



Grundlagen des Maschinellen Lernens

Dimensionsreduktion

Nils Schwenzfeier

02.06.2021
Prof. Dr. Volker Gruhn

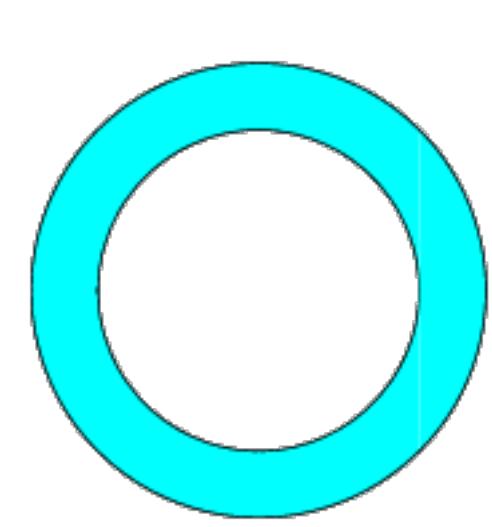
1. Motivation
2. Principal Component Analysis (PCA)
3. t -Distributed Stochastic Neighbourhood Embedding (t-SNE)

Motivation

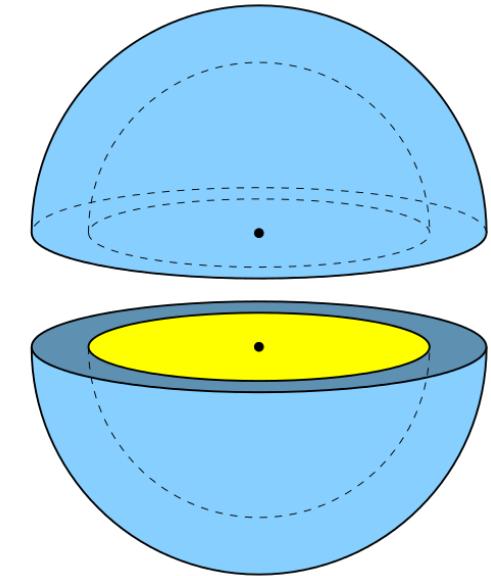
Das Problem mit den Dimensionen

► Motivation

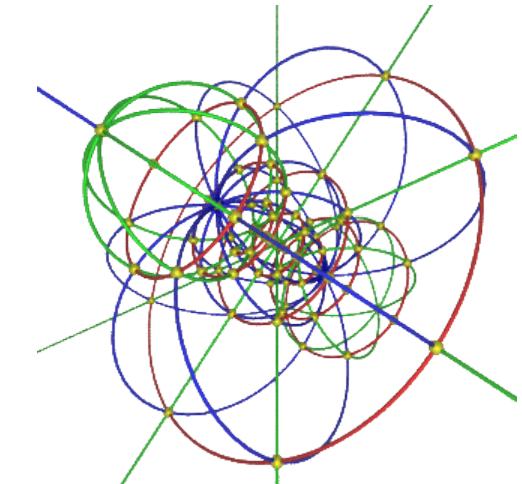
Wie viel vom Volumen einer Kugel macht die Schale aus, wenn ihre Dicke 0,1% vom Radius entspricht?



<https://mathworld.wolfram.com/Annulus.html>



<https://de.wikipedia.org/wiki/Datei:Kugelschale.svg>



https://commons.wikimedia.org/wiki/File:Hypersphere_coord.PNG

Dimension	$n = 2$	$n = 3$	$n = 4$	$n = 10$	$n = 100$	$n = 1000$	$n = 10000$
Anteil Volumen	0.2 %	0.3 %	0.4 %	1 %	9.5 %	63.2 %	> 99,99%

Eine 10000-dimensionale Pizza, die 2 Meter im Durchmesser misst und eine 1mm dicke Kruste besitzt, besteht zu über 99,99% aus Kruste!

Beobachtung:

In hochdimensionalen Räumen sind die meisten Punkte **Randpunkte** (haben extreme Features)

Das Problem mit den Dimensionen

► Motivation

Wie weit sind 1.000 zufällig in einer Kugel (Radius = 1) gewählte Punkte im Mittel voneinander entfernt?

Dimension	n = 2	n = 3	n = 4	n = 10	n = 100	n = 1000	n = 10000
Mittlerer Abstand	1	1.1	1.2	1.3	1.4	1.41	1.414
Standardabweichung	0.46	0.4	0.36	0.23	0.07	0.02	0.007

- Der mittlere Abstand wächst langsam und geht gegen $\sqrt{2}$
- Die Standardabweichung fällt und geht gegen 0

Beobachtung:

In hochdimensionalen Räumen sind verschiedene Punkte im Mittel fast **gleich weit** voneinander entfernt (nah und fern sind praktisch gleich)

- Je höher die Dimension des Feature-Raums, desto
 - mehr Punkte liegen am Rand des Raums,
 - größer ist der Abstand zwischen den Punkten
 - weniger Unterschied gibt es zwischen den Abständen der Punkte
- Hochdimensionale Räume haben sehr viel Volumen
- Bei gleichbleibender Menge an Daten liegen diese immer spärlicher im Raum

Dieses Problem nennt sich **Curse of Dimensionality**

Wenn Daten nur noch spärlich im Feature-Raum liegen, dann können Modelle nur schwierig an die Daten angepasst werden und gleichzeitig adäquat den Raum beschreiben.

Typische Phänomene:

- Overfitting wird zum Problem
- Höhere Rechenzeit/Ressourcenverbrauch
- Schlechtere Generalisierung

Dimensionsreduktion kann gute Idee sein!

Chancen

- Effizientere Resourcennutzung (z.B. Zeit, Speicher)
- Bessere Verallgemeinerung (overfitting)
- Entfernen von Rauschen
- Ermöglichung der Visualisierung im Zwei- oder Dreidimensionalen

Gefahren

- Verlust von Informationen in den (Trainings-)Daten
- Schlechtere Accuracy des Modells

Feature Selection

- Dimension wird reduziert, indem einzelne Features entfernt werden
- Welche Features für das Modell behalten werden sollen, kann z.B. durch *cross validation* ermittelt werden
- Lasso-Regularisierung ist auch eine Form von Feature Selection, da einzelne Koeffizienten von Features auf Null gesetzt werden und somit entfernt werden

Feature Extraction

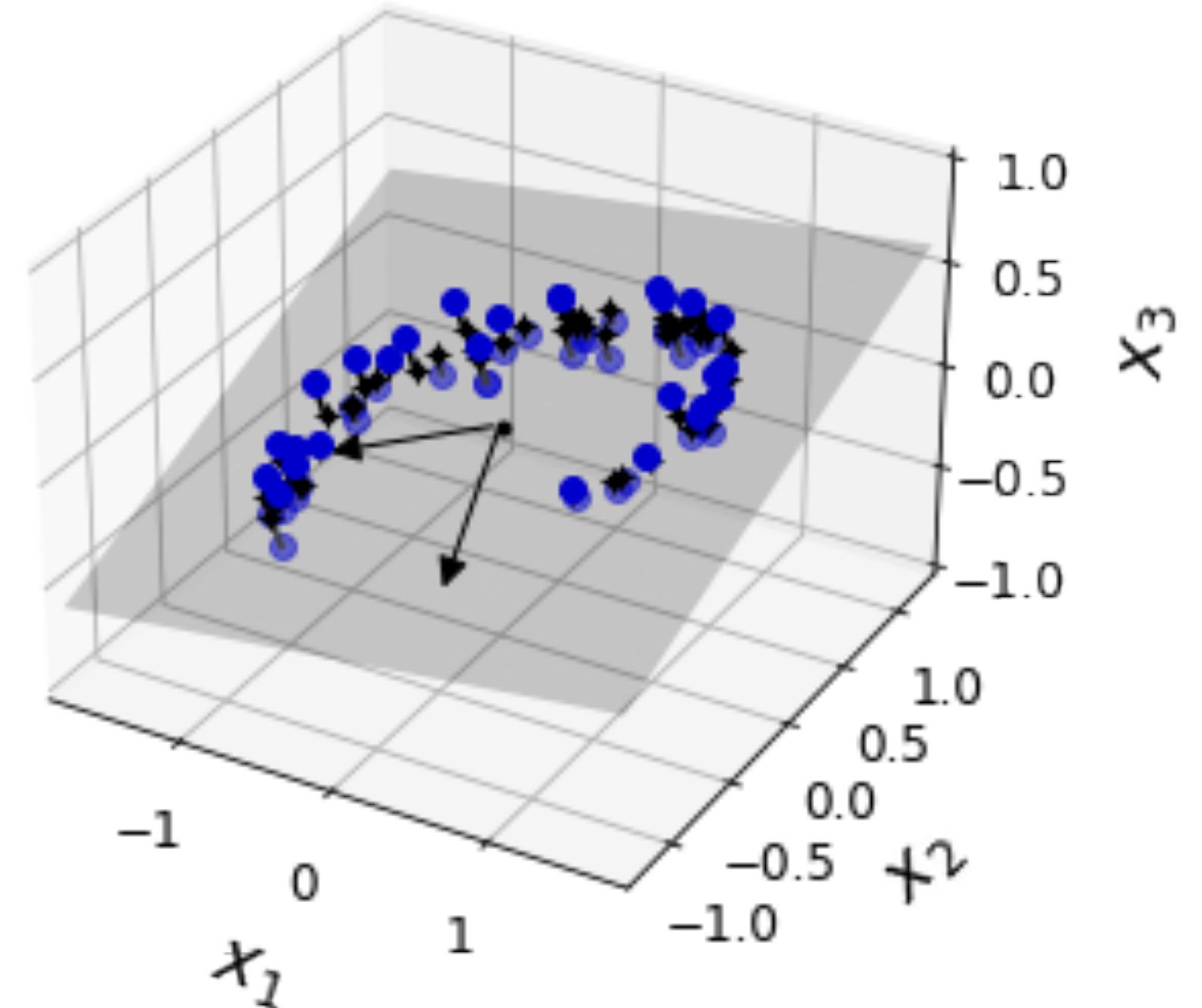
- Dimension wird reduziert, indem aus den ursprünglichen Features neue konstruiert werden
- Verglichen mit Feature Selection deutlich mehr Möglichkeiten der Dimensionsreduktion
- Ziel ist, nach der Feature Extraction möglichst wenige (relevante) Informationen zu verlieren

Annahme:

Daten sind nicht komplett gleichmäßig im Feature-Raum verteilt

Mögliche Folgen:

- Manche Features sind kaum von Bedeutung (weil fast konstant)
- Manche Features sind miteinander korreliert
- Datenpunkte liegen ungefähr in einem Unterraum



Quelle: https://github.com/ageron/handson-ml2/blob/master/08_dimensionality_reduction.ipynb

