Fachhochschule Graubünden
University of Applied Sciences

Methods & Algorithms

(formerly known as Introduction to Data Science)

FS 2023

Prof. Dr. rer. nat. habil. Ralf-Peter Mundani

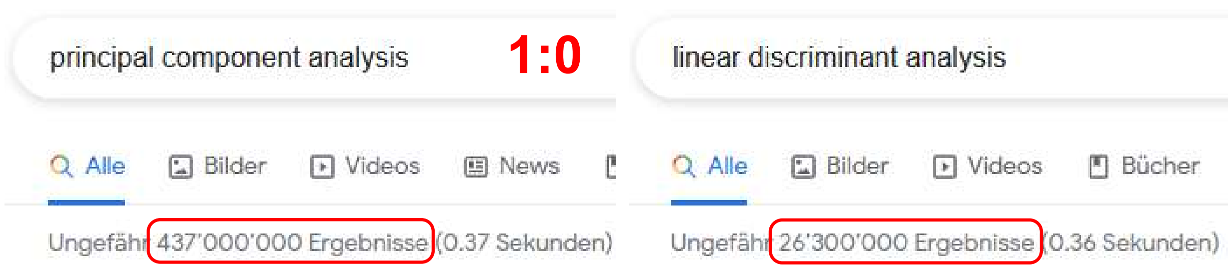
DAViS

PCA – darf’s noch ein bisschen mehr sein…?

- Hauptkomponentenzerlegung
  - mächtiges Werkzeug (aber schwer zu verstehen)
  - Dimensionsreduktion unter Erhaltung grösstmöglicher Varianz
  - projizierte Daten nicht immer für Klassifikation geeignet

- Diskriminanzanalyse
  - ähnliches Werkzeug («*same same, but different…*»)
  - erlaubt multivariate Klassifikation
  - guter Ausgangspunkte für weitere Ansätze (leider zu viele für diese Veranstaltung…)

- was sagt Google…?



Zur Abwechslung: mal wieder ein bisschen lineare Algebra ☺

- Nomenklatur
  - Skalare: *s*
  - Vektoren: $\vec{\mathbf{v}}$ (1D Reihung)
  - Matrizen: $\mathbf{A}$ (2D Reihung)

- Operationen
  - Skalarprodukt: $\vec{\mathbf{v}}^T \cdot \vec{\mathbf{w}} = s$
  - dyadisches Produkt: $\vec{\mathbf{v}} \cdot \vec{\mathbf{w}}^T = \mathbf{A}$
  - Matrix-Vektor-Multiplikation: $\mathbf{A} \cdot \vec{\mathbf{v}} = \vec{\mathbf{w}}$
  - Matrix-Matrix-Multiplikation: $\mathbf{A} \cdot \mathbf{B} = \mathbf{C}$ (brauchen wir nicht…)

Zur Abwechslung: mal wieder ein bisschen lineare Algebra ☺

- Skalarprodukt $\vec{\mathbf{v}}^T \cdot \vec{\mathbf{w}}$

$$\vec{\mathbf{v}}^T \cdot \vec{\mathbf{w}} = \begin{pmatrix} v_{x_1} & v_{x_2} & \cdots & v_{x_n} \end{pmatrix} \cdot \begin{pmatrix} w_{x_1} \\ w_{x_2} \\ \cdots \\ w_{x_n} \end{pmatrix} = v_{x_1} \cdot w_{x_1} + v_{x_2} \cdot w_{x_2} + \ldots + v_{x_n} \cdot w_{x_n} = \sum_{i=1}^{n} v_{x_i} \cdot w_{x_i}$$

- Beispiel: $\vec{\mathbf{v}} = \begin{pmatrix} 3 \\ 1 \\ 2 \end{pmatrix}, \vec{\mathbf{w}} = \begin{pmatrix} 4 \\ 0 \\ 3 \end{pmatrix}$

$$\vec{\mathbf{v}}^T \cdot \vec{\mathbf{w}} = \begin{pmatrix} 3 & 1 & 2 \end{pmatrix} \cdot \begin{pmatrix} 4 \\ 0 \\ 3 \end{pmatrix} = 3 \cdot 4 + 1 \cdot 0 + 2 \cdot 3 = 18$$

Zur Abwechslung: mal wieder ein bisschen lineare Algebra ☺

Kennen wir schon.

- Matrizen

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{pmatrix}$$

mit m Zeilen und n Spalten (kurz: $m \times n$)

Hauptdiagonale (für $m = n$)

- Transponierte $\mathbf{A}^T$ einer Matrix $\mathbf{A}$

$$\mathbf{A}^T = \begin{pmatrix} a_{11} & a_{21} & \cdots & a_{m1} \\ a_{12} & a_{22} & \cdots & a_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{mn} \end{pmatrix}$$

mit n Zeilen und m Spalten (kurz: $n \times m$)

Zur Abwechslung: mal wieder ein bisschen lineare Algebra ☺

- besondere Matrizen
  - symmetrische Matrix (d.h. $\mathbf{A} = \mathbf{A}^T$)
    - Beispiel (3×3)

$$\mathbf{A} = \begin{pmatrix} 4 & 1 & 3 \\ 1 & 2 & 1 \\ 3 & 1 & 5 \end{pmatrix} \quad \text{mit} \quad \mathbf{A}^T = \begin{pmatrix} 4 & 1 & 3 \\ 1 & 2 & 1 \\ 3 & 1 & 5 \end{pmatrix}$$

  - Diagonalmatrix

$$\mathbf{D} = \begin{pmatrix} * & & & \\ & * & & \\ & & \ddots & \\ & & & * \end{pmatrix} \qquad \text{z.B. (4×4): } \mathbf{D} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

(auch Identitätsmatrix $\mathbf{I}$)

Zur Abwechslung: mal wieder ein bisschen lineare Algebra ☺

- Matrix-Vektor-Multiplikation

$$\mathbf{A}\cdot\vec{\mathbf{v}} = \underbrace{\begin{pmatrix} \boxed{a_{11} \quad a_{12} \quad \cdots \quad a_{1n}} \\ a_{21} \quad a_{22} \quad \cdots \quad a_{2n} \\ \vdots \quad\quad \vdots \quad\quad \ddots \quad\quad \vdots \\ a_{m1} \quad a_{m2} \quad \cdots \quad a_{mn} \end{pmatrix}}_{|\,m \times n\,|} \cdot \underbrace{\begin{pmatrix} v_{x_1} \\ v_{x_2} \\ \vdots \\ v_{x_n} \end{pmatrix}}_{|\,n\,|} = \begin{pmatrix} a_{11}\cdot v_{x_1} + a_{12}\cdot v_{x_2} + \cdots + a_{1n}\cdot v_{x_n} \\ a_{21}\cdot v_{x_1} + a_{22}\cdot v_{x_2} + \cdots + a_{2n}\cdot v_{x_n} \\ \vdots \\ a_{m1}\cdot v_{x_1} + a_{m2}\cdot v_{x_2} + \cdots + a_{mn}\cdot v_{x_n} \end{pmatrix} = \underbrace{\begin{pmatrix} \boxed{w_{x_1}} \\ w_{x_2} \\ \vdots \\ w_{x_m} \end{pmatrix}}_{|\,m\,|} = \vec{\mathbf{w}}$$

