w}$? (diejenigen die das SSE minimiert)

$$\boldsymbol{w}^* = \text{argmin}_{\boldsymbol{w}} \, \text{SSE} = \text{argmin}_{\boldsymbol{w}} \, (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})^T (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w})$$

Bei einem Minimalwert einer Funktion ist ihre Ableitung gleich Null

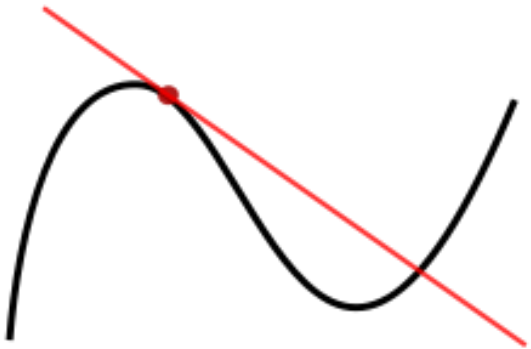
D. h., finde ein $\boldsymbol{w}$ bei dem $\dfrac{\partial \text{SSE}}{\partial \boldsymbol{w}} = \boldsymbol{0}^T$

# Matrix Calculus

(insbesondere Ableitungen bezüglich Vektoren und Matrizen)

# Differentialrechnung - Recap

„Die Ableitung einer Funktion einer reellen Variablen misst **die Empfindlichkeit einer Größe** (einem Funktionswert oder einer abhängigen Variablen) **gegenüber einer Änderung,** die durch eine andere Größe (die unabhängige Variable) bestimmt wird“ (Wikipedia)



Funktion: $f(x)$

Ableitung: $\frac{\partial f(x)}{\partial x}$

Minimum/Maximum: $\frac{\partial f(x)}{\partial x} = 0$

# Ableitungen bezüglich Vektoren

## Ableitungen einer skalaren Funktion in Bezug auf einen Vektor

- Ergibt den Gradientenvektor: $$\nabla_{\boldsymbol{x}} f = \frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial f(\boldsymbol{x})}{\partial x_1} \\ \vdots \\ \frac{\partial f(\boldsymbol{x})}{\partial x_d} \end{bmatrix}$$

Notation mit Spaltenvektor als Resultat ist typischer im maschinellen Lernen, aber es ginge auch eine Definition mit einem Zeilenvektor.

- Beispiel: Quadratische Form $\nabla_{\boldsymbol{x}} \boldsymbol{x}^T \boldsymbol{x} = 2\boldsymbol{x}$ $\nabla_{\boldsymbol{x}} \boldsymbol{x}^T \boldsymbol{A}\boldsymbol{x} = 2\boldsymbol{A}\boldsymbol{x}$ $(\text{ if } \boldsymbol{A} \text{ is symmetric})$

## Ableitungen einer vektorwertigen Funktion in Bezug auf einen Vektor

- Ergibt eine Matrix (die Jacobi-Matrix/Jacobian)
  - dies ist eigentlich die Transponierte der Jacobi-Matrix
  - da die Jacobi-Matrix typischerweise im Zählerlayout verwendet wird

$$\nabla_{\boldsymbol{x}} \boldsymbol{f} = \frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial f_1(\boldsymbol{x})}{\partial x_1} & \cdots & \frac{\partial f_k(\boldsymbol{x})}{\partial x_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_1(\boldsymbol{x})}{\partial x_d} & \cdots & \frac{\partial f_k(\boldsymbol{x})}{\partial x_d} \end{bmatrix}$$

- Beispiel: Lineare Form $\nabla_{\boldsymbol{x}} \boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^T$

# Ableitungen bezüglich Matrizen

## Ableitungen einer skalaren Funktion in Bezug auf eine Matrix

- … sind wiederum eine Matrix

$$\nabla_{\boldsymbol{W}} f = \frac{\partial f(\boldsymbol{W})}{\partial \boldsymbol{W}} = \begin{bmatrix} \frac{\partial f(\boldsymbol{W})}{\partial W_{11}} & \cdots & \frac{\partial f(\boldsymbol{W})}{\partial W_{1d}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f(\boldsymbol{W})}{\partial W_{k1}} & \cdots & \frac{\partial f(\boldsymbol{W})}{\partial W_{kd}} \end{bmatrix}$$

## Ableitungen einer vektorwertigen Funktion in Bezug auf eine Matrix

- … sind ein 3D-Tensor!
- Das wird etwas knifflig… zum Glück brauchen wir das (fast) nie

# Matrix Calculus - Kurzüberblick

Es lohnt sich einige Regeln gut zu kennen (siehe Wikipedia)

| | Skalar | Vektor | Matrix |
|---|---|---|---|
| • **Linear:** | $\frac{\partial ax}{\partial x} = a$ | $\nabla_{\boldsymbol{x}} \boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^T$<br>$\nabla_{\boldsymbol{x}} \boldsymbol{a}^T\boldsymbol{x} = \boldsymbol{a}$ | $\nabla_{\boldsymbol{X}}\ \boldsymbol{a}^T\boldsymbol{X}\boldsymbol{b} = \boldsymbol{a}\boldsymbol{b}^T$<br>$\nabla_{\boldsymbol{X}}\ \mathrm{tr}(\boldsymbol{A}\boldsymbol{X}\boldsymbol{B}) = \boldsymbol{A}^T\boldsymbol{B}^T$ |
| • **Quadratisch:** | $\frac{\partial x^2}{\partial x} = 2x$ | $\nabla_{\boldsymbol{x}} \boldsymbol{x}^T\boldsymbol{x} = 2\boldsymbol{x}$<br>$\nabla_{\boldsymbol{x}} \boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} = (\boldsymbol{A}^T + \boldsymbol{A})\boldsymbol{x}$<br>$= 2\boldsymbol{A}\boldsymbol{x}$<br>(if $\boldsymbol{A}$ is symmetric) | |

# Mit diesem Wissen können wir nun $w$ berechnen

$$\begin{aligned}\mathrm{SSE}(\boldsymbol{w}) &= (\boldsymbol{y}-\boldsymbol{X}\boldsymbol{w})^T(\boldsymbol{y}-\boldsymbol{X}\boldsymbol{w}) \\ &= \boldsymbol{w}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}^T\boldsymbol{X}\boldsymbol{w} - \boldsymbol{w}^T\boldsymbol{X}^T\boldsymbol{y} + \boldsymbol{y}^T\boldsymbol{y} \\ &= \boldsymbol{w}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} - 2\boldsymbol{y}^T\boldsymbol{X}\boldsymbol{w} + \boldsymbol{y}^T\boldsymbol{y}\end{aligned}$$

Wollen Minimum finden, daher Ableitung nach $\boldsymbol{w}$

$$\begin{aligned}\nabla_{\boldsymbol{w}}\mathrm{SSE}(\boldsymbol{w}) &= \frac{\partial}{\partial \boldsymbol{w}}\left\{\boldsymbol{w}^T\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} - 2\boldsymbol{y}^T\boldsymbol{X}\boldsymbol{w} + \boldsymbol{y}^T\boldsymbol{y}\right\} \\ &=\end{aligned}$$

Setzt man diesen Gradienten auf 0 so erhält man

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

Dies ist die kleinste Quadrate / Least Squares Lösung

Hier hilfreich:

$$\nabla_{\boldsymbol{x}}\boldsymbol{A}\boldsymbol{x} = \boldsymbol{A}^T$$

$$\begin{aligned}\nabla_{\boldsymbol{x}}\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} &= (\boldsymbol{A}^T + \boldsymbol{A})\boldsymbol{x} \\ &= 2\boldsymbol{A}\boldsymbol{x} \quad \text{(if } \boldsymbol{A} \text{ is symmetric)}\end{aligned}$$