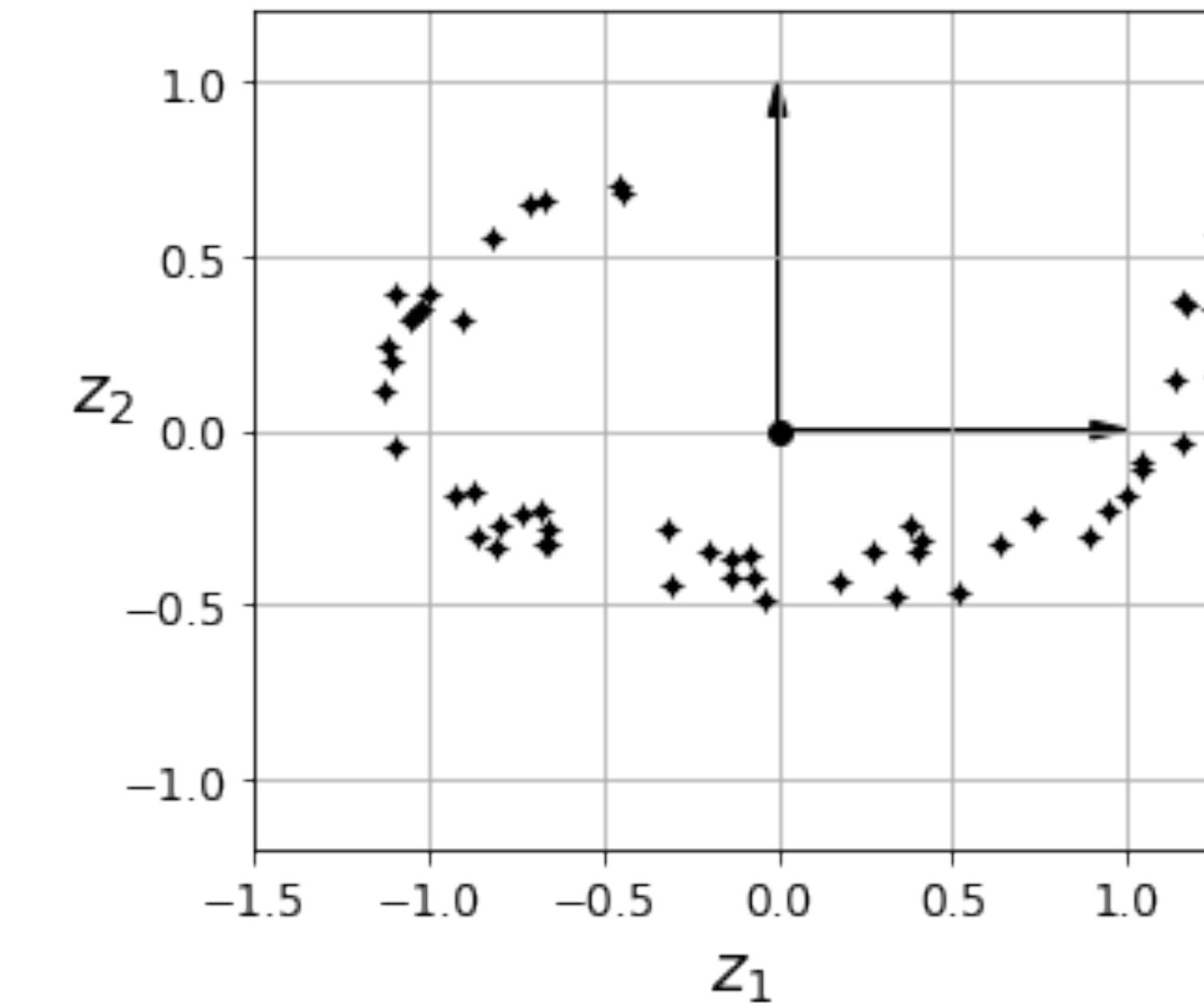
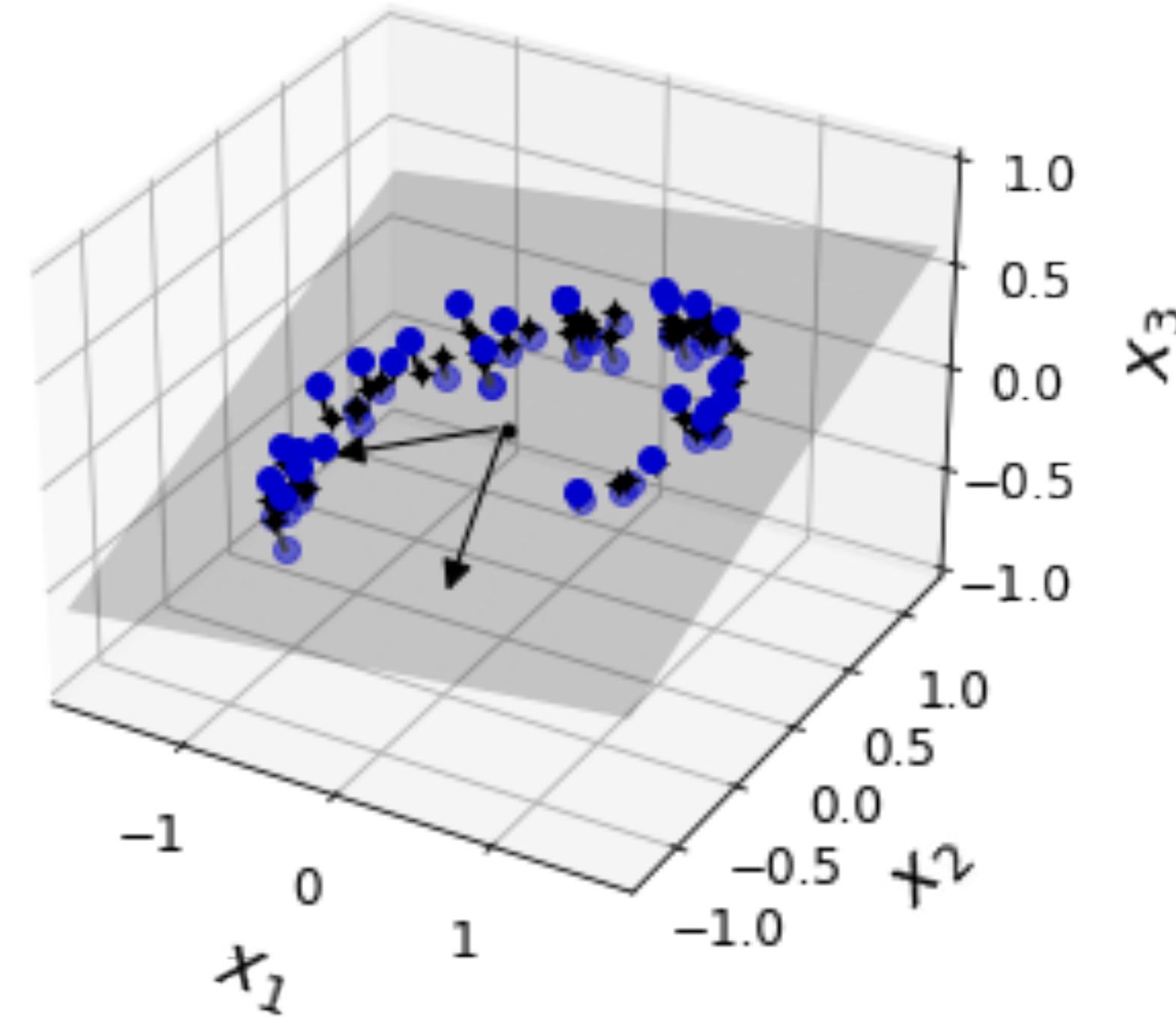
Daten liegen ungefähr in zweidimensionalem Unterraum
(graue Fläche)

Dimensionsreduktion durch Projektion der Datenpunkte auf den Unterraum

Dreidimensionaler Feature-Raum



Zweidimensionaler Feature-Raum



Quelle Bilder: https://github.com/ageron/handson-ml2/blob/master/08_dimensionality_reduction.ipynb

Principal Component Analysis (PCA)

1. Grundlegende Idee
2. Formale Herleitung
3. Praktische Überlegungen
4. Grenzen der PCA

Principal Component Analysis (dt. Hauptkomponentenanalyse)

- Wir wollen neue Koordinatenachsen (Komponenten) für die Daten finden
(Feature Extraction durch Koordinatentransformation)
- Entlang der Koordinatenachsen soll Varianz der Daten maximiert werden
- Hypothese: „Relevante“ Informationen werden durch hohe Varianz abgedeckt
- Koordinatenachsen sollen orthogonal zueinander stehen (orthogonale Transformation)
- Achsen sind dadurch unkorreliert – es kommt zu keiner Überlappung von Informationen zwischen den Achsen
- Achsen mit wenig Varianz werden verworfen, um Dimension zu reduzieren

Ausgangslage

- Es liegen n Datenpunkte mit D Features vor:

$$(x_n)_{n=1,\dots,N} \quad \text{mit } x_n \in \mathbb{R}^D$$

- Mittlerer Vektor der Datenpunkte ist

$$\bar{x} = \frac{1}{N} \sum_{n=1}^N x_n$$

Ziel

- Datenpunkte sollen auf eine neue Achse $u_1 \in \mathbb{R}^D$ projiziert werden, sodass die Varianz der Projektionen maximiert wird
- Da nur die Richtung von u_1 interessant ist, schränken wir die Suche auf Einheitsvektoren ein:

$$\|u_1\|^2 = u_1^\top u_1 = 1$$

Formale Herleitung

► PCA

- Projektion von x_n auf u_1 ist

$$\tilde{x}_n = \langle x_n, u_1 \rangle u_1 = (u_1^\top x_n) u_1$$

- Die neue Koordinate ist Koeffizient vor u_1 , d.h.

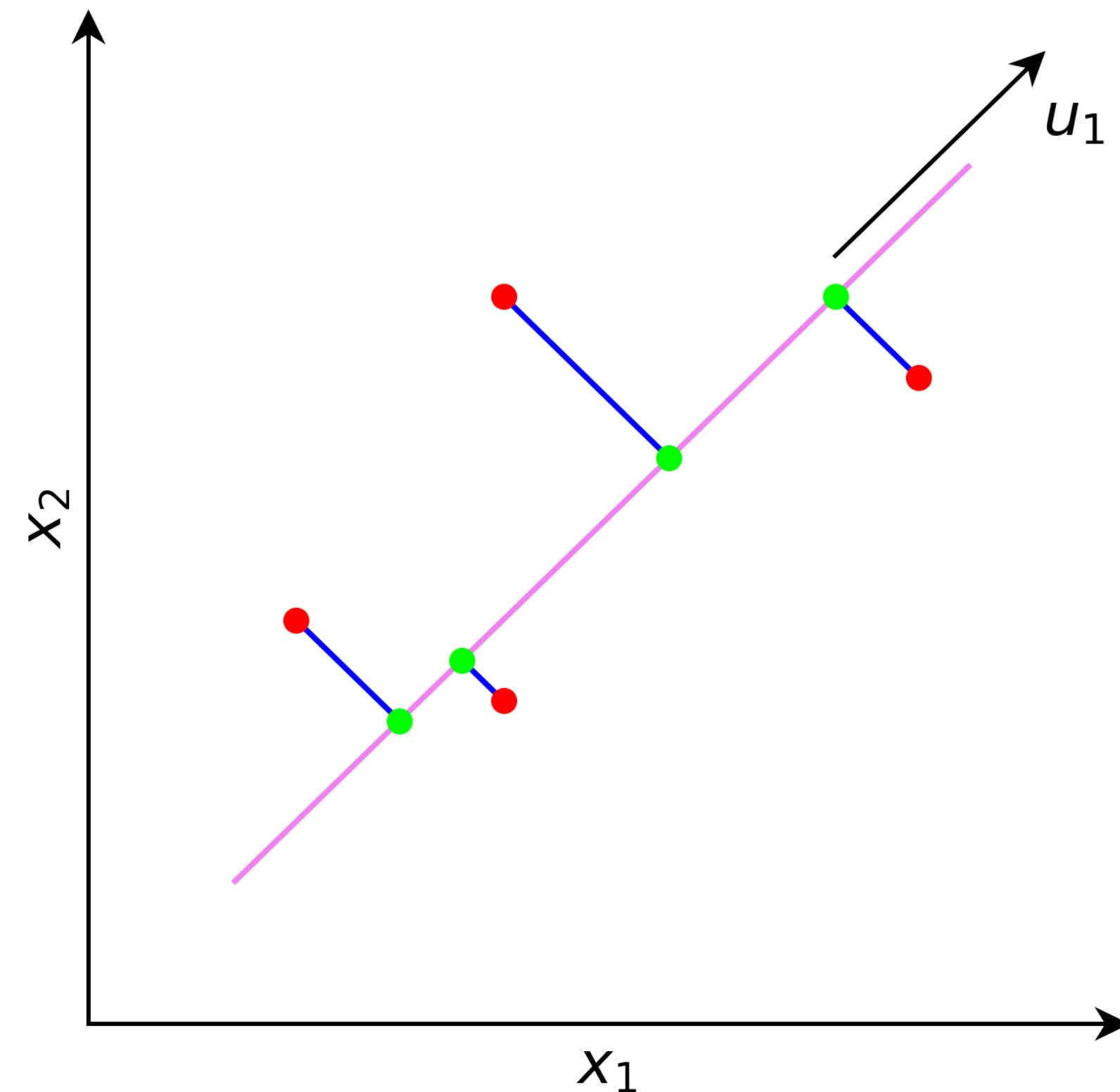
$$u_1^\top x_n$$

- Varianz der Projektionen ist:

$$\begin{aligned} \text{Var}(u_1^\top x_1, \dots, u_1^\top x_N) &= \frac{1}{N} \sum_{n=1}^N (u_1^\top x_n - u_1^\top \bar{x})^2 \\ &= \dots \\ &= u_1^\top S u_1 \end{aligned}$$