- Beispiel

$$\mathbf{A}\cdot\vec{\mathbf{v}} = \begin{pmatrix} 4 & 1 & 3 \\ 1 & 2 & 1 \\ 3 & 1 & 5 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ 1 \\ 3 \end{pmatrix} = \begin{pmatrix} 4\cdot 2 + 1\cdot 1 + 3\cdot 3 \\ 1\cdot 2 + 2\cdot 1 + 1\cdot 3 \\ 3\cdot 2 + 1\cdot 1 + 5\cdot 3 \end{pmatrix} = \begin{pmatrix} 18 \\ 7 \\ 22 \end{pmatrix} = \vec{\mathbf{w}}$$

Zur Abwechslung: mal wieder ein bisschen lineare Algebra ☺

- dyadisches (äusseres) Produkt $\vec{\mathbf{v}} \cdot \vec{\mathbf{w}}^T$

$$\vec{\mathbf{v}} \cdot \vec{\mathbf{w}}^T = \begin{pmatrix} v_{x_1} \\ v_{x_2} \\ \vdots \\ v_{x_m} \end{pmatrix} \cdot \begin{pmatrix} w_{x_1} & w_{x_2} & \cdots & w_{x_n} \end{pmatrix} = \underbrace{\begin{pmatrix} v_{x_1} \cdot w_{x_1} & v_{x_1} \cdot w_{x_2} & \cdots & v_{x_1} \cdot w_{x_n} \\ v_{x_2} \cdot w_{x_1} & v_{x_2} \cdot w_{x_2} & \cdots & v_{x_2} \cdot w_{x_n} \\ \vdots & \vdots & \ddots & \vdots \\ v_{x_m} \cdot w_{x_1} & v_{x_m} \cdot w_{x_2} & \cdots & v_{x_m} \cdot w_{x_n} \end{pmatrix}}_{|\, m \times n \,|} = \mathbf{A}$$

- Beispiel: $\vec{\mathbf{v}} = \begin{pmatrix} 3 \\ 1 \\ 2 \end{pmatrix}, \vec{\mathbf{w}} = \begin{pmatrix} 4 \\ 0 \\ 3 \end{pmatrix}$

$$\vec{\mathbf{v}} \cdot \vec{\mathbf{w}}^T = \begin{pmatrix} 3 \\ 1 \\ 2 \end{pmatrix} \cdot \begin{pmatrix} 4 & 0 & 3 \end{pmatrix} = \begin{pmatrix} 3 \cdot 4 & 3 \cdot 0 & 3 \cdot 3 \\ 1 \cdot 4 & 1 \cdot 0 & 1 \cdot 3 \\ 2 \cdot 4 & 2 \cdot 0 & 2 \cdot 3 \end{pmatrix} = \begin{pmatrix} 12 & 0 & 9 \\ 4 & 0 & 3 \\ 8 & 0 & 6 \end{pmatrix}$$

Richtig trennen – aber in welche Tonne…?

- Diskriminanzanalyse – ein vielseitiges Werkzeug für die
  - Dimensionsreduktion
  - robuste Klassifizierung (d.h. selbe Ergebnisse mit oder ohne Normalisierung der Daten)

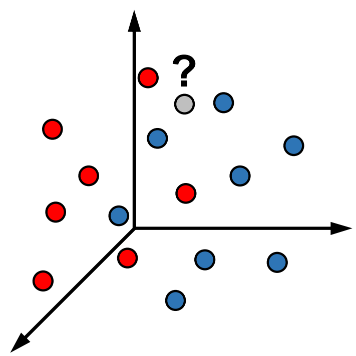
- prominente Anwendungsbeispiele
  - Konkursversagen mit 80–90% Genauigkeit (E. Altman, 1968) basierend auf 31 Jahren Daten
  - Unterscheidung von verschiedener Krankheitsstufen in biomedizinischen Anwendungen
  - Gesichtserkennung (George Orwell lässt grossen ☺)

- warum also noch ein Algorithmus…?
  - funktioniert sehr gut nicht nur im Fall binärer Daten (→ multivariate Analyse)
  - Vorverarbeitung von Daten zur Reduktion von (Berechnungs)Kosten
  - …

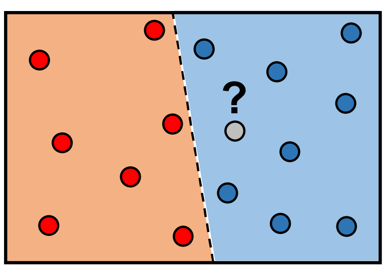
Richtig trennen – aber in welche Tonne…?

- Idee
  - Methode des überwachten Lernens
  - Ausgangspunkt: klassifizierte Daten mit d Merkmalen in k Klassen
  - Diskriminanzfunktion teilt (Sub)Raum in k zugehörige disjunkte Regionen
  - einfache Klassifikation: prüfe, in welcher Region das Datum liegt

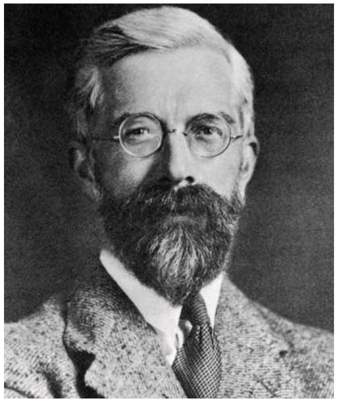
  - Beispiel ($d = 3$, $k = 2$)



$\Rightarrow$



Richtig trennen – aber in welche Tonne…?

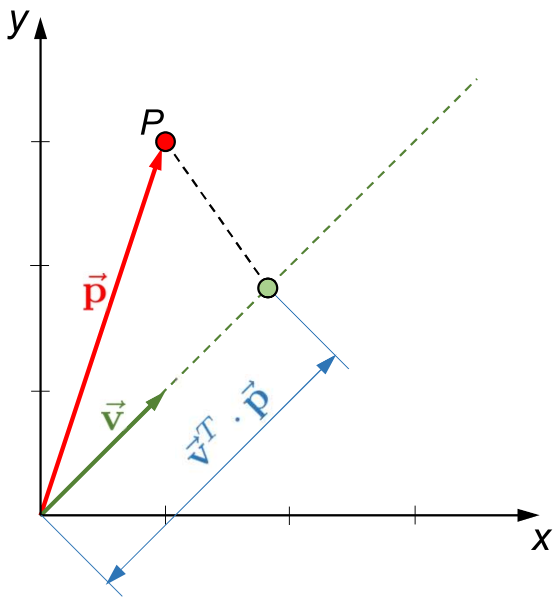


Quelle: bibmath.net

- …noch ein paar Gedanken
  - Diskriminanzanalyse setzt Normalverteilung der Daten voraus
  - allgemeiner Fall: Diskriminanzanalyse nach Ronald A. FISHER
    - britischer Statistiker (1890–1962)
    - berühmte Beiträge: *Maximum-Likelihood*-Prinzip, Varianzanalyse (ANOVA), *F*-Verteilung
  - simple Annahme: n Beobachtungen (mit d Merkmalen), die sich alle mindestens einer von k Klassen zuordnen lassen
  - geometrisches Verfahren: Beobachtungsraum wird durch eine Reihe von Hyperebenen derart partitioniert, dass diese die Grenzen zwischen jeweils zwei Klassen darstellen
  - im weiteren: Diskussion für $d = 2$, später dann Verallgemeinerung auf $d > 2$

Richtig trennen – aber in welche Tonne…?

