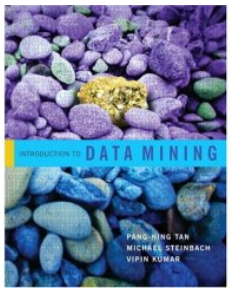
Informationssysteme

Sommersemester 2016

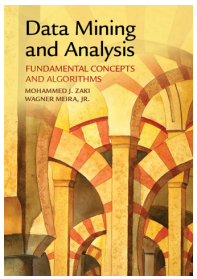
Prof. Dr.-Ing. Sebastian Michel
TU Kaiserslautern

smichel@cs.uni-kl.de

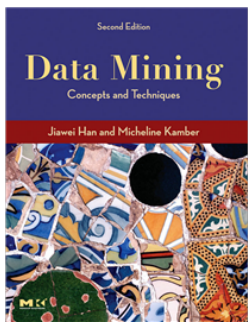
Literatur zu Data-Mining



Pang-Ning Tan, Michael Steinbach, Vipin Kuma. Introduction to Data Minig. Ein paar relevante Kapitel sind frei verfügbar unter
http://www-users.cs.umn.edu/ kumar/dmbook/index



Mohammed J. Zaki, Wagner Meira Jr. Data Mining and Analysis.
http://www.dataminingbook.info



Jiawei Han, Micheline Kamber. Data Mining. Concepts and Techniques.

Warenkorbanalyse

Objekte sind: Brot, Milch, Windeln, Bier, Eier
Transaktionen sind: 1:{Brot, Milch}, 2:{Brot, Windeln, Bier, Eier}, 3:{Milch, Windeln, Bier}, 4:{Brot, Milch, Windeln, Bier} und 5:{Brot, Milch, Windeln}

TID	Brot	Milch	Windeln	Bier	Eier
1	1	1			
2	1		1	1	1
3		1	1	1	
4	1	1	1	1	
5	1	1	1		

Welche Objekte (Items) werden häufig zusammen gekauft?
Können wir Regeln angeben der Form: Kunden die Windeln kaufen kaufen auch meist Bier?

Darstellung als Binärmatrix

TID	Brot	Milch	Windeln	Bier	Eier
1	1	1	0	0	0
2	1	0	1	1	1
3	0	1	1	1	0
4	1	1	1	1	0
5	1	1	1	0	0

Itemsets

{Brot, Milch}
{Brot, Windeln, Bier, Eier}
{Milch, Windeln, Bier}
{Brot, Milch, Windeln, Bier}
{Brot, Milch, Windeln}

- **Ein Itemset ist eine Menge von Objekten**
  - Eine **Transaktion** t ist ein Itemset mit dazugehöriger Transaktions ID, $t = (tid, I)$ wobei I das Itemset der Transaktion ist
- Eine Transaktion $t = (tid, I)$ **enthält ein Itemset** X falls $X \subseteq I$
- **Der Support von Itemset** X in einer Datenbank D ist die Anzahl der Transaktionen in D, die X enthalten:

$$\text{supp}(X, D) = |\{t \in D : t \text{ enthält } X\}|$$

- Die **relative Häufigkeit** von Itemset X in Datenbank D ist der Support relativ zur Größe der Datenbank, $\text{supp}(X,D)/|D|$
- **Ein Itemset ist häufig (frequent), falls** dessen relative Häufigkeit über einem bestimmten Schwellwert **minfreq** liegt.
- Alternativ kann man auch einen Schwellwert **minsupp** bzgl. des Supports betrachten.

Beispiel

TID	Brot	Milch	Windeln	Bier	Eier
1	1	1	0	0	0
2	1	0	1	1	1
3	0	1	1	1	0
4	1	1	1	1	0
5	1	1	1	0	0

Itemset {Brot, Milch} hat Support 3 und relative Häufigkeit $3/5$
Itemset {Brot, Milch, Eier} hat Support und relative Häufigkeit 0.

Für minfreq = 1/2 haben wir die folgenden frequent itemsets:
{Brot}, {Milch}, {Windeln}, {Bier}, {Brot, Milch}, {Brot, Windeln}, {Milch, Windeln} und {Windeln, Bier}.

Assoziationsregeln und Konfidenz

- Eine **Assoziationsregel** ist eine Regel der Form $X \rightarrow Y$, wobei X und Y disjunkte Itemsets sind (d.h. $X \cap Y = \emptyset$)
- **Idee:** Eine Transaktion, die Itemset X enthält, enthält (vermutlich) auch Itemset Y
- Der **Support einer Regel** $X \rightarrow Y$ in Datenbank D ist

$$\text{supp}(X \rightarrow Y, D) = \text{supp}(X \cup Y, D)$$

- Die **Konfidenz der Regel** $X \rightarrow Y$ in Datenbank D ist

$$\text{conf}(X \rightarrow Y, D) = \text{supp}(X \cup Y, D)/\text{supp}(X,D)$$

Mit anderen Worten: Die Konfidenz ist die bedingte Wahrscheinlichkeit, dass eine Transaktion Y enthält, wenn sie X enthält.

Beispiel

TID	Brot	Milch	Windeln	Bier	Eier
1	1	1	0	0	0
2	1	0	1	1	1
3	0	1	1	1	0
4	1	1	1	1	0
5	1	1	1	0	0

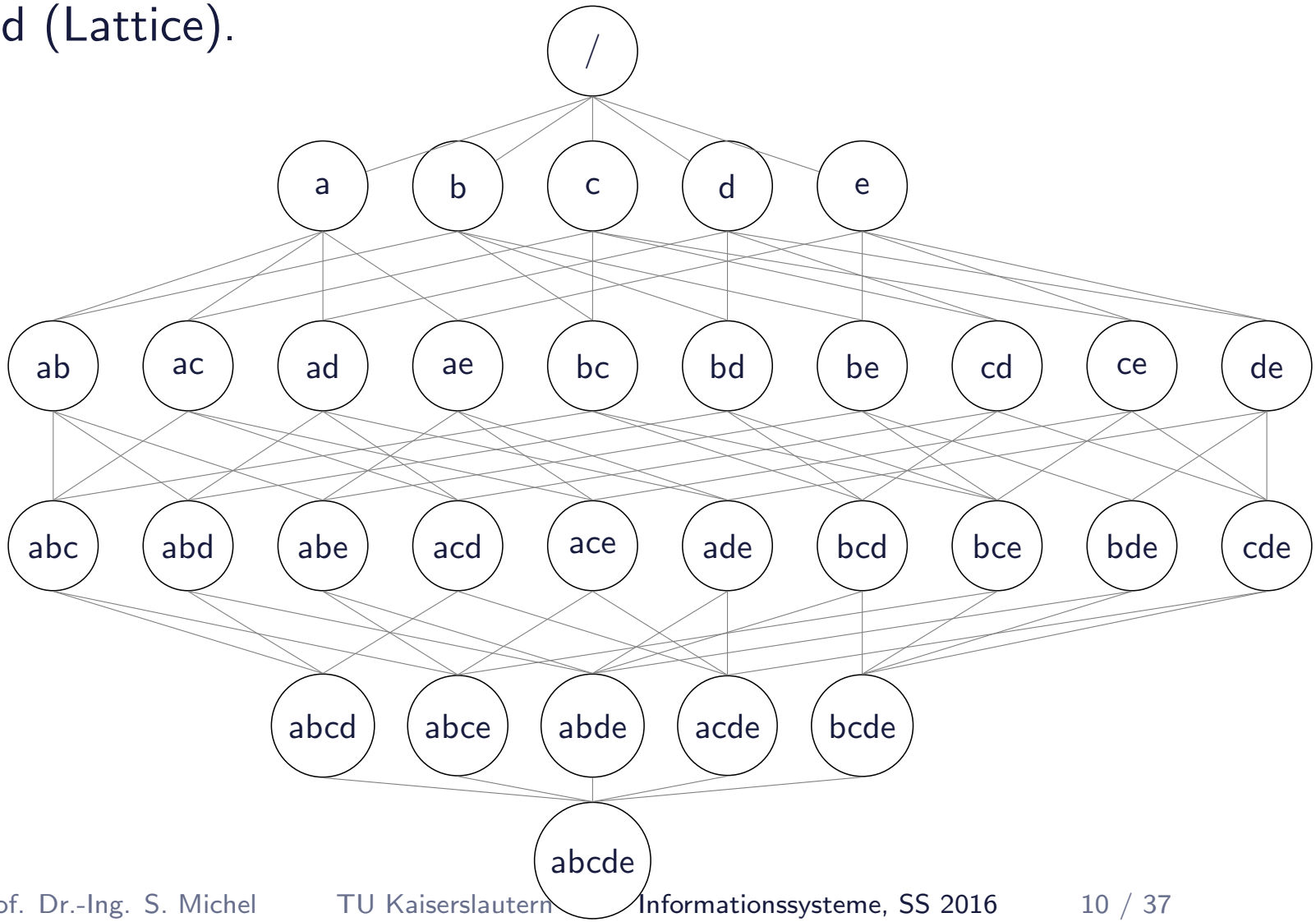
$\{\textbf{Brot, Milch}\} \rightarrow \{\textbf{Windeln}\}$ hat Support 2 und Konfidenz 2/3

$\{\textbf{Windeln}\} \rightarrow \{\textbf{Brot, Milch}\}$ hat Support 2 und Konfidenz 1/2

$\{\textbf{Eier}\} \rightarrow \{\textbf{Brot, Windeln, Bier}\}$ hat Support 1 und Konfidenz 1

Mögliche Itemset

- Was sind mögliche Itemset?
- Hier alle Itemsets für die Items {a,b,c,d,e} in der Darstellung als Verband (Lattice).



