Machine Learning 1

Dr. Harald Bögeholz

Oktober 2023

Was ist Machine Learning?

Computer lernen aus Daten, Aufgaben zu lösen, ohne explizit dafür programmiert zu sein.

Anwendungsbeispiele

- Spam-Filter
- Bilder klassifizieren
- Tumore auf Röntgenbildern erkennen
- Spracherkennung
- Umsatzprognosen treffen
- Kreditkartenbetrug erkennen
- Empfehlungssysteme
- Künstliche Intelligenz für Spiele (Schach, Go, ...)
- Autonomes Fahren
- Chatbots

Arten von Machine Learning

Überwachtes Lernen: Eingaben und die zugehörigen Ausgaben sind bekannt. Daraus lernt das System, zu unbekannten Eingaben korrekte Ausgaben zu generieren.

Unüberwachtes Lernen: Die Daten sind nicht mit zugehörigen Ausgaben versehen. Das System erkennt Zusammenhänge und Muster in den Daten.

Verstärkendes Lernen: Ein Agent lernt durch Beobachten seiner Umgebung die Wirkung seines Handelns und optimiert eine Belohnungsfunktion.

Überwachtes Lernen

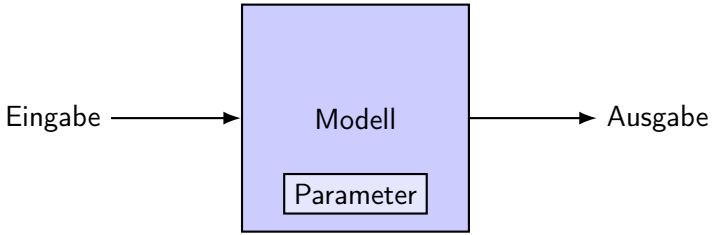
Klassifikation **Diskrete Zielgröße:** Die Ausgabe ist eine von endlich vielen Klassen (Hund, Katze, Maus, ...)

Regression **Kontinuierliche Zielgröße:** Die Ausgabe ist ein Zahlenwert (Preis, Umsatz, Temperatur, ...)

Herkömmliche Programmierung vs. Machine Learning

Herkömmliche Programmierung: Wir schreiben ein Programm.

Herkömmliche Programmierung vs. Machine Learning



Machine Learning:

- Wir *wählen* ein Modell.
- Wir *trainieren* das Modell anhand von Trainingsdaten, d.h. wir bestimmen die Parameter des Modells.
- Wir *evaluieren* das Modell anhand von Testdaten.

High-End-Beispiel: GPT-4

Sprachmodell GPT-4, die Basis von ChatGPT von OpenAI:

- Eingabe: Text
- Ausgabe: Text
- Parameter: 8 Modelle mit jeweils 220 Milliarden Parametern

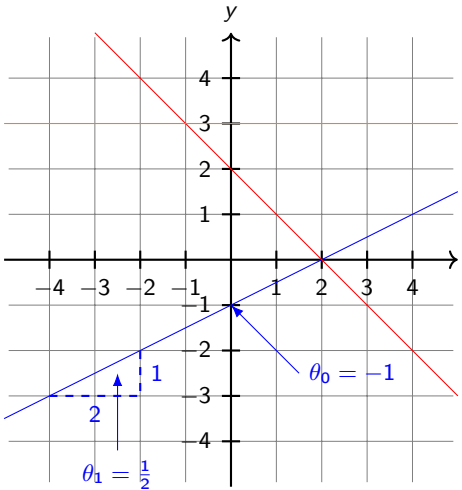
Quelle: https://medium.com/@mlubbad/the-ultimate-guide-to-gpt-4-parameters-everything-you-need-to-know-about-nlps-game-changer-109b8767855a

Einfaches Beispiel: Lineares Modell

Einfaches Beispiel: Lineares Modell $y = \theta_0 + \theta_1 x$.