- Vorbereitung: Projektion von Punkten auf eine Gerade (d.h. vom 2D auf 1D)



Beispiel:

$$P = (1, 3),\ \vec{\mathbf{p}} = \begin{pmatrix} 1 \\ 3 \end{pmatrix},\ \vec{\mathbf{v}} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

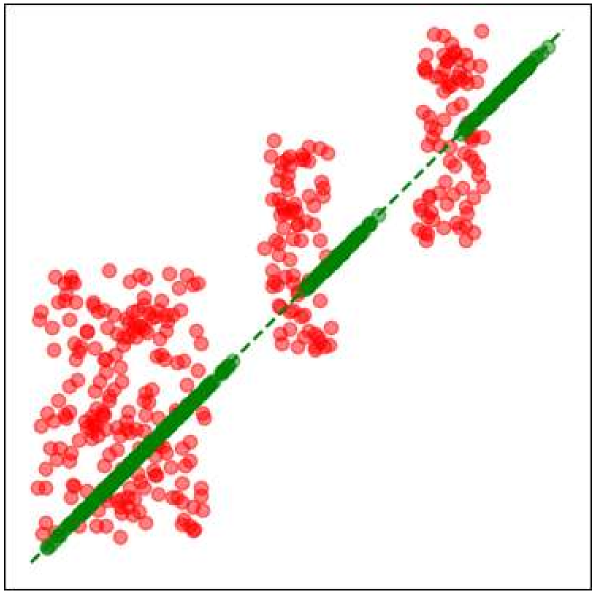
$$\vec{\mathbf{v}}^T \vec{\mathbf{p}} = (1 \ \ 1) \cdot \begin{pmatrix} 1 \\ 3 \end{pmatrix} = 1 \cdot 1 + 1 \cdot 3 = 4$$

- nach Projektion $\Rightarrow$ jeder Punkt hat Abstand (auf der Geraden durch $\vec{\mathbf{v}}$) vom Ursprung

Richtig trennen – aber in welche Tonne…?

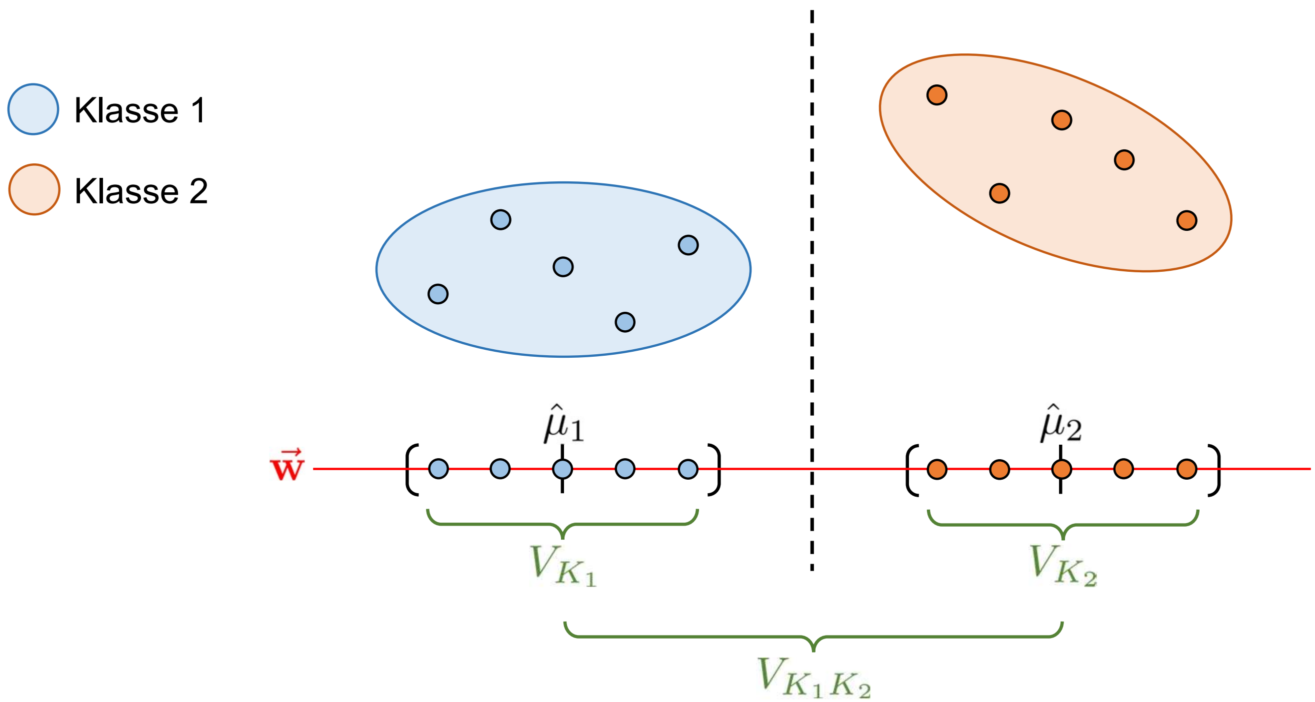
- kleine Fingerübung (mal wieder Datei 'clicks.dat')
  - zunächst das Skalarprodukt händisch (**np.dot()**) ausrechnen ☺

```
data = np.loadtxt('clicks.dat', delimiter=',')
v = np.array([1., 1.])
data_p = []
for p in data:
  data_p.append(np.dot(v, p))
plt.plot(data_p, data_p, 'ro')
plt.show()
```



