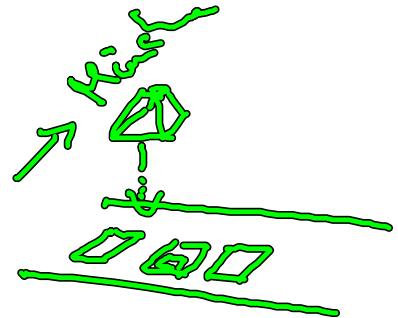


# Mustererkennung

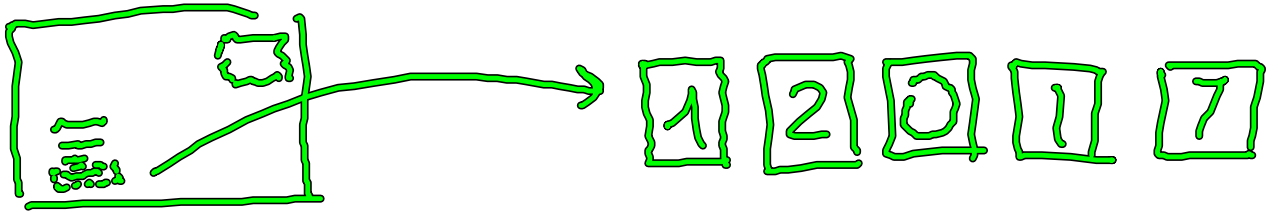
Gesichter, Schilder, ... viele Anwendungsgebiete



Bsp: Briefe → Postleitzahlen



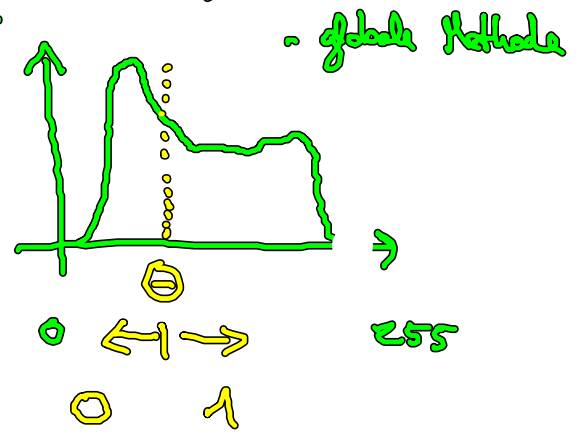
Vorverarbeitungsschritte:



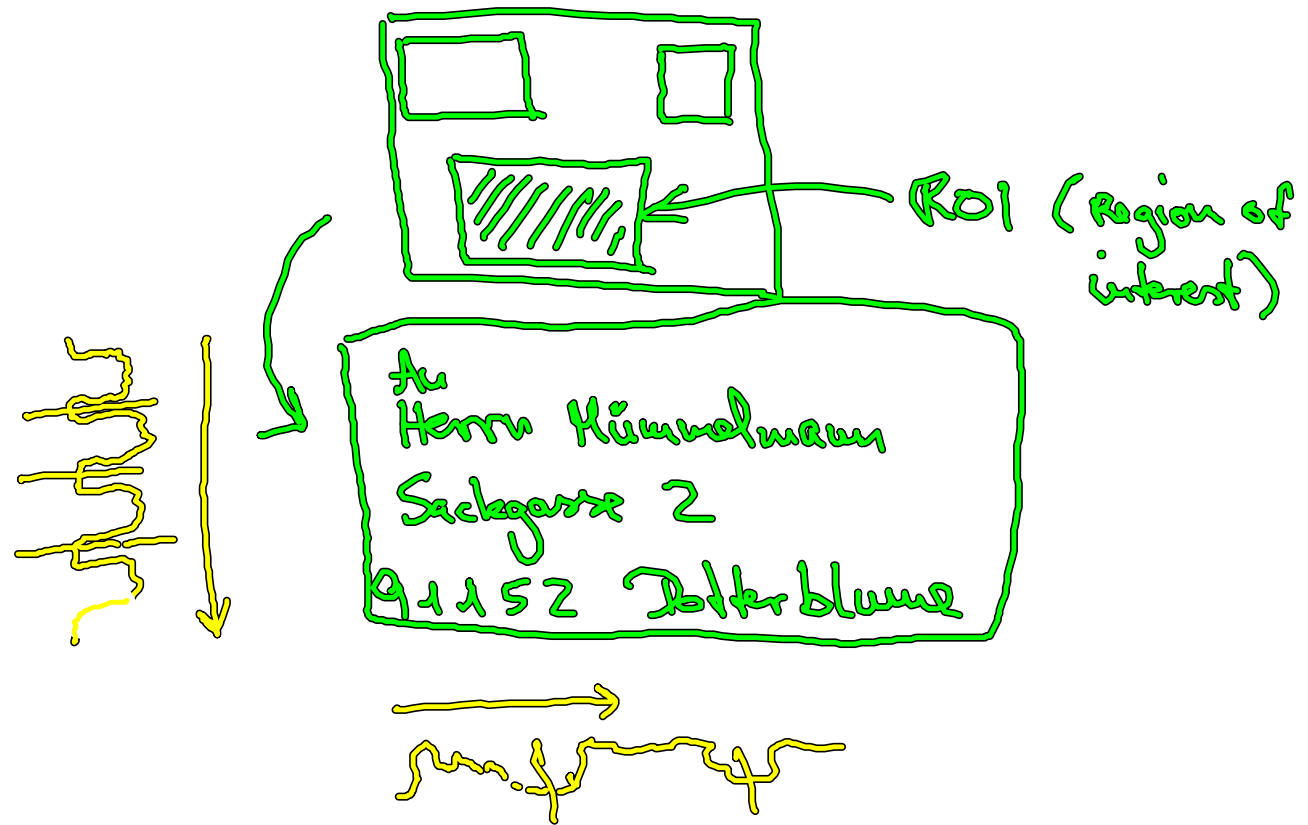
„Graubild“

- Binarisierung:

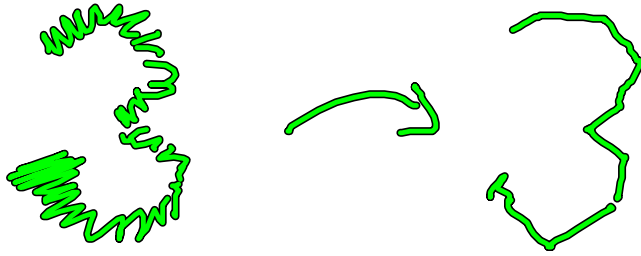
Histogramm



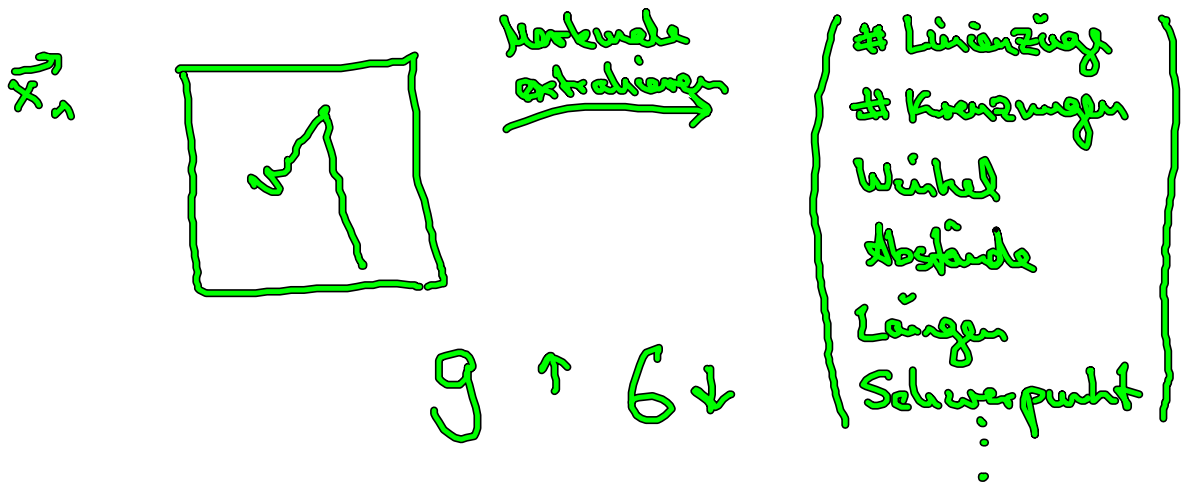
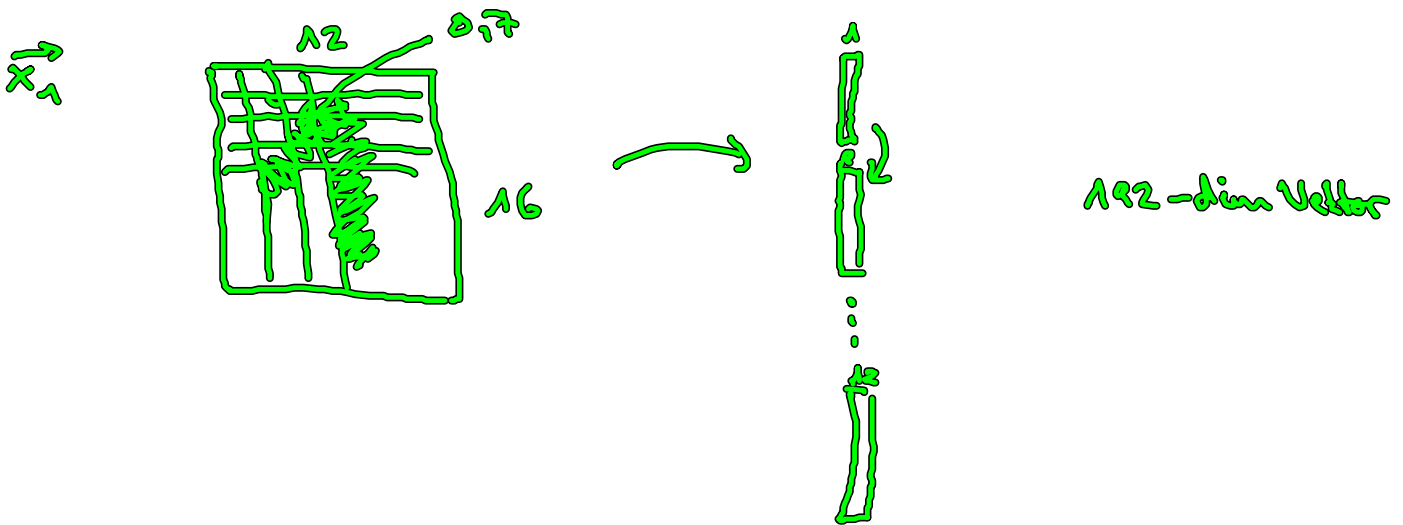
- Textur Analyse:

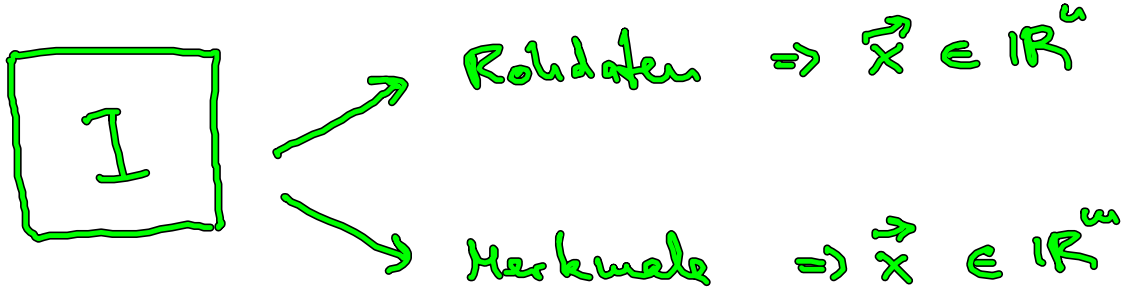


- Skelettierung



gaben: vorverarbeitete Ziffern  $\vec{x}_1, \dots, \vec{x}_n$   
 Trainingdatenbank