- Eingabe: Eine reelle Zahl x
- Ausgabe: Eine reelle Zahl y
- Parameter: Zwei reelle Zahlen θ_0, θ_1, die y-Achsenabschnitt und Steigung einer Geraden bestimmen.

Geradengleichung

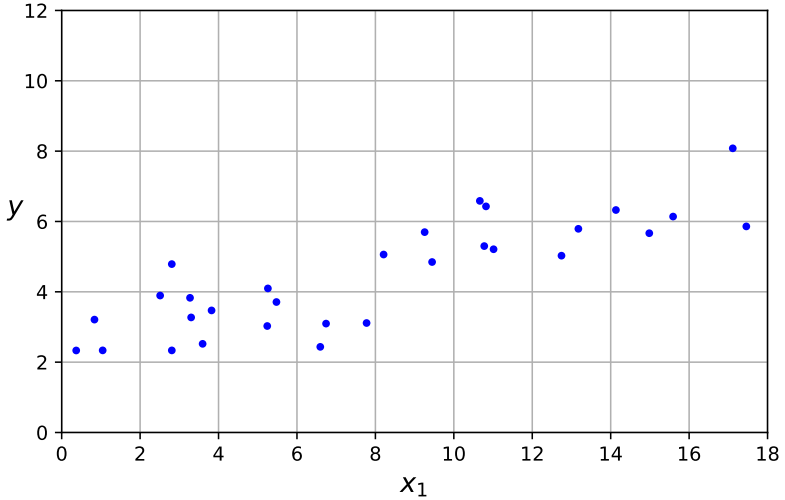


$y = 3 \quad (\theta_0 = 3,\ \theta_1 = 0)$

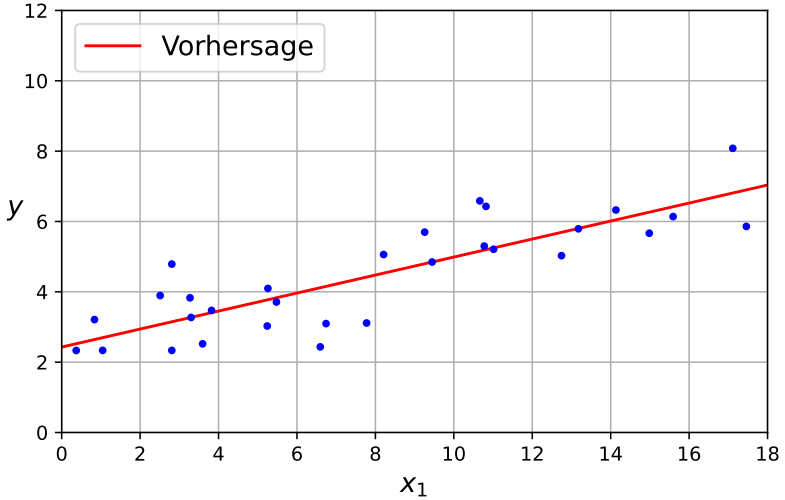
$y = -1 + \frac{1}{2}x \quad (\theta_0 = -1,\ \theta_1 = \frac{1}{2})$