- S ist die Kovarianzmatrix zu $(x_1^\top, \dots, x_N^\top)$:

$$S = \frac{1}{N} \sum_{n=1}^N (x_n - \bar{x})(x_n - \bar{x})^\top$$



- $\text{Var}(u_1^\top x_1, \dots, u_1^\top x_N) = u_1^\top S u_1$ soll maximiert werden
- Es soll $u_1^\top u_1 = 1$ gelten (Einheitsvektor)
- Zusammengefasst liegt folgendes Optimierungsproblem vor:

$$\max_{u_1} u_1^\top S u_1 \quad \text{unter der Nebenbedingung } u_1^\top u_1 = 1$$

Lagrange-Multiplikatoren

- Verfahren, um Optimierungsprobleme mit Nebenbedingung in „gewöhnliche“ Optimierungsprobleme zu überführen

$$\max_x f(x) \quad \text{unter der Nebenbedingung } g(x) = C$$

- Stelle Lagrange-Funktion L mit Lagrange-Multiplikator $\lambda \in \mathbb{R}$ auf:

$$L(x, \lambda) = f(x) + \lambda(g(x) - C)$$

- Lösung des Optimierungsproblem mit Nebenbedingung ist lokales Maximum der Lagrange-Funktion L
- Finde Maximum von L durch Ableiten und mit Null Gleichsetzen:

$$\nabla_x L(x, \lambda) = 0$$

- Lagrange-Funktion unseres Optimierungsproblems:

$$L(u_1, \lambda_1) = \underbrace{u_1^\top S u_1}_{f(u_1)} + \lambda_1 \underbrace{(1 - u_1^\top u_1)}_{C-g(u_1)}$$

- Ableiten und mit Null Gleichsetzen ergibt:

$$\nabla_{u_1} L(u_1, \lambda_1) \stackrel{!}{=} 0$$

$$S u_1 - \lambda_1 u_1 = 0$$

$$S u_1 = \lambda_1 u_1$$

- u_1 ist ein Eigenvektor von S zum Eigenwert λ_1

- Von links mit u_1^\top multiplizieren:

$$\begin{aligned} u_1^\top S u_1 &= \lambda_1 \underbrace{\overbrace{u_1^\top u_1}^{=1}} \\ u_1^\top S u_1 &= \lambda_1 \end{aligned}$$

Varianz der Projektionen
der Datenpunkte auf u_1

Erinnerung: Wir wollen die Varianz maximieren

Lösung: Wir wählen für u_1 den Eigenvektor von S zum größten Eigenwert λ_1

u_1 heißt erste **Hauptkomponente**

- Wir haben eine Richtung (u_1) gefunden, entlang derer die Varianz der Daten maximiert wird
- Wir suchen jetzt weitere Richtungen, die orthogonal zueinander und zu u_1 liegen und entlang derer die Varianz der Daten maximiert wird

Induktionsanfang: erste Hauptkomponente u_1

Induktionsannahme: u_1, \dots, u_m seien die Eigenvektoren von S zu den m größten Eigenwerten $\lambda_1, \dots, \lambda_m$

Induktionsschritt: Wir suchen u_{m+1} mit folgenden Eigenschaften:

- Varianz der Projektionen der Datenpunkte auf u_{m+1} ist maximal
- u_{m+1} ist orthogonal zu u_1, \dots, u_m
- u_{m+1} ist Einheitsvektor, d.h. $u_{m+1}^\top u_{m+1} = 1$

Optimierungsproblem:

$$\max_{u_{m+1}} u_{m+1}^\top S u_{m+1} \quad \text{unter den Nebenbedingungen}$$

- $\langle u_{m+1}, u_i \rangle = u_{m+1}^\top u_i = 0 \quad \text{für } i = 1, \dots, m \text{ und}$
- $u_{m+1}^\top u_{m+1} = 1$

Lagrange-Funktion:

$$L(u_{m+1}, \lambda_{m+1}, \eta_1, \dots, \eta_m) = u_{m+1}^\top S u_{m+1} + \lambda_{m+1}(1 - u_{m+1}^\top u_{m+1}) + \sum_{i=1}^m \eta_i u_{m+1}^\top u_i$$

mit Lagrange-Multiplikatoren $\lambda_{m+1}, \eta_1, \dots, \eta_m$

Lösung: u_{m+1} ist Eigenvektor von S zum größten Eigenwert λ_{m+1} unter den verbliebenen Eigenwerten

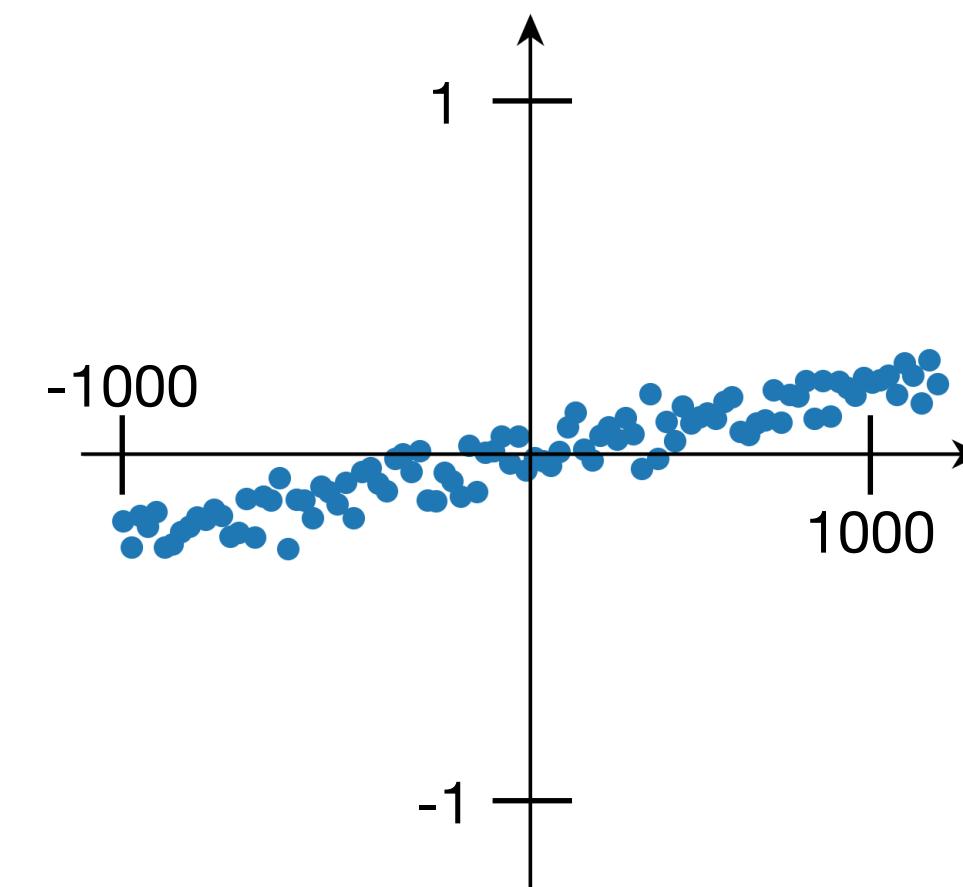
Beweis: Übung ;D

Durchführung der PCA

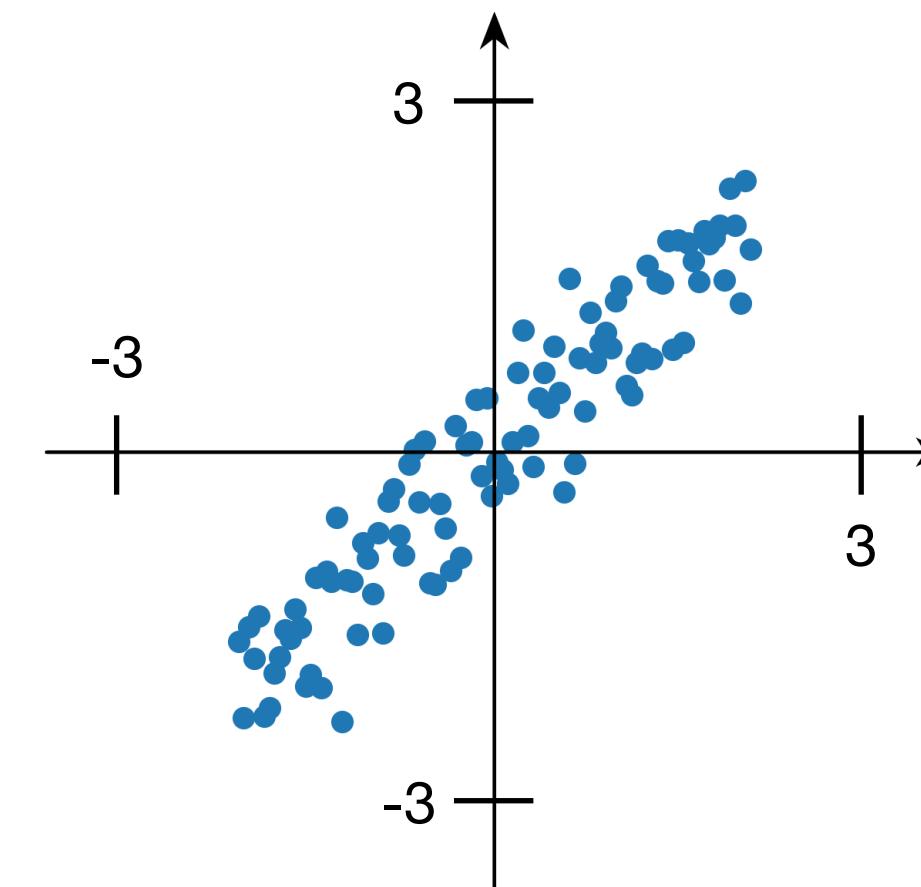
- Die Eigenwertzerlegung der Kovarianzmatrix S liefert Eigenwerte $\lambda_1, \dots, \lambda_D$ und dazugehörige Eigenvektoren u_1, \dots, u_D
- Werden die Eigenwerte (= Varianz entlang der Achsen) der Größe nach sortiert, dann ergeben die Eigenvektoren die Komponenten der PCA
- PCA-Transformation kann als Rotation des Koordinatensystems interpretiert werden: die alten Achsen werden auf die Komponenten der PCA gedreht

Praktische Überlegungen

▶ PCA



- Varianz entlang der x-Achse ist sehr hoch
- Varianz entlang der y-Achse ist niedrig
- Achsen der PCA unterscheiden sich kaum von bestehenden Achsen
- Die Daten stellen inhaltlich dasselbe dar, die PCA ergibt aber jeweils stark unterschiedliche Achsen
- Varianzen der Features sind für die PCA maßgeblich