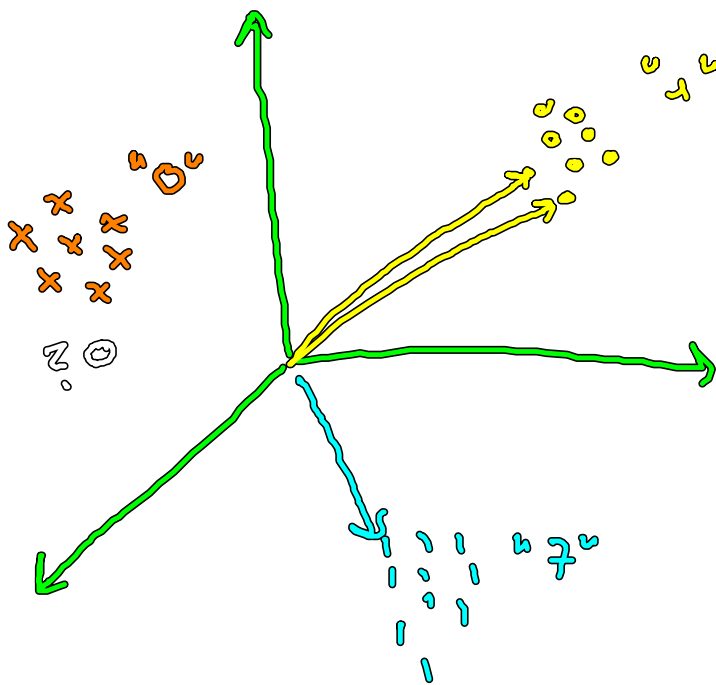




$m \ll n$

Um ein intelligentes System zu bauen benötigen wir bekannte Daten (Trainingsmenge) inklusive des Klassentyps.

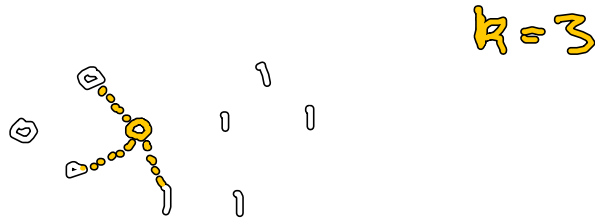
d.h.  $(\vec{x}_1, "0")$ ,  $(\vec{x}_2, "5")$ , ...



Raum der Daten  
 $0, 1, \dots, 9$

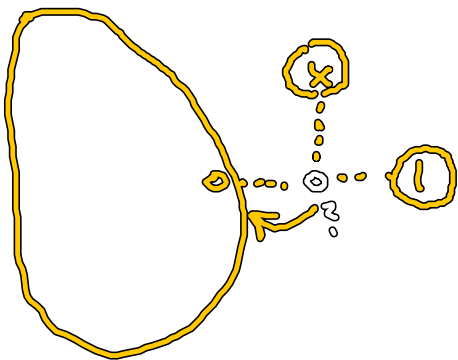
# k-nächste Nachbarn

gesucht ist die Klassenzugehörigkeit von  $\vec{y}$   
wir betrachten alle  $\vec{x}_1, \dots, \vec{x}_n$  und berechnen  
den Abstand zu  $\vec{y}$



Klasse von  $\vec{y}$  = Klasse von nächsten  
Nachbarn

= k-nächste Nachbarn (Mehrheitsentscheid)