$y = 2 - x \quad (\theta_0 = 2,\ \theta_1 = -1)$

Beispiel: Lineare Regression



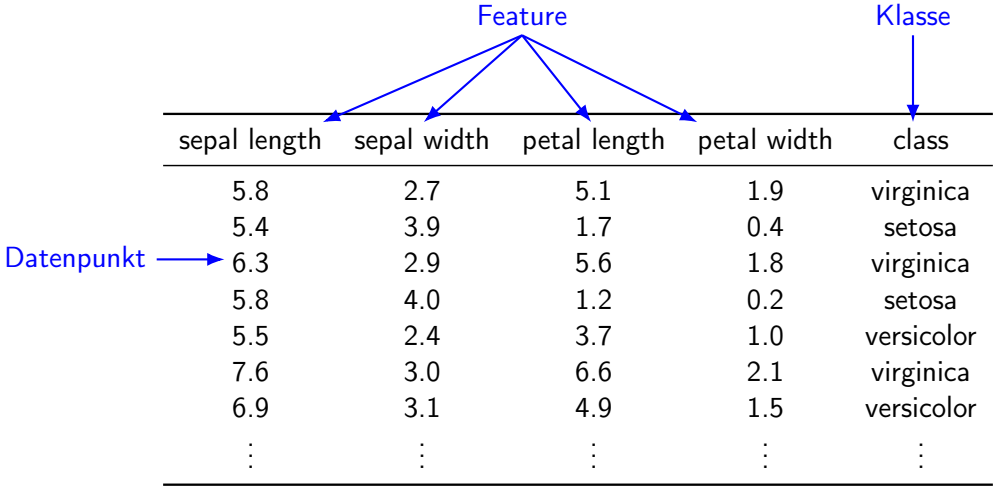
Beispiel: Lineare Regression



ML-Terminologie

Datensatz:	Eine Menge an Datenpunkten (Samples)
Datenpunkt (Sample):	Beim überwachten Lernen eine Eingabe mit zugehöriger Ausgabe
Merkmal (Feature):	Eine Eingabe kann aus mehreren Werten bestehen. Jeder einzelne ist ein Merkmal (Feature)
Klasse:	Ein möglicher Wert für die Zielgröße (bei Klassifizierung)

Beispiel: Der Iris-Datensatz



sepal length	sepal width	petal length	petal width	class
5.8	2.7	5.1	1.9	virginica
5.4	3.9	1.7	0.4	setosa
6.3	2.9	5.6	1.8	virginica
5.8	4.0	1.2	0.2	setosa
5.5	2.4	3.7	1.0	versicolor
7.6	3.0	6.6	2.1	virginica
6.9	3.1	4.9	1.5	versicolor
⋮	⋮	⋮	⋮	⋮

Notation

- n ist die Anzahl der Features
- m ist die Anzahl der Datenpunkte
- $x^{(i)}$ ist ein Vektor mit allen Features des i-ten Datenpunktes (Input-Variablen). Beispiel:

$$x^{(1)} = \begin{pmatrix} 5.8 \\ 2.7 \\ 5.1 \\ 1.9 \end{pmatrix}$$

- x^T steht für den transponierten Vektor x, d.h. aus einer Spalte wird eine Zeile:

$$(x^{(1)})^T = \begin{pmatrix} 5.8 & 2.7 & 5.1 & 1.9 \end{pmatrix}$$

Notation

- X ist eine Matrix mit den Feature-Werten aller Datenpunkte. Jeder Datenpunkt ist eine Zeile in der Matrix.

$$X = \begin{pmatrix} (x^{(1)})^T \\ (x^{(2)})^T \\ (x^{(3)})^T \\ \vdots \\ (x^{(m)})^T \end{pmatrix} = \begin{pmatrix} 5.8 & 2.7 & 5.1 & 1.9 \\ 5.4 & 3.9 & 1.7 & 0.4 \\ 6.3 & 2.9 & 5.6 & 1.8 \\ 5.8 & 4.0 & 1.2 & 0.2 \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix}$$

- $y^{(i)}$ ist der gewünschte Output (Label) des i-ten Datenpunktes. Beispiel:

$$y^{(1)} = 2 \quad \text{(der Label } \texttt{virginica} \text{ ist als die Zahl 2 codiert)}$$

- y ist der Vektor aller Outputs $y^{(i)}$.

