

Datenzentrierte Informatik

Prof. Dr. Elena Demidova

Arbeitsgruppe Data Science & Intelligente Systeme (DSIS)

Abteilung III: Informationssysteme und Künstliche Intelligenz
Institut für Informatik
Rheinische Friedrich-Wilhelms-Universität Bonn

WS 25/26



Überwachtes Lernen, Klassifikation und Modellauswahl

Thematische Einordnung und Lernziele

- Thematische Einordnung:
 - Datenzentrierte Informatik → Maschinelles Lernen
- notwendige Vorkenntnisse:
 - -
- Lernziele:
 - Überwachtes Lernen
 - Erlernen einer Klasse anhand von Beispielen
 - Erlernen multipler Klassen
 - Modellauswahl und Generalisierung
- Vorlesung basiert auf Kapitel 2 in [1]

„Maschinelles Lernen befasst sich mit Computerprogrammen, die ihre Leistung durch Erfahrung automatisch verbessern.“

– Herbert Alexander Simon

- „Erfahrung“ kann Daten, Interaktion mit der Umgebung usw. bedeuten
- maschinelles Lernen erfolgt durch die Angleichung mathematischer Modelle an Erfahrung (Daten)

Überwachtes Lernen: Lernen aus Beispielen

- **Ziel:** Erlernen einer Funktion zur Abbildung von Eingabewerten \mathbf{x} auf den Ausgabewert y (bzw. Ausgabevektor \mathbf{y})
 - **Klassifikation:** y ist ein Klassencode (z. B. 0/1)
 - **Regression:** y ist ein Zahlwert
- **Mashine-Learning-Ansatz**
 - Erlernen eines Vorhersagemodells $m(\cdot)$ mit Parametern θ

$$\hat{\mathbf{y}} = m(\mathbf{x}|\theta)$$

- **Ziel:** Minimierung der Fehlerrate
 - die Vorhersage $\hat{\mathbf{y}}$ soll möglichst nahe an der wahren Ausgabe \mathbf{y} liegen: $\hat{\mathbf{y}} \approx \mathbf{y}$
- **Lernprozess:**
 - Das Vorhersagemodell wird anhand einer Trainingsmenge $\mathcal{D} = \{(\mathbf{x}^1, \mathbf{y}^1), (\mathbf{x}^2, \mathbf{y}^2), \dots, (\mathbf{x}^n, \mathbf{y}^n)\}$ optimiert
 - Parameter θ werden dabei so optimiert, dass die empirische Fehlerrate des Modells auf der Trainingsmenge minimiert wird

Trainingsmenge

- **Trainingsmenge (engl. training set)** \mathcal{D} enthält N Beispiele der Form: $(\mathbf{x}^i, \mathbf{y}^i)$
 - \mathbf{x}^i : der Eingabevektor (beliebig dimensioniert)
 - Beispiel: Merkmale der Immobilien (z. B. Preis, Zimmerzahl, Lage, Wohnfläche)
 - \mathbf{y}^i : die gewünschte Ausgabe (Label)
 - bei zwei Klassen: ein binärer Wert $y^i \in \{0, 1\}$
 - bei K Klassen: ein K -dimensionaler Vektor $\mathbf{y}^i \in \{0, 1\}^K$ (One-Hot-Codierung)
- **Annahme: Unabhängigkeit und identische Verteilung** der Stichprobe \mathcal{D} (engl. independent and identically distributed)
 - alle Instanzen von \mathcal{D} wurden aus derselben gemeinsamen Verteilung $p(\mathbf{x}, \mathbf{y})$ zufällig ausgewählt
 - alle Instanzen von \mathcal{D} sind unabhängig voneinander

Erlernen einer Klasse anhand von Beispielen

Erlernen einer Klasse anhand von Beispielen

Erlernen einer Klasse \mathcal{C} : Beispiel „Familienhaus“

- **Gegeben:**
 - eine Menge an Beispielen: Häuser mit Merkmalen
 - z. B. Preis, Zimmerzahl, Fläche, Lage
 - eine Gruppe von Personen zur Annotation
 - z. B. Immobilienexperten
- **positive Beispiele (engl. positive examples):**
 - Häuser, die von Befragten als Familienhaus klassifiziert wurden
 - Beispiel: 200 T. EUR, 4 Zimmer, 120 m², ruhige Lage
- **negative Beispiele (engl. negative examples):**
 - alle anderen Häuser, die nicht als Familienhaus gelten
 - Beispiel: 120 T. EUR, 2 Zimmer, 50 m², zentrale Lage
- **Ziel des Lernens:**
 - ein Modell (mathematische Beschreibung) finden, das auf alle positiven und keine negativen Beispiele zutrifft

Was benötigen wir, um dieses Modell zu erstellen?

Eingaberepräsentation (engl. input representation)

- Welche **Merkmale/Attribute** (engl. **features**) eines Hauses helfen, ein Familienhaus von anderen Häusern zu unterscheiden?
 - Beispiel: Preis und Zimmerzahl sind oft aussagekräftig
 - andere Attribute werden als irrelevant angesehen
- **Eingaberepräsentation**
 - die ausgewählten Attribute bilden den **Eingabevektor x** für den Klassifikator:

$$\mathbf{x} = \begin{bmatrix} \textit{Preis} \\ \textit{Zimmerzahl} \end{bmatrix}$$

Eingaberepräsentation: Beispiel

Wie repräsentieren wir ein Haus in der Trainingsmenge?