Existiert die Inverse?
Siehe auch Ridge Regression

# Bewertung von (linearen) Regressionsmodellen

**Wie können wir die Qualität eines Modells einschätzen?**

- Der SSE kann problemspezifisch je nach Ausgabebereich beliebige Werte annehmen.
- Die Bewertung sollte unabhängig von der Varianz von $y$ sein

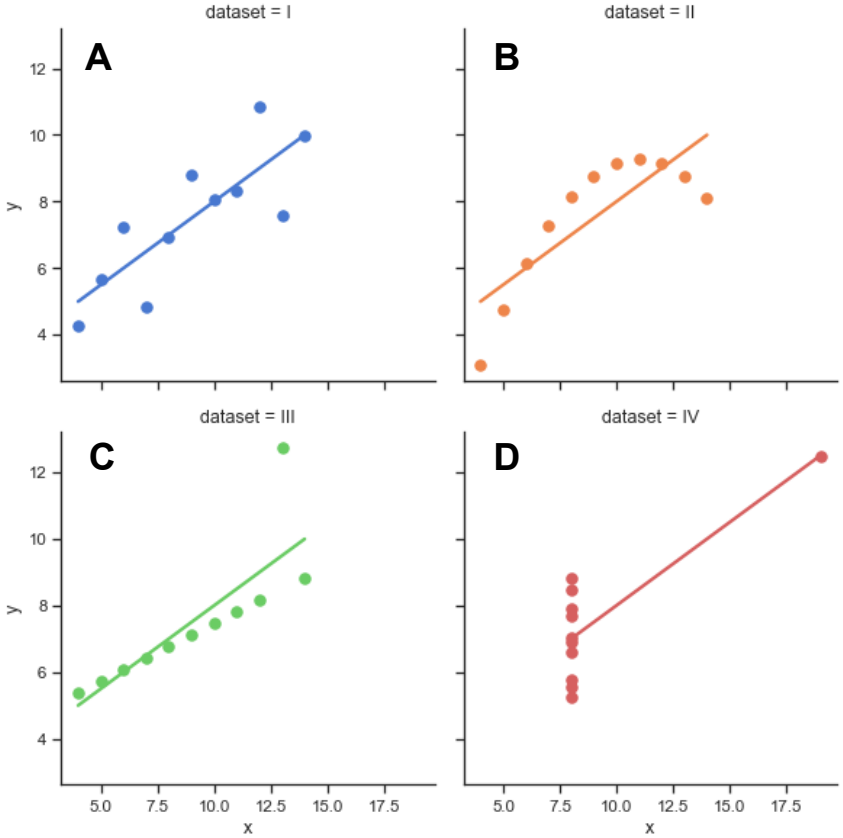
**R-Quadrat** (oder $R^2$) bestimmt, wie viel der Gesamtvariation in $y$ durch die Variation in $\boldsymbol{x}$ erklärt wird. Mathematisch lässt sich dies wie folgt ausdrücken

$$R^2 = 1 - \frac{\text{Regression sum of squares}}{\text{Total sum of squares}} = 1 - \frac{\sum_{n=1}^{N}(\hat{y}_n - y_n)^2}{\sum_{n=1}^{N}(\bar{y} - y_n)^2}$$

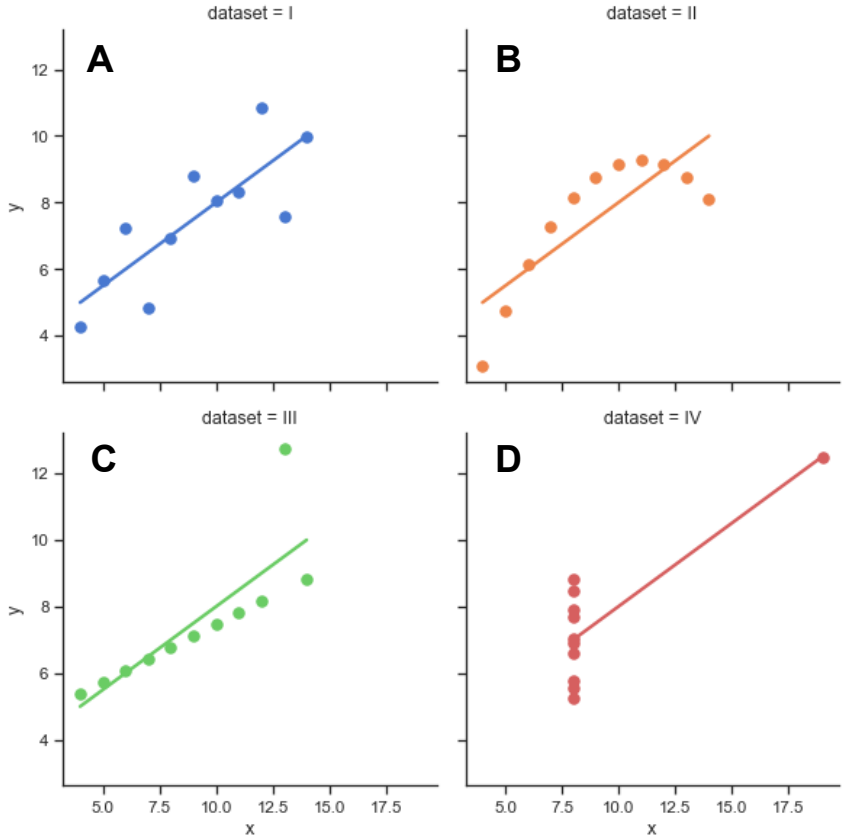
wobei $\bar{y}$ der Mittelwert der Ausgabewerte ist.

$R^2$ gibt an, wie gut die Regressionsgerade die tatsächlichen Datenpunkte approximiert. Ein $R^2$ von 1 bedeutet, dass die Regressionsgerade perfekt zu den Daten passt.

# Einige Beispiele für lineare Regression (gone wrong)



# Auch bekannt als Anscombe-Quartett

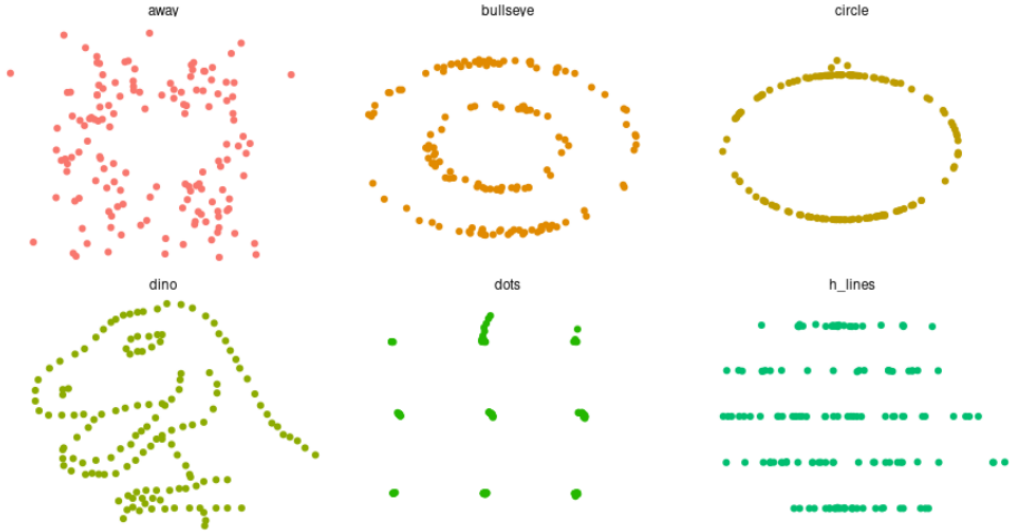



Fast identische Mittelwerte und Standardabweichungen in $x$ und $y$, als auch Korrelationen zwischen $x$ and $y$, bis auf mehrere Nachkommastellen.