Ein naiver Algorithms

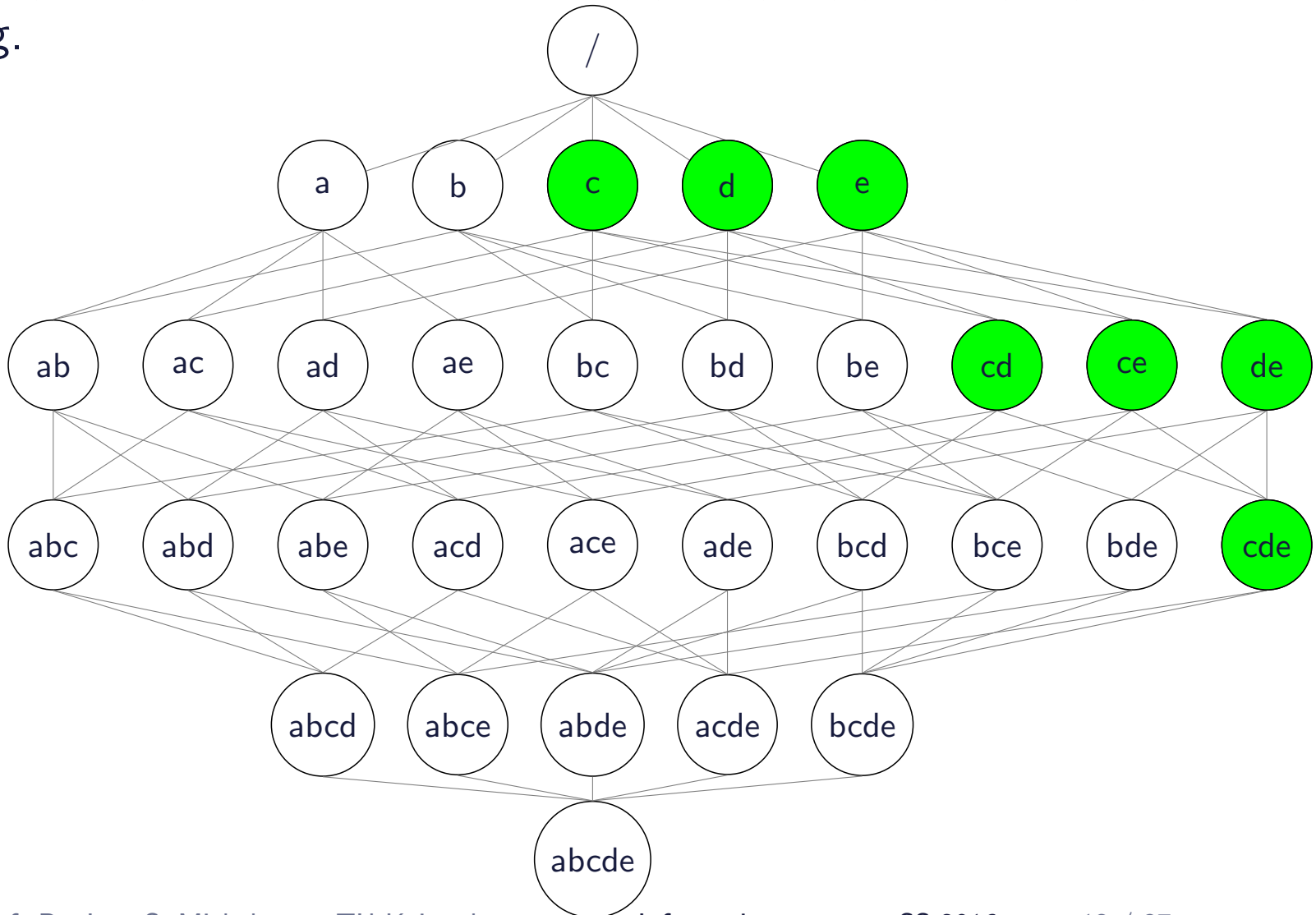
- **Betrachte jedes mögliche Itemset und teste ob es häufig ist.**
- **Wie berechnet man den Support?**

  Zähle für jedes Itemset in welchen Transaktionen es enthalten ist
- Berechnen des Support dauert $O(|I| \times |D|)$ und es gibt $2^{|I|}$ mögliche Itemsets, also im Worstcase: $O(|I| \times |D| \times 2^{|I|})$

Das Apriori-Prinzip

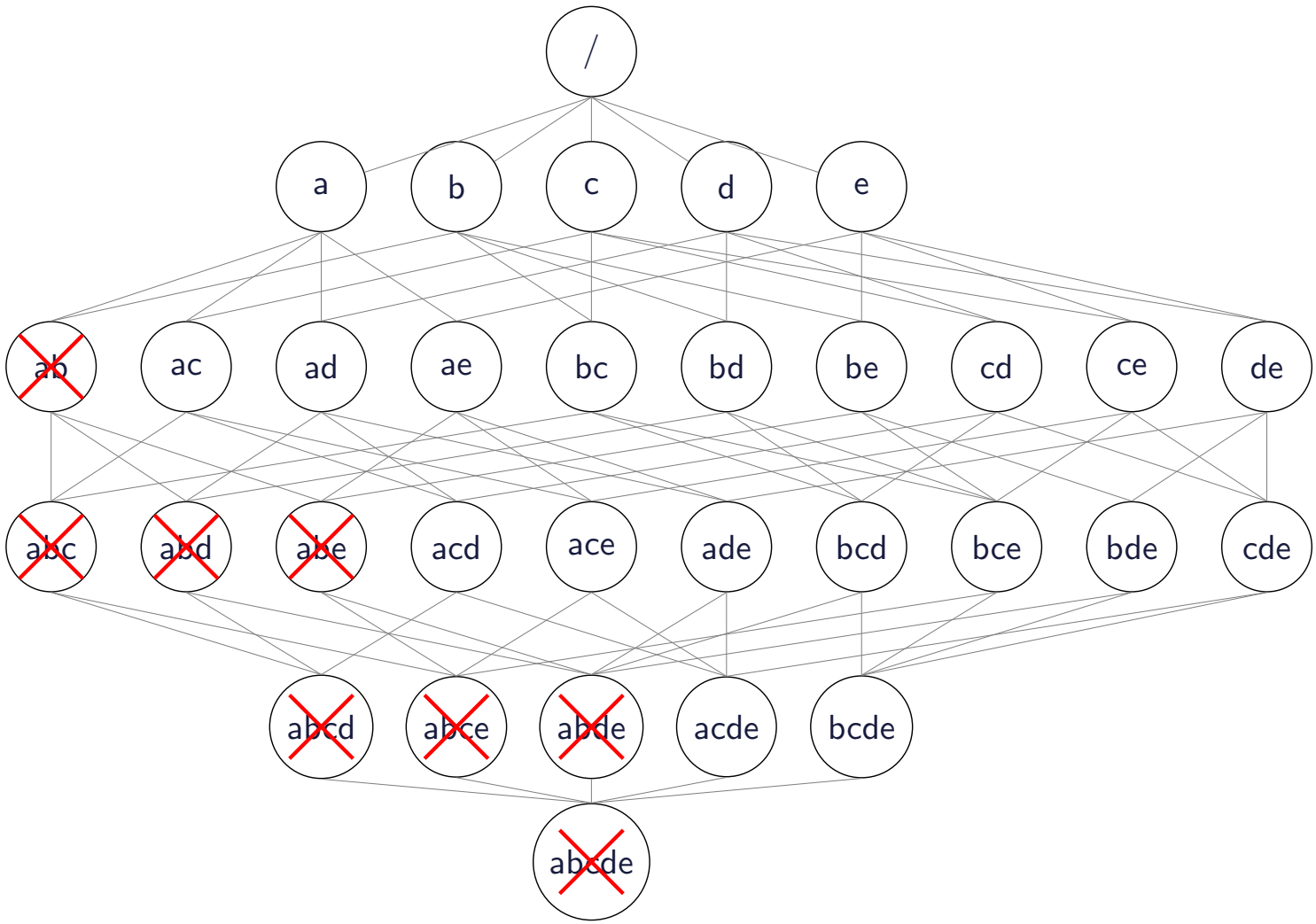
Falls ein Itemset häufig ist, so sind alle Teilmengen ebenfalls häufig.

Beispiel: Ist {c,d,e} häufig, so sind auch {c}, {d}, {e}, {c,d}, {c,e} und {d,e} häufig.



Das Apriori-Prinzip

Umgekehrt: falls {a,b} nicht häufig ist (Englisch: infrequent), so sind auch alle Supermengen von {a,b} nicht häufig.



Anti-Monotonie

Sei I eine Menge von Items und sei $J = 2^I$ die Potenzmenge von I. Ein Maß f ist **monoton** (**oder aufwärts geschlossen**) falls

$$\forall X,Y \in J : (X \subseteq Y) \Rightarrow f(X) \leq f(Y)$$

Im Gegensatz, f ist **anti-monoton** (oder abwärts geschlossen) falls

$$\forall X,Y \in J : (X \subseteq Y) \Rightarrow f(Y) \leq f(X)$$

Ist Support monoton oder anti-monoton?

Anti-Monotonie

Sei I eine Menge von Items und sei $J = 2^I$ die Potenzmenge von I. Ein Maß f ist **monoton (oder aufwärts geschlossen)** falls

$$\forall X,Y \in J : (X \subseteq Y) \Rightarrow f(X) \leq f(Y)$$

Im Gegensatz, f ist **anti-monoton** (oder abwärts geschlossen) falls

$$\forall X,Y \in J : (X \subseteq Y) \Rightarrow f(Y) \leq f(X)$$

Support ist anti-monoton:

Für Itemsets X und Y mit $X \subseteq Y$ gilt $\text{supp}(X) \geq \text{supp}(Y)$. D.h. wenn X nicht häufig ist (infrequent), dann sind auch alle Obermengen von X nicht häufig.

Beispiel

Minimum Support Schwellwert = 3

Kandidaten 1-Itemsets

Item	**Count**
Bier	3
Brot	4
Cola	2
Windeln	4
Eier	2
Milch	4

Beispiel

Minimum Support Schwellwert = 3

Rot markierte Itemsets sind unter Schwellwert und werden eliminiert.

Kandidaten 1-Itemsets

Item	Count
Bier	3
Brot	4
Cola	**2**
Windeln	4
Eier	**2**
Milch	4

Beispiel

Minimum Support Schwellwert = 3

Rot markierte Itemsets sind unter Schwellwert und werden eliminiert.

Kandidaten 1-Itemsets

Item	Count
Bier	3
Brot	4
Cola	**2**
Windeln	4
Eier	**2**
Milch	4

Kandidaten 2-Itemsets

Itemset	Count
{Bier,Brot}	2
{Bier, Windeln}	3
{Bier, Milch}	2
{Brot, Windeln}	3
{Brot, Milch}	3
{Windeln, Milch}	3

Beispiel

Minimum Support Schwellwert = 3

Rot markierte Itemsets sind unter Schwellwert und werden eliminiert.

Kandidaten 1-Itemsets

Item	Count
Bier	3
Brot	4
Cola	**2**
Windeln	4
Eier	**2**
Milch	4

Kandidaten 2-Itemsets

Itemset	Count
{Bier,Brot}	**2**
{Bier, Windeln}	3
{Bier, Milch}	**2**
{Brot, Windeln}	3
{Brot, Milch}	3
{Windeln, Milch}	3

Beispiel

Minimum Support Schwellwert = 3

Rot markierte Itemsets sind unter Schwellwert und werden eliminiert.

Kandidaten 1-Itemsets

Item	Count
Bier	3
Brot	4
Cola	**2**
Windeln	4
Eier	**2**
Milch	4

Kandidaten 2-Itemsets

Itemset	Count
{Bier,Brot}	**2**
{Bier, Windeln}	3
{Bier, Milch}	**2**
{Brot, Windeln}	3
{Brot, Milch}	3
{Windeln, Milch}	3

Kandidaten 3-Itemsets

Itemset	Count
{Brot,Windeln, Milch}	3

Der Apriori-Algorithms

Der **Apriori-Algorithmus benutzt die Anti-Monotonie des Support-Maßes**, um die Menge an zu betrachtenden Itemsets einzuschränken.