- Jedes Haus i in der Trainingsmenge wird durch ein geordnetes Paar $(\mathbf{x}^i; y^i)$ repräsentiert
- Eingabevektor \mathbf{x} enthält die Attributwerte eines Hauses:

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

- Attribute in diesem Beispiel:
 - x_1 : Preis
 - x_2 : Zimmerzahl
- Kennung (engl. label) y gibt die Klasse an:

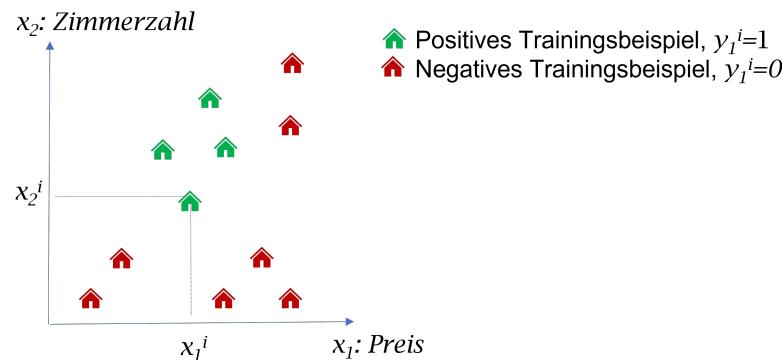
$$y = \begin{cases} 1, & \text{falls } \mathbf{x} \text{ ein positives Beispiel der Klasse } \mathcal{C} \text{ ist} \\ 0, & \text{sonst} \end{cases}$$

- Beispiel:
 - $\mathbf{x} = \begin{bmatrix} 200000 \\ 4 \end{bmatrix}, y = 1.$

Trainingsdaten im 2D-Raum

Trainingsdaten mit zwei Dimensionen können im 2D-Raum ($x_1; x_2$) abgebildet werden

- jede Instanz ($\mathbf{x}^i; y^i$) ist ein Punkt mit:
 - Koordinaten ($x_1^i; x_2^i$): Attributwerte
 - Label y^i : Klassenzugehörigkeit
- Beispiel: ein 2D-Raum, der durch die Merkmale Preis und Zimmerzahl aufgespannt wird
 - Legende: Farbliche Kennzeichnung der Klassen



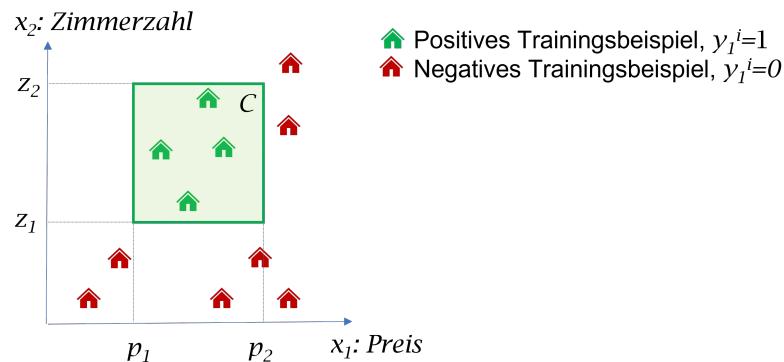
Hypothesenklasse

Hypothesenklasse (engl. hypothesis class) \mathcal{H} definiert die Struktur des Vorhersagemodells $m(\cdot)$

- Annahme: die Klasse \mathcal{C} (Familienhaus) kann als ein Rechteck im 2D-Preis-Zimmerzahl-Raum repräsentiert werden
- In diesem Fall: \mathcal{H} ist die Menge aller Rechtecke in diesem 2D-Raum, bestimmt durch die Parameter:

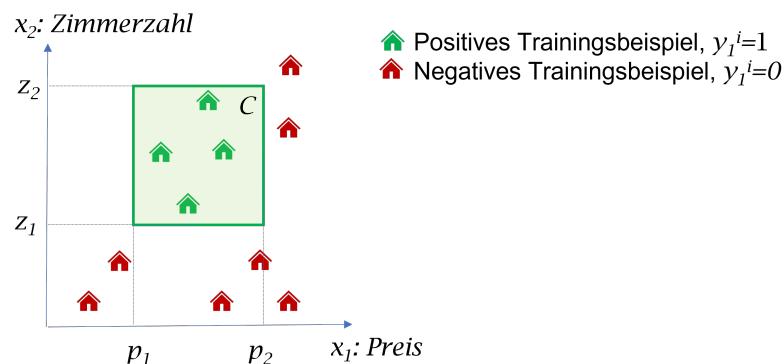
$$\theta = (p_1, p_2, z_1, z_2),$$

$$(p_1 \leq Preis \leq p_2) \text{ AND } (z_1 \leq Zimmerzahl \leq z_2)$$



Hypothese

- **Annahme:**
 - Klasse \mathcal{C} wurde aus der Hypothesenklasse \mathcal{H} gezogen
- **Ziel des Lernalgorithmus:**
 - finde spezielle Hypothese (engl. hypothesis) $h \in \mathcal{H}$, die \mathcal{C} möglichst genau approximiert
 - hier: bestimme die Parameter $\theta = (p_1, p_2, z_1, z_2)$, die h definieren
 - das Erlernen der Klasse wird auf die Suche nach den optimalen Parametern θ^* reduziert
- **Beispiel-Parametervektor:**
 - $\theta = (180000, 450000, 3, 5)$

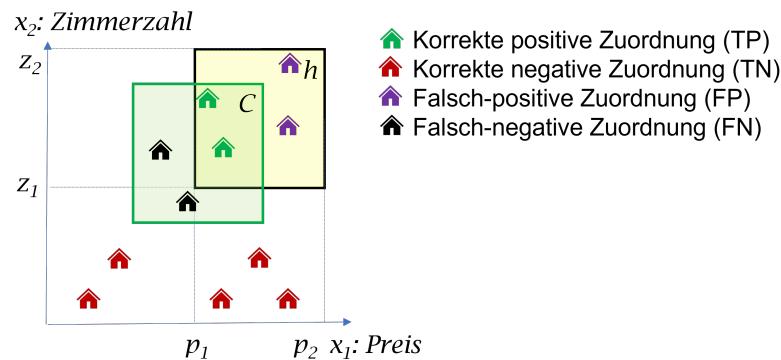


Falsch positive/falsch negative Zuordnung

- Eine Hypothese h trifft eine Vorhersage für eine Instanz $\mathbf{x} \in \mathcal{D}$:

$$h(\mathbf{x}) = \begin{cases} 1, & \text{falls } h \text{ den } \mathbf{x} \text{ als positives Beispiel von } \mathcal{C} \text{ einstuft} \\ 0, & \text{falls } h \text{ den } \mathbf{x} \text{ als negatives Beispiel einstuft} \end{cases}$$