**Take-home message:**

Falls möglich, Resultate immer visualisieren!

# Noch mehr Anscombe-artige Daten...




```
## # A tibble: 13 × 6
##    dataset    mean_x mean_y std_dev_x std_dev_y corr_x_y
##    <chr>       <dbl>  <dbl>     <dbl>     <dbl>    <dbl>
##  1 away         54.3   47.8      16.8      26.9  -0.0641
##  2 bullseye     54.3   47.8      16.8      26.9  -0.0686
##  3 circle       54.3   47.8      16.8      26.9  -0.0683
##  4 dino         54.3   47.8      16.8      26.9  -0.0645
##  5 dots         54.3   47.8      16.8      26.9  -0.0603
##  6 h_lines      54.3   47.8      16.8      26.9  -0.0617
```

https://cran.r-project.org/web/packages/datasauRus/vignettes/Datasaurus.html

# Zusammenfassung lineare Regression

Wir haben nun unseren ersten (ML-)Algorithmus hergeleitet: **die lineare Regression!**

- Die Lösung wird als Lösung der kleinsten Quadrate bezeichnet.
- Dies ist einer der seltenen Fälle, in denen wir eine geschlossene Lösung erhalten können.

**Dies war nur möglich, weil:**

- Die Kostenfunktion (SSE) für lineares $f(x)$ konvex ist
  - Es gibt nur ein Minimum (!)
- Die Kostenfunktion ist quadratisch in $\boldsymbol{w}$
  - Das Minimum ist leicht zu erhalten

# Fragen?

# Verallgemeinerte Lineare Regression

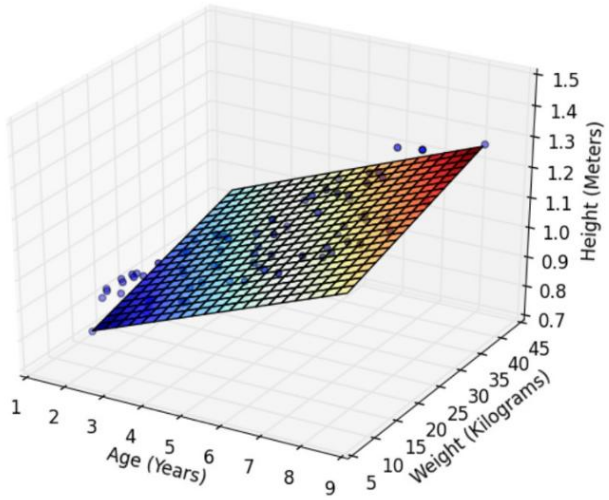
## Generalized Linear Regression

# Lineare Funktionen

Bisher haben wir unsere Funktion $f$ als linear in $\boldsymbol{x}$ und $\boldsymbol{w}$ modelliert

$$f(\boldsymbol{x}) = \tilde{\boldsymbol{x}}^T \boldsymbol{w}$$

Diese Gleichung kann jedoch nur Hyperflächen im d-dimensionalen Eingaberaum darstellen



# Verallgemeinerte Form

Mit einer kleinen Umformulierung, können wir dieses Modell flexibler machen

$$f(\boldsymbol{x}) = \boldsymbol{\phi}(\boldsymbol{x})^T \boldsymbol{w}$$

wobei $\boldsymbol{\phi}(\boldsymbol{x})$ eine vektorwertige Funktion des Eingabevektors $\boldsymbol{x}$ ist.

Dies wird auch als lineares Basisfunktionsmodell bezeichnet, wobei $\phi_i$ als Basisfunktionen bekannt sind.

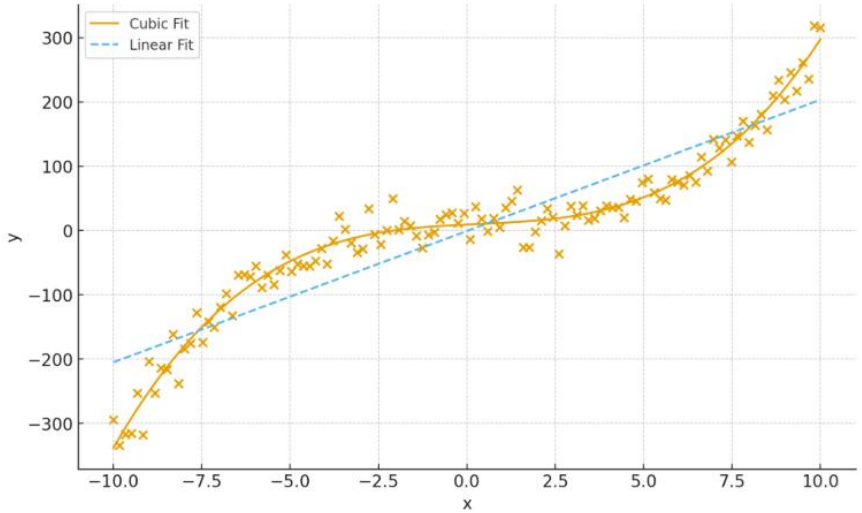
Das Modell ist linear in Bezug auf die Parameter $\boldsymbol{w}$, aber nicht unbedingt in Bezug auf $\boldsymbol{x}$.

# Beispiel: polynomische Kurvenanpassung

$$f(\boldsymbol{x}) = \boldsymbol{\phi}(\boldsymbol{x})^T \boldsymbol{w}$$

wobei

$$\boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{bmatrix}, \qquad \boldsymbol{\phi}(\boldsymbol{x}) = \begin{bmatrix} 1 \\ x \\ x^2 \\ x^3 \end{bmatrix}$$

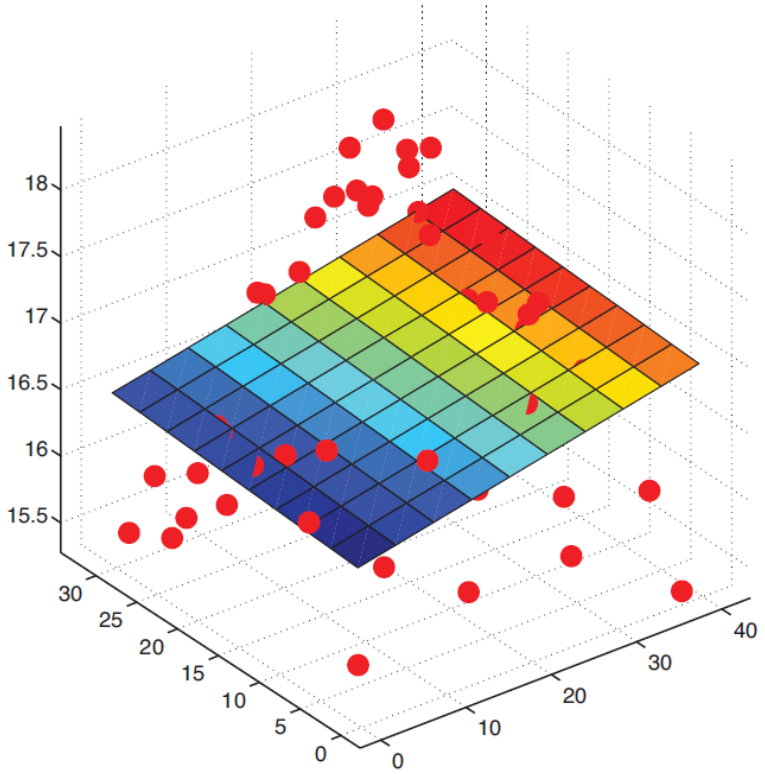


# Multiple lineare Regression in diesem Format

$$f(\boldsymbol{x}) = \boldsymbol{\phi}(\boldsymbol{x})^T \boldsymbol{w}$$

wobei

$$\boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix}, \qquad \boldsymbol{\phi}(\boldsymbol{x}) = \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix}$$