Lineare Regression

- Lineare Regression: $\hat{y} = h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$.
- Die Schreibweise $\hat{y}$ bezeichnet den vom Modell vorhergesagten Wert, während y der tatsächliche Wert eines Datenpunktes ist.
- Die Funktion $h_\theta(x)$ heißt auch Hypothesen-Funktion. Sie ist parameterisiert durch den Vektor

$$\theta = \begin{pmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \vdots \\ \theta_n \end{pmatrix}$$

Die Hypothesen-Funktion der linearen Regression lässt sich auch als Skalarprodukt von Vektoren bzw. als Matrixmultiplikation formulieren. Wir ergänzen dazu den Feature-Vektor x um ein Element $x_0 = 1$. Dann haben wir

$$\hat{y} = h_\theta(x) = \theta^T x = \begin{pmatrix} \theta_0 & \theta_1 & \theta_2 & \cdots & \theta_n \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$$

Mittlerer quadratischer Fehler

Ein Maß für den Fehler eines Regressionsmodells ist die *mittlere quadratische Abweichung* (Mean Square Error, MSE):

$$\mathrm{MSE}(X, h_\theta) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

Besser interpretierbar ist die Wurzel der mittleren quadratischen Abweichung (Root Mean Square Error, RMSE):

$$\mathrm{RMSE}(X, h_\theta) = \sqrt{\mathrm{MSE}(X, h_\theta)} = \sqrt{\frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2}$$

Mittlere absolute Abweichung

Ein anderes Maß ist die *mittlere absolute Abweichung* (Mean Absolute Error, MAE):

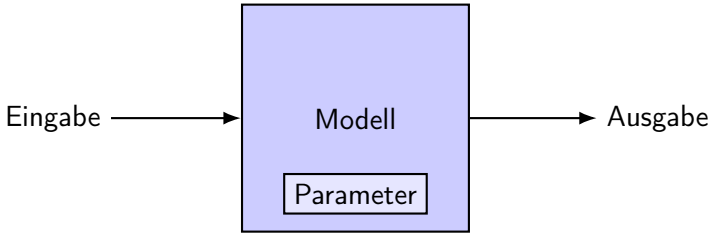
$$\mathrm{MAE}(X, h_\theta) = \frac{1}{m} \sum_{i=1}^{m} \left| h_\theta(x^{(i)}) - y^{(i)} \right|$$

Es gilt immer

$$\mathrm{MAE}(X, h_\theta) \leq \mathrm{RMSE}(X, h_\theta)$$

Der mittlere quadratische Fehler gewichtet größere Abweichungen stärker als kleinere. Er ist das bevorzugte Gütemaß bei Regression; im Falle der linearen Regression führt er auch zu einer eleganten mathematischen Lösung.

Erinnerung: Machine Learning



Machine Learning:

- Wir *wählen* ein Modell.
- Wir *trainieren* das Modell anhand von Trainingsdaten, d.h. wir bestimmen die Parameter des Modells.
- Wir *evaluieren* das Modell anhand von Testdaten.

- Lineares Modell: $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$
- Trainieren heißt: Bestimme den Parameter-Vektor θ so, dass der mittlere quadratische Fehler $\mathrm{MSE}(X, h_\theta)$ minimiert wird.
- Hierfür fügen wir der Feature-Matrix X ein Dummy-Feature hinzu, das überall den Wert 1 hat:

$$X = \begin{pmatrix} 1 & x_1^{(1)} & x_2^{(1)} & x_3^{(1)} & \cdots & x_n^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & x_3^{(2)} & \cdots & x_n^{(2)} \\ 1 & x_1^{(3)} & x_2^{(3)} & x_3^{(3)} & \cdots & x_n^{(3)} \\ 1 & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(m)} & x_2^{(m)} & x_3^{(m)} & \cdots & x_n^{(m)} \end{pmatrix}$$

Für die Minimierung des MSE gibt es eine geschlossene Formel. Das Optimum $\hat{\theta}$ errechnet sich als

$$\hat{\theta} = (X^T X)^{-1} X^T y.$$

- Die einzelnen Faktoren sind Matrizen, die per Matrix-Multiplikation multipliziert werden. **In Numpy:** Der Operator `@` multipliziert Matrizen.
- X^T steht für die zu X transponierte Matrix, d.h. die Matrix wird an der Diagonalen gespiegelt, Zeilen und Spalten tauschen ihre Rollen. **In Numpy:** `X.T`.
- X^{-1} ist die Inverse der Matrix X. **In Numpy:** `np.linalg.inv(X)`.