- $y = \mathcal{C}(\mathbf{x})$ ist die tatsächliche Klasse von Instanz \mathbf{x} , $h(\mathbf{x})$ ist die Vorhersage der Hypothese
 - korrekte positive/negative Zuordnung: $h(\mathbf{x}) = \mathcal{C}(\mathbf{x})$
 - falsch-negative Zuordnung (FN): $\mathcal{C}(\mathbf{x}) = 1$, aber $h(\mathbf{x}) = 0$
 - falsch-positive Zuordnung (FP): $\mathcal{C}(\mathbf{x}) = 0$, aber $h(\mathbf{x}) = 1$



Empirische Fehlerrate

- **Empirische Fehlerrate (engl. empirical error)**: relative Klassifikationsfehler für eine endliche Anzahl von Datenpunkten
 - der Anteil der Datenpunkte, bei denen die Vorhersage $\hat{y}^i = h(\mathbf{x}^i)$ nicht mit den tatsächlichen Werten y^i aus \mathcal{D} übereinstimmen

$$E(h|\mathcal{D}) = \frac{1}{N} \sum_{i=1}^N \mathcal{L}(h(\mathbf{x}^i), y^i)$$

- $\mathcal{L}(\cdot)$: Verlustfunktion, die den Fehler zwischen der Vorhersage \hat{y}^i des Modells $h(\mathbf{x}^i)$ und dem tatsächlichen Wert y^i berechnet

- **Verlustfunktion/Fehlerfunktion (engl. loss function) $\mathcal{L}(\cdot)$**
 - bewertet die Qualität der Approximation
 - berechnet den Unterschied zwischen der Modellvorhersage $\hat{y} = h(\mathbf{x})$ und dem Zielwert y
- **0-1-Verlustfunktion (engl. zero-one loss)**

$$\mathcal{L}_{0-1}(\hat{y}, y) = \begin{cases} 1, & \text{wenn } \hat{y} \neq y \\ 0, & \text{wenn } \hat{y} = y \end{cases}$$

- einfach und direkt, oft in theoretischen Analysen verwendet
- nicht differenzierbar, daher nicht für die Optimierung in maschinellen Lernalgorithmen geeignet
- **quadratische Verlustfunktion (engl. squared loss, L2 loss)**

$$\mathcal{L}_2(\hat{y}, y) = (\hat{y} - y)^2$$

- differenzierbar, eignet sich zur Berechnung von Gradienten
- häufig verwendet in Regressionsproblemen

Verlustfunktionen

- **Binäre Kreuzentropie (engl. binary cross-entropy loss)**
 - eine Verlustfunktion für binäre Klassifikation, wenn die Ausgabe als Wahrscheinlichkeit interpretiert wird

$$\mathcal{L}_{BCE}(\hat{y}, y) = -(y \cdot \log(\hat{y}) + (1 - y) \cdot \log(1 - \hat{y}))$$

- $y \in \{0, 1\}$: wahrer Label
- $\hat{y} \in (0, 1)$: Ausgabe des Modells
 - interpretiert als Wahrscheinlichkeit, dass $y = 1$

Beispiel, berechnet mit \ln

y	\hat{y}	\mathcal{L}_{BCE}
1	0,9	$-\log(0,9) \approx 0,105$
1	0,1	$-\log(0,1) \approx 2,303$
0	0,1	$-\log(1 - 0,1) = -\log(0,9) \approx 0,105$
0	0,9	$-\log(1 - 0,9) = -\log(0,1) \approx 2,303$

Ziel der Klassifikation: Minimierung der Fehlerrate auf **neuen, unbekannten Daten** – die **Generalisierungsfähigkeit** des Modells

- **Trainingsdaten-Fehlerrate**

- Fehlerrate auf dem **Trainingsdatensatz**, der zum Lernen des Modells verwendet wurde
- **Risiko:** Überanpassung (engl. overfitting)

- **Testdaten-Fehlerrate**

- Fehlerrate auf einem **separaten Testdatensatz**, der **nicht** in den Lernprozess eingeflossen ist
- **Standardmaß** für die Bewertung von Modellen

- **Wahre Fehlerrate**

- der theoretische Fehler, den das Modell auf allen neuen, bislang nicht gesehenen Datenpunkten macht
- Grenzwert des empirischen Fehlers für eine zunehmende Anzahl neuer Datenpunkte
- ist nicht direkt messbar

Auswahl der Hypothese

Wie wählt man eine Hypothese h aus?

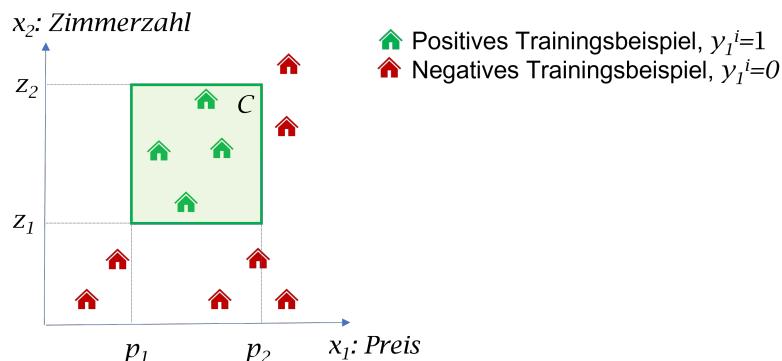
- **Gegeben:**

- Hypothesenklasse \mathcal{H} : die Menge aller möglichen Rechtecke im 2D
- Trainingsdatensatz \mathcal{D}