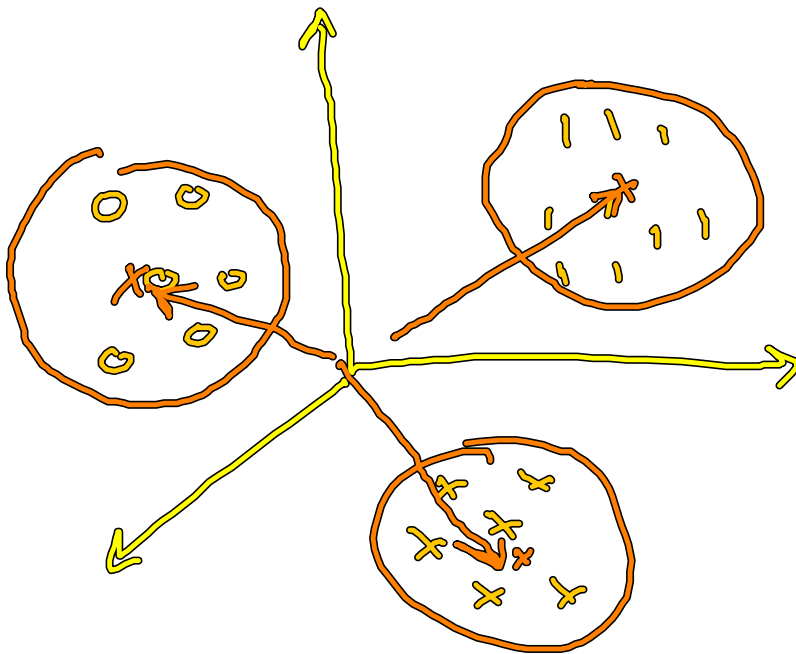


Nachteil: Laufzeit und Qualität abhängig von den Daten

## Clustering

„ nicht überwachtes Lernen“

IDEE: Daten werden zunächst nach Ähnlichkeiten zusammengefügt und anschließend entschieden, welche Klasse damit repräsentiert wird



# LBG-Algorithmus (k-means)

Daten:  $\vec{x}_1, \dots, \vec{x}_n \in \mathbb{R}^m$  gehören zu  $k$  Klassen  
gesucht sind  $k$  Prototypen ( $\vec{p}_1, \dots, \vec{p}_k$ )  