- **Apriori generiert niemals ein Kandidaten-Itemset, dass nicht-häufige Teilmengen besitzt.**

Der Apriori-Algorithms: Pseudocode

/* Notation: mit σ bezeichnen wir den Support eines Itemsets */

1. $k = 1$
2. $F_k = \{i \mid i \in I \wedge \sigma(\{i\}) \geq \text{minsupp}\}$ /*Häufige 1-Itemsets*/
3. **repeat**
4. $k = k + 1$
5. $C_k = \text{apriori-gen}(F_{k-1})$ /*Generiere Kandidaten*/
6. **for** each transaction $t \in T$ **do**
7. $C_t = subset(C_k,t)$ /*Betrachte Kandidaten die in TA*/
8. **for** each candidate itemset $c \in C_t$ **do**
9. $\sigma(c) = \sigma(c) + 1$ /*Erhöhe Support-Zähler*/
10. **end for**
11. **end for**
12. $F_k = \{c \mid c \in C_k \wedge \sigma(c) \geq \text{minsupp}\}$ /*Finde häufige Itemsets*/
13. **until** $F_k = \emptyset$
14. Result $= \bigcup F_k$

Der Apriori-Algorithms: Pseudocode (2)

- C_k ist die Menge der k-Itemsets
- F_k ist die Menge der häufigen k-Itemsets

Zuerst wird ein Mal über die Daten gelaufen, um den **Support jedes einzelnen Items** zu finden (Schritt 2). Dann kennen wir also F_1.

Danach werden iterativ **neue Kandidaten k-Itemsets berechnet, basierend auf den häufigen $(k-1)$-Itemsets** (Schritt 5). Die Methode dafür nennt man **apriori-gen**(....)

Nun wird **für jedes Kandidaten-Itemset der Support berechnet**, indem ein Mal über die Daten (Transaktionen) gelaufen wird (Schritt 6–10).

Anschließend werden nicht-häufige Itemsets entfernt (Schritt 12).

Der Algorithmus terminiert sobald $F_k = \emptyset$ (Schritt 13).

Generierung und Eliminierung von Kandidaten

1. Generierung von Kandidaten

Generiert k-Itemsets (d.h. Itemsets der Länge k) basierend auf den Itemsets der vorherigen Iteration(en).

2. Eliminierung von Kandidaten

Finde und eliminiere unnütze Kandidaten k-Itemsets.

Generierung von Kandidaten: Ziele

Ziele:

- **Vollständigkeit:** Es müssen alle häufigen k-Itemsets erzeugt werden.
- **Effizienz:** Es sollte vermieden werden unnütze Itemsets zu erzeugen, d.h. solche die ein nicht-häufiges Itemset enthalten.
- Ebenso sollten **Itemsets nicht mehrfach generiert** werden.

Erfüllen die nachfolgenden Ansätze diese Ziele?

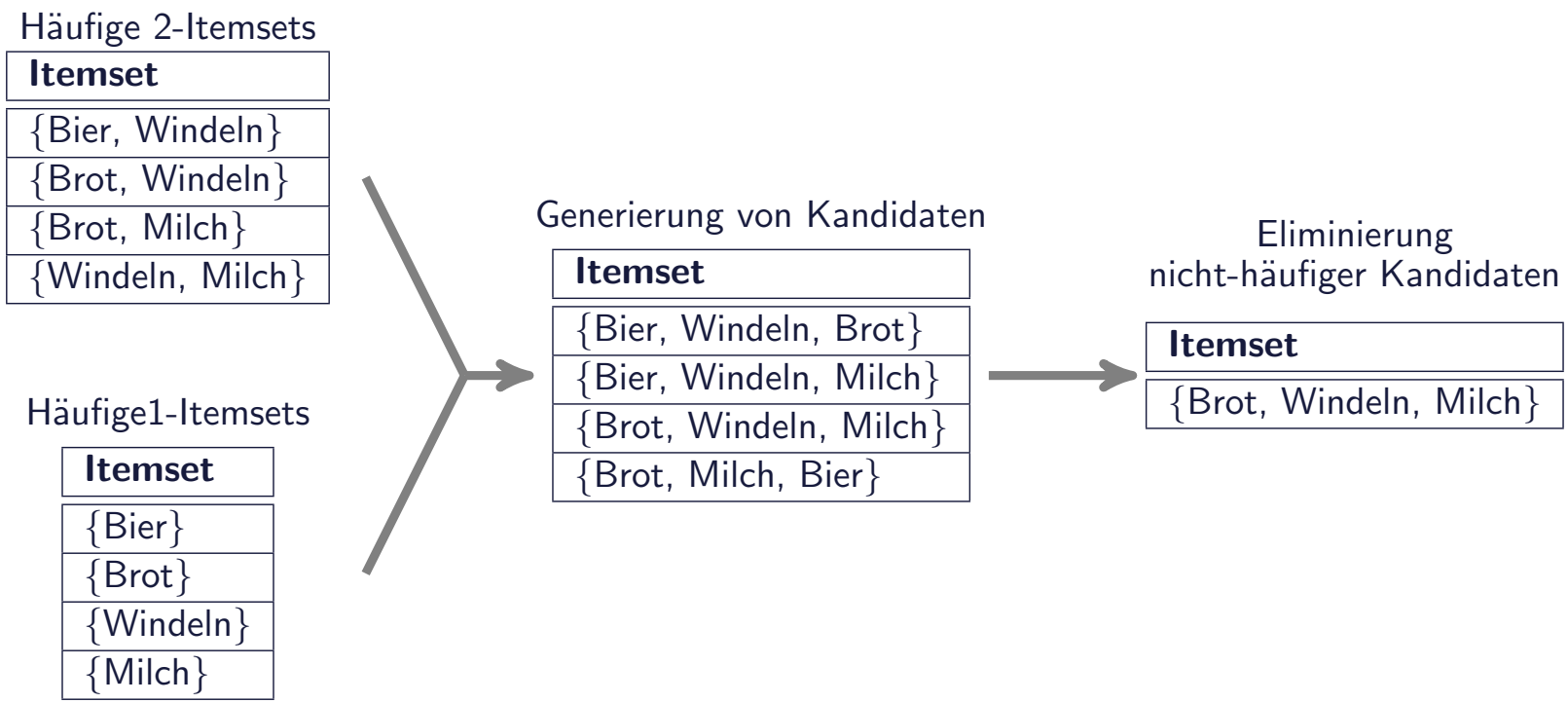
Generierung von Kandidaten: Brute-Force Ansatz

- **Schritt 1: Generiere alle Kandidaten.** Dies sind $\binom{d}{k}$ viele für d verschiedene Items.
- **Schritt 2**: Dann **entferne die nicht-häufigen Itemsets**.

Verbesserte Variante: Betrachte nur die Items aus F_1

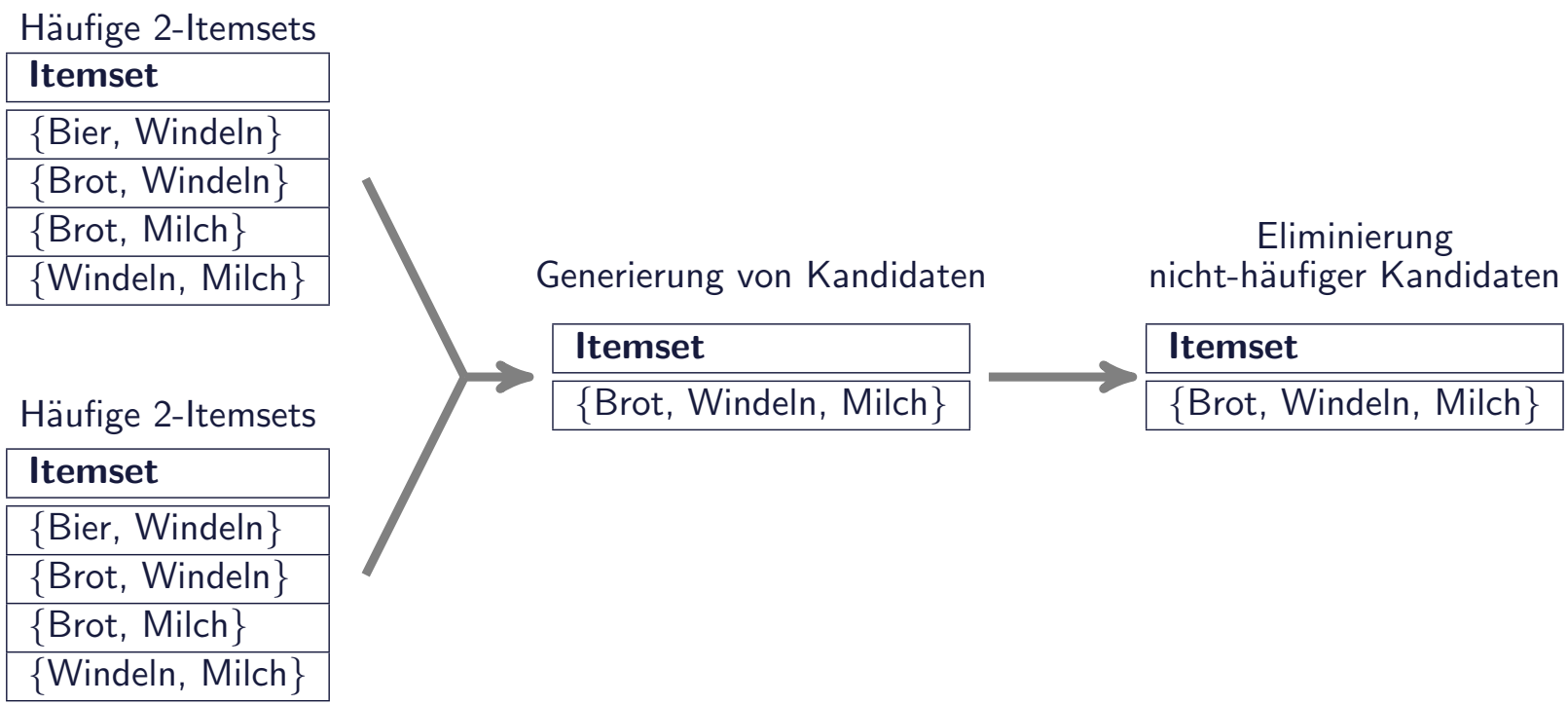
Generierung von Kandidaten: "$F_{k-1} \times F_1$"

- **Verknüpfe die häufigen $(k-1)$-Itemsets mit häufigen Items.**
- Dann entferne die nicht häufigen resultierenden Itemsets.
- Verbesserung: Erlaube nur Ergänzung durch 1-Itemset, wenn dies lexikographisch größer als Items des (k-1)-Itemsets ist.