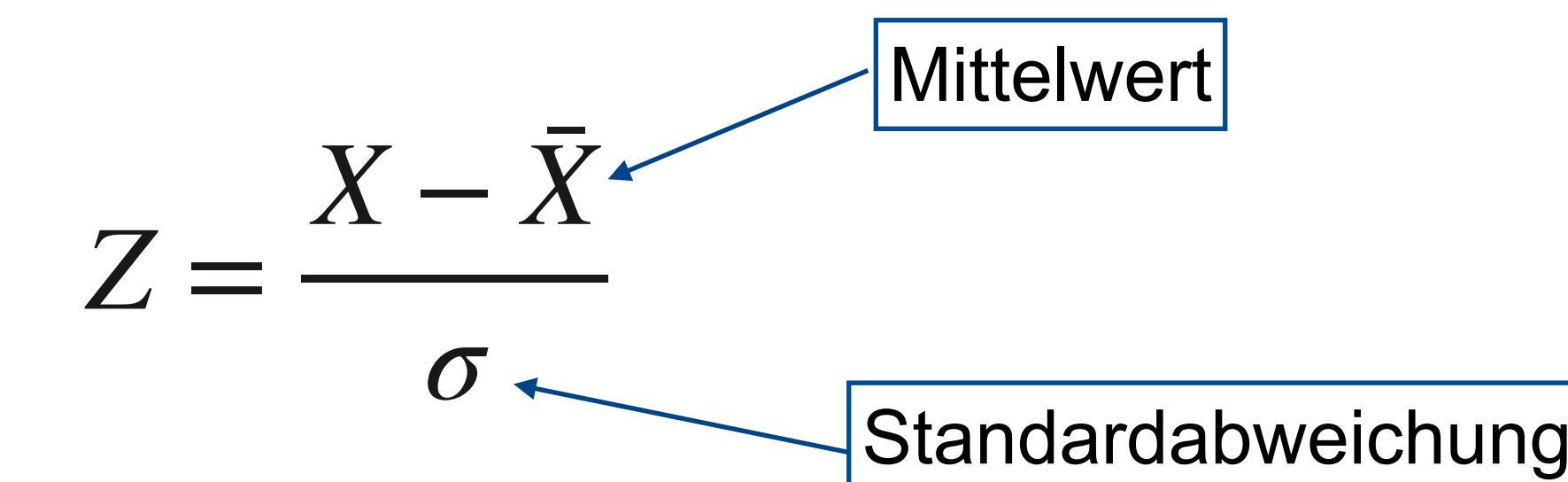
- Varianz entlang x- und y-Achse sind gleich
- Achsen der PCA sind um ca. 45° gedreht

- Willkürliche Wahl der Skalierung kann PCA verzerrnen
- Solange Features nicht mit derselben Einheit gemessen werden, sollten Skalierungen vereinheitlicht werden

Empfehlung: Features auf Varianz 1 skalieren

- Varianz der Daten verändert sich nicht, wenn sie verschoben werden
- Kovarianzmatrix ist translationsinvariant, also auch die PCA
- Sind die Daten nicht zentriert, haben sie nach einer PCA-Transformation einen Offset (unschön!)
- Deswegen werden Daten in der Regel zentriert

(zusammengefasste) Empfehlung: Features normalisieren

$$Z = \frac{X - \bar{X}}{\sigma}$$


Mittelwert
Standardabweichung

Dimensionsreduktion mit PCA

- **Annahme:** Komponenten mit der größten Varianz enthalten die meisten Informationen
- **Idee:** Dimension reduzieren, indem Komponenten mit wenig Varianz verworfen werden
- Gegeben:
 - Datenpunkte $x_n \in \mathbb{R}^D, n = 1, \dots, N$
 - PCA-Komponenten u_1, \dots, u_D
- Wenn k Komponenten verworfen werden, so sind die neuen Daten

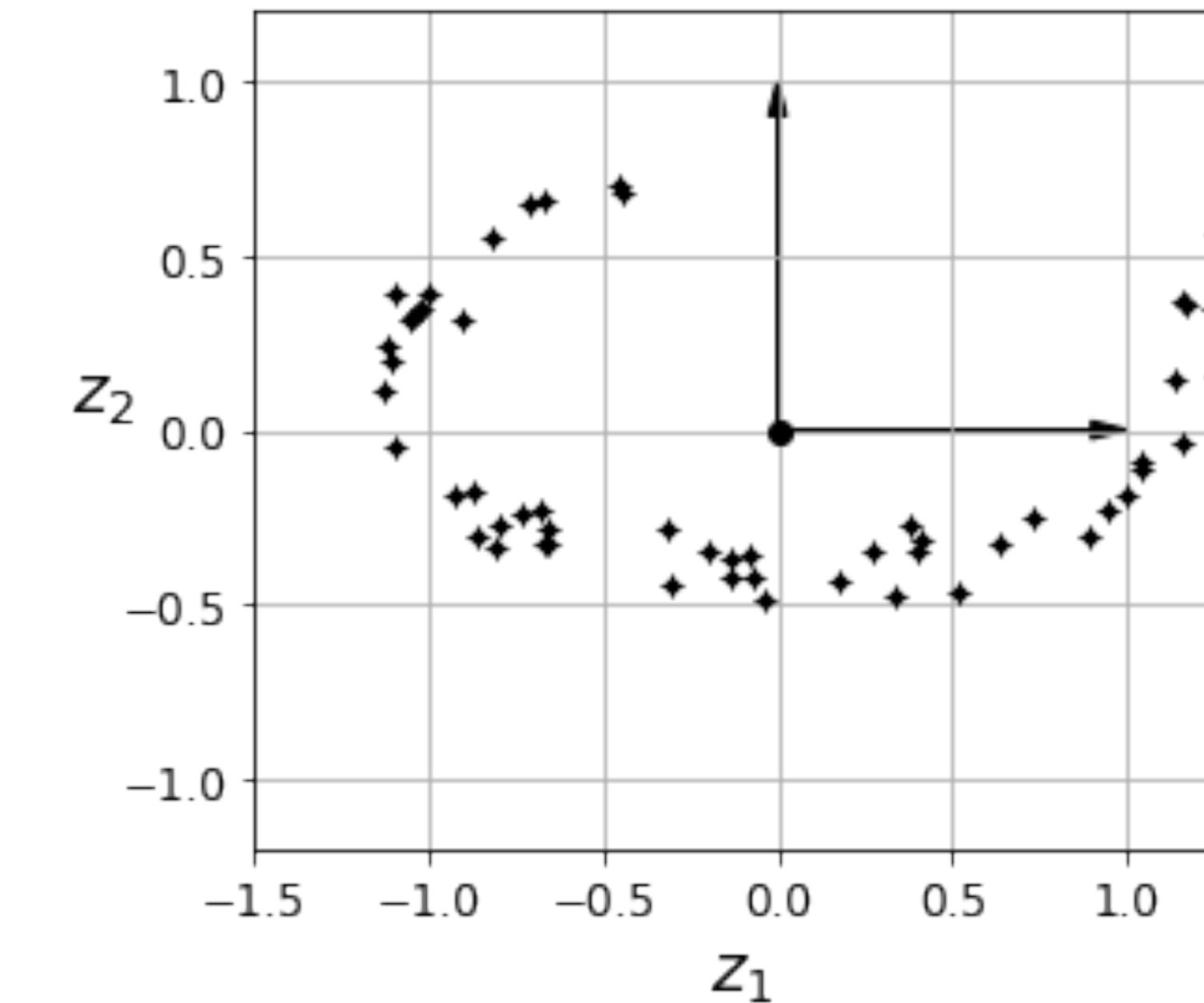
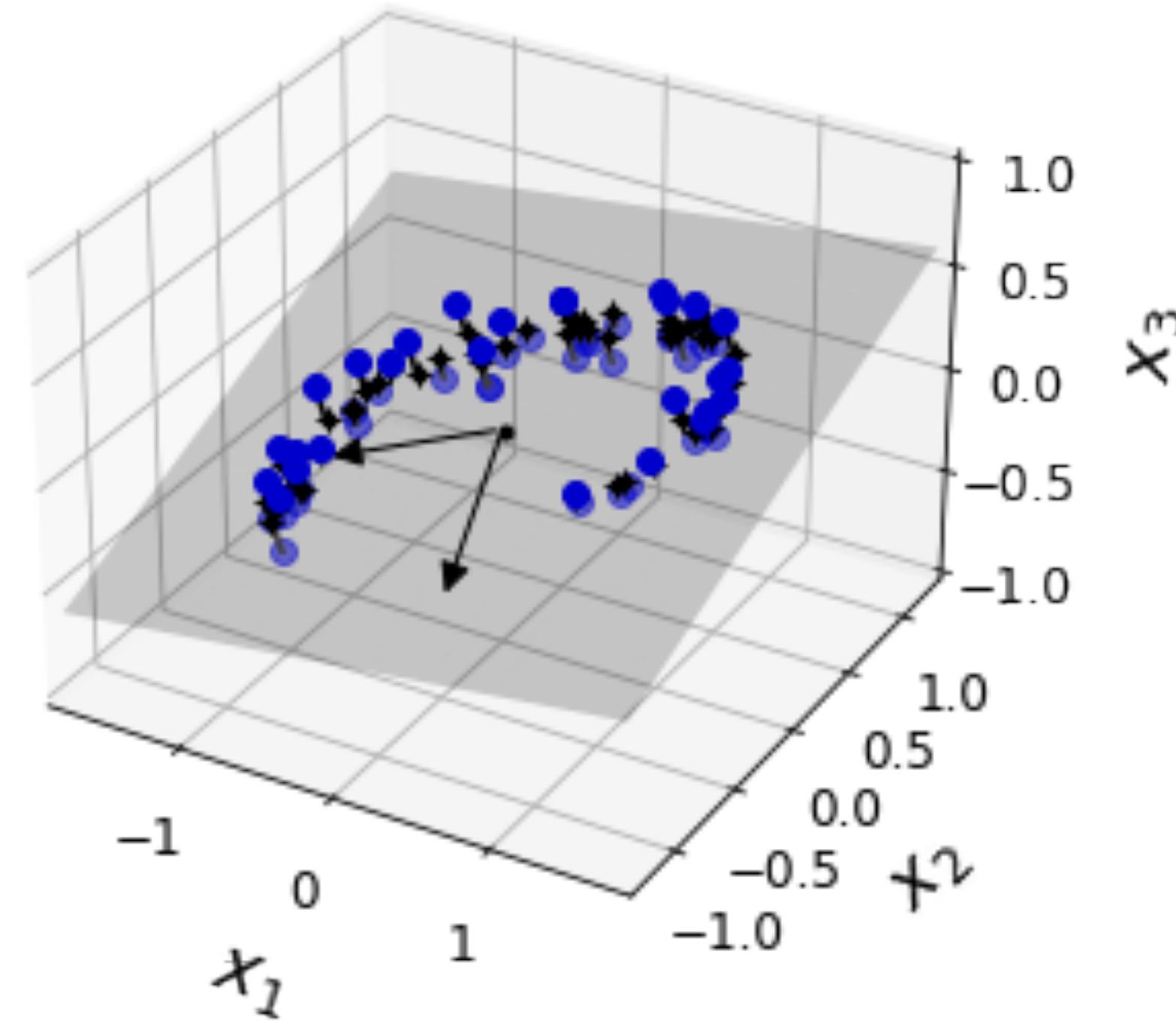
$$z_n = \begin{pmatrix} u_1^\top \\ \vdots \\ u_{D-k}^\top \end{pmatrix} x_n, \quad n = 1, \dots, N$$

Dimensionsreduktion durch Projektion der Datenpunkte auf den Unterraum

Dreidimensionaler Feature-Raum



Zweidimensionaler Feature-Raum



Quelle Bilder: https://github.com/ageron/handson-ml2/blob/master/08_dimensionality_reduction.ipynb

Frage: Wie wähle ich k , sprich welche Komponenten verwerfe ich?