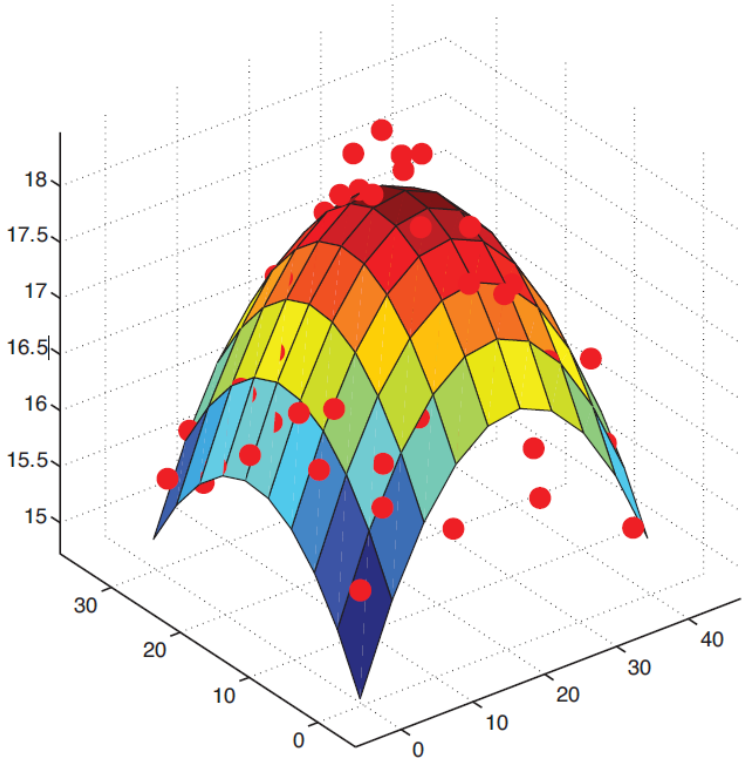


# Beispiel: quadratisches Polynom, zwei unabhängige Variablen

$$f(\boldsymbol{x}) = \boldsymbol{\phi}(\boldsymbol{x})^T \boldsymbol{w}$$

wobei

$$\boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \\ w_4 \end{bmatrix}, \qquad \boldsymbol{\phi}(\boldsymbol{x}) = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_1^2 \\ x_2^2 \end{bmatrix}$$



# Lösung für die verallgemeinerte lineare Regression

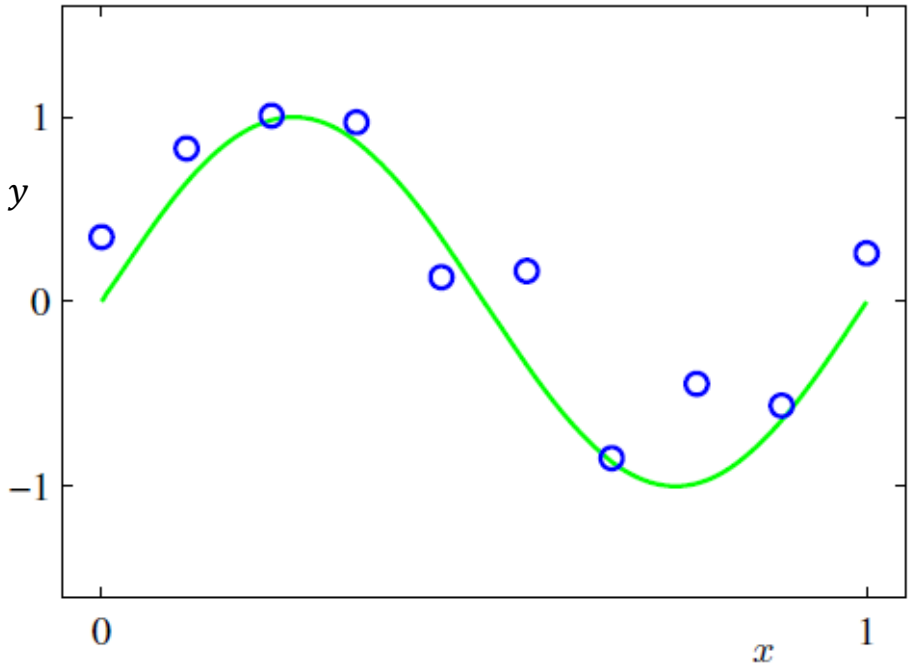
Die Herleitungen bleiben genau die gleichen, nur die Datenmatrix wird nun durch die Basisfunktionsmatrix ersetzt, d. h.:

$$\boldsymbol{w}^* = (\boldsymbol{\Phi}^T \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^T \boldsymbol{y},$$

mit $\boldsymbol{\Phi} = \begin{bmatrix} \boldsymbol{\phi}_1^T \\ \vdots \\ \boldsymbol{\phi}_n^T \end{bmatrix}$

Im Prinzip, können wir damit **jede nichtlineare Funktion lernen**, wenn wir geeignete Basisfunktionen kennen (was in der Regel nicht der Fall ist).