Generierung von Kandidaten: "$F_{k-1} \times F_{k-1}$"

- **Kombiniere Paare von häufigen $(k-1)$-Itemsets** falls diese in den ersten (k-2) Items übereinstimmen.
- Betrachte lexikographische Sortierung der Itemsets.
- D.h., $A = \{a_1, a_2, \ldots a_{k-1}\}$ und $B = \{b_1, b_2, \ldots, b_{k-1}\}$ können kombiniert werden falls $a_i = b_i$ (für $i = 1,2, \ldots, k-2$) und $a_{k-1} \neq b_{k-1}$



Assoziationsregeln (Association Rules)

- **Basierend auf den häufigen Itemsets** können wir nun **Assoziationsregeln** generieren.
- Falls Z ein häufiges Itemset ist und $X \subset Z$ ist eine echte Teilmenge von Z, dann haben wir eine Regel $X \rightarrow Y$, mit $Y = Z \backslash X$.
- **Diese Regeln sind häufig da**

$$\text{supp}(X \rightarrow Y) = \text{supp}(X \cup Y) = \text{supp}(Z)$$

Assoziationsregeln: Konfidenz

- Für eine Regel $X \rightarrow Y$ betrachten wir die **Konfidenz**, die wie folgt definiert ist

$$\text{conf}(X \rightarrow Y) = \frac{\text{supp}(X \cup Y)}{\text{supp}(X)}$$

- **Eine Regel $X \rightarrow Y$ mit conf$(X \rightarrow Y) \geq minconf$ wird als confident bezeichnet.**

- **Ist eine Regel $X \rightarrow Z \backslash X$ nicht confident, so** kann keine Regel $W \rightarrow Z \backslash W$ mit $W \subseteq X$ confident sein.

Assoziationsregeln: Berechnung

Input: Menge F von häufigen Itemsets, minconf Schwellwert.

1. **foreach** $Z \in F$ mit $|Z| \geq 2$ **do**
2. $A = \{X \mid X \subset Z, X \neq \emptyset\}$
3. **while** $A \neq \emptyset$ **do**
4. $X =$ größtes Itemset aus A
5. $A = A \backslash X$
6. $c = \text{supp}(Z)/\text{supp}(X)$ /* Berechnung Konfidenz */
7. **if** $c \geq minconf$ **then**
8. print $X \rightarrow Y$, supp(Z), c /* Ausgabe, wobei $Y = Z \backslash X$ */
9. **else**
10. $A = A \backslash \{W \mid W \subset X\}$
11. **end**
12. **end**
13. **end**

Beispiel: Daten

Wir haben folgende Transaktionen

1:{Brot, Milch},
2:{Brot, Windeln, Bier, Eier},
3:{Milch, Windeln, Bier},
4:{Brot, Milch, Windeln, Bier}
5:{Brot, Milch, Windeln}

Mit minfreq=0.05 haben wir die folgenden häufigen Itemsets:

{Brot},{Milch},{Windeln},{Bier},{Brot, Milch},{Brot, Windeln},{Brot, Bier},{Milch, Windeln},{Milch, Bier},{Windeln, Bier},{Brot, Milch, Windeln},{Brot, Windeln, Bier},{Milch, Windeln, Bier}

Notation: Auf der folgenden Folie ist $A^{(i)}$ die Menge A in Iteration i

minfreq = 0.05, minconf = 0.5 und für Z={**Milch, Windeln, Bier**}

$A^{(0)}$ ={{Milch}, {Windeln}, {Bier}, {Milch, Windeln}, {Milch, Bier}, {Windeln, Bi

X ={Windeln, Bier}

Ausgabe: {Windeln, Bier} → {Milch} 2 0.667

$A^{(1)}$ ={{Milch}, {Windeln}, {Bier}, {Milch, Windeln}, {Milch, Bier}}

X ={Milch, Bier}

Ausgabe: {Milch, Bier} → {Windeln} 2 1.0

$A^{(2)}$ ={{Milch}, {Windeln}, {Bier}, {Milch, Windeln}}

X ={Milch, Windeln}

Ausgabe: {Milch, Windeln} → {Bier} 2 0.667

$A^{(3)}$ ={{Milch}, {Windeln}, {Bier}}

X ={Milch}

Ausgabe: {Milch} → {Windeln, Bier} 2 0.5

$A^{(4)}$ ={{Windeln}, {Bier}}

X ={Windeln}

Ausgabe: {Windeln} → {Milch, Bier} 2 0.5

$A^{(5)}$ ={{Bier}}

X ={Bier}

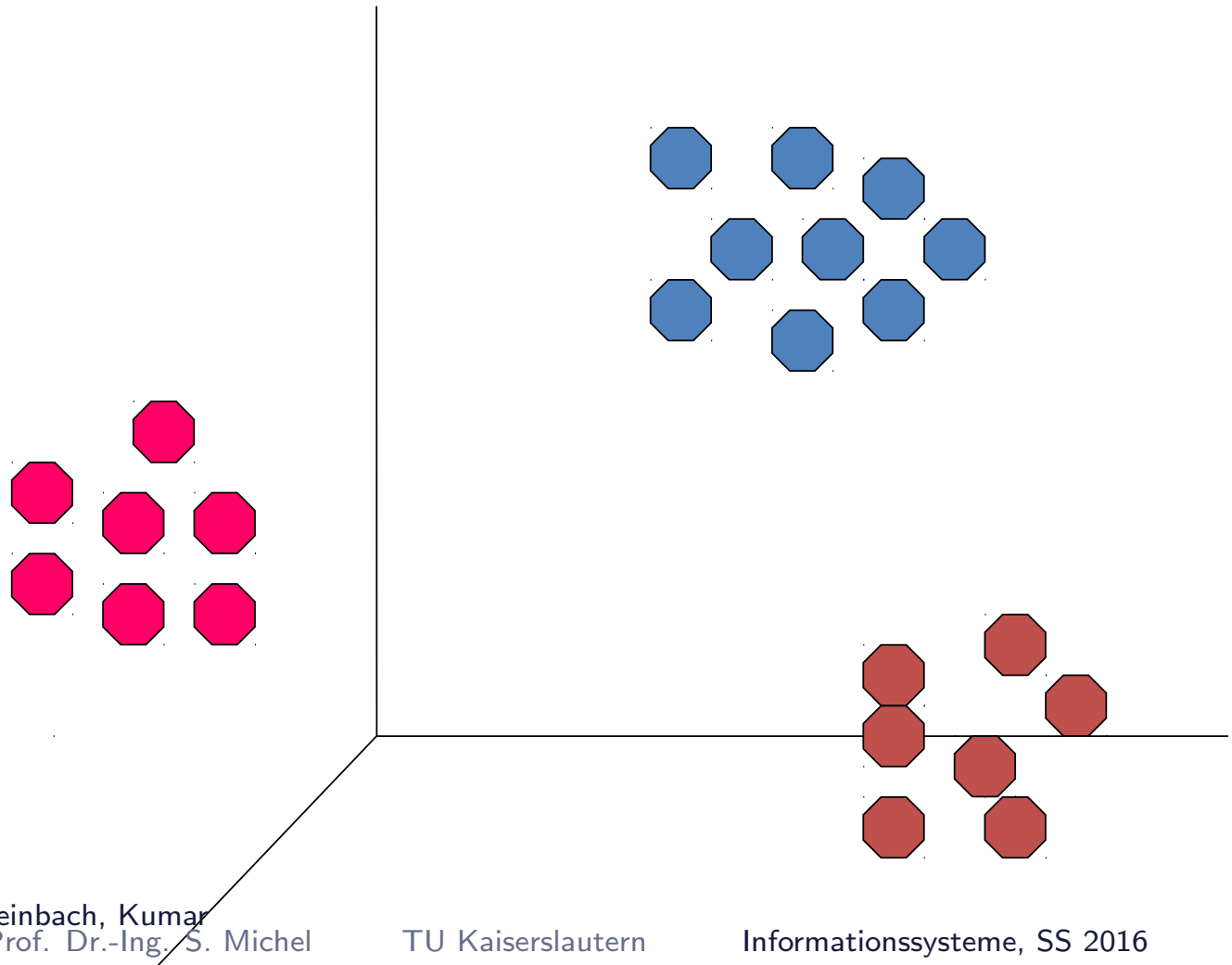
Ausgabe: {Bier} → {Milch, Windeln} 2 0.667

Zusammenfassung Itemsetmining

- **Warenkorbanalyse ist klassisches Beispiel für Data-Mining**
- Dabei werden **Transaktionen** bestehend aus Items auf **häufig zusammen auftretende Items** sowie nach **Assoziationsregeln** der Form "Wer Brot kauf kauft auch Bier" durchsucht.
- **Apriori Algorithmus** generiert Itemsets bottom-up basierend auf häufigen Itemsets kleinerer Länge.
- Dies funktioniert aufgrund der "**Anti-Monotonie**" des Supports.
- Assoziationsregeln werden basierend auf häufigen Itemsets berechnet.

Clustering

Gegeben eine Menge von Objekten. Ziel: Finden eines guten Clusterings (Gruppierung) der Objekte anhand ihrer Eigenschaften. Hier, anhand ihrer 3D Koordinaten.



Abbildung, Tan,Steinbach, Kumar

Das Clustering-Problem (2)

- Gegeben eine **Menge U von Objekten** und eine **Distanzfunktion** $d : U \times U \rightarrow \mathbb{R}^+$
- **Gruppiere Objekte** aus U in Cluster (Teilmengen), so dass die Distanz zwischen den Punkten eines Clusters klein ist und die Distanz zwischen den einzelnen Clustern groß ist.

