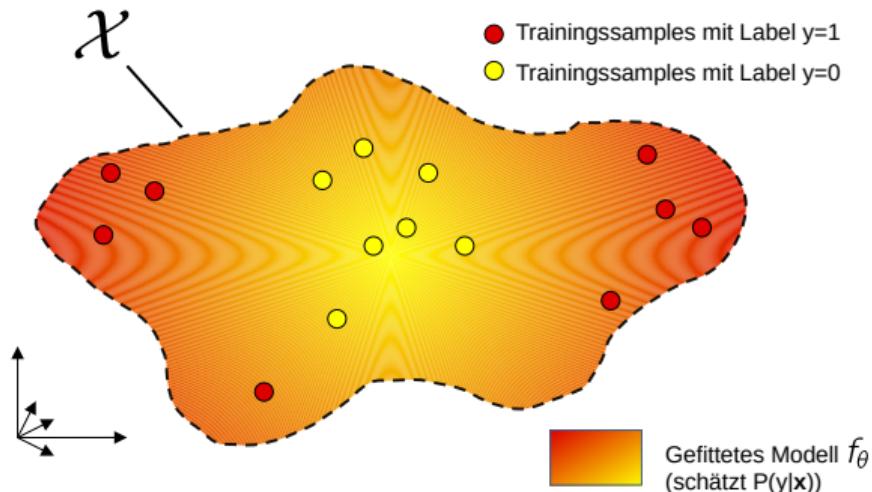


Künstliche Intelligenz (Sommersemester 2024)

Kapitel 03: Probabilistische Modelle

Prof. Dr. Adrian Ulges

ML: Der Merkmalsraum (“Feature Space”)



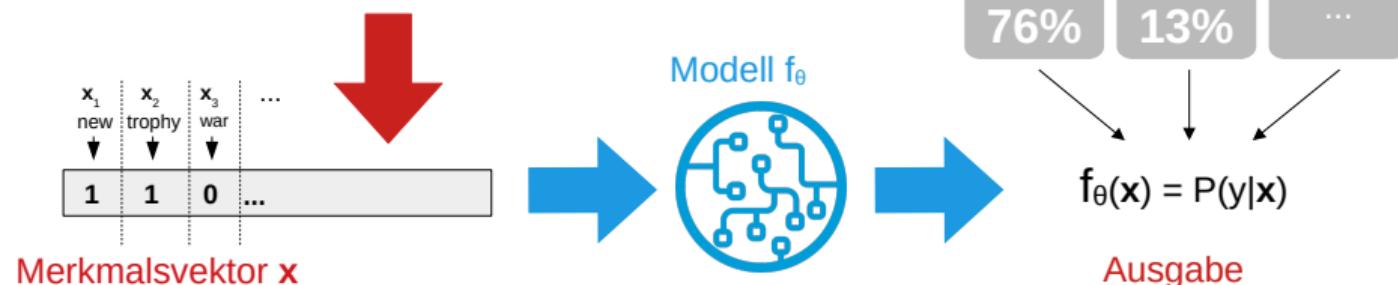
- Wir können uns ein Modell $f_\theta : \mathcal{X} \rightarrow \mathbb{R}$ als einen **Farbverlauf** vorstellen, der jedem Punkt x einen Wert $f_\theta(x)$ zuweist.
- Das Modell f_θ schätzt die Klassenzugehörigkeit eines Objekts x , d.h. $f_\theta(x) \approx P(Y=1 | X=x)$.
- Ziel: Lerne mittels der Trainingsmenge Parameter θ , so dass das Modell gut “passt”.

Beispiel in diesem Kapitel: Dokumentklassifikation

- ▶ **Anwendungen:** Spam Filtering, News-Kategorisierung, Suchmaschinen, Software-Issue-Diagnose, Sentiment-Analyse, ...
- ▶ **Eingabe:** Dokument. **Ausgabe:** Klassenwahrscheinlichkeiten.

Eingabedokument

"Oregon quarterback Marcus Mariota won the Heisman Trophy on Saturday in New York, becoming the first Hawaii native and the first Ducks player to ..."