# Beispiel: Fitten einer Sinus-Kurve mit einem Polynom



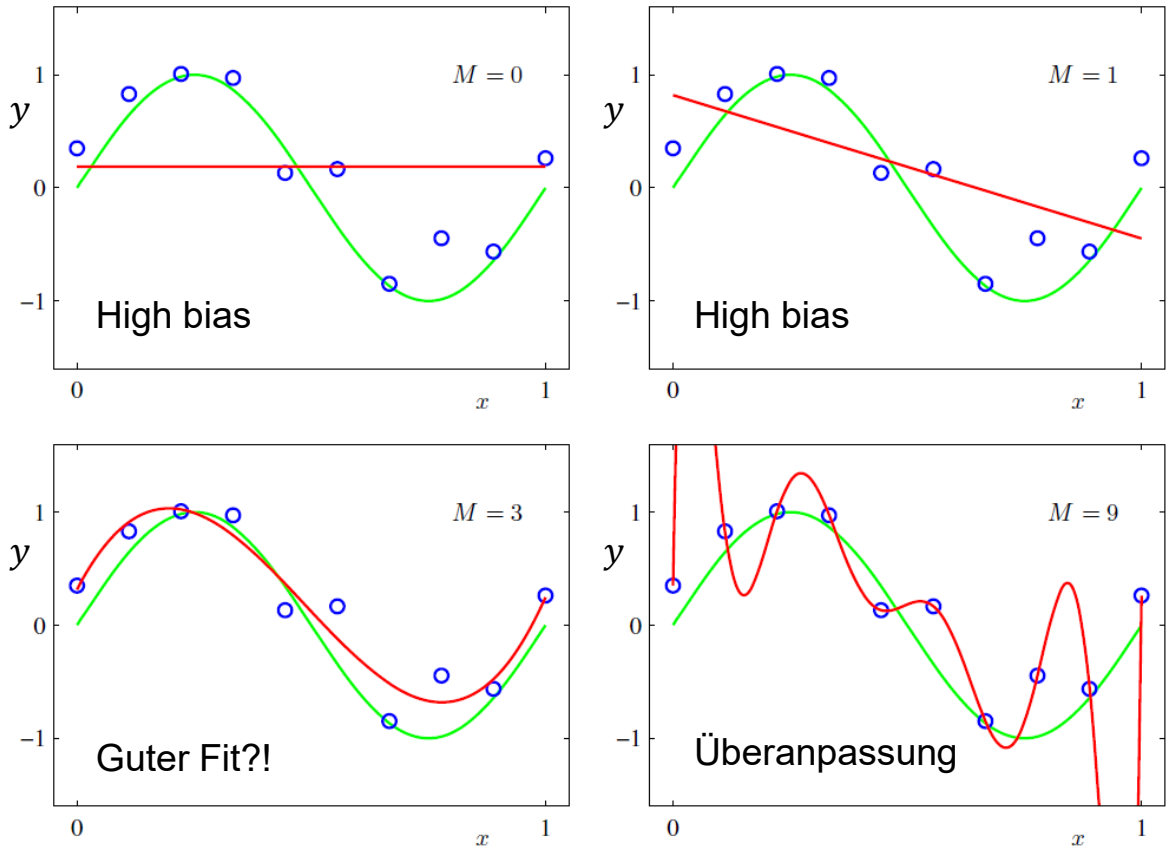

Wahre Sinus-Kurve plus Rauschen liefert 10 Trainingsdatenpunkte.

Wir wollen ein Polynom *M*-ter Ordnung als verallgemeinertes lineares Modell “lernen” um die Kurve möglichst genau nachzubilden, ohne die eigentliche Form der zugrunde liegenden Funktion zu kennen.

$$y(x, w) = \sum_{j=0}^{M} w_j x^j$$

Bishop, Pattern Recognition and Machine Learning (2006)

# Beispiel: Fitten einer Sinus-Kurve mit einem Polynom




In ML ist die Wahl der richtigen Modellkomplexität eine zentrale Herausforderung
→ **mehr z.B. in Vorlesung 6!**

# Überanpassung bei der polynomischen Regression

Der Fehler auf dem Trainingssatz **ist kein Hinweis auf eine gute Anpassung**!

Wir benötigen immer einen **unabhängigen Testsatz!**

**Überanpassung** (Overfitting/high variance):

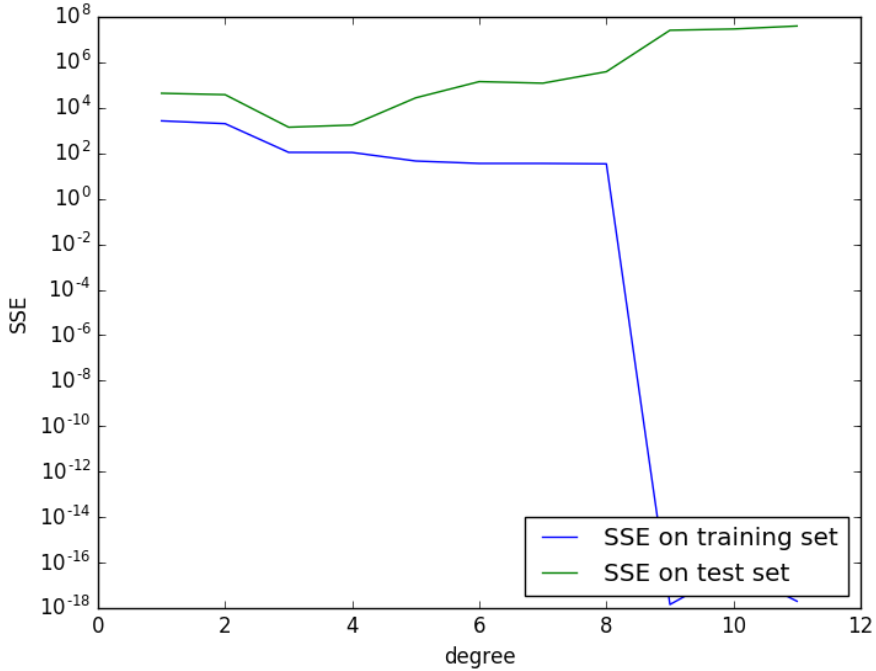
- Der Trainingsfehler sinkt
- Der Testfehler steigt

Das Modell ist zu komplex. Es passt sich dem Rauschen an und weist zwischen den Trainingspunkten ein nicht spezifiziertes Verhalten auf.

**Unteranpassung** (Underfitting/high bias):

- Trainings- und Testfehler sind hoch

Das Modell ist zu einfach um die Daten zu fitten.

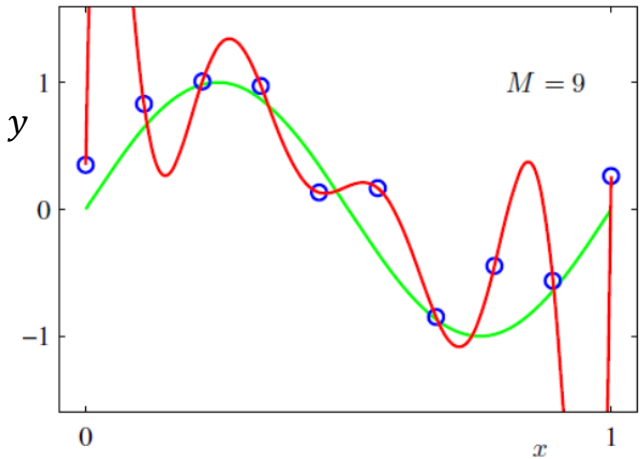


# Fragen?

# Ridge Regression

# Was passiert im Fall der Überanpassung?

Empirisch und theoretisch nachweisbar, die Gewichte $\boldsymbol{w}$ werden sehr groß um sich den Daten (inklusive dem Rauschen) perfekt anpassen zu können.




| | $M=0$ | $M=1$ | $M=6$ | $M=9$ |
|---|---|---|---|---|
| $w_0^\star$ | 0.19 | 0.82 | 0.31 | 0.35 |
| $w_1^\star$ | | -1.27 | 7.99 | 232.37 |
| $w_2^\star$ | | | -25.43 | -5321.83 |
| $w_3^\star$ | | | 17.37 | 48568.31 |
| $w_4^\star$ | | | | -231639.30 |
| $w_5^\star$ | | | | 640042.26 |
| $w_6^\star$ | | | | -1061800.52 |
| $w_7^\star$ | | | | 1042400.18 |
| $w_8^\star$ | | | | -557682.99 |
| $w_9^\star$ | | | | 125201.43 |

Eine Idee der **Regularisierung:** Könnte man z.B. der Kostenfunktion (Least Squares) einen Term hinzufügen, der das Anwachsen dieser Gewichte effektiv einschränkt?