- Im Supervised Learning: Validierung! k ist Hyperparameter
- Ansonsten: Ad-hoc-Entscheidung, Heuristiken, nach (Bauch-)Gefühl
- Möglicher Ansatz: *Proportion of Variance Explained*

Proportion of Variance Explained

- Varianz entlang Hauptkomponente u_i ist λ_i (Eigenwert der Kovarianzmatrix):

$$\text{Var}(u_i^\top x_n) = \lambda_i$$

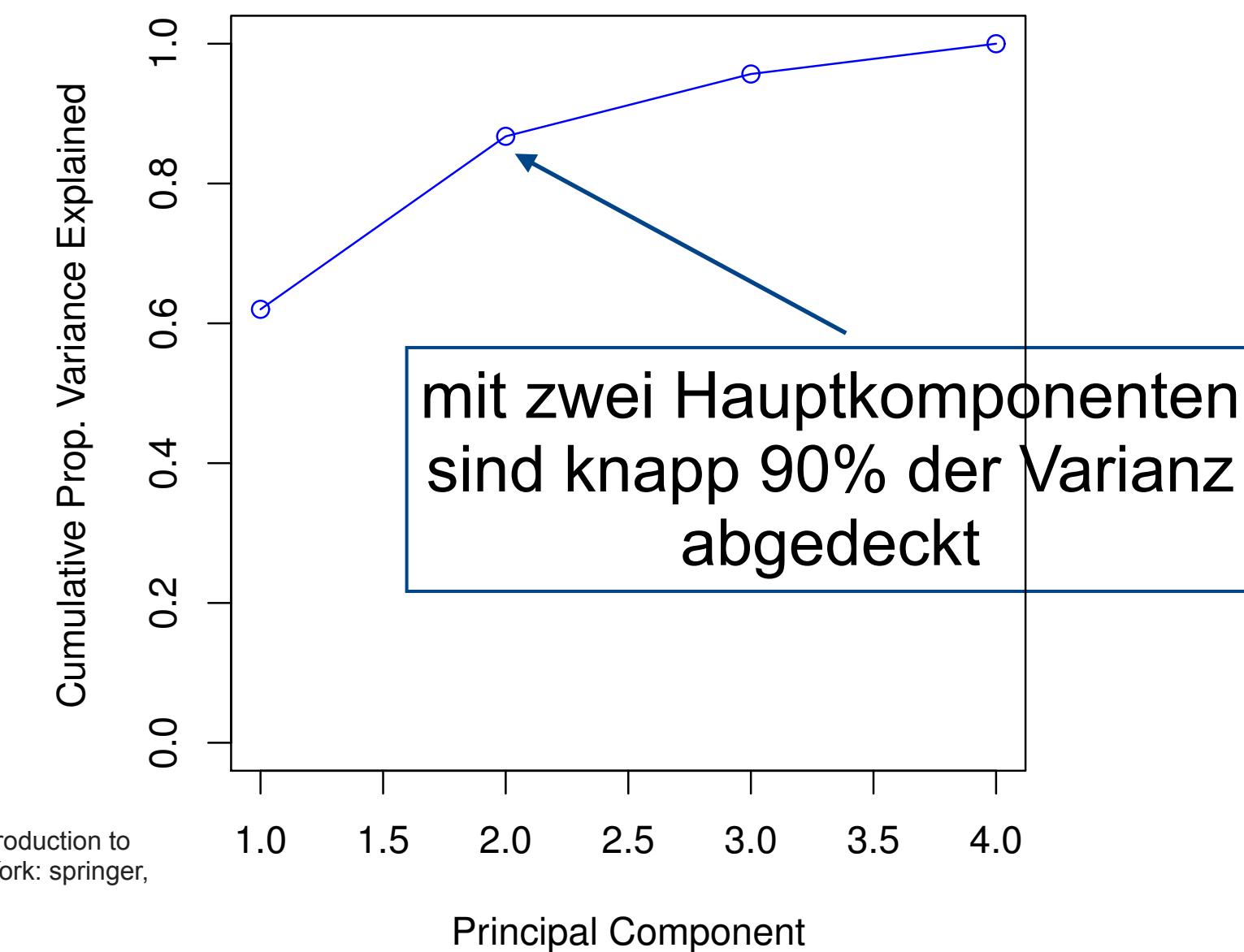
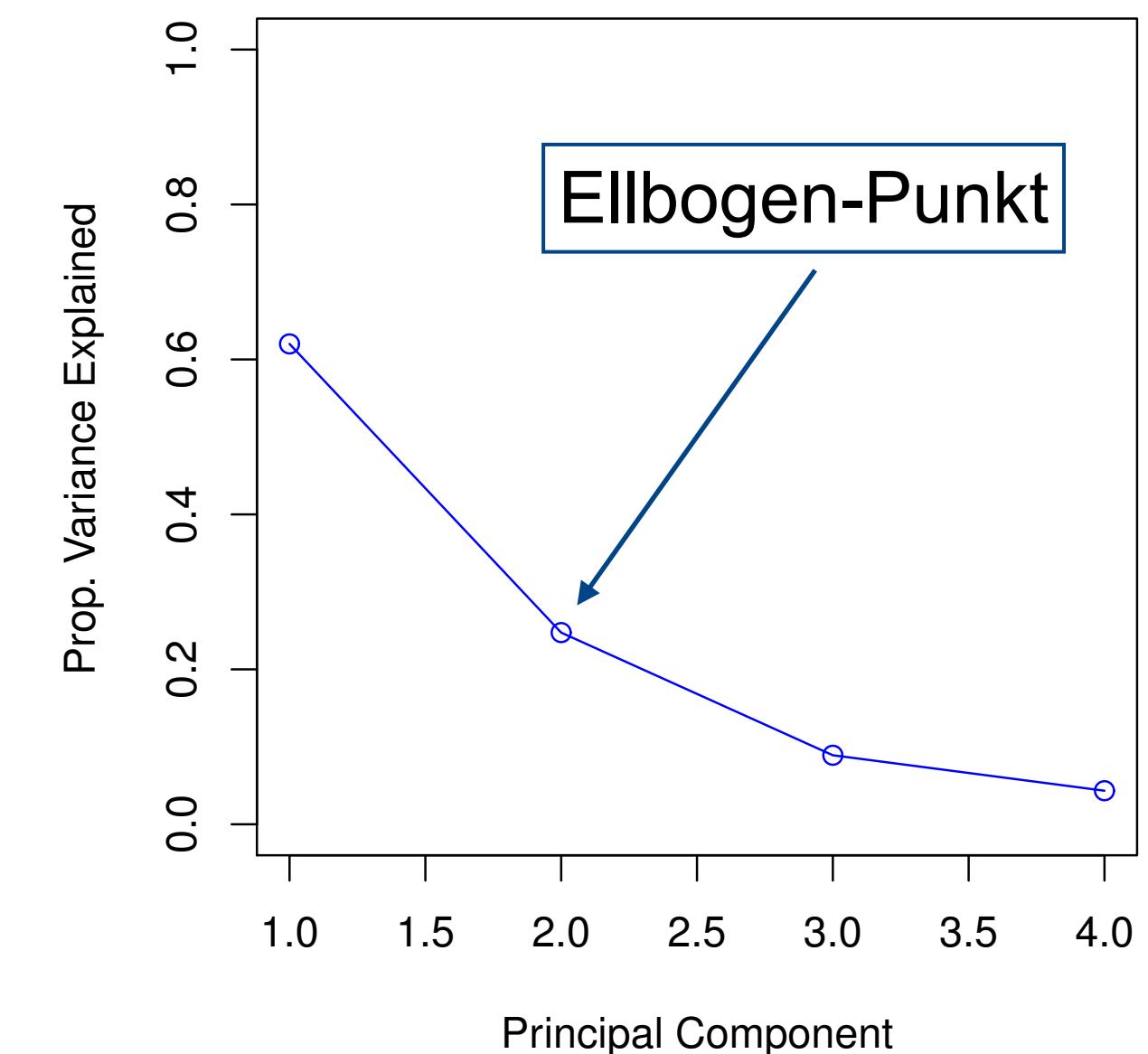
- Gesamtvarianz über alle Komponenten ist $\text{Var}_{\text{total}}$:

$$\text{Var}_{\text{total}} = \sum_{i=1}^D \text{Var}(u_i^\top x_n) = \sum_{i=1}^D \lambda_i$$

- Proportion of Variance Explained ist Verhältnis dieser Werte:

$$\text{PVE}(j) = \frac{\text{Var}(u_j^\top x_n)}{\text{Var}_{\text{total}}} = \frac{\lambda_j}{\sum_{i=1}^D \lambda_i}$$

- **Ziel** ist es, mit möglichst wenigen Komponenten eine möglichst hohe kumulierte PVE zu erreichen