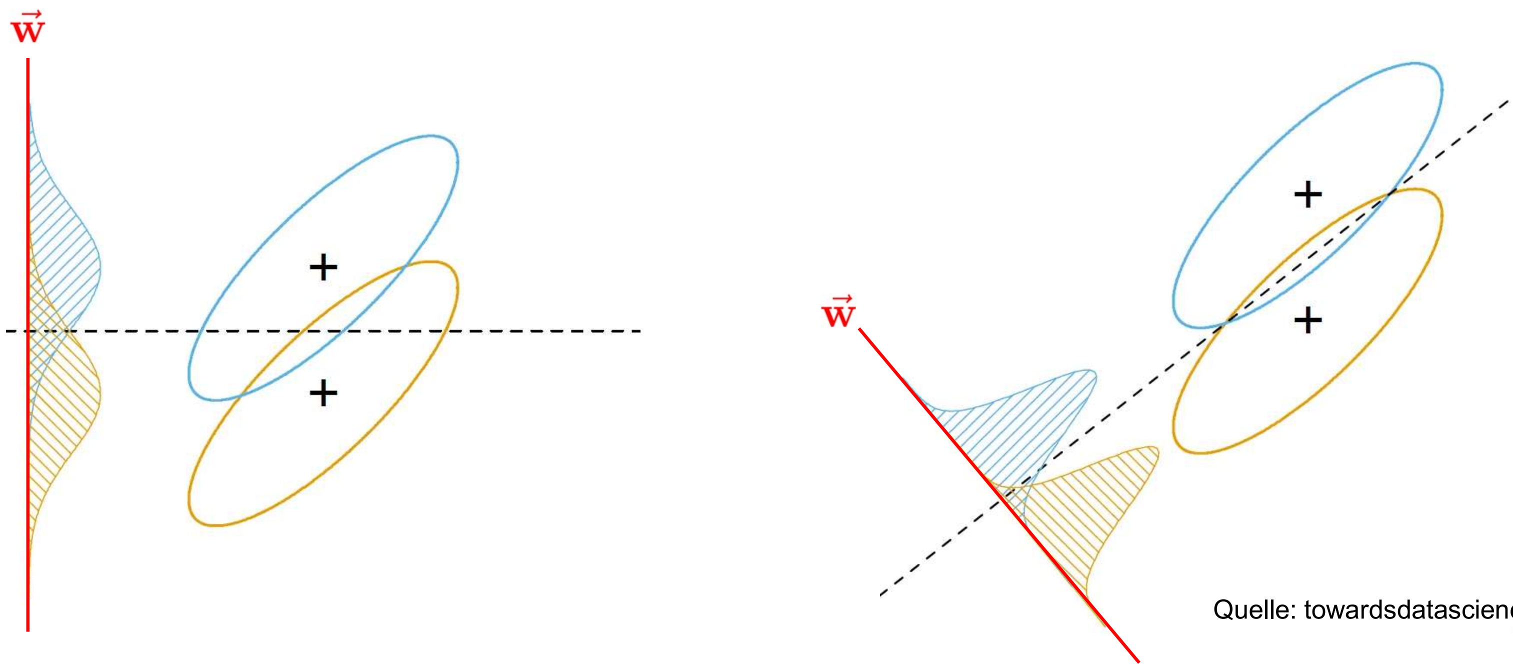
Linear Discriminant Analysis (LDA)

- multivariate Diskriminanzanalyse nach R. Fisher (1936)
  - Idee: Projektion *d*-dimensionaler Daten in einen Unterraum (< *d*), sodass Variabilität zwischen den Klassen maximal und Variabilität innerhalb einer Klasse minimal wird



Linear Discriminant Analysis (LDA)

- multivariate Diskriminanzanalyse nach R. Fisher (1936)
  - etwas anschaulicher: finde optimales $\vec{\mathbf{w}}$, sodass V_{K_1}, V_{K_2} minimal und $V_{K_1 K_2}$ maximal



Quelle: towardsdatascience.com

Linear Discriminant Analysis (LDA)

- multivariate Diskriminanzanalyse nach R. Fisher (1936)
    - kurze Auffrischung Statistik ☺

$$\mu_x = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad \sigma_x^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \mu_x)^2 \qquad \sigma_{x,y} = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \mu_x)(y_i - \mu_y)$$

(Mittelwert) (Varianz) (Kovarianz)

- Preisfrage: was ist Unterschied zwischen Kovarianz und Korrelation…?

- Anmerkung: Cov(*X*, *X*) = Var(*X*) $\Leftrightarrow$ $\sigma_{x,x} = \sigma_x^2$

Linear Discriminant Analysis (LDA)

- multivariate Diskriminanzanalyse nach R. Fisher (1936)
    - Vorhang auf für die Kovarianzmatrix Σ…

$$\Sigma_{(1D)} = \begin{pmatrix} \sigma_x^2 \end{pmatrix} \qquad \Sigma_{(2D)} = \begin{pmatrix} \sigma_x^2 & \sigma_{x,y} \\ \sigma_{y,x} & \sigma_y^2 \end{pmatrix} \qquad \Sigma_{(3D)} = \begin{pmatrix} \sigma_x^2 & \sigma_{x,y} & \sigma_{x,z} \\ \sigma_{y,x} & \sigma_y^2 & \sigma_{y,z} \\ \sigma_{z,x} & \sigma_{z,y} & \sigma_z^2 \end{pmatrix} \qquad \Sigma_{(nD)} = \ldots$$

- unnützes Wissen: Σ ist symmetrisch (da $\sigma_{x,y} = \sigma_{y,x}$) $\Rightarrow$ Vorteile für die Berechnung der LDA
- damit Berechnung des optimalen $\vec{\mathrm{w}}$ möglich

Linear Discriminant Analysis (LDA)

Mathematik

- multivariate Diskriminanzanalyse nach R. Fisher (1936)
  - Ziel: maximale Variabilität zwischen Klassen bei minimaler Variabilität innerhalb jeder Klasse

$$\max \frac{V^2_{K_1 K_2}}{V^2_{K_1} + V^2_{K_2}} \;=\; \max \frac{(\hat{\mu}_1 - \hat{\mu}_2)^2}{V^2_{K_1} + V^2_{K_2}}$$

  - Betrachtung Zähler

$$(\hat{\mu}_1 - \hat{\mu}_2)^2 \;=\; (\vec{\mathbf{w}}^T \mu_1 - \vec{\mathbf{w}}^T \mu_2)(\vec{\mathbf{w}}^T \mu_1 - \vec{\mathbf{w}}^T \mu_2)^T \;=\; \vec{\mathbf{w}}^T (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T \vec{\mathbf{w}} \;=:\; \vec{\mathbf{w}}^T S_B \, \vec{\mathbf{w}}$$

  - Betrachtung Nenner ($\hat{x}_i = \vec{\mathbf{w}}^T x_i$)

$$V^2_K \;=\; \sum (\hat{x}_i - \hat{\mu})(\hat{x}_i - \hat{\mu})^T \;=\; \sum (\vec{\mathbf{w}}^T x_i - \vec{\mathbf{w}}^T \mu)(\vec{\mathbf{w}}^T x_i - \vec{\mathbf{w}}^T \mu)^T \;=\; \sum \vec{\mathbf{w}}^T (x_i - \mu)(x_i - \mu)^T \vec{\mathbf{w}} \;=:\; \vec{\mathbf{w}}^T \Sigma \, \vec{\mathbf{w}}$$

$$\Rightarrow\; V^2_{K_1} + V^2_{K_2} \;=\; \vec{\mathbf{w}}^T \Sigma_1 \vec{\mathbf{w}} + \vec{\mathbf{w}}^T \Sigma_2 \vec{\mathbf{w}} \;=\; \vec{\mathbf{w}}^T (\Sigma_1 + \Sigma_2) \vec{\mathbf{w}} \;=:\; \vec{\mathbf{w}}^T S_W \, \vec{\mathbf{w}}$$