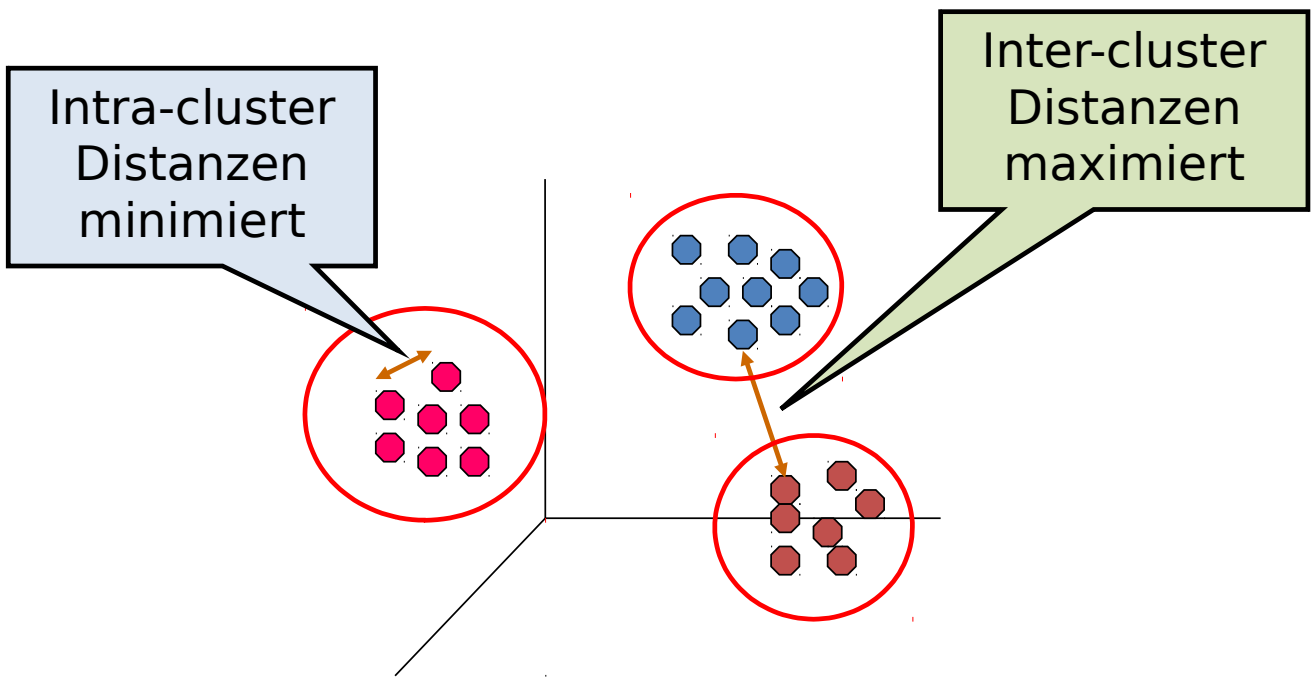


Abbildung nach Tan,Steinbach, Kumar

Partitionen und Prototypen

Wir betrachten hier **exklusives Clustering**, d.h. **ein Objekt ist genau einem Cluster zugeordnet**:

- Die Menge U ist partitioniert in k Clusters $C_1, C_2, \ldots, C_k$ mit

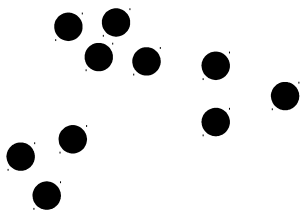
$$\bigcup_i C_i = U \qquad \text{und} \qquad C_i \cap C_j = \emptyset \text{ für } i \neq j$$

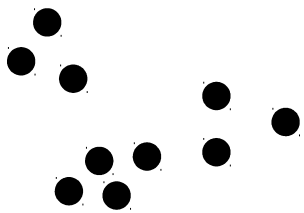
- Jedes Cluster C_i wird von einem sogenannten **Prototypen** μ_i repräsentiert (aka. **Schwerpunkt/Centroid oder Mitte/Durchschnitt**)
  - Dieser Prototyp μ_i muss nicht notwendigerweise eines der Objekte aus C_i sein

- Die **Qualität des Clusterings** wird dann in der Regel berechnet als den **quadratischen Fehler** zwischen den Objekten eines Clusters und dem Prototypen eines Clusters (hier für d-dimensionale Daten):

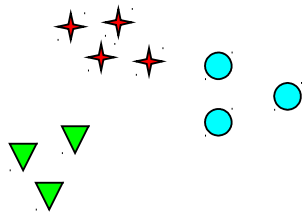
$$\sum_{i=1}^{k} \sum_{x_j \in C_i} ||x_j - \mu_i||_2^2 = \sum_{i=1}^{k} \sum_{x_j \in C_i} \sum_{l=1}^{d} (x_{jl} - \mu_{il})^2$$

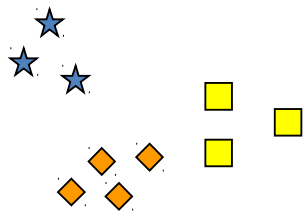
Clustering nicht (immer) eindeutig



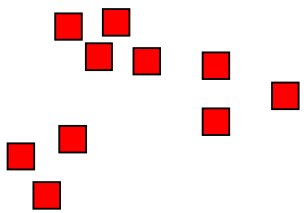


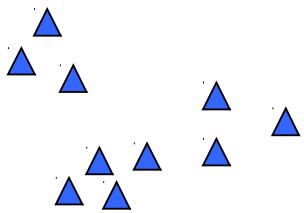
Wie viele Cluster?



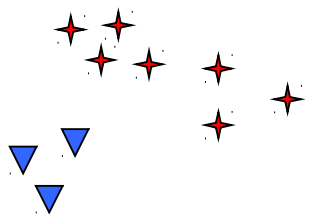


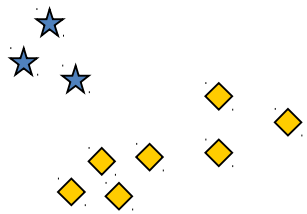
Sechs Cluster





Zwei Clusters





Vier Cluster

Abbildung nach Tan,Steinbach, Kumar

Ein Naiver (Brute-Force) Ansatz

1. Generiere **alle möglichen Clusterings**, eins nach dem anderen
2. **Berechne den quadratischen Fehler**
3. **Wähle das Cluster mit dem kleinsten Fehler aus**

Dieser Ansatz ist leider unbrauchbar: Es gibt viel zu viele mögliche Clusterings, die ausprobiert werden müssen.

- Es gibt k^n Möglichkeiten k Cluster zu erzeugen bei n Objekten. Davon können einige Cluster leer sein. Also für 50 Objekte und 3 Cluster gibt es $3^{50} = 717897987691852588770249$ Möglichkeiten.
- Die Anzahl der Möglichkeiten diese n Punkte in k nicht leere Cluster aufzuteilen ist die Stirling-Zahl der zweiten Art.
  $S(50,3) = 11964966405235881137373 0.$
  Nur zur Info, hier die Definition Stirling-Zahl der zweiten Art:

$$S(n,k) = \left\{ {n \atop k} \right\} = \frac{1}{k!} \sum_{j=0}^{k} (-1)^j \binom{k}{j} (k-j)^n$$

K-Means Clustering

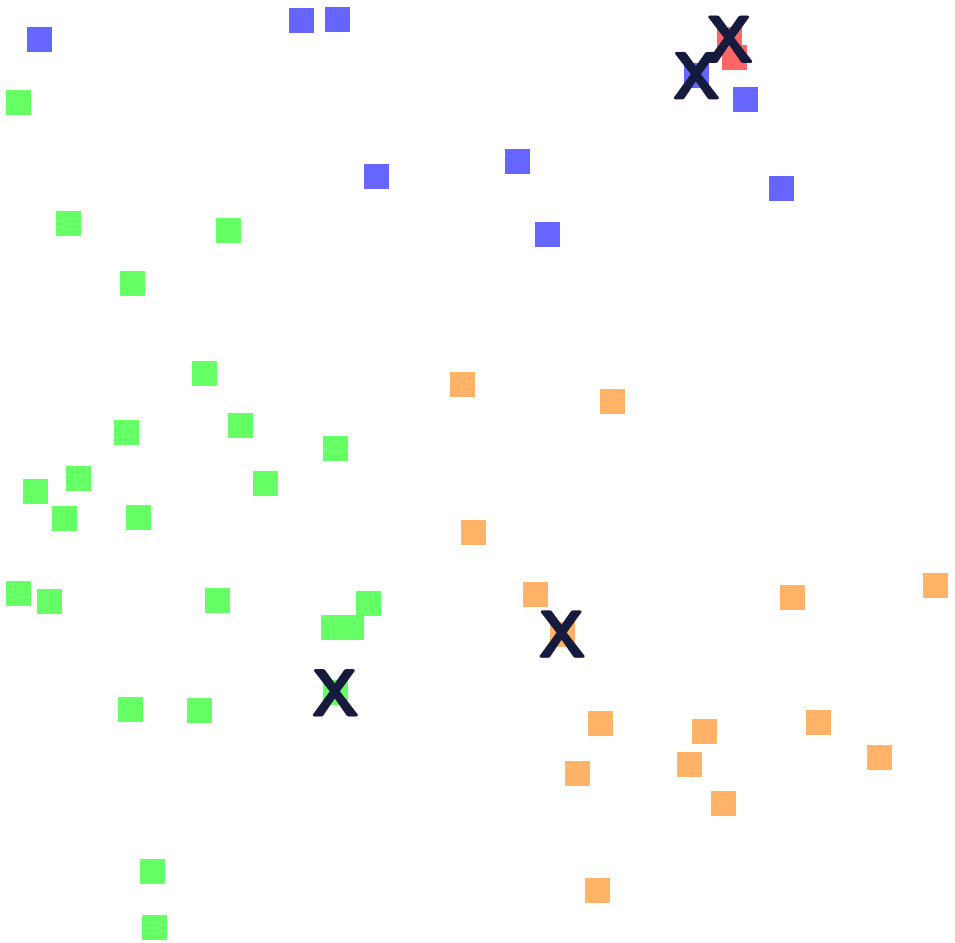
- **Jedes Cluster wird durch einen Mittelpunkt (Centroid) repräsentiert**
- **Ein Objekt wird dem Centroid mit der geringsten Distanz zugewiesen**
- **Es gibt k Cluster. k ist ein Parameter.**

Algorithmus:

1. Wähle zufällig k Objekte als initiale Centroids aus.
2. **repeat**
3. Ordne Objekte dem jeweils nächstgelegenen Centroid zu
4. Berechne für jedes Cluster den neuen Centroid.
5. **until** die Centroids ändern sich nicht mehr