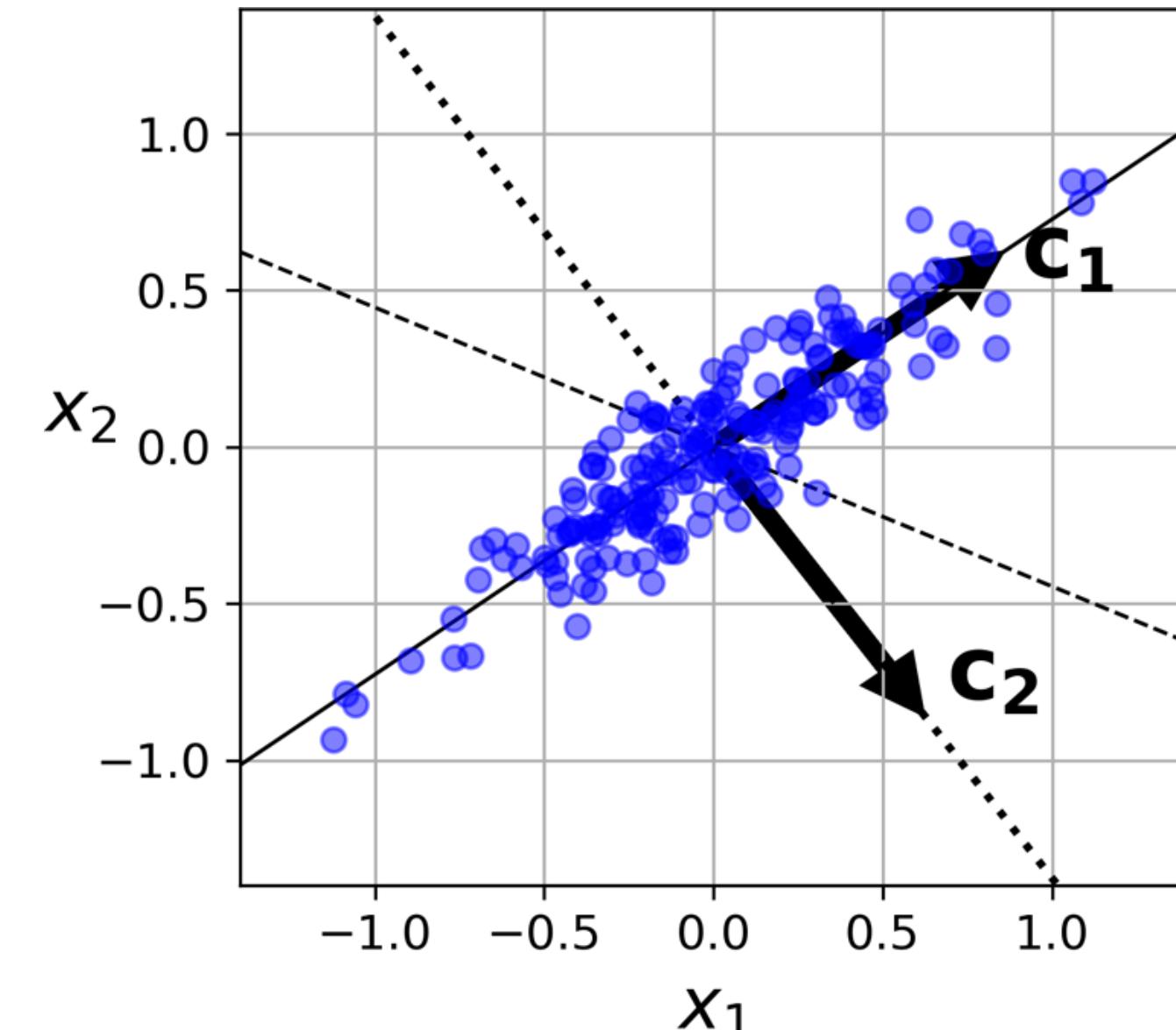


Weitere Interpretation der PCA

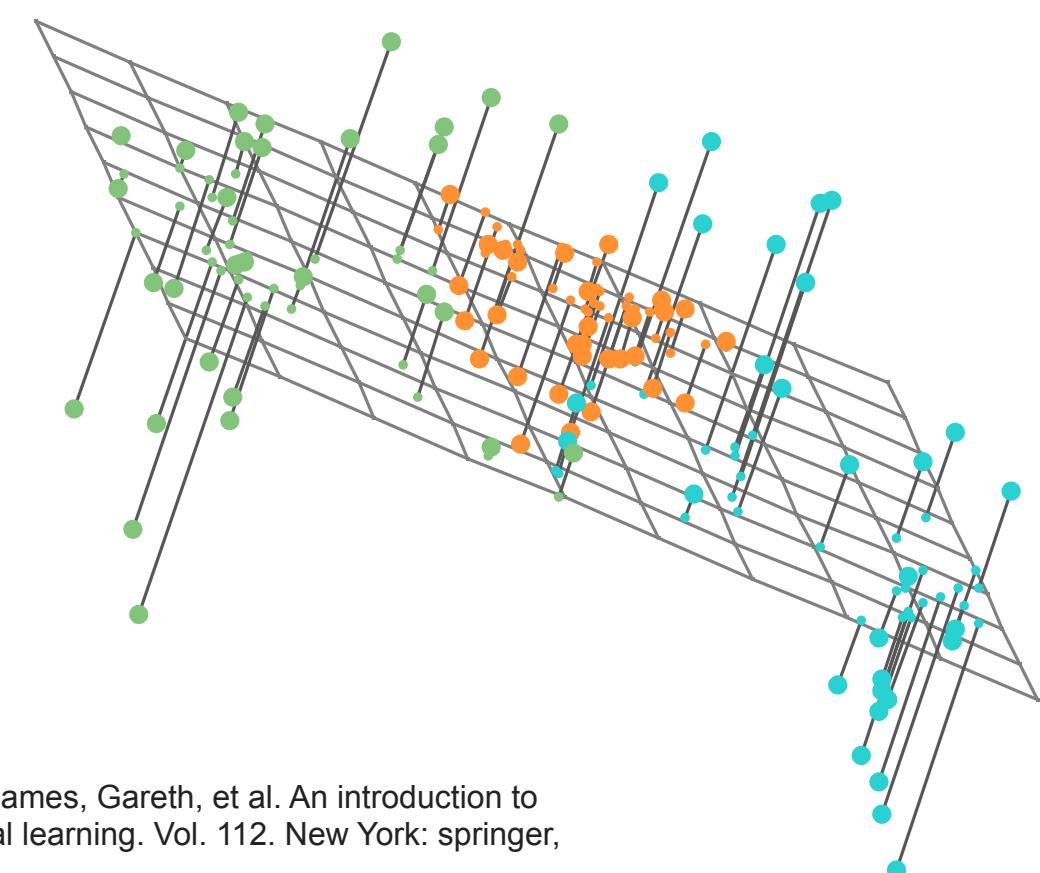
- PCA erzeugt Komponenten, entlang derer Varianz der Daten maximiert wird
- Diese Komponenten spannen Unterräume auf, genauer:

Die ersten d Hauptkomponenten spannen den d -dimensionalen Unterraum auf, der den Daten am nächsten liegt

- Dieser Unterraum minimiert die Summe der quadrierten Abstände zu den Datenpunkten

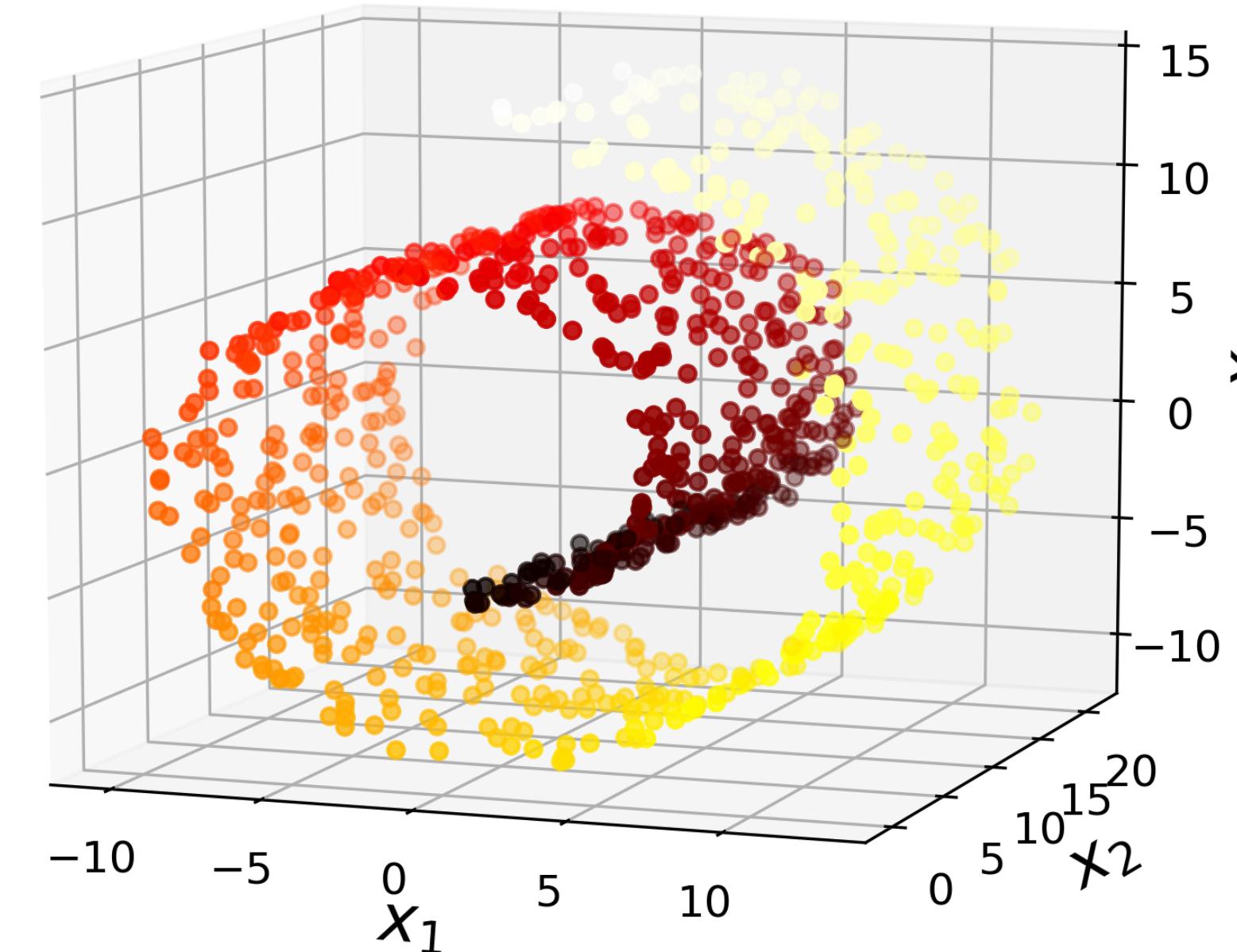


Quelle: https://github.com/ageron/handson-ml2/blob/master/08_dimensionality_reduction.ipynb



Grafik: James, Gareth, et al. An introduction to statistical learning. Vol. 112. New York: Springer, 2013.

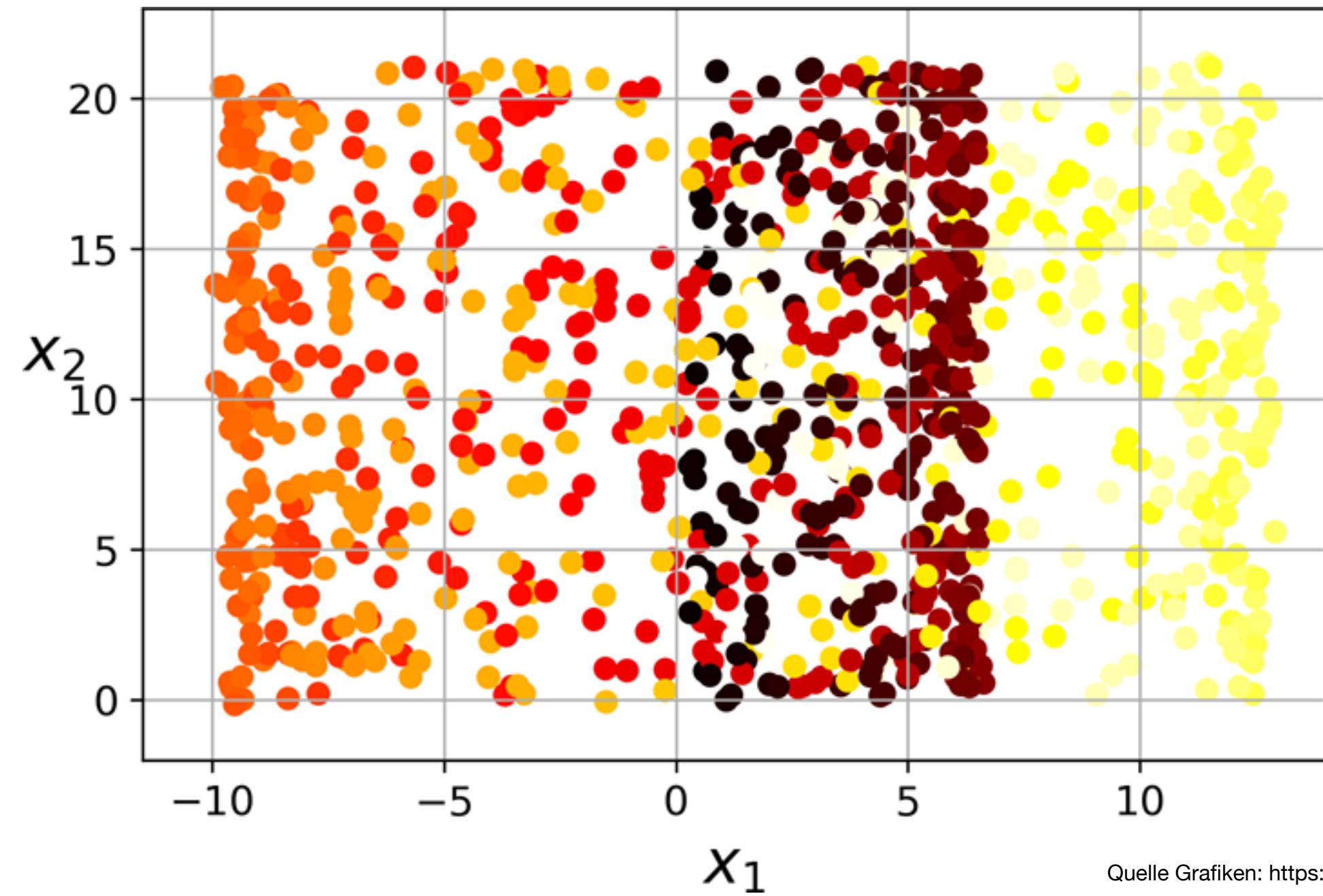
- PCA findet *lineare* Unterräume, die die Daten approximieren
- Liegen die Daten in *nichtlinearen* Unterräumen, kann durch eine PCA-Transformation die wesentliche Struktur verloren gehen
- Beispiel ist die *swiss roll*: eine zweidimensionale nichtlineare Mannigfaltigkeit (eng. *manifold*) im Dreidimensionalen



Quelle: https://github.com/ageron/handson-ml2/blob/master/08_dimensionality_reduction.ipynb