Linear Discriminant Analysis (LDA)

- multivariate Diskriminanzanalyse nach R. Fisher (1936)
  - damit folgt schliesslich ein Optimierungsproblem ($\Rightarrow$ `fit()`)

$$\vec{\mathbf{w}} \; := \; \arg\max \; \frac{\vec{\mathbf{w}}^T S_B \, \vec{\mathbf{w}}}{\vec{\mathbf{w}}^T S_W \, \vec{\mathbf{w}}}$$

mit

$$S_B = (\mu_1 - \mu_2)(\mu_1 - \mu_2)^T$$

$$S_W = \Sigma_1 + \Sigma_2$$

  - unnützes Wissen: Lösung des Optimierungsproblems mithilfe eines Lagrange-Multiplikators

$$\mathcal{L} = \vec{\mathbf{w}}^T S_B \, \vec{\mathbf{w}} - \lambda(\vec{\mathbf{w}}^T S_W \, \vec{\mathbf{w}} - 1) \quad \text{mit} \quad \frac{\partial \mathcal{L}}{\partial \vec{\mathbf{w}}} = 0$$

Linear Discriminant Analysis (LDA)

- multivariate Diskriminanzanalyse nach R. Fisher (1936)
  - was bei mehr als zwei Klassen…?
  - für k Klassen sei $N = n_1 + n_2 + \ldots + n_k$ die Gesamtanzahl aller Datenpunkte

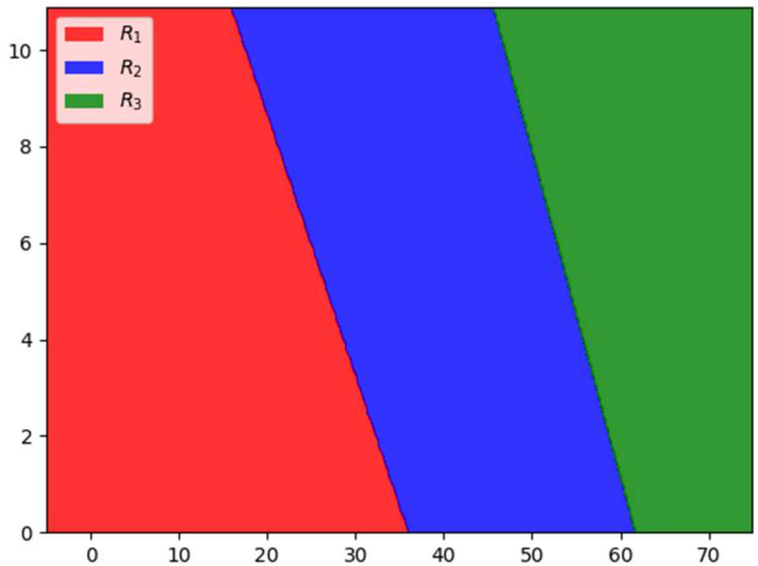
aus $\mu_k = \frac{1}{n_k} \sum_{x_i \in K_k} x_i$ mit $\mu = \frac{1}{N} \sum_{i=1}^{N} x_i$ folgt $S_B = \frac{1}{N} \sum_{i=1}^{k} n_i (\mu_i - \mu)(\mu_i - \mu)^T$

$$S_W = \Sigma_1 + \Sigma_2 + \ldots + \Sigma_k$$

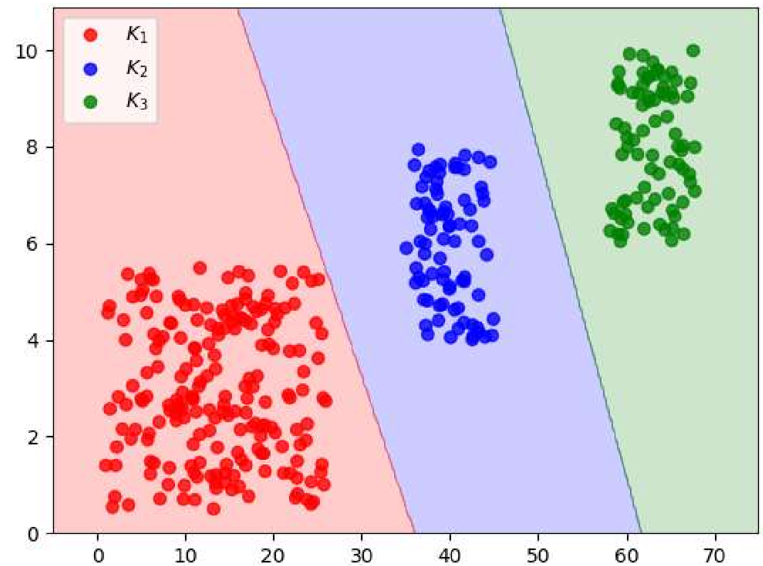
  - und damit $\vec{\mathbf{w}} = eig(S_W^{-1} S_B)$

Linear Discriminant Analysis (LDA)

- multivariate Diskriminanzanalyse nach R. Fisher (1936)
  - mit optimaler Projektionsebene (aufgespannt durch $\vec{w}$) ist auch Diskriminanzfunktion bekannt $\rightarrow$ Klassifikation durch Zuordnung: Punkt x gehört zu Klasse K_i falls x in Region R_i
  - Beispiel: Datei 'clicks.dat'



$\Rightarrow$



Praktische Übung

- Beispiel: Datensatz **`wine`** aus Paket **`sklearn.datasets`**
  - Datensatz enthält 178 Samples in 13 Dimensionen mit 3 Klassen

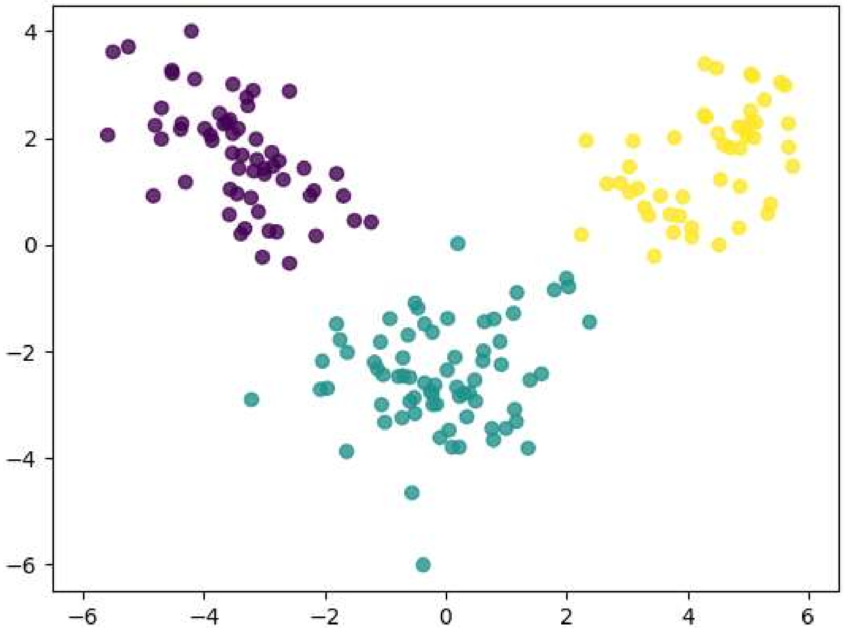
```
wine = datasets.load_wine()
x = wine.data
y_true = wine.target
```