- **Aufgabe:**

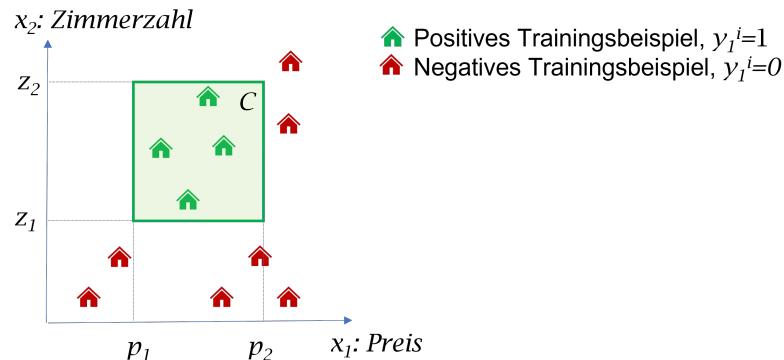
- Hypothese $h \in \mathcal{H}$: ein Rechteck im 2D-Raum
 - definiert durch 4 Parameter: $\theta = (p_1^h, p_2^h, z_1^h, z_2^h)$
- wähle die Parameter θ , sodass die Fehlerrate $E(h|\mathcal{D}) = 0$ ist
- d. h., h schließt alle positiven und keins der negativen Beispiele ein

- Falls x_1, x_2 reelle Werte annehmen: unendlich viele h mit $E = 0$



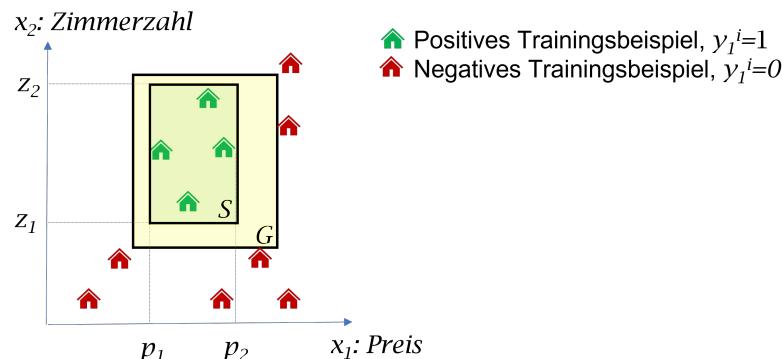
Auswahl der Hypothese: Generalisierung

- Wie wählt man eine Hypothese h , die auch **zukünftige Beispiele**, die nicht Teil der Trainingsmenge \mathcal{D} sind, korrekt klassifiziert?
- Ziel: **Generalisierung** auf unbekannten Daten
 - zukünftige Beispiele können nahe den Grenzen zwischen positiven und negativen Beispielen liegen
- Problem: unterschiedliche Hypothesen können für diese Beispiele unterschiedliche Vorhersagen treffen



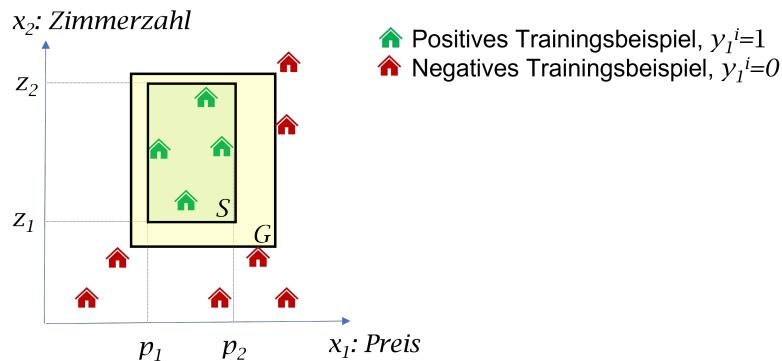
Spezifischste und allgemeinste Hypothesen

- **Spezifischste Hypothese (engl. most specific hypothesis) S**
 - S ist das kleinste Rechteck, das alle positiven und keines der negativen Beispiele einschließt
 - repräsentiert die genaueste Abgrenzung von \mathcal{C} in den Trainingsdaten
- **Allgemeinste Hypothese (engl. most general hypothesis) G**
 - G ist das größte Rechteck, das alle positiven und keines der negativen Beispiele einschließt
 - G umfasst möglicherweise mehr Datenpunkte als notwendig



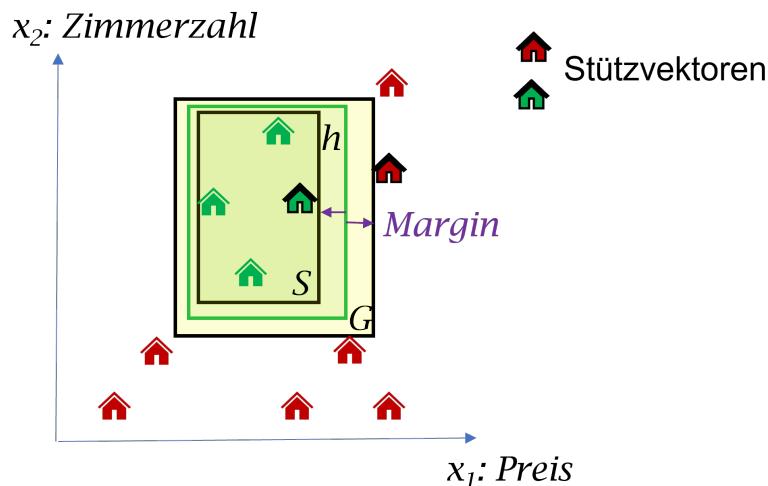
Versionsraum

- **Versionsraum (engl. version space)**: die Menge aller Hypothesen $h \in \mathcal{H}$, die zwischen der S und G liegen
 - gültige Hypothesen ohne Fehler, konsistent zur Trainingsmenge \mathcal{D}
 - der Versionsraum stellt alle möglichen Hypothesen dar, die die Trainingsdaten korrekt klassifizieren
- Gegeben eine andere Trainingsmenge \mathcal{D}' , können S und G variieren
 - wir nehmen an, dass \mathcal{D} hinreichend groß ist, damit S und G eindeutig und fest definiert sind