# Ridge Regression

Ein erstes Beispiel einer wichtigen Idee im maschinellen Lernen - **Regularisierung:**

- Schränken Sie das Modell so ein, dass es sich nicht mehr perfekt an die Trainingsdaten anpassen kann.

**Einfache Form der Regularisierung:** Erzwingen kleiner Gewichte $\boldsymbol{w}$

- Kleine Gewichte führen zu einer glatteren Funktion
- Führen einen „Regularisierungsterm" in die Kostenfunktion ein

$$E_D(\mathbf{w}) + \lambda E_W(\mathbf{w})$$

Datenterm + Regularisierungsterm

- wobei $\lambda$ der sogenannte Regularisierungsparameter ist; ein *Hyperparameter*.
- Dieser muss in der Regel durch objektive Validierungsformen manuell getuned werden (mehr dazu in zukünftigen Vorlesungen).

# Regularized Least Squares / Ridge regression

Mit der Least Squares Fehlerfunktion und einem quadratischen Regularisierer erhalten wir

$$L_{\text{ridge}} = (\boldsymbol{y} - \boldsymbol{\Phi w})^T(\boldsymbol{y} - \boldsymbol{\Phi w}) + \lambda \boldsymbol{w}^T \boldsymbol{w}$$

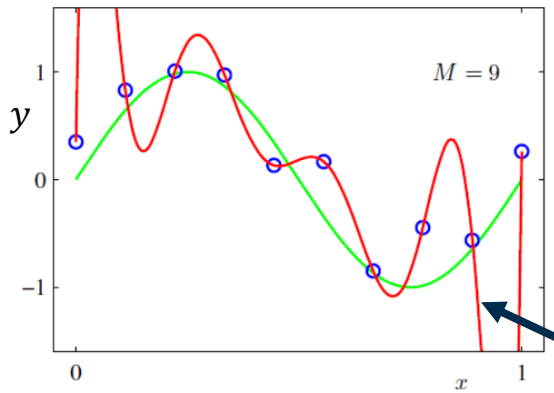
- Diese spezielle Wahl des Regularisierers wird oft als **weight decay** bezeichnet, gerade beim Regularisieren neuronaler Netzwerke, da er in sequenziellen Lernalgorithmen dazu führt, dass die Gewichtswerte gegen Null abfallen, sofern sie nicht durch die Daten gestützt werden.
- Inbesondere in der Statistik wird dies oft als **shrinkage** und **Ridge Regression** bezeichnet.

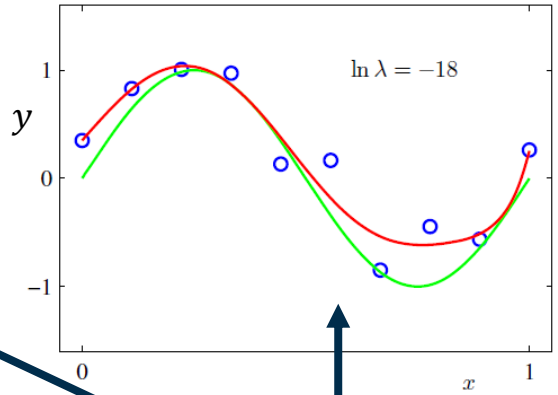
Herleitungen können ähnlich wie zuvor durchgeführt werden. Die Lösung ist

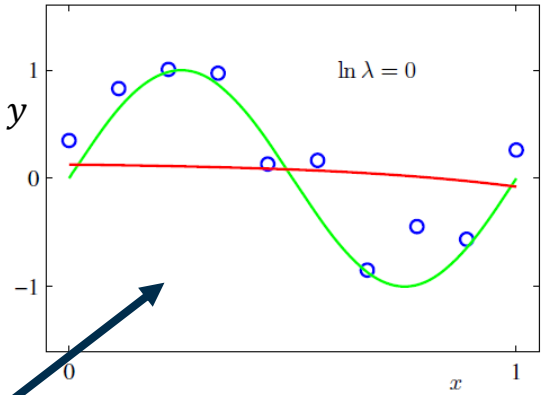
$$\boldsymbol{w}^*_{\text{ridge}} = (\boldsymbol{\Phi}^T \boldsymbol{\Phi} + \lambda \boldsymbol{I})^{-1} \boldsymbol{\Phi}^T \boldsymbol{y}$$

- $\boldsymbol{I}$ ist die Identitätsmatrix
- Wichtiger Vorteil: Die Matrix $(\boldsymbol{\Phi}^T \boldsymbol{\Phi} + \lambda \boldsymbol{I})$ hat nun vollen Rang und kann leichter invertiert werden.

# Einfluss des Regularisierungsparameters auf Polynom 9-ter Ordnung

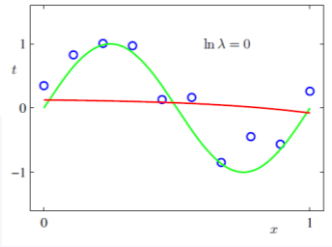



| | $\ln \lambda = -\infty$ | $\ln \lambda = -18$ | $\ln \lambda = 0$ |
|---|---|---|---|
| $w_0^\star$ | 0.35 | 0.35 | 0.13 |
| $w_1^\star$ | 232.37 | 4.74 | -0.05 |
| $w_2^\star$ | -5321.83 | -0.77 | -0.06 |
| $w_3^\star$ | 48568.31 | -31.97 | -0.05 |
| $w_4^\star$ | -231639.30 | -3.89 | -0.03 |
| $w_5^\star$ | 640042.26 | 55.28 | -0.02 |
| $w_6^\star$ | -1061800.52 | 41.32 | -0.01 |
| $w_7^\star$ | 1042400.18 | -45.95 | -0.00 |
| $w_8^\star$ | -557682.99 | -91.53 | 0.00 |
| $w_9^\star$ | 125201.43 | 72.68 | 0.01 |

# Einfluss des Regularisierungsparameters




You cannot vote anymore

Was passiert in diesem Fall lambda =1? (mehrere Antworten können richtig sein)

| | | | |
|---|---|---|---|
| 1 | Unteranpassung / High-bias | 59% | 80 |
| 2 | Überanpassung / Overfitting | 9% | 12 |
| 3 | Wir fitten nun ein Polynom 1. Ordnung | 16% | 22 |
| 4 | Die effektive Komplexität des Models wurde erhöht | 4% | 6 |
| 5 | Die Gewichte wurden so stark beschränkt, dass kein guter Fit mehr möglich ist | 76% | 103 |

# Überblick

## Lineare Regression

- Einführung in die Regression
- Differentialrechnung mit Matrizen und Vektoren (Matrix Calculus)
- Lösung nach der Methode der Kleinsten Quadrate (Least Squares)
- Verallgemeinerte Lineare Regression (Generalized Linear Regression)
- Ridge Regression

## Wahrscheinlichkeitstheorie

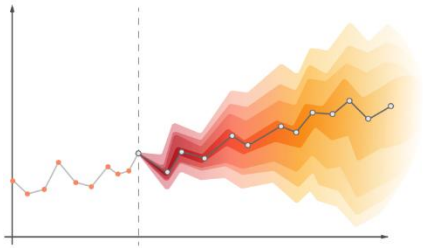
- Probabilistische Modelle
- Bayes'sche Regel (Bayes‘ Rule/Bayes‘ Theorem)
- Erwartungswerte und Monte-Carlo-Schätzung
- Maximum-Likelihood-Schätzung (Maximum Likelihood Estimation)