  - Diskriminanzanalyse mit Projektion (**`transform`**) & Klassifikation (**`predict`**) der Daten

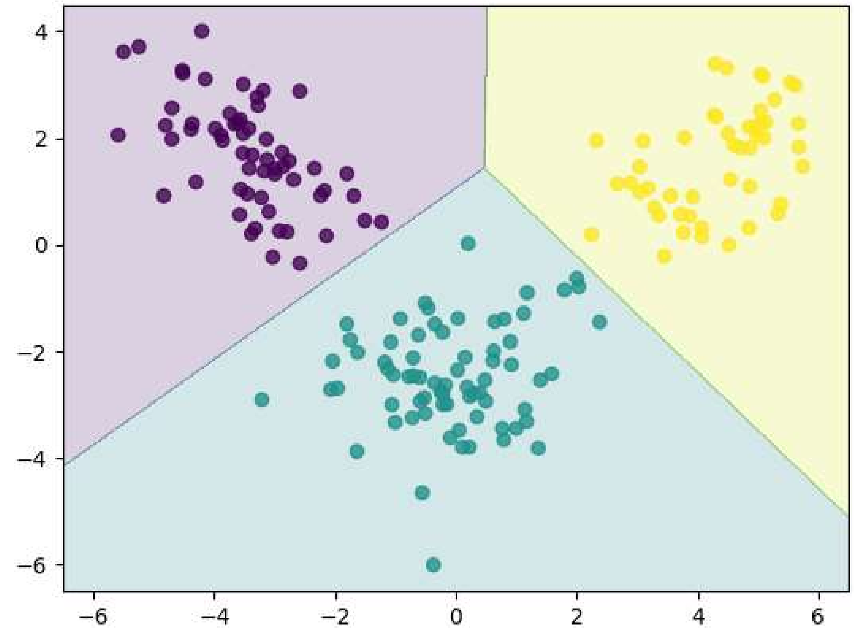
```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
model = LinearDiscriminantAnalysis(n_components=2)
model.fit(x, y_true)
x_proj = model.transform(x)
y_pred = model.predict(x)
```

Praktische Übung

- Beispiel: Datensatz **`wine`** aus Paket **`sklearn.datasets`**
  - Darstellung des projizierten Datensatzes (2D)



Scatterplot mit $c = y\_true$

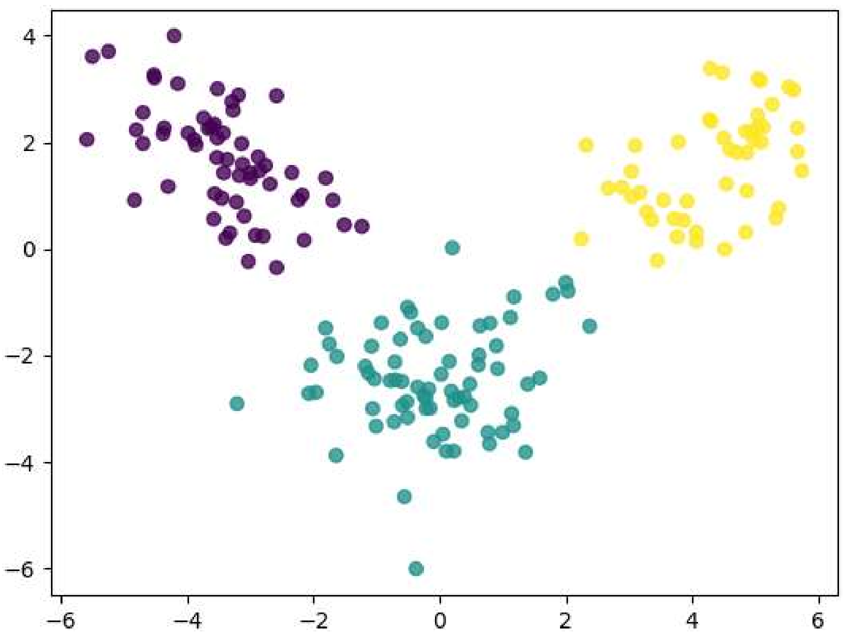


Scatterplot mit $c = y\_pred$

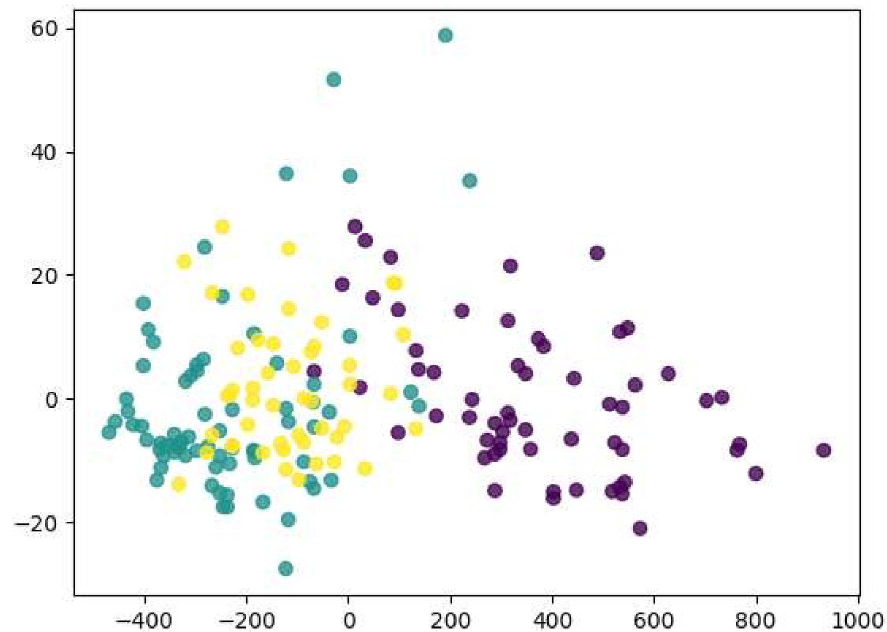
#Originalklassifierung, wie sie richtig ist (2D) `Vorhersage Klassifizierung`

Praktische Übung

- Beispiel: Datensatz **`wine`** aus Paket **`sklearn.datasets`**
  - Vergleich LDA und PCA (in zwei Dimensionen)