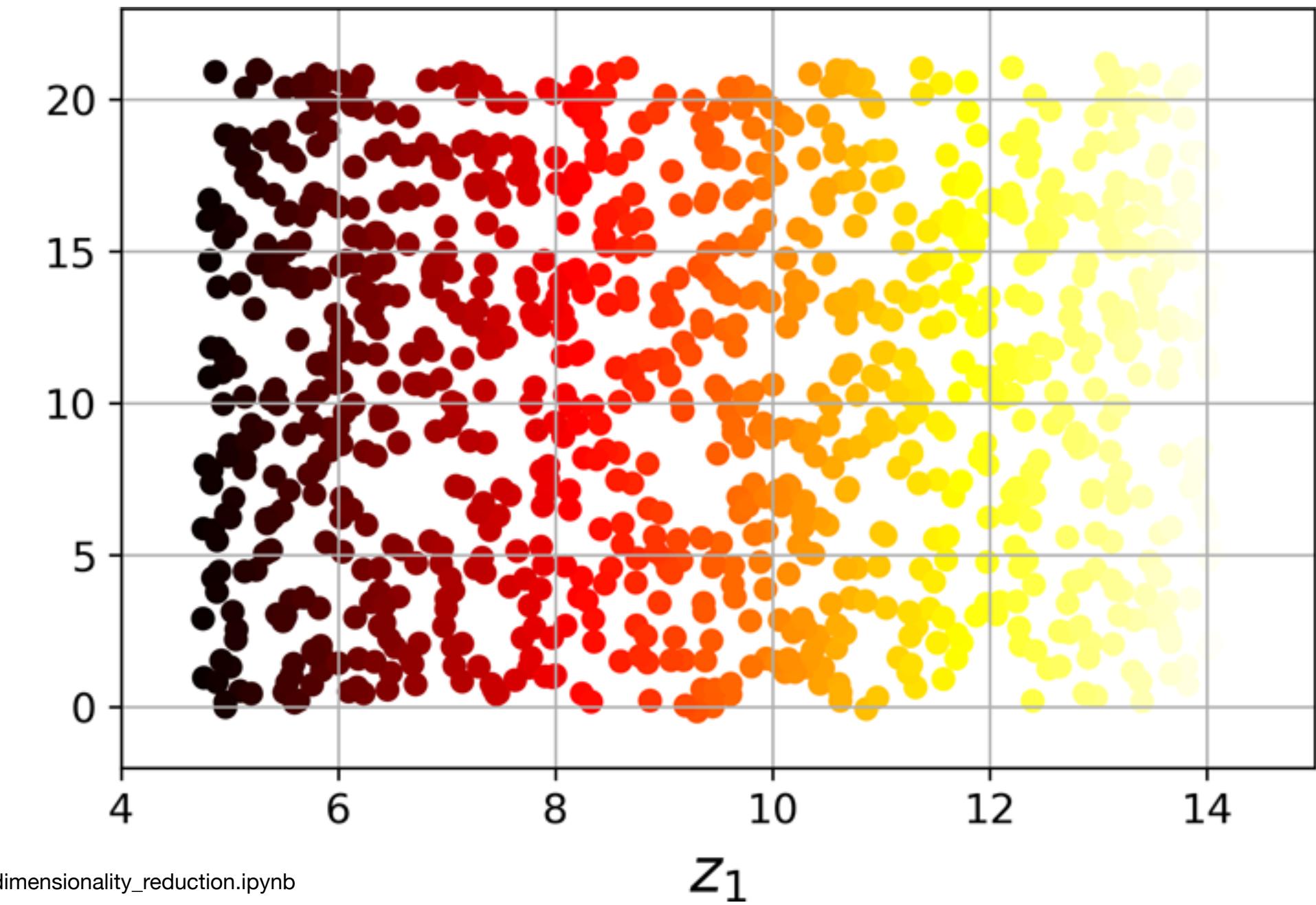
Grenzen der PCA

▶ PCA



Quelle Grafiken: https://github.com/ageron/handson-ml2/blob/master/08_dimensionality_reduction.ipynb

- Lineare Projektion auf x_1 - x_2 -Ebene
- Verschiedene Positionen der Rolle liegen übereinander: wesentliche Struktur geht verloren



- Mögliche nichtlineare Projektion: Rolle wurde „auseinander gerollt“
- Wesentliche Struktur bleibt erhalten

***t*-Distributed Stochastic Neighbourhood Embedding (t-SNE)**

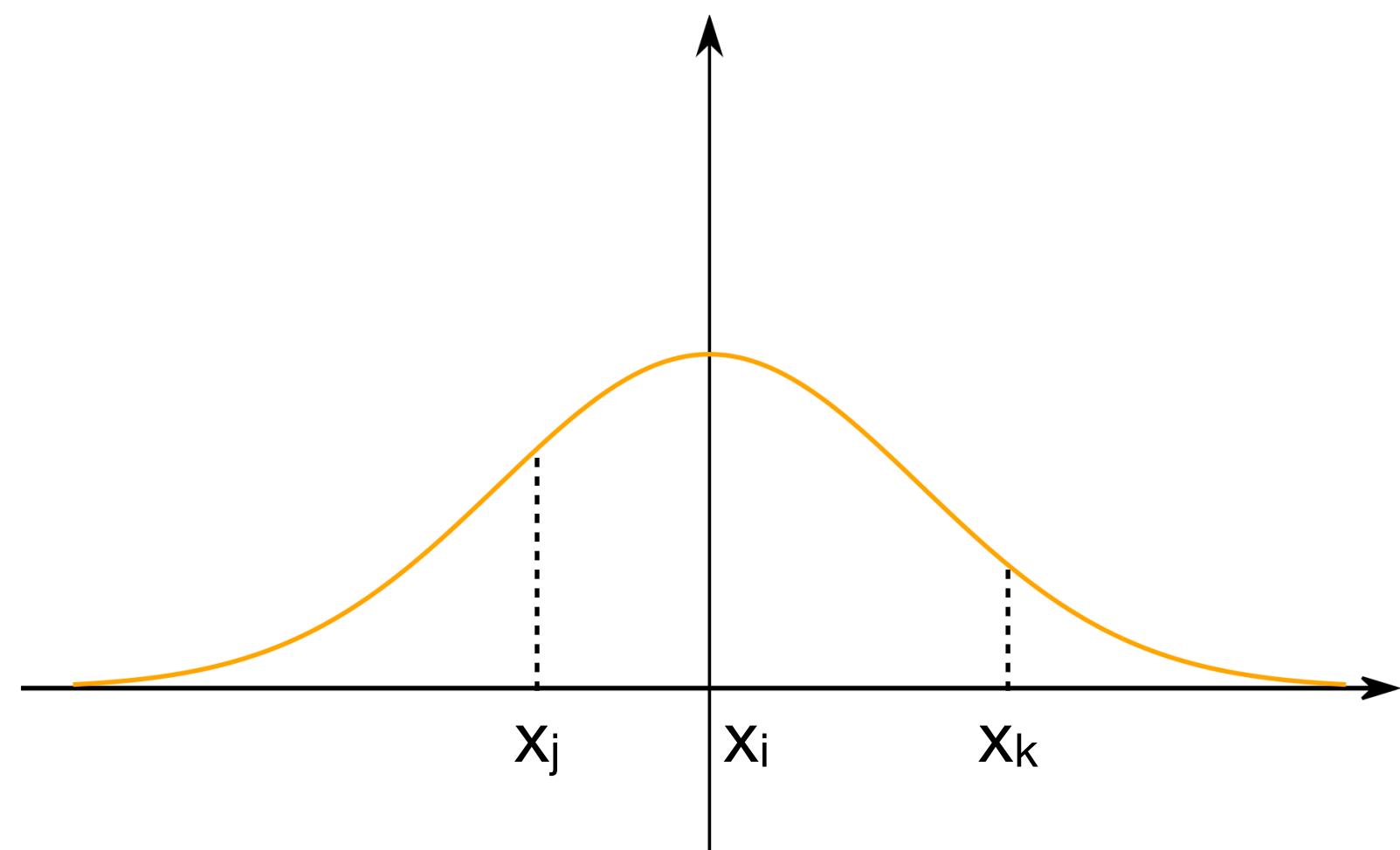
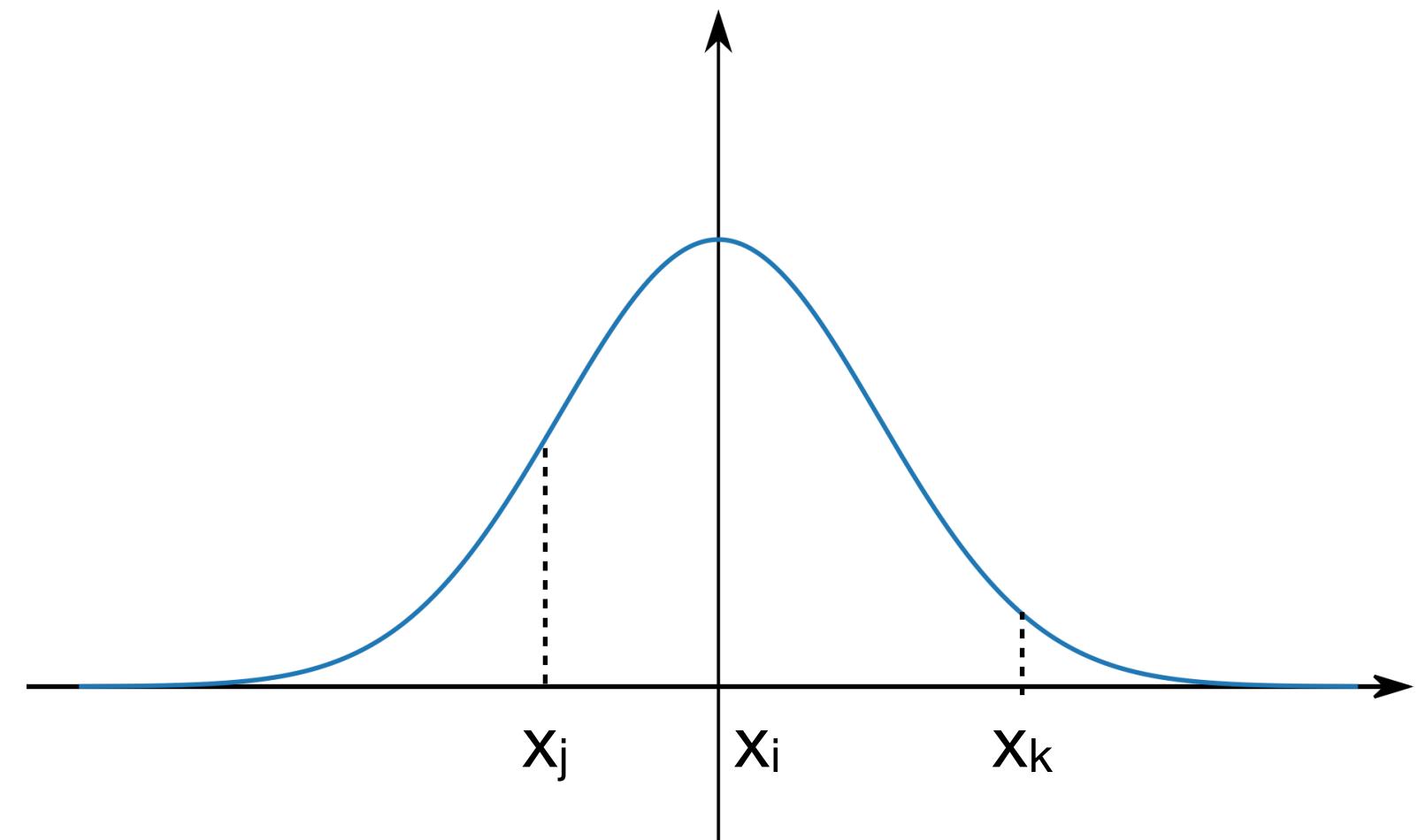
t-SNE

- Dimensionen reduzieren, um Daten visuell darzustellen (2D o. 3D)
- Kann mit Hunderten und auch Tausenden Dimensionen umgehen
- Ähnliche Datenpunkte sollen nah beieinander bleiben, unähnliche Datenpunkte weit auseinander
- Gut, um Cluster von Punkten zu erkennen
- Wird nicht als Pre-Processing für Pipelines verwendet
 - Globale Struktur geht verloren

- Ob zwei Punkte ähnlich zueinander sind, wird durch Wahrscheinlichkeitsverteilungen modelliert
- Im Ausgangsraum gibt p_{ij} an, wie hoch die Wahrscheinlichkeit ist, dass die Datenpunkte x_i und x_j ähnlich zueinander sind (daher der Begriff *stochastic neighbourhood*)
- Im (zwei- bzw. dreidimensionalen) Zielraum sollen Punkte y_n liegen, die den Datenpunkten x_n entsprechen
- Die Wahrscheinlichkeit, dass y_i und y_j ähnlich zueinander sind, wird durch q_{ij} angegeben
- **Ziel** ist es, solche Punkte y_n zu finden, sodass die Wahrscheinlichkeiten p_{ij} und q_{ij} möglichst gleich sind

- Der Wert $p_{j|i}$ modelliert, wie wahrscheinlich der Punkt x_i den Punkt x_j als seinen Nachbar ansieht (sprich Ähnlichkeit)
- Ähnlichkeit wird mit einer Normalverteilung modelliert
 - x_i liegt genau in der Mitte der Glockenkurve
 - Je weiter weg x_j von x_i liegt, desto weniger ähnlich ist der Punkt
- Der Wert von x_j in der Glockenkurve ist

$$\exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma_i^2}\right)$$



- Die Wahrscheinlichkeit $p_{j|i}$ berechnet sich durch

$$p_{j|i} = \begin{cases} \frac{\exp(-\|x_i - x_j\|^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|x_i - x_k\|^2/2\sigma_i^2)}, & i \neq j \\ 0, & i = j \end{cases}$$

- Damit symmetrische Werte entstehen, setzen wir

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$$

- Wie wählen wir jetzt die Varianzen σ_i ?