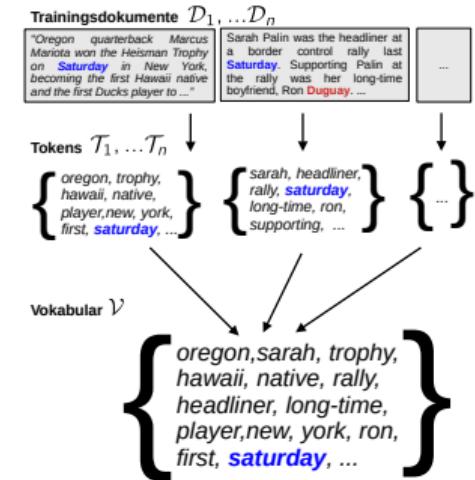


Schritt 1: Merkmalsextraktion

- Wir müssen zunächst eine Transformation definieren, die jedes Dokument \mathcal{D} in einen Merkmalvektor $\mathbf{x} \in \mathbb{R}^d$ verwandelt.
- Wir verwenden hier sogenannte 'Bag-of-Words'-Merkmale.
- Grundlage: Eine Menge von Trainingsdokumenten $\mathcal{D}_1, \dots, \mathcal{D}_n$.

Schritt 1: Vorverarbeitung

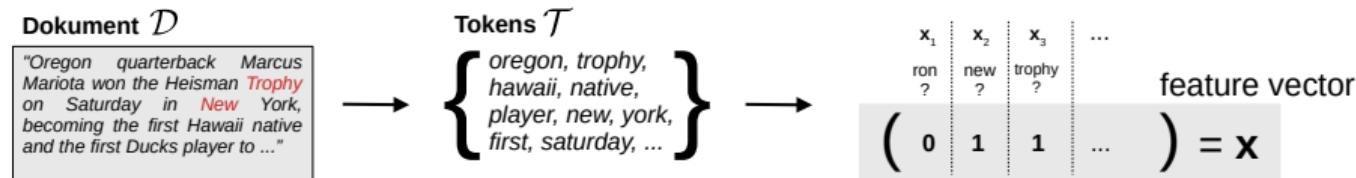
- Wir zerlegen alle Trainingsdokumente in Einzelwörter (*Tokenization*).
- Wir **normalisieren** Wörter per Lowercasing (Sagtest → sagtest).
- Wir **filtern** uninteressante Wörter (z.B. das, in, nicht ...).
- Wir **filtern** zu seltene Wörter (arachnophobia). Später mehr dazu...
- Aus jedem Dokument \mathcal{D} wird hierdurch eine Menge von Tokens \mathcal{T} .



Schritt 2: Vokabular

- Wir vereinigen sämtliche Tokenmengen $\mathcal{T}_1, \dots, \mathcal{T}_n$ zu einem sogenannten **Vokabular** \mathcal{V} .
- \mathcal{V} enthält also alle uns bekannten und relevanten Wörter (in normalisierter Form).

Schritt 1: Merkmalsextraktion (cont'd)



Merkalsvektor \mathbf{x} berechnen

- ▶ Gegeben: Ein Vokabular mit m Wörtern, und ein beliebiges **Eingabedokument** \mathcal{D} .
- ▶ Wir bringen die Wörter des Vokabulars in eine feste Reihenfolge t_1, \dots, t_m .
- ▶ \mathcal{D} wird in einen **booleschen Vektor** $\mathbf{x} = (x_1, \dots, x_m) \in \{0, 1\}^m$ transformiert.
- ▶ Jeder Eintrag x_i ist genau dann 1, wenn der Begriff t_i im Dokument erscheint (*und* $x_i = 0$ sonst).
- ▶ Beachten Sie, dass die *Reihenfolge der Begriffe* verloren geht!

Anmerkungen

- ▶ Wir wenden die obige Transformation auf alle Trainingsdokumente an, und erhalten Trainingsvektoren $\mathbf{x}_1, \dots, \mathbf{x}_n$.