LDA



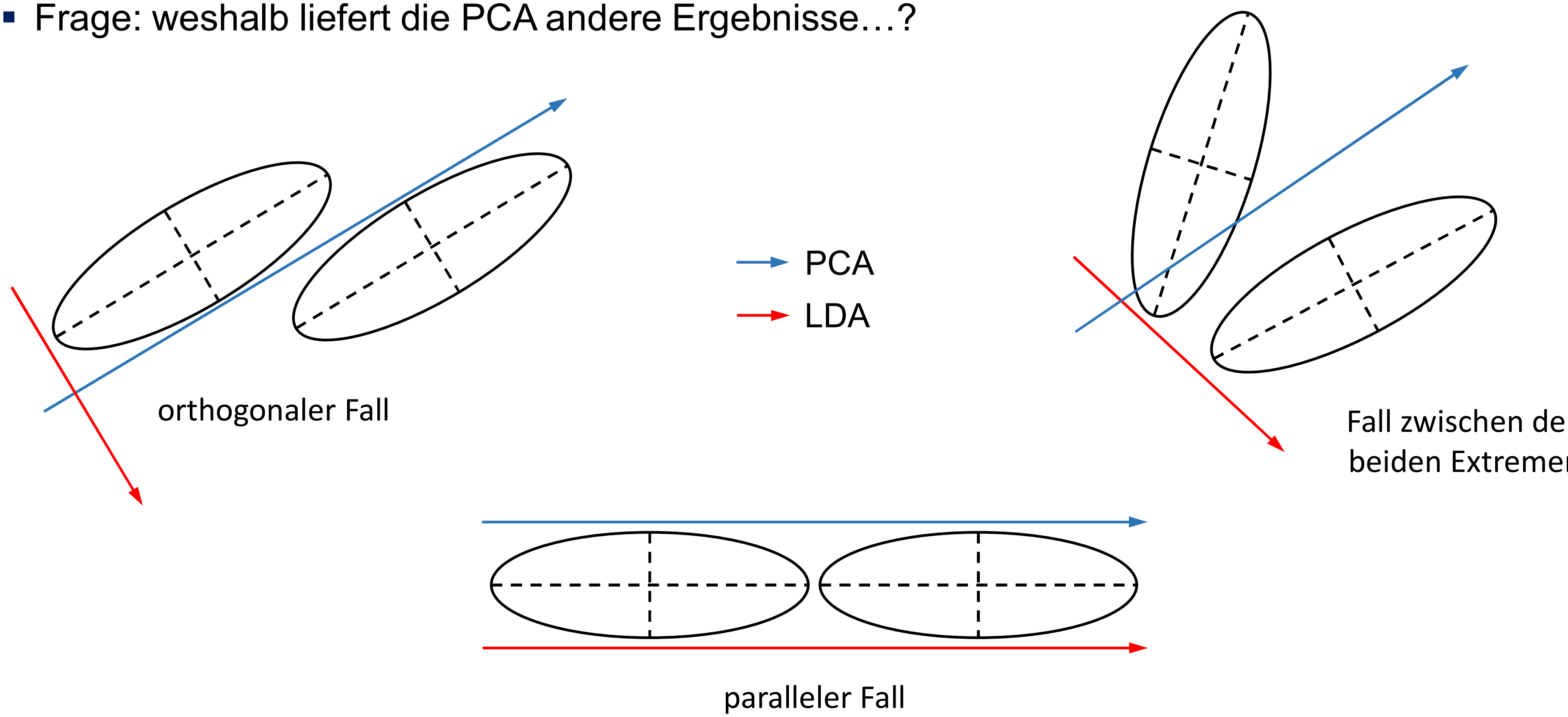
PCA


```
Varianz / Streuung minimieren

aus Originaldaten nur Klassen in
dimensionalen Raum möglich getrennt
```

```
Varianz zu erhalten (Streuung erhalten)

Originaldaten möglichst behalten in einem
reduzierten Raum
```

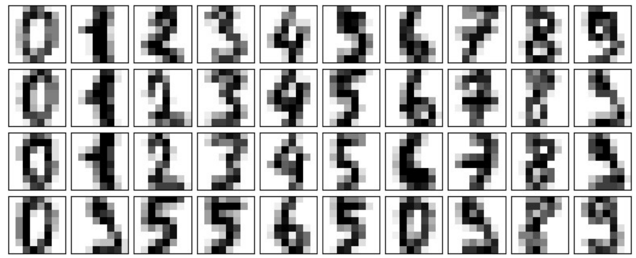
Praktische Übung

- Frage: weshalb liefert die PCA andere Ergebnisse…?



Praktische Übung

- Datensatz: handschriftlicher Ziffern (Paket **`sklearn`**)
  - Datensatz enthält 1797 Bilder mit Auflösung 8×8



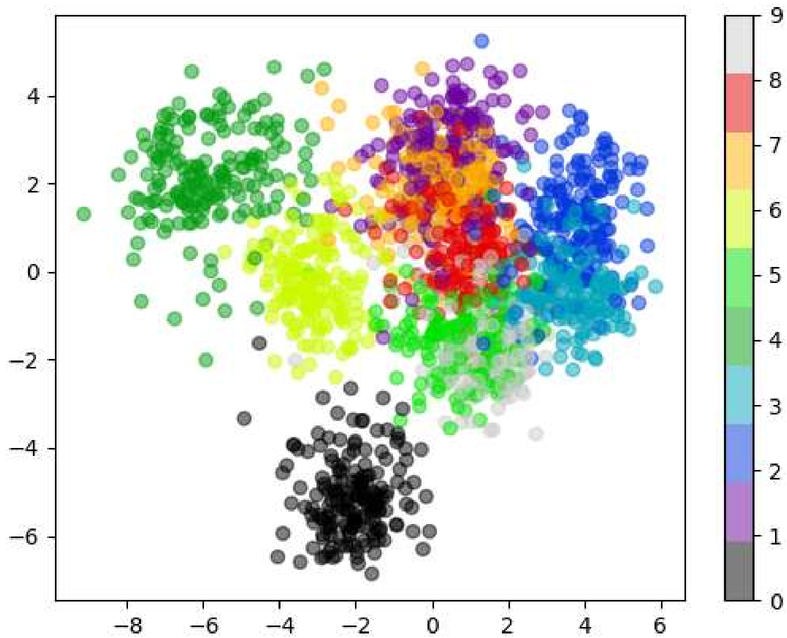
```
digits = datasets.load_digits()
x = digits.data
y_true = digits.target
```

  - Diskriminanzanalyse (Projektion auf 2D/3D)