# Wahrscheinlichkeitstheorie

- Grundlagen und Formulierung probabilistischer Modelle
- Maximum Likelihood Estimation (MLE)

# Probabilistische Modelle – Motivation

Alan Mosca



**Probabilistische Modelle $p_\theta(y|x)$ ermöglichen es uns zusätzlich die Unsicherheit der Vorhersage zu erfassen.**

- Oft ist es wichtig zu wissen, wie ***(un)sicher*** die Vorhersage tatsächlich ist
- Ein großer Teil der ML-Forschung befasst sich mit der Frage, wie gute probabilistische Modelle geschätzt werden können
- Die zugrunde liegenden Herleitungen ähneln sehr dem „optimization view“

– Es ist sehr informativ, die Zusammenhänge/Unterschiede zu verstehen

- Sie werden uns als Grundlage für den zweiten und dritten Teil der Vorlesung dienen

# Notation (bewusst einfach gehalten)

- Eine **Zufallsvariable** $X$ stellt ungewisse Zustände oder Ausprägungen der Welt dar
- Wir schreiben $p(x)$ für die Wahrscheinlichkeit, dass $X$ den Wert $x$ annimmt
- Der Ereignisraum ist die Menge aller möglichen Werte, die $X$ annehmen kann
  - Dieser kann diskret, kontinuierlich oder gemischt sein

- $p(x)$ ist die **Wahrscheinlichkeitsmasse-/dichtefunktion**
  - Weist jedem Punkt des Ereignisraums eine Zahl zu
  - Nicht-negativ, summiert (integriert) sich zu 1
  - Intuitiv: Wie oft tritt $x$ auf? Wie sehr glauben wir an $x$?

# Wahrscheinlichkeitsverteilungen

- **Gemeinsame Verteilung**
  (joint distribution)

  $p(x, y)$

  Wahrscheinlichkeit, dass $X = x$ und $Y = y$

- **Bedingte Verteilung**
  (conditional distribution)

  $p(x|y)$

  Wahrscheinlichkeit, dass $X = x$ gegeben $Y = y$

Conditional Distributions

$P(W|T)$

$P(W|T = hot)$

| W | P |
|---|---|
| sun | 0.8 |
| rain | 0.2 |

$P(W|T = cold)$

| W | P |
|---|---|
| sun | 0.4 |
| rain | 0.6 |

Joint Distribution

$P(T, W)$

| T | W | P |
|---|---|---|
| hot | sun | 0.4 |
| hot | rain | 0.1 |
| cold | sun | 0.2 |
| cold | rain | 0.3 |

# Regeln der Wahrscheinlichkeitsrechnung

- **Summenregel (Marginalisierung/Integration):**

$$p(x) = \sum_y p(x, y)$$

$$p(x_1) = \sum_{x_2} \sum_{x_3} \cdots \sum_{x_D} p(x_1, \ldots, x_D)$$

  - **Hinweis:** Bei kontinuierlichen Verteilungen werden die Summen durch Integrale ersetzt.

$P(T, W)$

| T | W | P |
|---|---|---|
| hot | sun | 0.4 |
| hot | rain | 0.1 |
| cold | sun | 0.2 |
| cold | rain | 0.3 |

$P(t) = \sum_w P(t, w)$

$P(T)$

| T | P |
|---|---|
| hot | 0.5 |
| cold | 0.5 |

$P(w) = \sum_t P(t, w)$

$P(W)$

| W | P |
|---|---|
| sun | 0.6 |
| rain | 0.4 |

# Regeln der Wahrscheinlichkeitsrechnung

- **Produktregel**

$$p(x, y) = p(x|y)p(y)$$

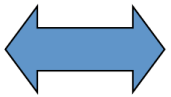
$$p(x_1, \cdots, x_D) = p(x_1)p(x_2|x_1)\ldots p(x_D|x_1, \ldots, x_{D-1})$$

$P(W)$

| W | P |
|---|---|
| sun | 0.8 |
| rain | 0.2 |

$P(D|W)$

| D | W | P |
|---|---|---|
| wet | sun | 0.1 |
| dry | sun | 0.9 |
| wet | rain | 0.7 |
| dry | rain | 0.3 |



$P(D, W)$

| D | W | P |
|---|---|---|
| wet | sun | 0.08 |
| dry | sun | 0.72 |
| wet | rain | 0.14 |
| dry | rain | 0.06 |

# Satz von Bayes

**Der Satz von Bayes** ist eine der wichtigsten Gleichungen in der Wahrscheinlichkeitstheorie und im maschinellen Lernen

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)} = \frac{p(y|x)p(x)}{\sum_{x'} p(y|x')p(x')}$$

- Methode zur „Umkehrung” der bedingten Wahrscheinlichkeiten
- Oft ist eine Bedingung schwierig, die andere hingegen einfach

Typisch in maschinellem Lernen:

$$p(\mathbf{w}|\mathcal{D}) = \frac{p(\mathcal{D}|\mathbf{w})p(\mathbf{w})}{p(\mathcal{D})}$$

mit Parametern $\boldsymbol{w}$ und Beobachtungsdaten $D$

# Erwartungswerte

Der durchschnittliche Wert einer Funktion $f(x)$ unter einer Wahrscheinlichkeitsverteilung $p(x)$ wird der Erwartungswert $\mathbb{E}[f(x)]$ genannt.

Für kontinuierliche Variablen ist dieser durch das folgende Integral gegeben

$$\mathbb{E}_p[f(x)] = \int p(x)f(x)dx$$

Ein **bedingter Erwartungswert** (conditional expectation) ist gegeben durch

$$\mathbb{E}_p[f(x)|Y = y] = \int p(x|y)f(x)dx$$

**Produktregel für Erwartungswerte:**

$$\mathbb{E}_{p(x,y)}[f(x)] = \int p(y)\mathbb{E}[f(x)|Y = y]dy$$

# Monte-Carlo-Schätzung

**Erwartungswerte können immer durch Stichproben approximiert werden:**

$$\mathbb{E}_p[f(x)] = \int p(x)f(x)dx \approx \frac{1}{N} \sum_{x_i \sim p(x)} f(x_i)$$

Notwendig, wenn keine analytische Lösung zur Berechnung des Integrals existiert (ein sehr typischer Fall!)

# Statistische Momente

**Momente sind Erwartungenswerte:**

- 1. Moment (Mittelwert): $\boldsymbol{\mu} = \mathbb{E}_p[\boldsymbol{x}]$
- 2. Moment: $\boldsymbol{M}_2 = \mathbb{E}_p[\boldsymbol{x}\boldsymbol{x}^T]$

**Zentrale Momente werden immer relativ zum Mittelwert berechnet:**

- 2. zentrales Moment (Kovarianz):

$$\boldsymbol{\Sigma} = \mathbb{E}_p[(\boldsymbol{x} - \boldsymbol{\mu})(\boldsymbol{x} - \boldsymbol{\mu})^T]$$

- Erfasst Variabilität (diagonale Einträge) und Korrelation (außerdiagonale Einträge)

# Wahrscheinlichkeitsverteilungen

**Bernoulli-Verteilung:**

- Binäre Zufallsvariable $X \in \{0, 1\}$
- Ein Parameter $p(X = 1) = \mu$
- Wahrscheinlichkeitsverteilung $p(x) = \mu^x (1 - \mu)^{(1-x)}$

- Kann mich sich vorstellen wie das Werfen einer Münze

# Wahrscheinlichkeitsverteilungen

**Multinomiale / kategoriale Verteilung:**

- K verschiedene Ereignisse: $C \in \{1, \ldots, K\}$
- Direkte Angabe der Wahrscheinlichkeiten: $p(C = k) = \mu_k, \quad \mu_k \geq 0, \quad \sum_{k=1}^{K} \mu_k = 1$
- Oder geschrieben mit “one-hot-encoding” (ohne „if“-Klausel)

$$p(c) = \prod_{k=1}^{K} \mu_k^{\boldsymbol{h}_{c,k}}$$

wobei $\boldsymbol{h}_c$ der K-dimensionale one-hot-encoding Vektor ist, der für die Dimension $c = k$ den Wert 1 und ansonsten den Wert 0 hat. $\boldsymbol{h}_{c,k}$ ist das k-te Element dieses Vektors.