Margin und Stützvektoren

- **Margin (engl. margin):** der Abstand zwischen der Entscheidungsgrenze und den ihr am nächsten liegenden Instanzen
 - ein intuitiver Ansatz zur Verbesserung der Generalisierung:
 - die Hypothese h so wählen, dass der Margin maximiert wird
 - d.h., der Abstand zwischen der Grenze und den Instanzen, die ihr am nächsten liegen, soll möglichst groß sein
- **Stützvektoren (engl. support vectors):** die Instanzen, die den Margin definieren und die Grenze stützen



Verlustfunktion: Hinge loss

- **Ziel:** Minimierung des Verlusts für die Hypothese h mit maximalem Margin
- **Hinge-Verlustfunktion (engl. hinge loss)**

- Zielvariable $y = \{-1, 1\}$ (binäre Klassifikation)
- \hat{y}_i : Ausgabe des Klassifikators (numerisch)

$$\mathcal{L}_{hinge}(\hat{y}_i, y) = \max(0, 1 - y \cdot \hat{y}_i)$$

- Interpretation
 - wenn die Vorhersage \hat{y}_i das korrekte Vorzeichen hat und der Betrag der Vorhersage größer ist als 1, dann Verlust = 0
 - wenn $y \cdot \hat{y}_i < 1$, ist der Verlust proportional zum Abstand von 1
- Vorteil der Hinge-Verlustfunktion
 - fördert die Maximierung des Margins
- wird oft in Support Vector Machines (SVM) verwendet

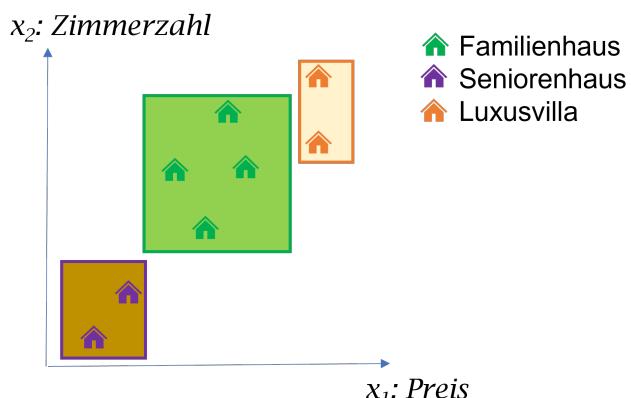
Erlernen mehrerer Klassen

Erlernen multipler Klassen

Erlernen multipler Klassen (engl. learning multiple classes)

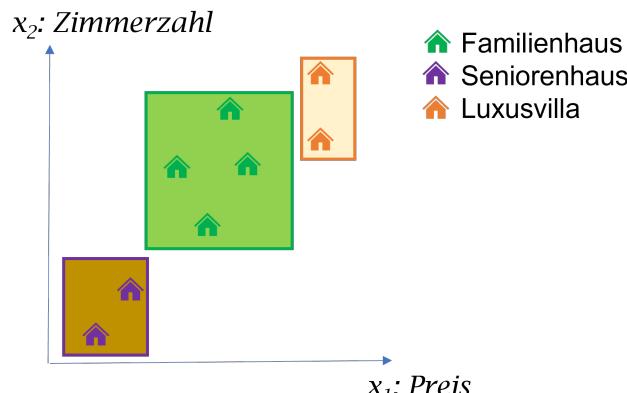
- bisher: Zweiklassenproblem
 - Beispiel: Klassifikation in {Familienhaus, kein Familienhaus}
- im allgemeinen Fall: K Klassen: $\mathcal{C}_k, k = 1, \dots, K$
 - Beispiel: {Familienhaus, Seniorenhaus, Luxusvilla}
 - eine Eingabeinstanz \mathbf{x}^i gehört zu genau einer Klasse
 - $\mathbf{y}^i \in \{0, 1\}^K$ ist ein K -dimensionaler Binärvektor (z. B. [1,0,0]):

$$y_k^i = \begin{cases} 1, & \text{falls } \mathbf{x}^i \in \mathcal{C}_k \\ 0, & \text{sonst} \end{cases}$$



Klassifikation mit K Klassen als K Zweiklassenprobleme

- **Idee:** Wir betrachten Klassifikation mit K Klassen als K Zweiklassenprobleme
 - für jede Klasse \mathcal{C}_k wird eine Hypothese h_k gelernt
- **Trainingsdaten für h_k**
 - Positive Instanzen: Trainingsbeispiele der Klasse \mathcal{C}_k
 - alle $\mathbf{x}^i \in \mathcal{C}_k$
 - Negative Instanzen: Instanzen aller anderen Klassen
 - alle $\mathbf{x}^i \in \mathcal{C}_j, j \neq k$
- **Vorteil:** Einfache Anwendung von Zweiklassen-Modellen
- **Beispiel:** Klassifikation von Immobilien in drei Klassen



Empirische Fehlerrate bei multipler Klassifikation

- **Fehlerrate für eine Klasse \mathcal{C}_k :**

- durchschnittliche Fehler über alle Instanzen bzgl. dieser Klasse

$$E(h_k|\mathcal{D}) = \frac{1}{N} \sum_{i=1}^N \mathcal{L}_{0-1}(h_k(\mathbf{x}^i), y_k^i)$$

- $h_k(\mathbf{x}^i)$: Vorhersage der Hypothese h_k für Instanz \mathbf{x}^i
- y_k^i : Zielwert der Instanz \mathbf{x}^i für die Klasse \mathcal{C}_k
- \mathcal{L}_{0-1} : 0-1-Verlustfunktion