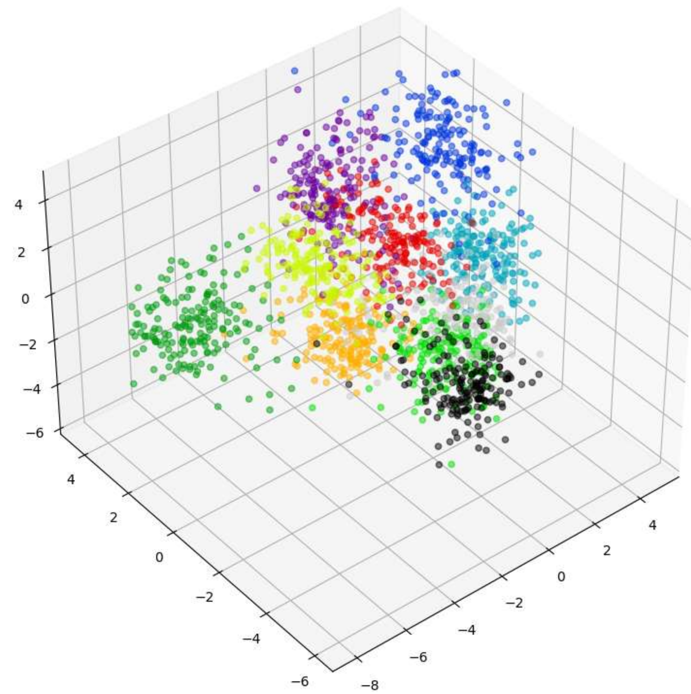
```
model = LinearDiscriminantAnalysis(n_components=2|3)
model.fit(x, y_true)
proj = model.transform(x)
y_pred = model.predict(x)
```

Praktische Übung

- Datensatz: handschriftlicher Ziffern (Paket **`sklearn`**)



$n\_components = 2$



$n\_components = 3$

Und noch ein kleiner Blick über den Tellerrand…

- Gesichtserkennung: Olivetti-Datensatz (Paket **`sklearn`**)
  - Bilder (Grösse 64×64) von 40 Personen mit jeweils 10 Gesichtsausdrücken

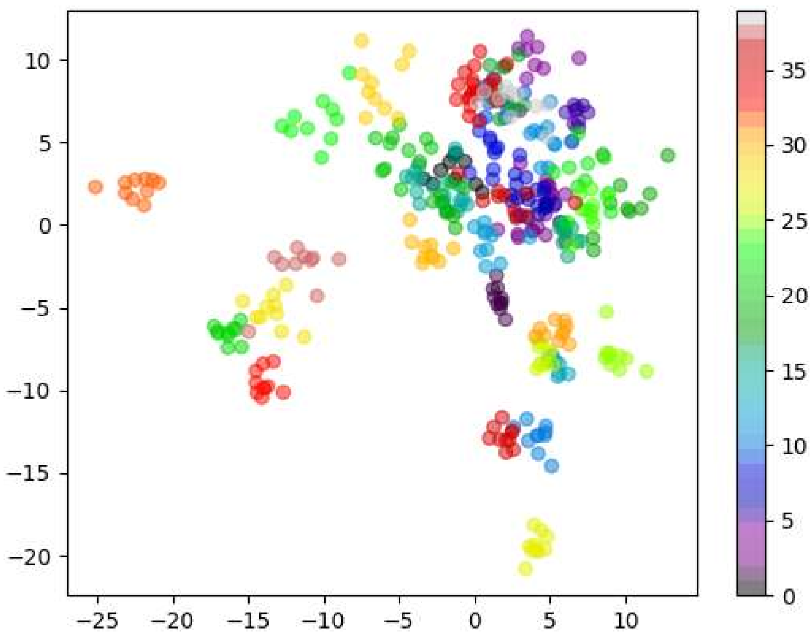
```
faces = datasets.fetch_olivetti_faces()
x = faces.data
y_true = faces.target
```

  - Bilder anzeigen (Datei **`plot.py`**)

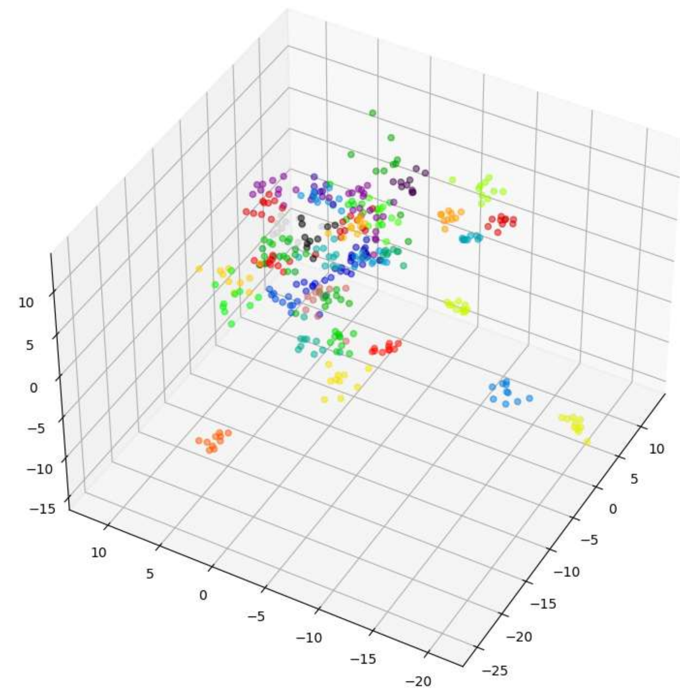
```
from plot import plot_faces
plot_faces(faces.images, 4, 6)
```

Und noch ein kleiner Blick über den Tellerrand…

- Gesichtserkennung: Olivetti-Datensatz (Paket **`sklearn`**)
  - Diskriminanzanalyse (Projektion auf 2D/3D)



$n\_components = 2$

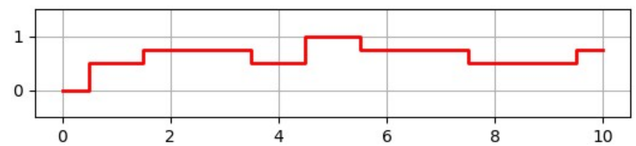


$n\_components = 3$

Und noch ein kleiner Blick über den Tellerrand…

- Gesichtserkennung: Olivetti-Datensatz (Paket **`sklearn`**)
  - Aufgabe
    - Aufteilung von **`x`** und **`y_true`** in Training- und Testdaten (4 Bilder $\rightarrow$ 1%)
    - **`fit(`**$n\_components = 2, 3, 5$**`)`** und **`transform()`** der Trainingsdaten mittels LDA
    - **`transform()`** der Testdaten mittels LDA
    - Anlernen (**`fit()`**) eines *k-NN* auf den projizierten Trainingsdaten für $k \in [1,10]$ sowie Klassifikation (**`predict()`**) mittels projizierter Testdaten
    - Bestimmung und Ausgabe der Treffsicherheit (**`accuracy_score()`**)
  - Ergebnis

$n\_components = 2$



$n\_components = 3$

$n\_components = 5$

Und noch ein kleiner Blick über den Tellerrand…

- Gesichtserkennung: Olivetti-Datensatz (Paket **`sklearn`**)
  - Rücktransformation der projizierten Trainingsdaten (sogenannte *Fisherfaces*)

n_components = 39

Fragen…?