↑ (es ist oft besser die # Prototypen  
>  $k$  zu wählen)

0) Initialisierung:

setze  $\vec{p}_1, \dots, \vec{p}_k$  zufällig  
(auf  $k$  zufällige  
Daten verteilen)

1) Expectation-Schritt

setze  $\vec{x}_i \in$  Klasse ( $j$ ), falls  
 $\vec{x}_i$  am nächsten zu  $\vec{p}_j$  liegt

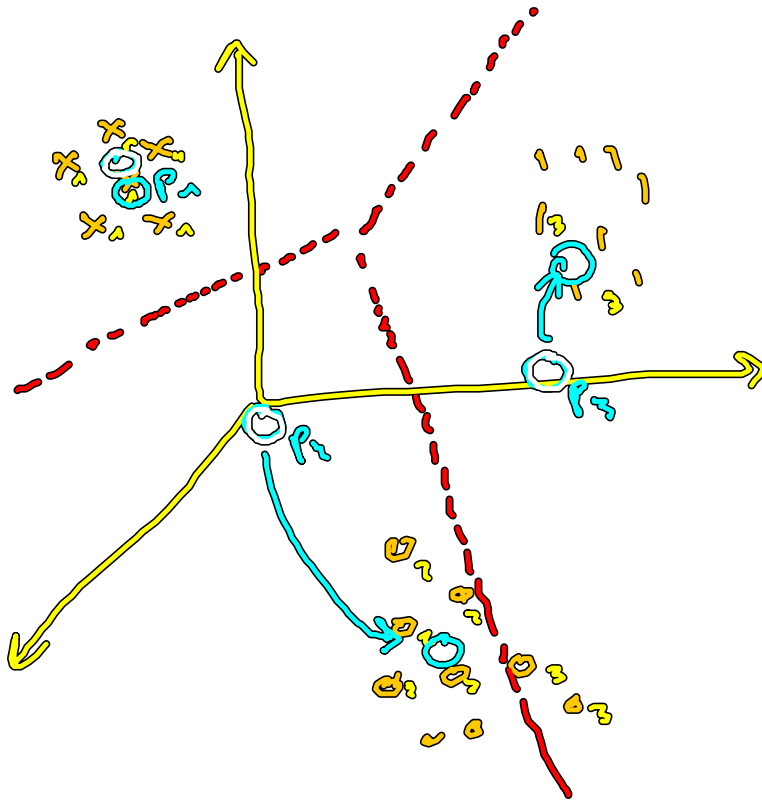
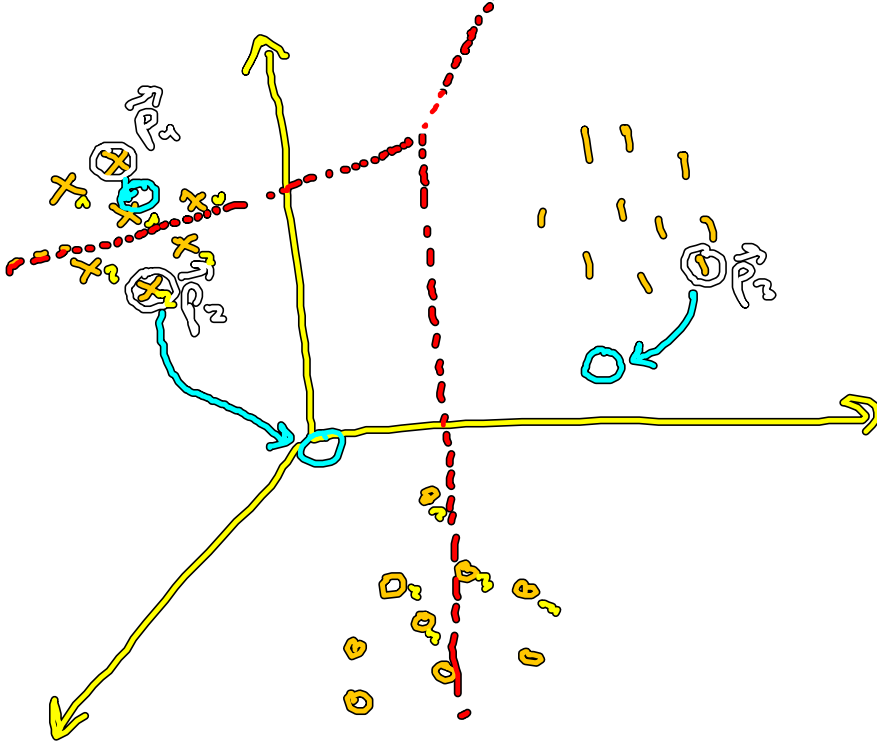
2) Maximization-Schritt