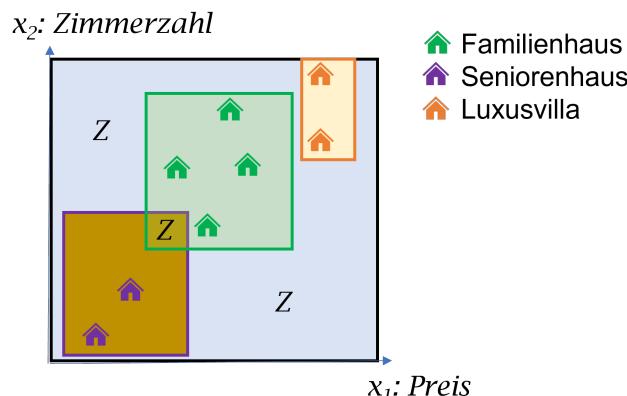
- **Gesamtfehlerrate (engl. the total empirical error)**

- durchschnittliche Fehler über alle Instanzen und alle Klassen

$$E(\{h_k\}_{k=1}^K|\mathcal{D}) = \frac{1}{N \cdot K} \sum_{i=1}^N \sum_{k=1}^K \mathcal{L}_{0-1}(h_k(\mathbf{x}^i), y_k^i)$$

Zweifelsfall bei multipler Klassifikation

- **Idealfall:** eindeutige Klassifikation
 - für ein gegebenes \mathbf{x} ist genau ein $h_k(\mathbf{x}) = 1$
 - der Klassifikator kann die Klasse eindeutig bestimmen: $\hat{y} = \mathcal{C}_k$
- **Zweifelsfall (engl. doubt)**
 - der Klassifikator kann keine Klasse eindeutig bestimmen
 - kein $h_k(\mathbf{x}) = 1 \rightarrow$ **keine Klasse** wird gewählt
 - mehr als ein $h_k(\mathbf{x}) = 1 \rightarrow$ **mehrere Klassen** werden gewählt
 - Zweifelsfälle können vom Klassifikator **abgelehnt** werden
 - **Alternativ:** Verwendung von Wahrscheinlichkeiten (z. B. Softmax)
- **Beispiel:** Z repräsentiert Zweifelsfall-Regionen



Modellauswahl und Generalisierung

Wie mächtig ist eine Hypothesenklasse?

Ist Hypothesenklasse \mathcal{H} flexibel genug, bzw. hat sie ausreichend „Kapazität“, um die Klasse \mathcal{C} zu erlernen?

- **Fall 1:** $\mathcal{C} \in \mathcal{H}$

- die Klasse \mathcal{C} ist in der Hypothesenklasse \mathcal{H} enthalten
- es existiert ein $h \in \mathcal{H}$, sodass $E(h|\mathcal{D}) = 0$
- **Interpretation:** Das Modell hat genügend „Kapazität“, um die Struktur der Daten zu erfassen

- **Fall 2:** $\mathcal{C} \notin \mathcal{H}$

- die Klasse \mathcal{C} ist **nicht** in \mathcal{H} enthalten
- es existiert kein $h \in \mathcal{H}$ mit $E(h|\mathcal{D}) = 0$
- **Interpretation:** Das Modell hat nicht genug „Kapazität“, um die Struktur der Daten zu erfassen, und kann die Klassifikation somit **nicht genau lernen**

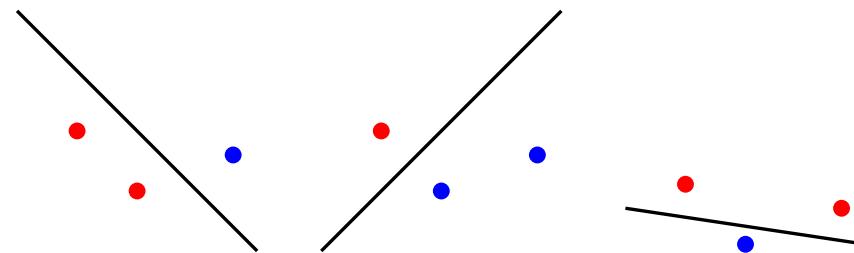
Wie mächtig ist eine Hypothesenklasse?

Kapazität einer Hypothesenklasse: Kann sie alle Einteilungen zerlegen?

- **Gegeben:** Eine Datenmenge mit N Punkten
 - jeder Punkt kann binär klassifiziert werden (z. B. + oder -)
- **Anzahl möglicher Einteilungen:**
 - N Punkte können in 2^N Weisen binär eingeteilt werden
 - Beispiel: $N = 3 \rightarrow 2^3 = 8$ verschiedene Klassifikationen
- **Zerlegung durch \mathcal{H} :**
 - eine Hypothesenklasse \mathcal{H} **zerlegt** die N Punkte, wenn für **jede** der 2^N möglichen Einteilungen eine Hypothese $h \in \mathcal{H}$ existiert, die diese Trennung realisiert
 - das bedeutet, dass \mathcal{H} in der Lage ist, jede beliebige Einteilung der Daten in zwei Klassen zu ermöglichen

Vapnik-Chervonenkis-Dimension (VC-Dimension)

- **VC-Dimension** von \mathcal{H} , $VC(\mathcal{H})$
 - die **maximale Anzahl** von Datenpunkten, die von \mathcal{H} korrekt getrennt (zerlegt) werden können
 - Maß der **Kapazität** (engl. capacity) der Hypothesenklasse \mathcal{H}
- **Beispiel:** Lineare Trennlinien in 2D
 - eine Gerade zerlegt drei nicht kollineare Datenpunkte
 - aber: vier Punkte in konvexer Hülle (z. B. Viereck) können nicht beliebig klassifiziert werden
 - $\rightarrow VC(\mathcal{H}) = 3$
 - Beispiel: $2^3 = 8$ mögliche binäre Anordnungen von 3 Datenpunkten (drei davon sind illustriert):