Wahrscheinlichkeitstheorie meets Machine Learning

- ▶ Machine Learning - Modelle müssen häufig **Entscheidungen unter Unsicherheit** treffen. Es ist sehr nützlich, die Konfidenz des Modells in Form von W'keiten auszudrücken!
- ▶ Wir modellieren die **Merkmale** als Zufallsvariablen X_1, \dots, X_d und erhalten einen Zufallsvektor $\mathbf{X} = (X_1, \dots, X_d)$.
- ▶ Wir modellieren auch das **Target** als Zufallsvariable Y .
- ▶ Um optimale Entscheidungen zu treffen, müssen wir $P(Y=y | \mathbf{X}=\mathbf{x})$ abschätzen
(Gegeben ein Objekt mit Merkmalsvektor \mathbf{x} , was ist die W'keit dass das Objekt zur Klasse y gehört?)
- ▶ Wir kürzen ab: $\cancel{P(Y=y | \mathbf{X}=\mathbf{x})} = P(y | \mathbf{x})$.

Wahrscheinlichkeitstheorie meets Machine Learning

- Beobachtung: Wir können diese Wahrscheinlichkeit mit Bayes' Regel "umkehren".

$$P(y|x) = \frac{P(y) \cdot P(x|y)}{P(x)}$$

? *ist dieselbe für Klassen alle*

$$\propto P(y) \cdot P(x|y)$$

Anmerkungen

- Dies wurde als "wichtigste Gleichung im ML" bezeichnet (S. Marsland).
- Die Komponenten der Gleichung haben Namen:
 - $P(y)$: Der sog. **prior** (*die allgemeine Verteilung der Klassen, ohne Kenntnis von x*)
 "23 Prozent aller Dokumente gehören zur Kategorie Politik."
 - $P(y|x)$: Der **posterior** (*die Verteilung mit Kenntnis von x – gesucht!*)
 "Das gegebene Dokument gehört mit 78 Prozent W'keit zur Kategorie Politik."
 - $P(x|y)$: Die **class-conditional density (CCD)** (*die Verteilung der Daten innerhalb einer Klasse*)
 "Politik-Dokumente enthalten mit 10% W'keit folgende Worte: ..."

Diskriminative vs. Generative Modelle

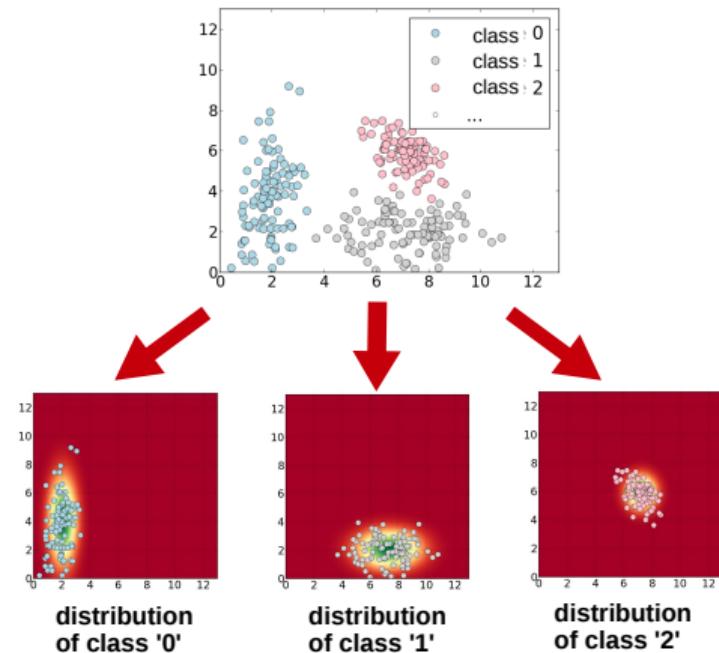
$$P(y|x) = \dots \propto P(y) \cdot P(x|y)$$

Es gibt zwei allgemeine Arten von ML-Modellen:

1. **Diskriminative** Modelle: verwenden ein direktes Modell für $P(y|x)$ (*oder die Entscheidungsgrenze, entsprechend*).
2. **Generative** Modelle: berechnen $P(y)$ und $P(x|y)$ und setzen sie in das Bayes-Theorem ein.