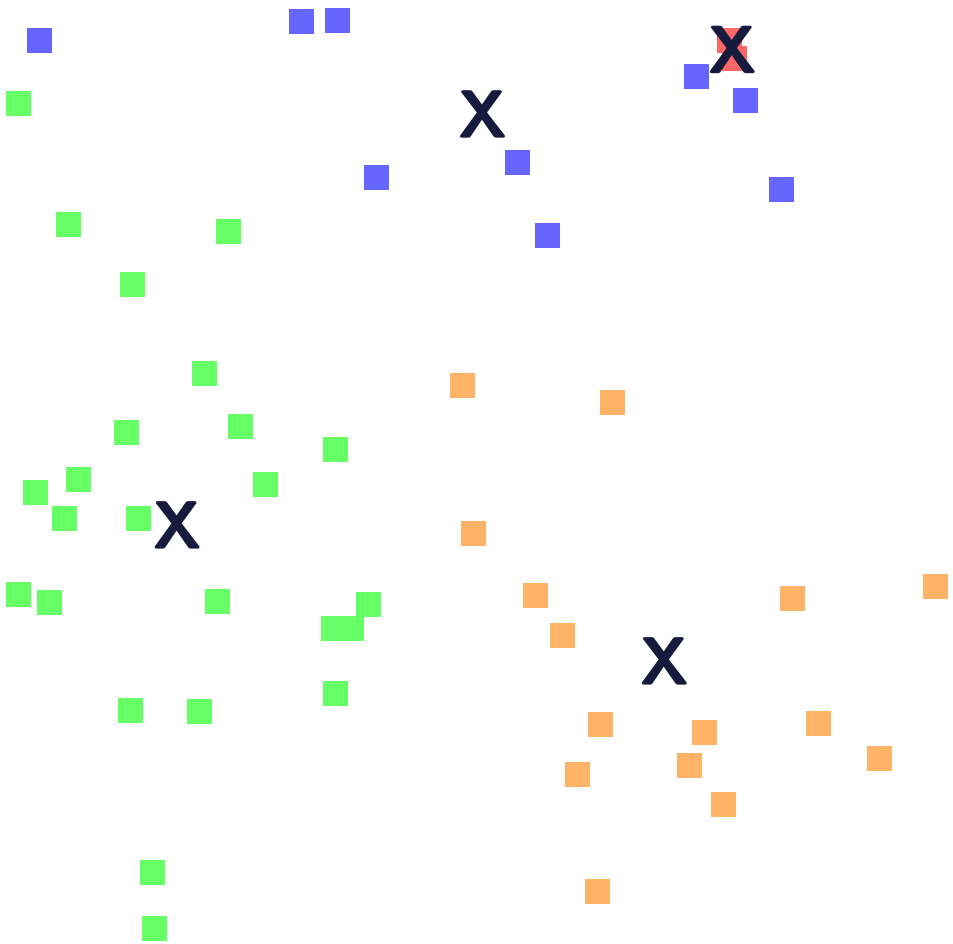
K-Means: Beispiel

Wähle zufällig $k = 4$ Centroids aus und ordne Objekte zu



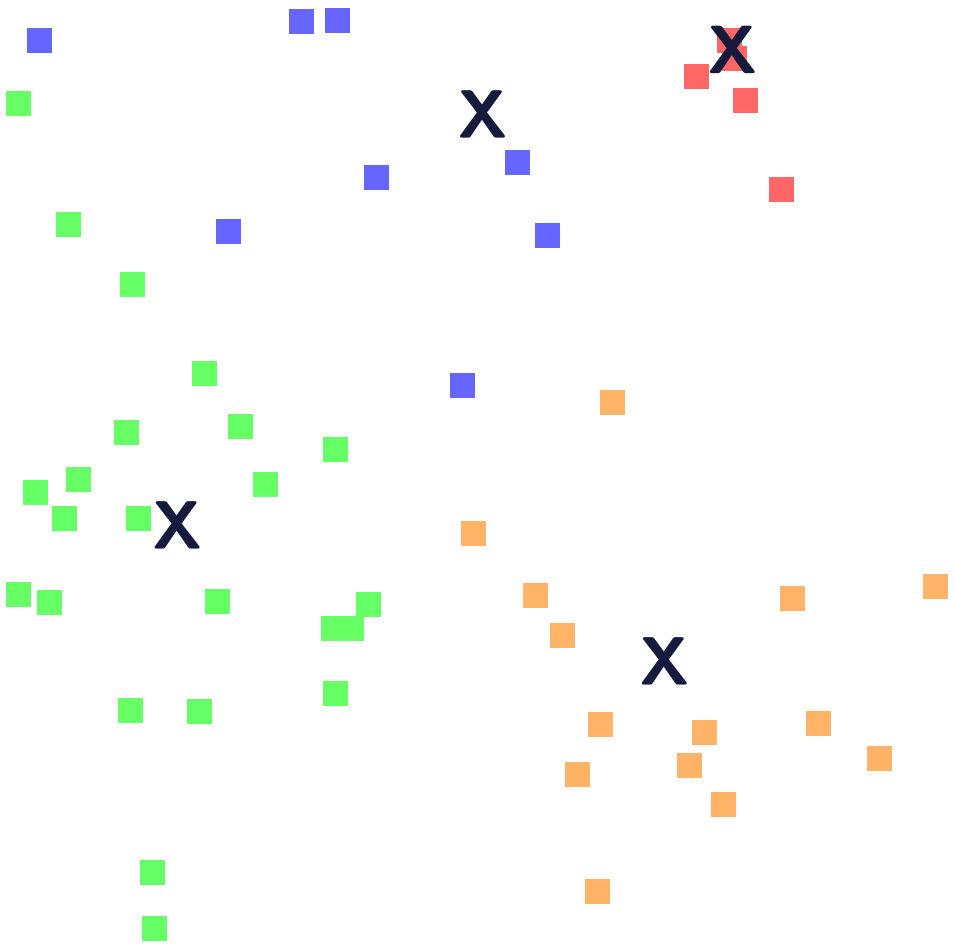
K-Means: Beispiel

Berechne Centroid jedes Clusters neu



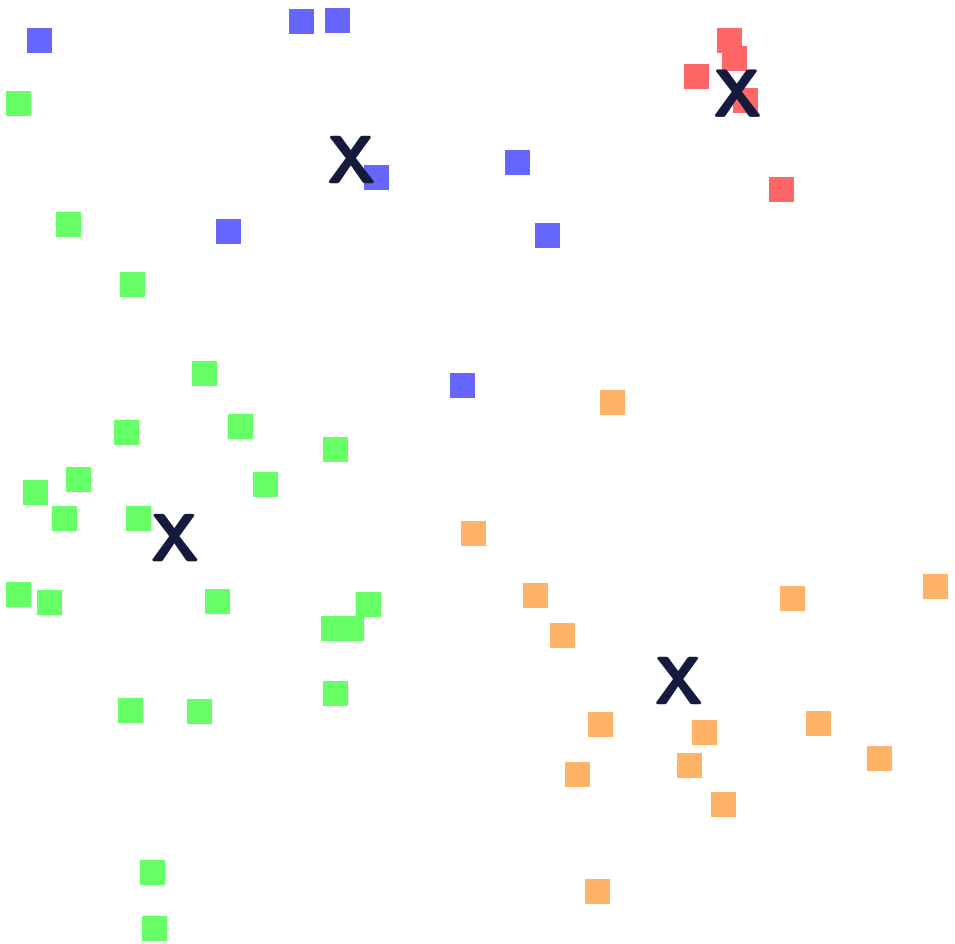
K-Means: Beispiel

Ordne Objekte neu zu



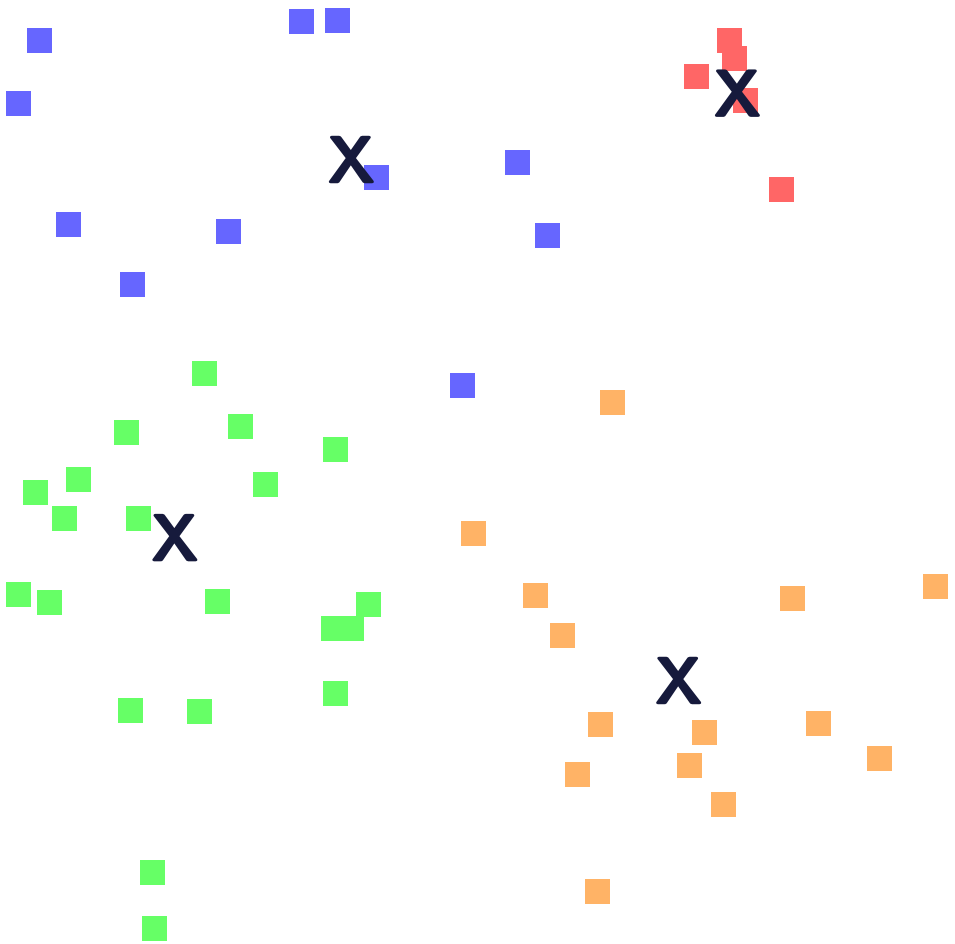
K-Means: Beispiel

Berechne Centroid jedes Clusters neu



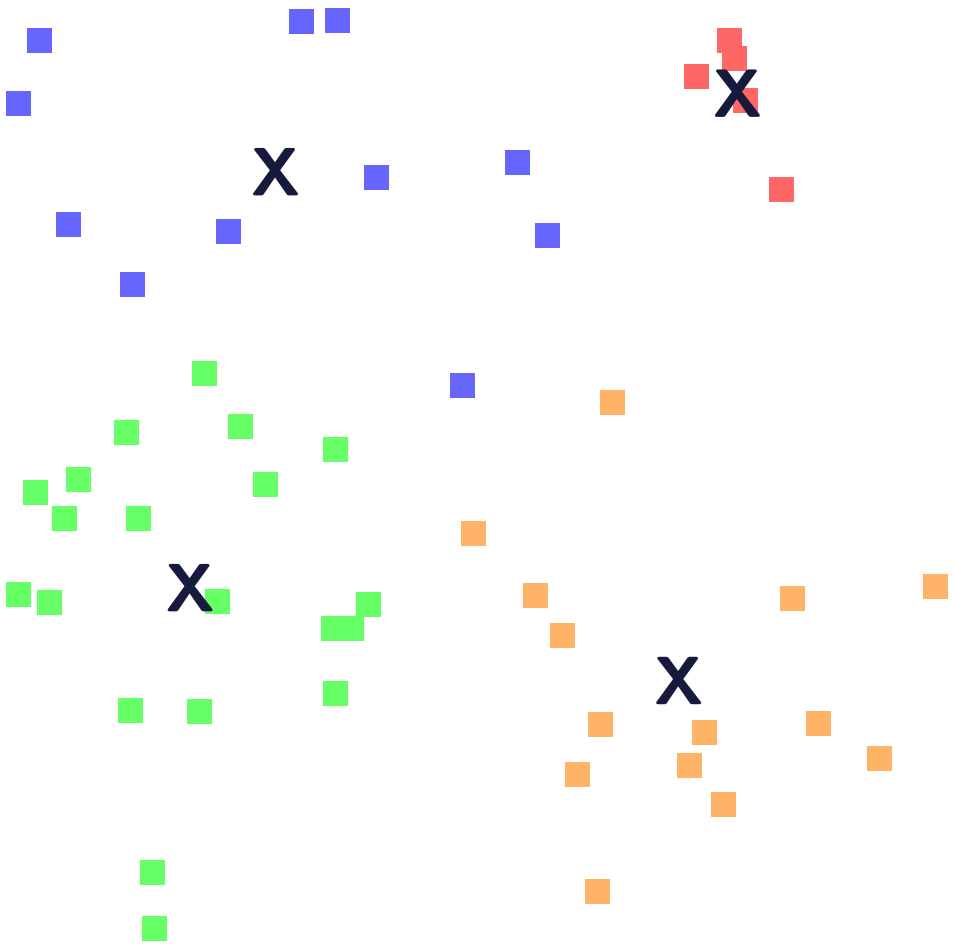
K-Means: Beispiel

Ordne Objekte neu zu



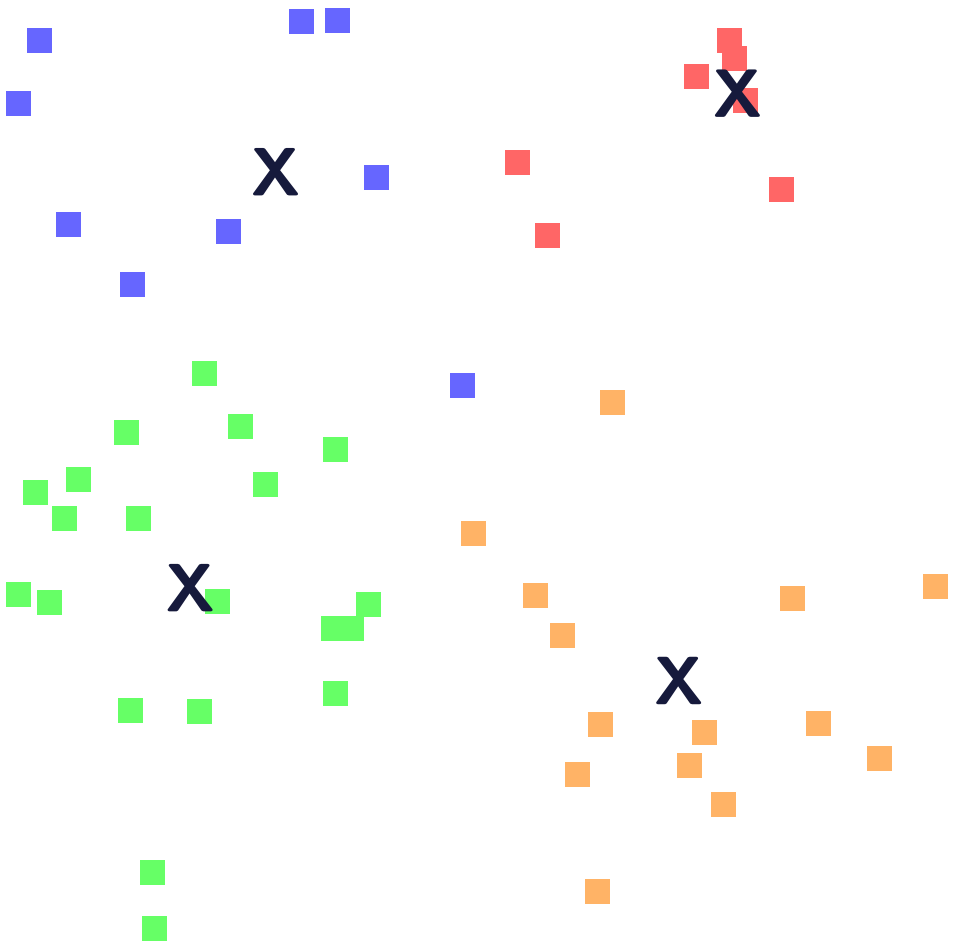
K-Means: Beispiel

Berechne Centroid jedes Clusters neu