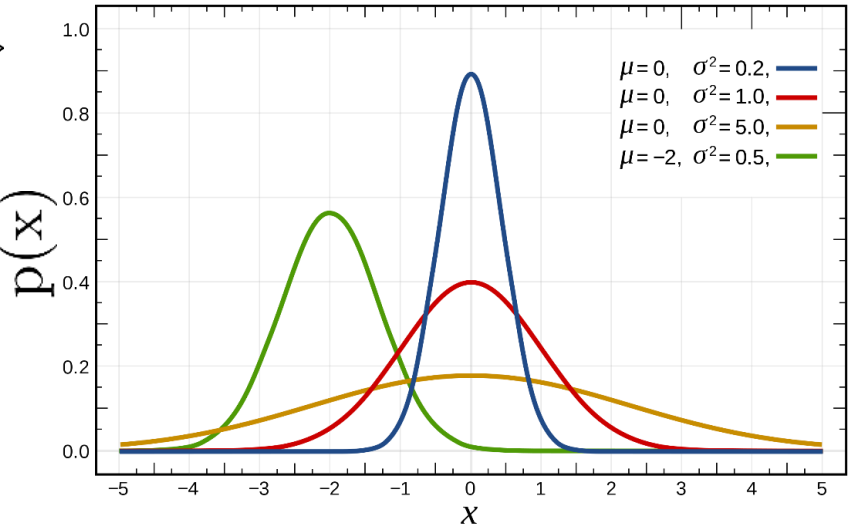
- Kann man sich vorstellen wie würfeln

# Wahrscheinlichkeitsverteilungen

**Gauß-Verteilung:**

- Kontinuierliche Zufallsvariable: $X \in \mathbb{R}$
- Die Verteilung ist vollständig durch den Mittelwert $\mu$ und Varianz $\sigma^2$ definiert

$$p(x) = \mathcal{N}(x|\mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\}$$



# Wahrscheinlichkeitstheorie

- Grundlagen und Formulierung probabilistischer Modelle
- Maximum Likelihood Estimation (MLE)

# Maximum Likelihood Estimation (MLE)

- Gegeben: **identisch unabhängig verteilte (iid)** Trainingsdaten $D = \{(x_i, y_i)\}_{i=1\ldots N}$ aus der der Datenverteilung $p_{\text{data}}$
- Sei $p_{\boldsymbol{\theta}}(x, y)$ eine Familie von Verteilungen, parametrisiert durch $\boldsymbol{\theta} \in \boldsymbol{\Theta}$
- Wir möchten $\boldsymbol{\theta}$ so finden, dass $p$ die Daten gut fittet

**Likelihood von $\theta$ für einen einzelnen Datenpunkt:**
(könnte z.B. einer Gauß-Verteilung folgen)

$$\text{lik}(\boldsymbol{\theta}; x_i, y_i) = p_{\boldsymbol{\theta}}(x_i, y_i)$$

**Likelihood von $\theta$ für den gesamten Datensatz (iid-Annahme):**

$$\text{lik}(\boldsymbol{\theta}; D) = \prod_i p_{\boldsymbol{\theta}}(x_i, y_i)$$

# Maximum Likelihood Estimation (MLE)

**Die Log-Likelihood lässt sich leichter optimieren:**

$$\text{loglik}(\boldsymbol{\theta}; D) = \sum_i \log p_{\boldsymbol{\theta}}(x_i, y_i)$$

- Log ist monoton zunehmende Funktion → gleiches Optimum
- Summen lassen sich „besser" optimieren als Produkte
- Der Logarithmus hebt die Exponentialform auf (die meisten Verteilungen gehören zur Exponentialfamilie, z.B. Gauß-Verteilung)

**Die MLE-Lösung ergibt sich dann aus:**

$$\boldsymbol{\theta}_{\text{ML}} = \text{argmax}_{\boldsymbol{\theta}} \text{loglik}(\boldsymbol{\theta}; D)$$

# Beispiel: lineares Gauß-Modell

$$p(y|\boldsymbol{x}) = \mathcal{N}(y|\boldsymbol{w}^T\tilde{\boldsymbol{x}}, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{(y - \boldsymbol{w}^T\tilde{\boldsymbol{x}})^2}{2\sigma^2}\right\}$$

$$\log p(y|\boldsymbol{x}) = -\log(\sqrt{2\pi\sigma^2}) - \frac{(y - \boldsymbol{w}^T\tilde{\boldsymbol{x}})^2}{2\sigma^2}$$

Wir betrachten das folgende bedingte Gaußsche Modell:

$$p_{\boldsymbol{\theta}}(y|\boldsymbol{x}) = \mathcal{N}(y|\boldsymbol{w}^T\tilde{\boldsymbol{x}}, \sigma^2), \quad \boldsymbol{\theta} = \{\boldsymbol{w}\}$$

**Log-Likelihood:**

$$\text{loglik}(\boldsymbol{\theta}; D) = -N \log \sqrt{2\pi\sigma^2} - \sum_i \frac{(y_i - \boldsymbol{w}^T\tilde{\boldsymbol{x}}_i)^2}{2\sigma^2}$$

- Um ***w*** zu erhalten, sind nur die quadratischen Fehler von Bedeutung, d. h.

$$\text{loglik}(\boldsymbol{\theta}; D) = \text{const}_1 - \text{const}_2 \sum_i (y_i - \boldsymbol{w}^T\tilde{\boldsymbol{x}}_i)^2$$

Maximieren hinsichtlich ***w***
*Alternative: minimiere das negative log-likelihood*

- Daher entspricht die MLE-Lösung der Lösung der kleinsten Quadrate!
- **Aber:** Wir können auch die Varianz $\sigma^2$ durch MLE erhalten!

# Zusammenfassung

Sie sind nun in der Lage

- Grundlagen des Matrix Calculus und der Wahrscheinlichkeitstheorie zu nutzen.
- Regressionsprobleme und den zugrunde liegenden Lernprozess zu beschreiben.
- Die Lösung der kleinsten Quadrate in geschlossener Form herzuleiten
  - Nur möglich, wenn die Kostenfunktion quadratisch in den Gewichten ist.
- Verallgemeinerte lineare Regressionsmodelle zu formulieren
  - Nichtlineare Funktionen in $x$ sind in Ordnung, solange sie linear in $w$ sind.
- Eine erste Idee zu erklären um Überanpassung zu vermeiden, indem die Gewichte durch (quadratische) Regularisierung klein gehalten werden.

**Nächste Woche:** Klassifikation, Bewertung von Algorithmen, Logistische Regression

# Weiteres Beispiel: Auswahl Ordnung eines Polynoms