Ähnlichkeit von x_j zu x_i

Summe aller Ähnlichkeiten

- Die Varianz σ_i wird durch die Umgebung von x_i bestimmt
 - Hat x_i viele nahe Nachbarn, so soll die Glockenkurve eng sein \rightarrow Varianz σ_i fällt klein aus
 - Hat x_i wenige nahe Nachbarn, so soll die Glockenkurve weit sein \rightarrow Varianz σ_i fällt groß aus

- Die Berechnung der Varianz wird durch die zu wählende Perplexität bestimmt:

$$\text{Perp}(P_i) = 2^{H(P_i)}$$

- $H(P_i)$ ist die Shannon-Entropie der Verteilung P_i :

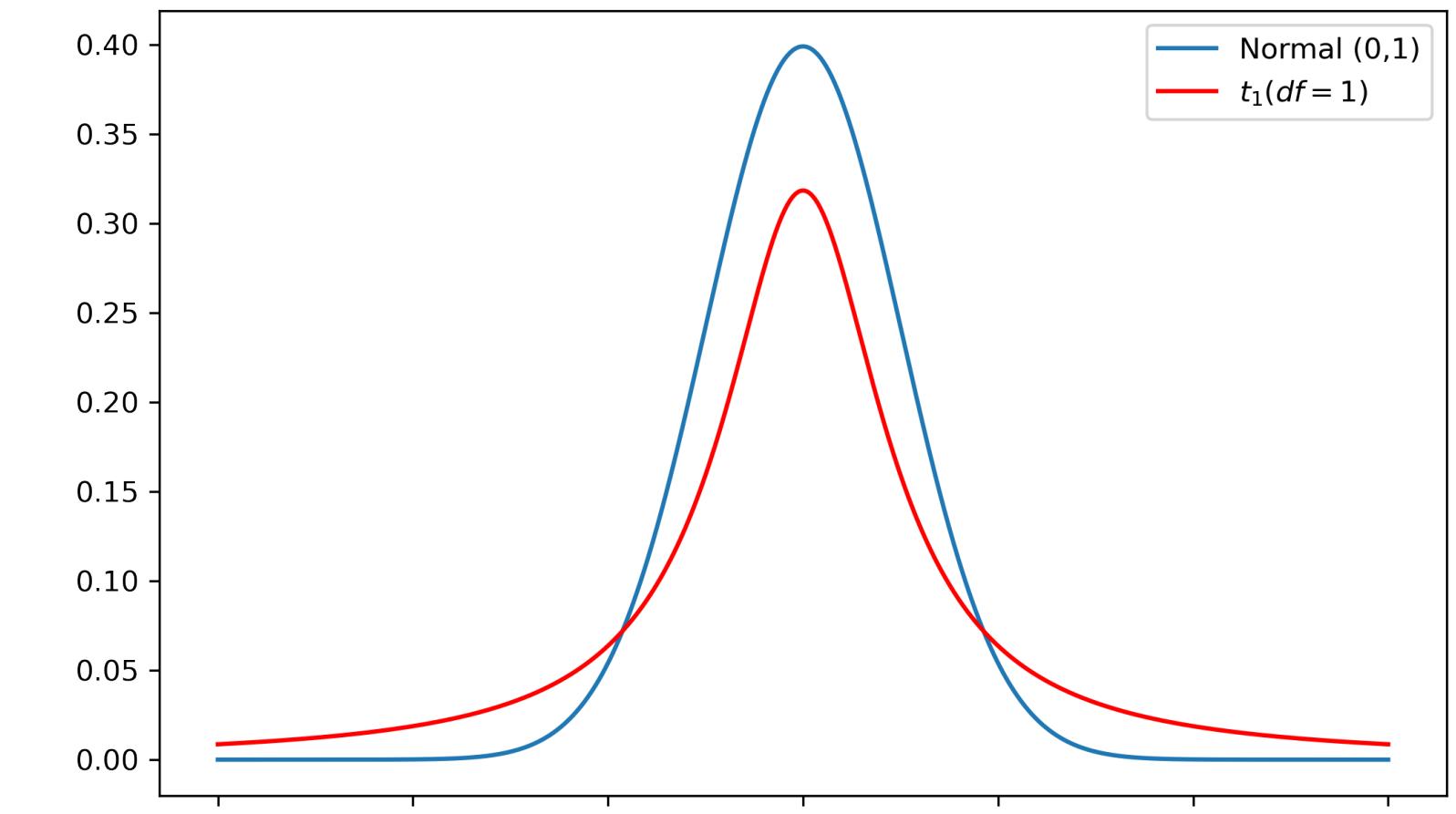
$$H(P_i) = \sum_j p_{j|i} \log_2 p_{j|i}$$

- Anschaulich gibt die Perplexität an, wie viele nahe Nachbarn ein Punkt hat

- In der Praxis werden für die Perplexität Werte zwischen 5 und 50 gewählt. Die Wahl hängt von den Daten ab.

- Prinzipiell können die Wahrscheinlichkeiten q_{ij} für die Punkte y_i im Zielraum auch mit Normalverteilungen modelliert werden
- Problem aber: die Punkte liegen dann alle zu nah beieinander
 - Erinnerung: Räume mit niedriger Dimension haben weniger Volumen!
 - Dieses Phänomen nennt sich **crowding problem**
- Statt der Normalverteilung wählen wir die t -Verteilung
 - Ränder der t -Verteilung haben mehr Masse
 - Punkte werden dadurch auseinander gezogen
- Die Wahrscheinlichkeit q_{ij} berechnet sich durch

$$q_{ij} = \begin{cases} \frac{(1 + \|y_i - y_j\|^2)^{-1}}{\sum_k \sum_{l \neq k} (1 + \|y_k - y_l\|^2)^{-1}}, & i \neq j \\ 0, & i = j \end{cases}$$



$$f(t) \sim (1 + t^2)^{-1}$$

In rot die t -Verteilung
(deswegen das t in t-SNE)

- Ziel ist es, die q_{ij} an die Werte p_{ij} anzupassen (indem man die Punkte y_n passend wählt)
- Wie gut bzw. schlecht eine Wahrscheinlichkeitsverteilung die andere abbildet, wird durch die *Kullback-Leibler-Divergenz* gemessen:

$$D_{\text{KL}}(P \parallel Q) = \sum_{i,j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

- Je niedriger die Divergenz, desto besser bildet die Verteilung Q die Verteilung P ab
- **Ziel** ist, die Divergenz zu minimieren: wir betrachten die Divergenz als Kostenfunktion C

- Wir verwenden *gradient descent*, um die Kostenfunktion C zu minimieren

$$\frac{\partial C}{\partial y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j)(1 + \|y_i - y_j\|^2)^{-1}$$

- Die y_n werden zufällig initialisiert und mithilfe des Gradienten angepasst
- Lösungen sind nicht eindeutig, da mehrere lokale Minima existieren können
 - Bei verschiedenen Initialisierungen können verschiedene Visualisierungen herauskommen
- Es existieren verschiedene Optimierungen, um die Kostenfunktion schneller zu minimieren

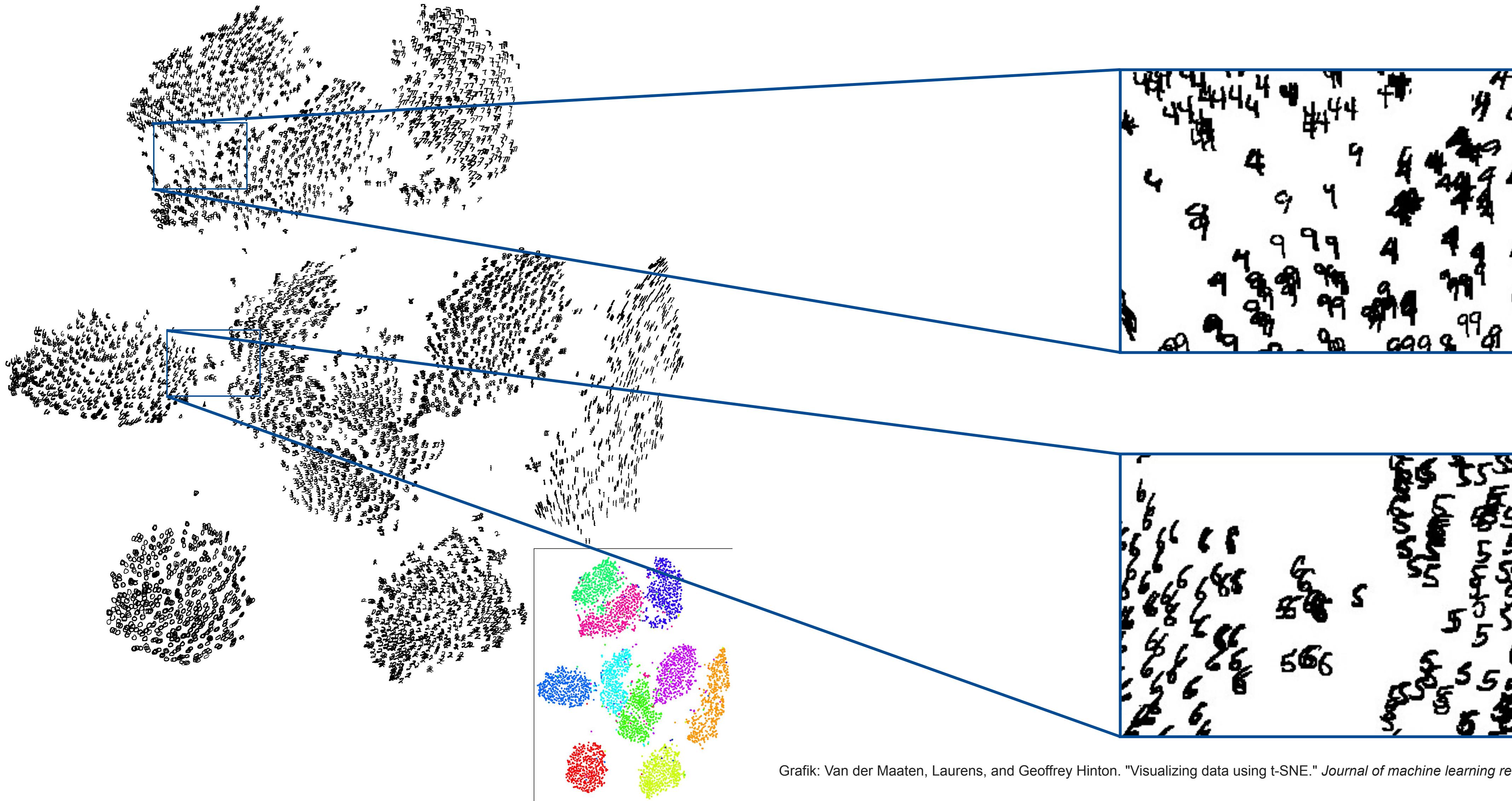
MNIST-Datenbank

- Datensatz mit handgeschriebenen Ziffern
- Bilder sind 28x28 Pixel groß und haben 256 Graustufen
- Daten haben dementsprechend 784 Dimensionen
- De facto „Hello World“-Datensatz für Bilderkennung

0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9

Beispiel

► t-SNE



Grafik: Van der Maaten, Laurens, and Geoffrey Hinton. "Visualizing data using t-SNE." *Journal of machine learning research* 9.11 (2008).