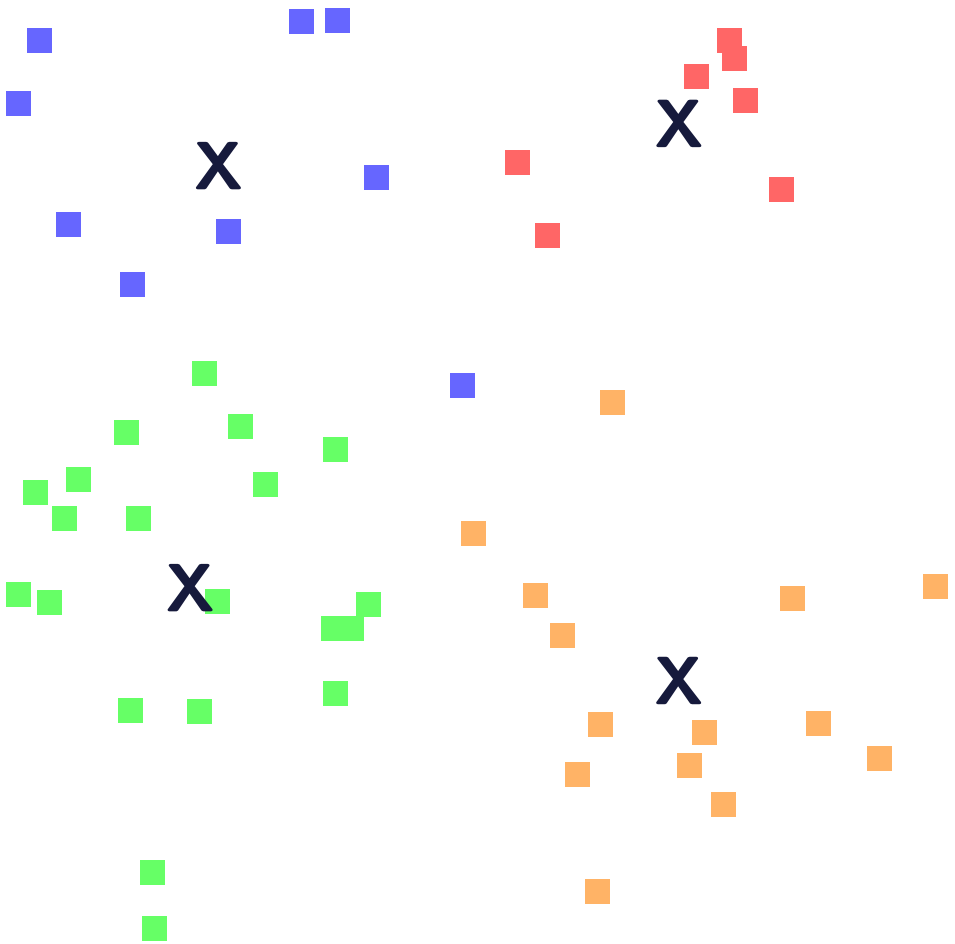
K-Means: Beispiel

Ordne Objekte neu zu



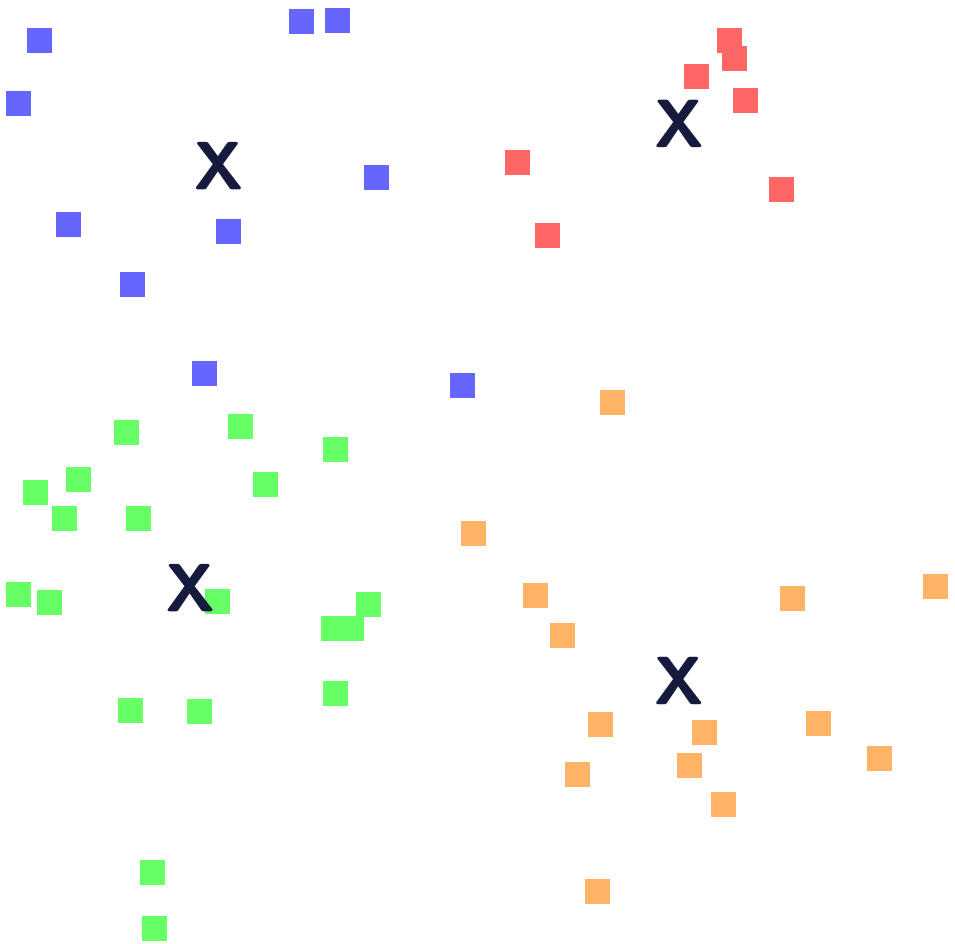
K-Means: Beispiel

Berechne Centroid jedes Clusters neu



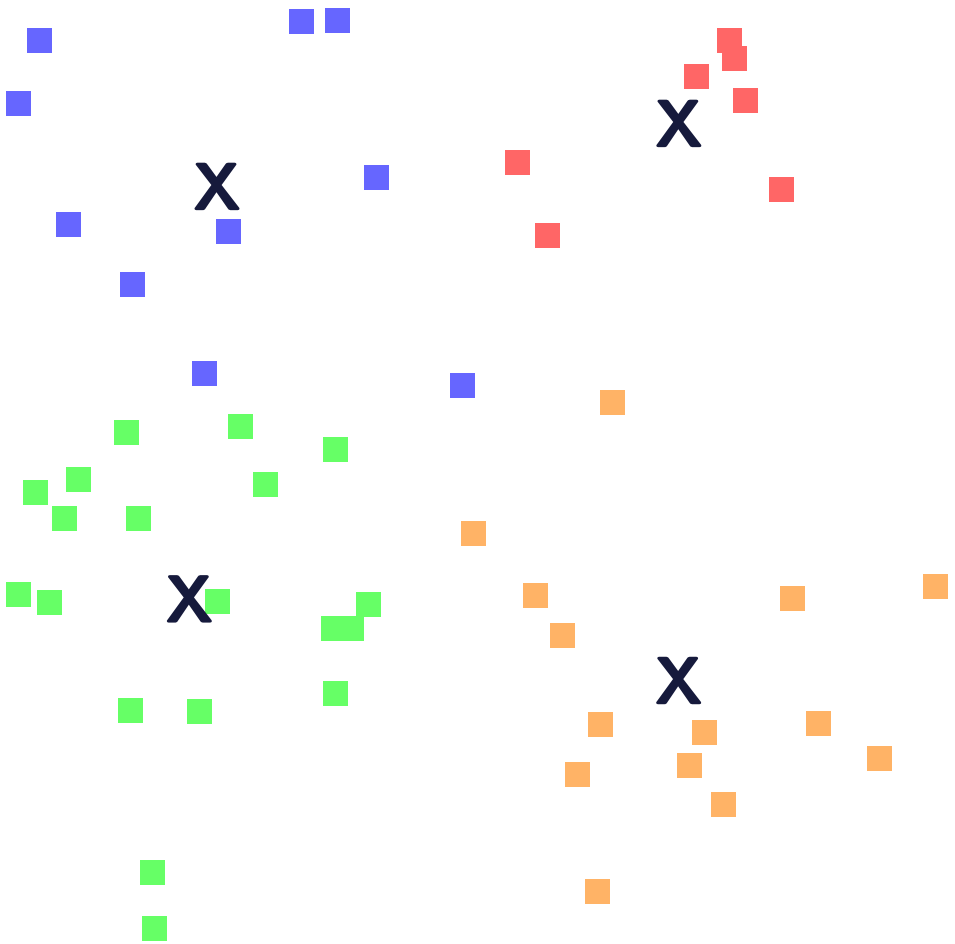
K-Means: Beispiel

Ordne Objekte neu zu



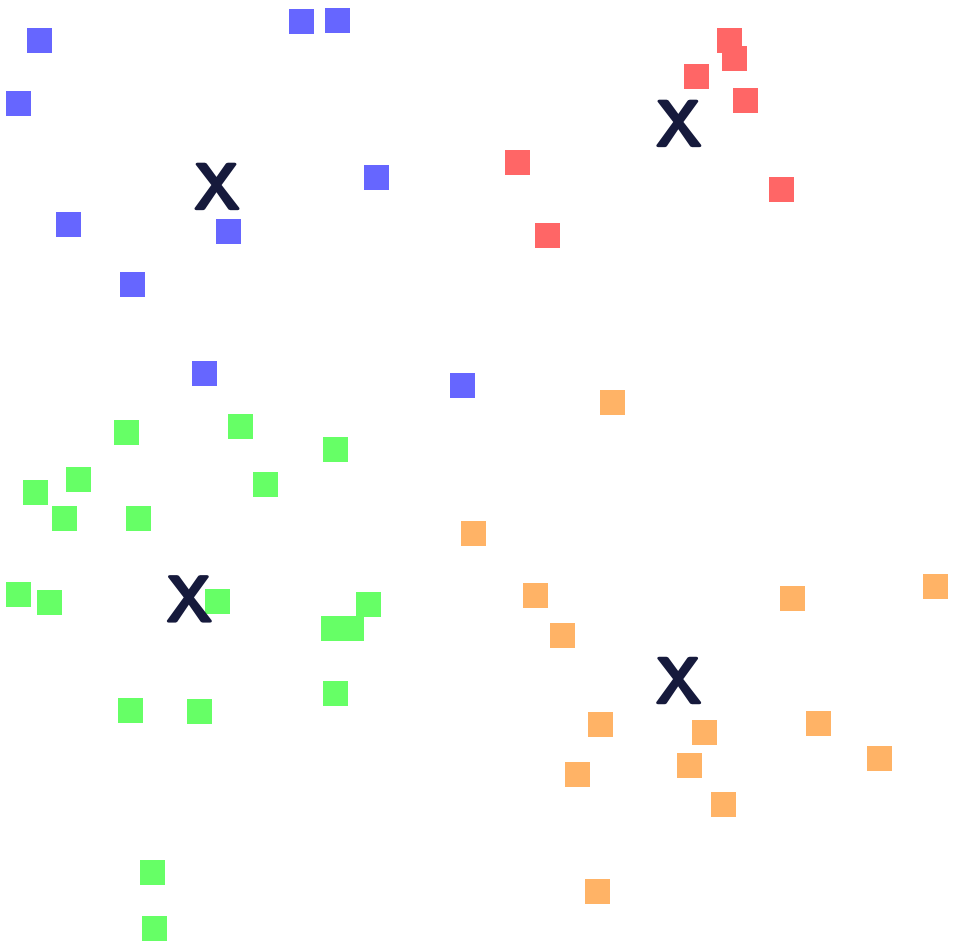
K-Means: Beispiel

Berechne Centroid jedes Clusters neu



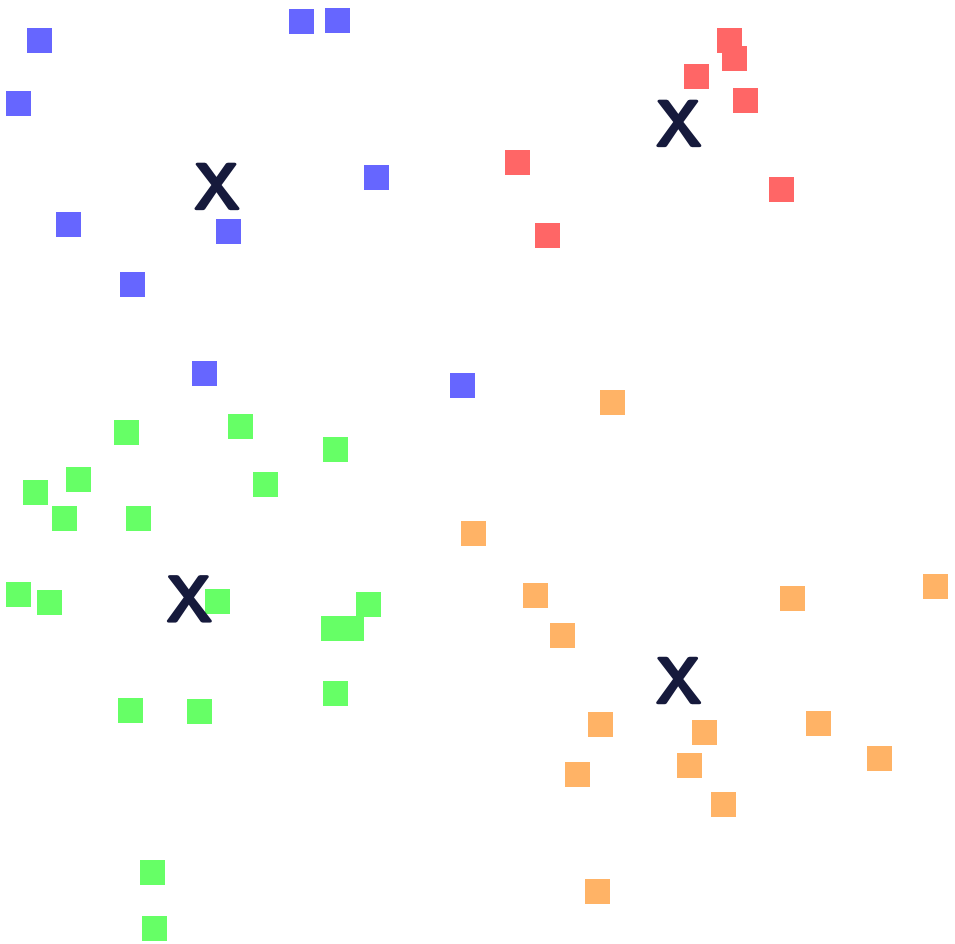
K-Means: Beispiel

Ordne Objekte neu zu



K-Means: Beispiel

Berechne Centroid jedes Clusters neu: Hat sich nichts geändert!



K-Means-Clustering

- Die initialen Centroids werden normalerweise zufällig ausgewählt. Dadurch können verschiedene Durchläufe auf den gleichen Daten unterschiedliche Cluster erzeugen.
- Als Centroid benutzt man typischerweise den Mittelwert (Mean) der Objekte eines Clusters.
- Als Distanzmaß wird z.B. die Euklidische Distanz benutzt

- Der K-Means-Algorithmus konvergiert
- In den ersten Iterationen sind die Änderungen des Clusterings am deutlichsten
- Abbruchkriterium auch: "Bis nur noch sehr wenige Objekte das Cluster wechseln"
- Komplexität ist $O(n \times k \times I \times d)$. n =Anzahl Objekte, k =Anzahl Cluster, I =Anzahl Iterationen, d =Dimensionalität der Daten.