- VC-Dimension berücksichtigt die Verteilung der Daten nicht

- **Rauschen (engl. noise)**

- ungewollte Anomalien in Daten, die das Lernen von Modellen erschweren und die Modellgenauigkeit beeinträchtigen können

- **Formen von Rauschen:**

- **Messrauschen (engl. input noise)**

- Ungenauigkeit bei der Erfassung der Eingabedaten
 - Beispiel: Sensorfehler, Rundungsfehler, Messfehler

- **Label-Rauschen (engl. teacher noise)**

- falsche Zuweisung von Instanzen zu Klassen
 - Beispiel: Fehler bei der Annotation durch Menschen

- **Unbeobachtetes Rauschen**

- Faktoren, die für die Klassifikation relevant, aber nicht in den aufgezeichneten Daten enthalten sind
 - Beispiel: fehlende Informationen über die Nachbarschaft, die den Hauspreis beeinflussen

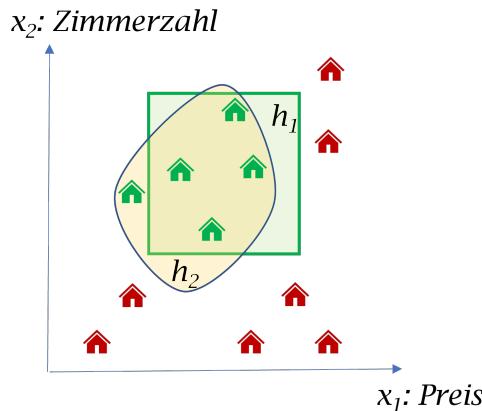
Rauschen in den Daten

- **Problem:**

- bedingt durch Rauschen kann eine Klasse schwerer zu erlernen sein
- ein Fehler gleich null ist mit einer einfachen Hypothesenklasse ist möglicherweise nicht zu erzielen

- **Konsequenz:**

- genaue Abgrenzung erfordert eine Hypothesenklasse von größerer Mächtigkeit
 - ein komplexeres Modell mit einer viel größeren Zahl an Parametern
- Aber: Risiko von **Overfitting** – das Modell lernt das Rauschen statt die wahre Struktur



Ockhams Rasiermesser: Einfachheit hat Vorrang

- **Prinzip:**

- Ockhams Rasiermesser (engl. Occam's razor): nach Wilhelm von Ockham (1288–1347) benanntes Prinzip
- von mehreren hinreichenden möglichen Erklärungen soll die einfachste vorgezogen werden

- **Anwendung im maschinellen Lernen:**

- einfache Modelle haben eine höhere Plausibilität
- einfache Modelle sind oft robuster und besser generalisierbar
- bei zwei Modellen mit ähnlicher Leistung: wähle das Modell mit weniger Parametern, oder weniger Komplexität

• Vorteile einfacher Modelle

- weniger Parameter → schneller zu trainieren
- geringere Varianz → stabileres Modell bei kleinen Änderungen in den Trainingsdaten
- robuster gegen Rauschen → weniger Overfitting
- leichter zu implementieren → schnelle Bereitstellung
- einfacher zu interpretieren → Transparenz
- bessere Generalisierung auf neuen Daten

• Nachteile einfacher Modelle

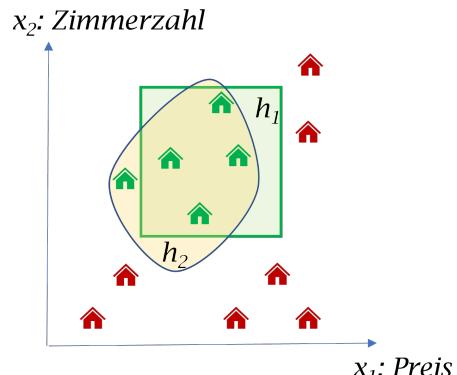
- stärkere Vereinfachung der Realität → können die zugrunde liegende Komplexität des Problems nicht vollständig abbilden
- Starrheit → unflexibel bei komplexen Beziehungen
- höherer Bias (Verzerrung) → führt zu systematischen Fehlern

Wohlformulierte und unbestimmte Probleme

- **Wohlformuliertes Problem (engl. well-posed problem)**
 - Probleme, die eine eindeutige Lösung besitzen, die stabil auf sich ändernde Eingabebedingungen reagiert
 - definiert durch Jacques Hadamard (1865-1963)
- **Unbestimmtes (inkorrekt gestelltes) Problem (engl. ill-posed problem)**
 - die vorhandenen Daten allein nicht ausreichen, um eine eindeutige Lösung zu finden
- **Lösungsstrategie im maschinellen Lernen:**
 - wir müssen zusätzliche Annahmen treffen, um für die vorliegenden Daten eine eindeutige Lösung zu finden
 - Hypothesenklasse $\mathcal{H} \rightarrow$ begrenzt den Suchraum auf sinnvolle Lösungen

Induktive Verzerrung

- **Induktive Verzerrung (engl. *inductive bias*)** des Lernalgorithmus ist die Menge an **Annahmen**, die wir treffen, um das Lernen zu ermöglichen
 - **Beispiel:** Annahme einer Hypothesenklasse \mathcal{H}
- Beispiele für die induktive Verzerrung:
 - Annahme der Form eines Rechtecks für die Klasse „Familienhaus“
 - Wahl des Rechtecks mit der größten Margin
- einfachere, starre Modelle haben eine größere Verzerrung