Das Bestimmtheitsmaß

- Zum Beurteilen der **Qualität** des Modells kann man das *Bestimmtheitsmaß* (Determinationskoeffizient, Coefficient of determination, R^2) heranziehen.
- **Idee:** Vergleiche die Fehlerquadratsumme des Modells mit der Varianz der Daten
- **Definition:**

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^{m} \left(y^{(i)} - \hat{y}^{(i)}\right)^2}{\sum_{i=1}^{m} \left(y^{(i)} - \bar{y}\right)^2}$$

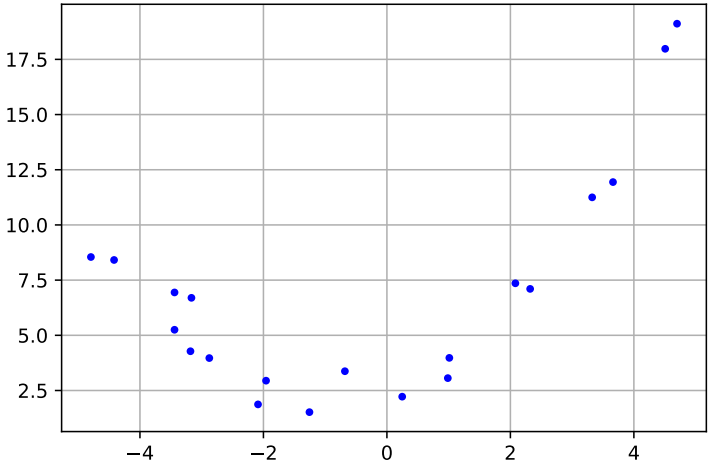
  Dabei ist $\bar{y}$ der Mittelwert des Vektors y.
- **Vorteil:** Es ist ein Prozent-Wert unabhängig von der Größenordnung der Daten, daher leichter interpretierbar

Das Bestimmtheitsmaß

Interpretation des Bestimmungsmaßes R^2:

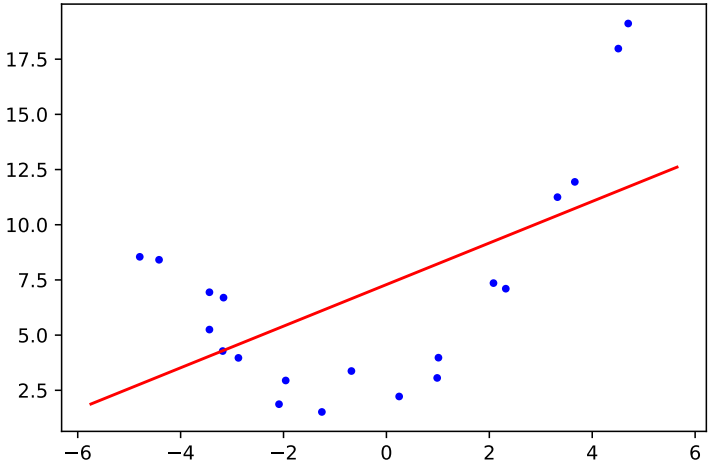
- $R^2 = 0$: Modell ist nur so gut wie der Mittelwert.
- $R^2 > 0$: Modell ist besser als der Mittelwert.
- $R^2 = 1$: Modell ist perfekt.
- $R^2 < 0$: Negative Werte sind möglich; Modell ist schlechter als Mittelwert.

Polynomiale Regression



Polynomiale Regression

Eine Gerade passt nicht auf jeden Datensatz:



Polynomiale Regression

Wenn x ein einzelnes Feature ist (kein Vektor), können wir als Modell ein Polynom n-ten Grades wählen:

$$h_\theta(x) = \theta_0 + \theta_1 x + \theta_2 x^2 + \cdots + \theta_n x^n$$

Wir bestimmen die Parameter θ_i, indem wir lineare Regression auf die Features x, x^2, $\ldots$, x^n anwenden.