Heinrich Heine Universität Düsseldorf

Algorithmen in der Bioinformatik

8. Cluster & Cliquen

Prof. Dr. Gunnar Klau

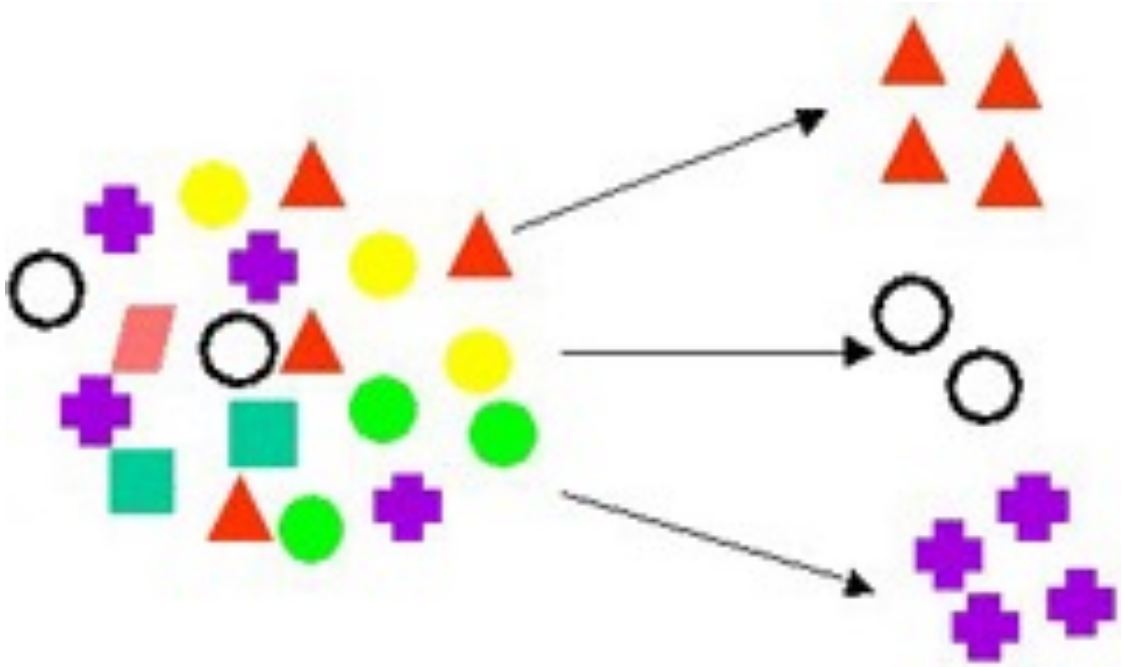
Nach Jones & Pevzner: An Introduction to Bioinformatics Algorithms

Clustern

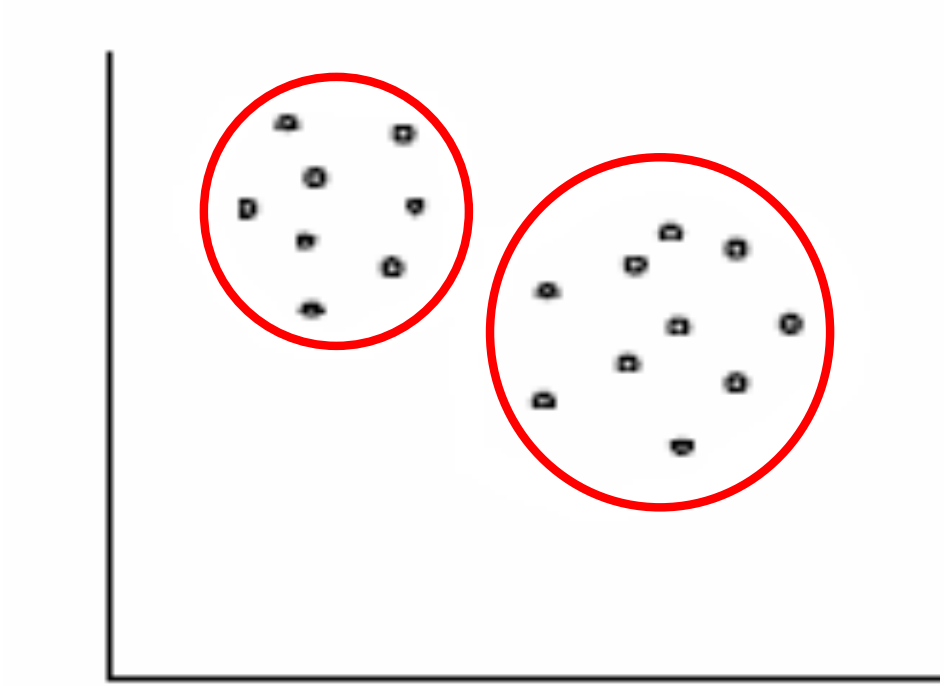
= Klassifizierung von Objekten in ‘zusammengehörige’ Gruppen
(Suche nach Ordnung in unordentlichen Daten)