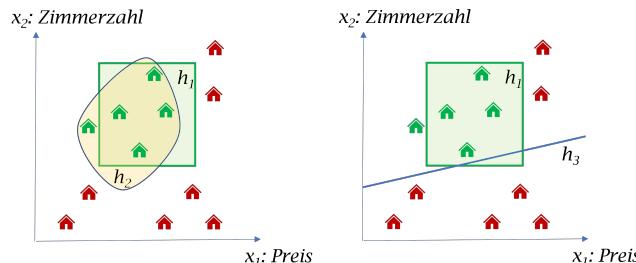


- **Modellauswahl**
 - Ziel: Finden eines Modells, das auf neuen, unbekannten Daten korrekte Vorhersagen trifft
 - Hypothesenklasse \mathcal{H} bestimmt, welche Muster das Modell lernen kann
- **Generalisierung (engl. generalization):**
 - beschreibt die Fähigkeit eines trainierten Modells, auf neue Instanzen außerhalb der Trainingsmenge richtig zu reagieren
- **Zusammenhang: Trainingsfehler vs. Generalisierung**
 - für eine gegebene Hypothesenklasse \mathcal{H} kann man $h \in \mathcal{H}$ mit minimalem Trainingsfehler bestimmen
 - Aber: Generalisierung von h hängt von der Wahl der \mathcal{H} ab
- **Komplexität der Hypothesenklasse**
 - für die bestmögliche Generalisierung muss die Komplexität der Hypothesenklasse an die Komplexität der den Daten zugrundeliegenden Funktion angepasst werden

Unteranpassung und Überanpassung

Das Dilemma der Modellkomplexität

- **Überanpassung (engl. overfitting)**: das Modell ist zu komplex für die zugrundeliegende Struktur der Daten
 - Beispiel: Anpassung von einem Polynom auf (verrauschte) Daten aus einer Gerade
 - Problem: das Modell lernt möglicherweise das Rauschen
 - Folge: schlechte Generalisierung
- **Unteranpassung (engl. underfitting)**: das Modell ist zu einfach für die zugrundeliegende Struktur der Daten
 - Beispiel: eine Gerade (lineare Funktion) approximiert Daten, die einem Polynom dritten Grades folgen
 - Problem: das Modell erfasst wichtige Muster nicht
 - Folge: systematische Fehler



Dreifacher Kompromiss

Der Dreifache Kompromiss (engl. triple trade-off) bei Lernalgorithmen zwischen drei Faktoren:

- **Komplexität der Hypothesenklasse \mathcal{H}**

- **Effekt:**

- wenn die Komplexität der \mathcal{H} sich erhöht, verringert sich der Generalisierungsfehler zunächst, nimmt aber dann wieder zu
 - zu einfach → Unteranpassung → hoher Generalisierungsfehler
 - zu komplex → Überanpassung → hoher Generalisierungsfehler

- **Menge an Trainingsdaten**

- **Effekt:**

- mit zunehmender Menge an Trainingsdaten nimmt der Generalisierungsfehler ab
 - komplexe Modelle profitieren stärker von mehr Daten

- **Generalisierungsfehler**

- **Strategien zur Reduzierung:**

- Reduzierung der Komplexität von \mathcal{H}
 - Erhöhung der Menge an Trainingsdaten

Komponenten eines Algorithmus für überwachtes Lernen

Drei zentrale Komponenten des überwachten Lernens

Gegeben die Trainingsmenge $\mathcal{D} = \{(\mathbf{x}^i, \mathbf{y}^i)\}$, besteht das Ziel des Lernens darin, eine nützliche Approximation $\hat{\mathbf{y}}^i$ für \mathbf{y}^i unter Verwendung des Modells $m(\mathbf{x}^i|\theta)$ zu finden

- **Modell (engl. model) $m(\mathbf{x}^i|\theta)$**
 - $m(\cdot)$: eine Hypothese h , definiert durch die Hypothesenklasse \mathcal{H}
 - \mathcal{H} wird vom Entwickler festgelegt (z. B. SVM, neuronales Netz)
 - ein bestimmter Parameterwert θ instanziert eine Hypothese $h \in \mathcal{H}$
 - der Lernalgorithmus lernt die optimalen Parameter θ^* aus den Trainingsdaten \mathcal{D}
- **Verlustfunktion (engl. loss function) $\mathcal{L}(\cdot)$**
 - misst den Unterschied zwischen der tatsächlichen Ausgabe \mathbf{y}^i und der Vorhersage $\hat{\mathbf{y}}^i = m(\mathbf{x}^i|\theta)$ durch das Modell
 - Beispiel: Kreuzentropie
- **Optimierungsprozedur (engl. optimization procedure)**
 - sucht den Wert θ^* , der die empirische Fehlerrate minimiert:
 - $\theta^* = \arg \min_{\theta} E(\theta|\mathcal{D})$

Die Studierenden sollen in der Lage sein

- Überwachtes Lernen zu beschreiben und zu diskutieren
 - Erlernen einer Klasse anhand von Beispielen
 - Erlernen multipler Klassen
 - Modellauswahl und Generalisierung
- (*) ML-Algorithmen für überwachtes Lernen in der Praxis zu verwenden

Literatur und Vorlesungsfolien

Maschinelles Lernen:

- ① Ethem Alpaydin: „Maschinelles Lernen“. De Gruyter Studium. 2. Auflage, (2019). (Kapitel 2)
- ② Aurélien Géron: „Praxiseinstieg Machine Learning mit Scikit-Learn, Keras und TensorFlow: Konzepte, Tools und Techniken für intelligente Systeme“. O'Reilly, 2. Auflage, (2020).

ML-Bücher sind auch elektronisch in der Bibliothek vorhanden!

<https://bonnus.ulb.uni-bonn.de/>

- Vorlesungsfolien: s. eCampus
 - Die Vorlesungsmaterialien, einschließlich aller Vorlesungsfolien, Übungsmaterialien und Prüfungsfragen, werden ausschließlich für die Teilnehmer*innen des Moduls "BA-INF 035 Datenzentrierte Informatik" an der Universität Bonn im WS 2025/2026 bereitgestellt. Die Weitergabe an Dritte, die Veröffentlichung und die Verbreitung der Vorlesungsmaterialien sind untersagt.