Generative Methoden: Lernen

- ▶ "Lernen" bei generativen Methoden bedeutet, $P(y)$ und $P(x|y)$ zu schätzen.
- ▶ Die Prior $P(y)$ ist normalerweise einfach (*wie viele Samples werden pro Klasse erwartet?*).
- ▶ Für $P(x|y)$ wählen wir oft eine **Form/Verteilung** aus (*z.B. Gaussverteilungen*) und schätzen Parameter (*z.B. mit ML-Verfahren*).



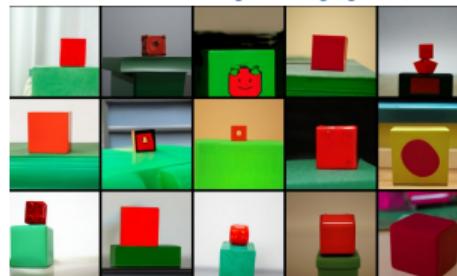
Generative Methoden: Bemerkungen

- ▶ Generative Modelle können **Stichproben** des Zufallsprozesses erzeugen:
 1. Ziehe eine Klasse $y \sim P(y)$
 2. Ziehe $x \sim P(x|y)$.
- ▶ Viele kürzliche **Deep Learning - Modelle** sind generativ, z.B.
 - ▶ GANs (generative adversarial networks)
 - ▶ VAEs (variational auto-encoders)
 - ▶ auto-regressive Sprachmodelle
 - ▶ Diffusion-Modelle
- ▶ Diese Modelle funktionieren erschreckend gut...

Beispiel: Dall·E generiert Bilder aus textuellen Captions (OpenAI, 2021) [1]

„Generate images of ...

... a small red block sitting on a large green block."



... a storefront saying ‚openai‘."



... an armchair imitating an avocado."



Zurück zu Textklassifikation

Beispiel: Spam Classifier

- Das Vokabular enthält zwei Worte: $\mathcal{V} = \{\text{mom}, \text{viagra}\}$.
- Es gibt zwei Klassen, **Spam** und **Ham**.
- Wir nehmen an, dass $P(Y=\text{spam}) = 10\%$.
- $P(x|y)$ ist eine **Wahrscheinlichkeitstabelle** (rechts).



Klasse $y = \text{SPAM}$

$$P(y|x) \propto P(y) \cdot P(x|y)$$

$$0,1 \cdot 0,2 = 0,02 \quad \star 5\%$$

Klasse $y = \text{HAM}$

$$P(y|x) \propto 0,9 \cdot 0,02 = 0,018$$

$$P(y|x) = \frac{0,02}{0,02 + 0,18} \quad \text{für SPAM}$$

Klasse 1 (Spam)			Klasse 2 (Ham)		
$x_1(\text{"mom"})$	$x_2(\text{"viagra"})$	$P(x \text{spam})$	$x_1(\text{"mom"})$	$x_2(\text{"viagra"})$	$P(x \text{ham})$
0	0	0.77	0	0	0.18
0	1	0.20	0	1	0.02
1	0	0.01	1	0	0.78
1	1	0.02	1	1	0.02

„20% aller Spam-Mails enthalten nicht das Wort „mom“, aber enthalten das Wort „viagra“.

$$\frac{0,02}{0,02 + 0,18} = 5\%$$

Textklassifikation (cont'd)

Anwendung des Modells

- Eine neue E-Mail enthalte 'viagra', aber nicht 'mom' ($\mathbf{x} = (0, 1)$).

		Klasse 1 (Spam)		Klasse 2 (Ham)
		x_1 ("mom")	x_2 ("viagra")	$P(\mathbf{x} \text{spam})$
		0	0	0.77
0	1	0.20		0 0 0.18
1	0	0.01		1 0 0.78
1	1	0.02		1 1 0.02

„20% aller Spam-Mails enthalten nicht das Wort „mom“, aber enthalten das Wort „viagra“.

Naive Bayes

Problem mit diesem Ansatz?