Ziel: maximiere gleichzeitig:

- Homogenität (Punkte desselben Clusters sind ähnlich)
- Separation (Punkte verschiedener Cluster sind unähnlich)



Gute Cluster, schlechte Cluster



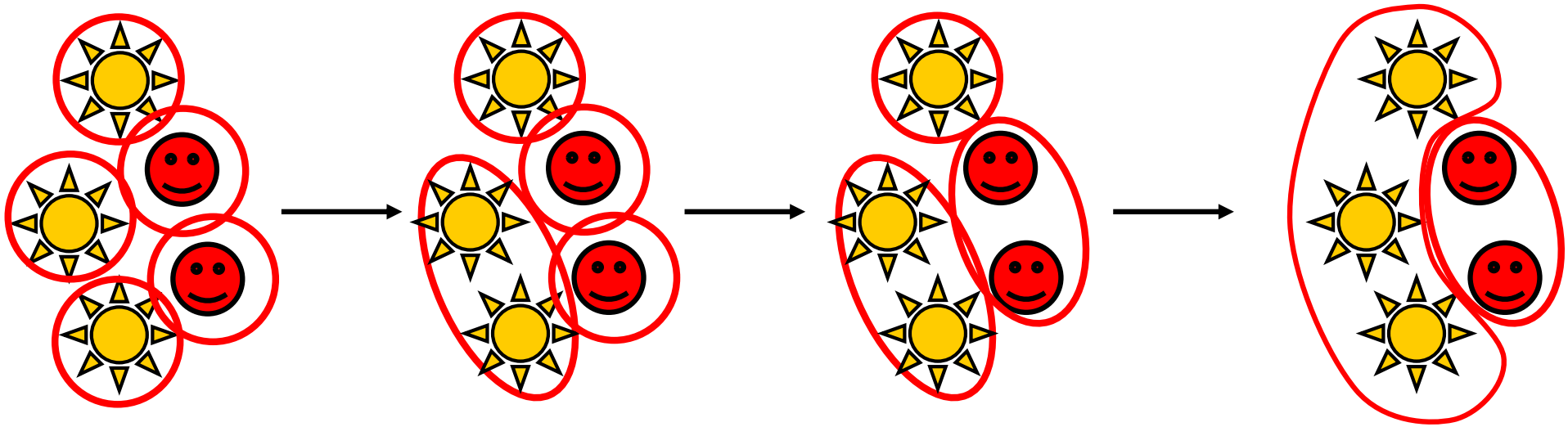
- Maximale Homogenität
- Maximale Separation

- Geringe Homogenität
- Geringe Separation

Techniken

Agglomerativ:

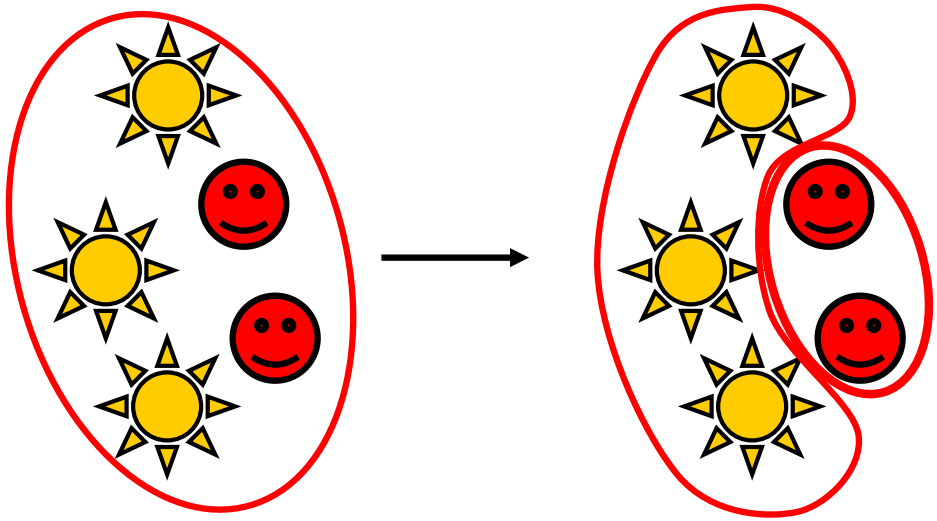
Starte mit jedem Element in seinem eigenen Cluster
Verbinde Cluster iterativ