setze  $\vec{p}_i$  auf den Schwerpunkt  
der Klasse ( $i$ ) für  $i=1 \dots k$

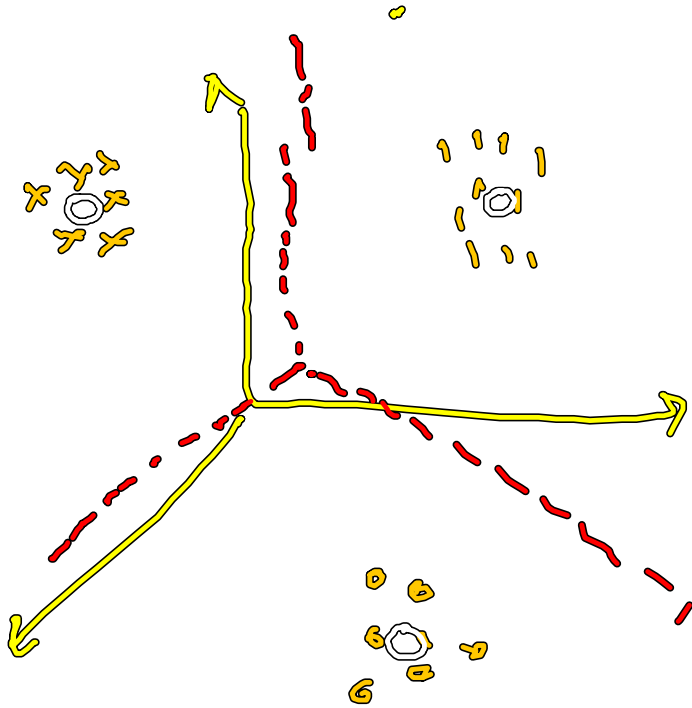
3) Abbruchkriterium

gehe zu (1) bis sich nichts  
mehr ändert

$P_1, P_2, P_3$

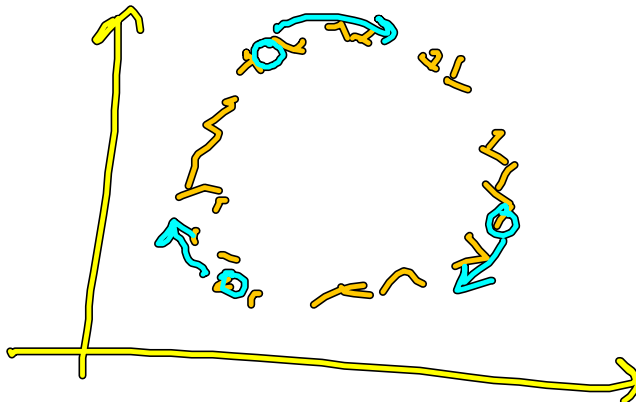
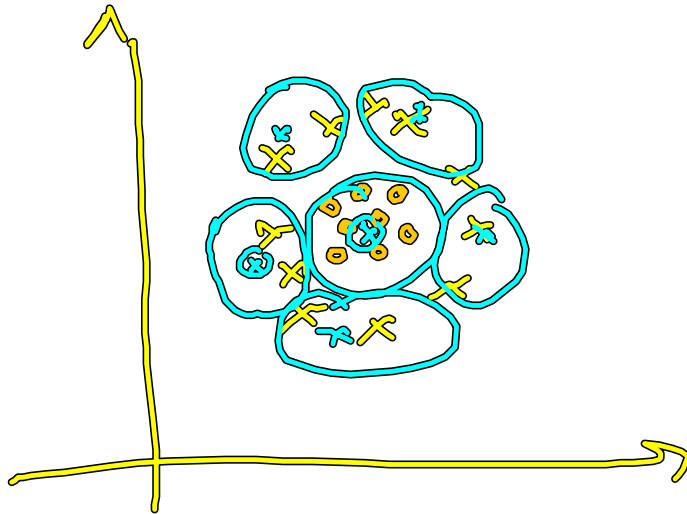