- In der Praxis wäre x ein boolescher Vektor mit > 10.000 Einträgen (*welche Begriffe erscheinen in der E-Mail, welche nicht?*).
- Wenn das Vokabular \mathcal{V} so groß ist, ist die Wahrscheinlichkeitstabelle **enorm** (2^m Einträge)!
- Die Wahrscheinlichkeitstabellen hätten $2^{10.000}$ Einträge!

m=1

„mom“			$P(x \text{spam})$
0	0	0.78	
1	0	0.22	

m=2

„mom“			$P(x \text{spam})$
„viagra“			
0	0	0.58	
0	1	0.22	
1	0	0.16	
1	1	0.04	

m=3

„mom“			$P(x \text{spam})$
„viagra“			
„buy“			
0	0	0	0.35
0	0	1	0.13
0	1	0	0.17
0	1	1	0.03
1	0	0	0.19
1	0	1	0.03
1	1	0	0.09
1	1	1	0.01

...

m=10,000 ?

Jeder dieser Einträge müsste aus einer begrenzten Trainingsmenge gelernt werden!



Naive Bayes

$$P(\text{Wort1}|\text{Wort2}) = P(\text{Wort1}) \quad P(A|B) = P(A)$$

*

- Unser Ziel ist es, $P(x|y)$ zu vereinfachen!
- Ansatz: Wir nehmen an, dass die einzelnen Wort-Einträge in x unabhängig sind. (daher Naive Bayes). alle Wörter

$$P(x|y) = P(x_1|y) \cdot P(x_2|y) \cdot \dots \cdot P(x_m|y).$$

ron new trophy

- Die Entscheidungsregel wird:

$$y^* = \arg \max_y P(y|x)$$

$$= \arg \max_y \frac{P(y) \cdot P(x|y)}{P(x)} \quad // \text{Bayes'sche Regel}$$

$$\propto \arg \max_y P(y) \cdot P(x|y) \quad // P(x) beeinflusst y^* nicht$$

$$= \arg \max_y P(y) \cdot \prod_{i=1}^m P(x_i|y). \quad // \text{Unabhängigkeit der Merkmale}$$

In 30% aller
Sport-Dokumente
kommt
"trophy"
vor

Dokumentenklassifikator (mit Naive Bayes)

- ▶ Gleiche Einstellungen wie oben, aber jetzt **Naive Bayes**.
- ▶ Es gibt **eine kleine** Wahrscheinlichkeitstabelle pro Merkmal: $P(x_1|y)$, $P(x_2|y)$.

		Class 1 (Spam)	Class 2 (Ham)
		$x_1(\text{„mom“})$	$x_1(\text{„mom“})$
		$P(x_1 \text{spam})$	$P(x_1 \text{ham})$
0	0.95	0	0.70
	0.05	1	0.30
		$x_2(\text{„viagra“})$	$x_2(\text{„viagra“})$
		$P(x_2 \text{spam})$	$P(x_2 \text{ham})$
0	0.66	0	0.99
	0.34	1	0.01

1 0.34

„34% aller Spam-Mails enthalten das Wort „viagra“ (unabhängig vom Wort „mom“).“

Naive Bayes

Anmerkungen

- Die zu erlernenden Einträge **verringern** sich von 2^m auf ... $2m$ (bzw. eigentlich m).
- Wir können diese Einträge zuverlässig aus **begrenzten Trainingsdaten** schätzen. ☺
- Beachten Sie, dass die Unabhängigkeitsannahme in Texten normalerweise **stark verletzt** wird, da Begriffe sich gegenseitig **beeinflussen!** ☹
(z.B. $P(\text{superbowl} = 1) \ll P(\text{superbowl} = 1 \mid \text{chiefs} = 1)$).

Naive Bayes: Quiz

Laut Einführung sind ML-Modelle Funktionen $f_\theta : \mathcal{X} \rightarrow \mathcal{Y}$.

Wie passt Naive Bayes auf diese Definition?

- ▶ Was ist der Merkmalsraum \mathcal{X} ?
- ▶ Was ist \mathcal{Y} ?
- ▶ Welche Form hat f_θ ?
- ▶ Was sind die Parameter θ ?





References I

- [1] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever.
Zero-shot text-to-image generation.
[CoRR, abs/2102.12092, 2021.](#)