das wünschen wir uns

Probleme



4) Klassen der Prototypen bestimmen

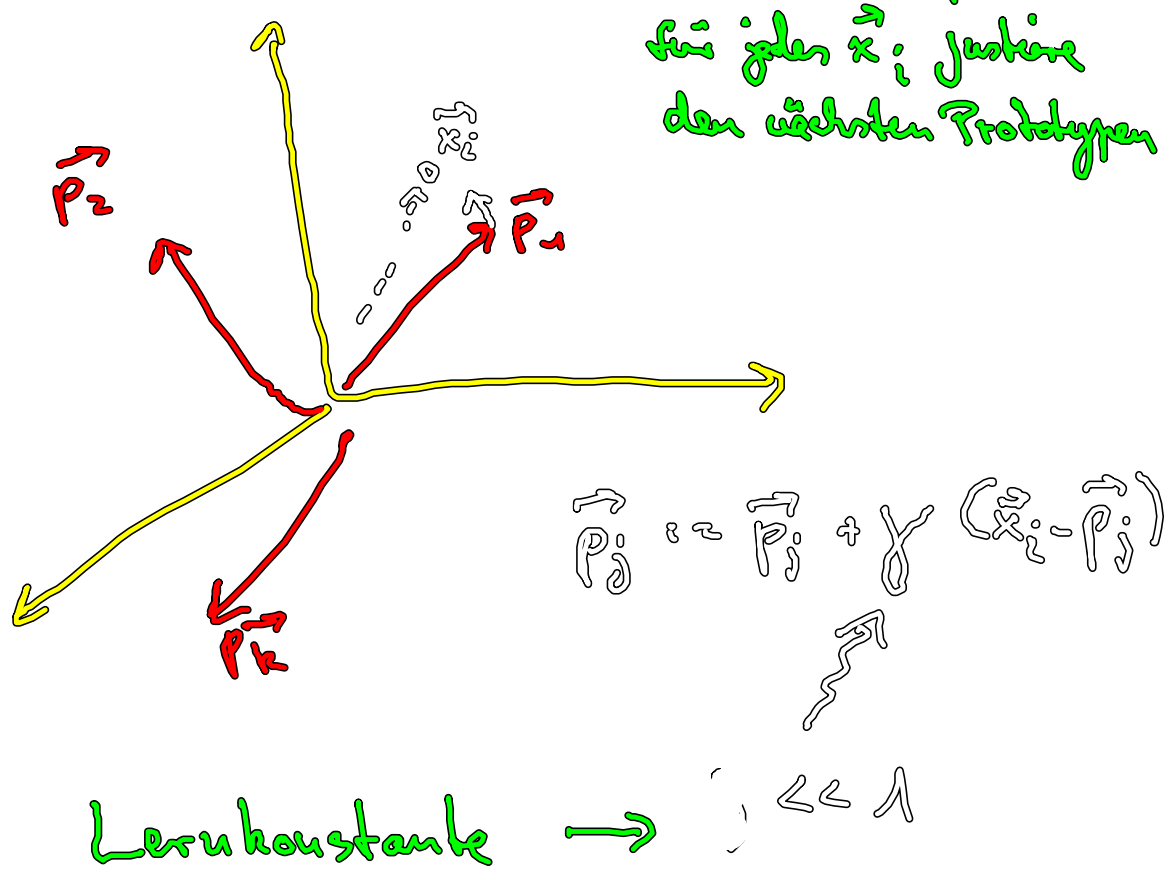
→ Mehrheitsentscheid

$\vec{p}_1, \dots, \vec{p}_k$  sind „gelobt“

Wenn wir nun ein Auftrags  $\vec{y}$  haben, berechnen wir die Abstände zu den Prototypen und wählen den nächsten.

→ viel Zeit wird für Lernen benötigt  
aber die Erkennung ist abschließend  
wesentlich schneller als k-means.

Online - Clustering



## Zwei Varianten von k-means:

„variabler k-means“

Daten:  $\vec{x}_1, \dots, \vec{x}_n \in \mathbb{R}^d$  gehören zu  $k$  Klassen  
 gesucht  $l$  Prototypen  $(\vec{p}_1, \dots, \vec{p}_l)$  und  $l$   
 mit bestmöglicher Erkennungsrate (Qualität  
 versus  
 Effizienz?)

$$l = k - 1$$

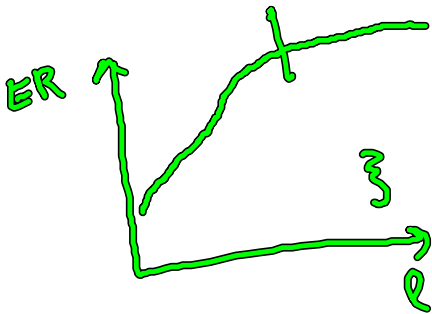
while (qualitätsausreißend) { # Clusters

l++;

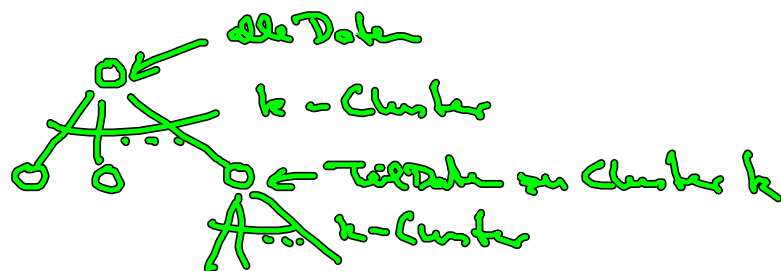
$(\vec{p}_1, \dots, \vec{p}_k) \leftarrow k\text{-means}(\vec{x}_1, \dots, \vec{x}_n, l)$

ER  $\leftarrow$  prüfe  $\vec{x}_1, \dots, \vec{x}_n$  mit  $\vec{p}_1, \dots, \vec{p}_k$

wenn ER sich nicht mehr  
signifikant ändert  $\rightarrow$  qualitätsausreißend



"rekursives k-means"



Daten:  $\vec{x}_1, \dots, \vec{x}_n \in \mathbb{R}^E$

festes  $l, t$  (Tiefe)

$k\text{-means-rec}(\vec{x}_1, \dots, \vec{x}_n, l, t)$

if ( $t=0$ )

return

(0)

(1)

(2)

(3)

k-means

for  $i=1$  to  $Q$

$k$ -means-rec(Daten von  $\vec{p}_i, Q, t-1$ )