Techniken

Divisiv:

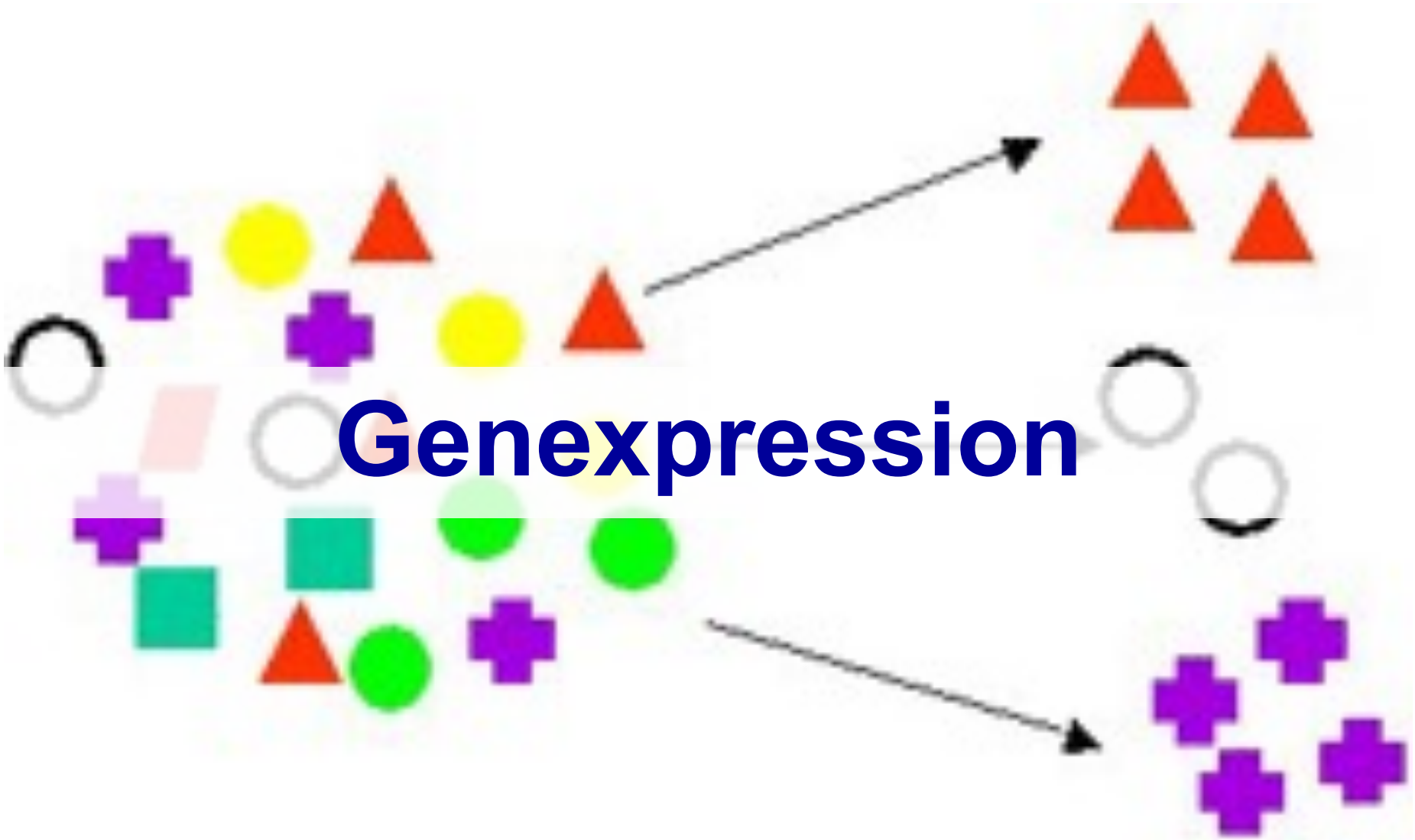
Starte mit einem allumfassenden Cluster
Teile Cluster iterativ



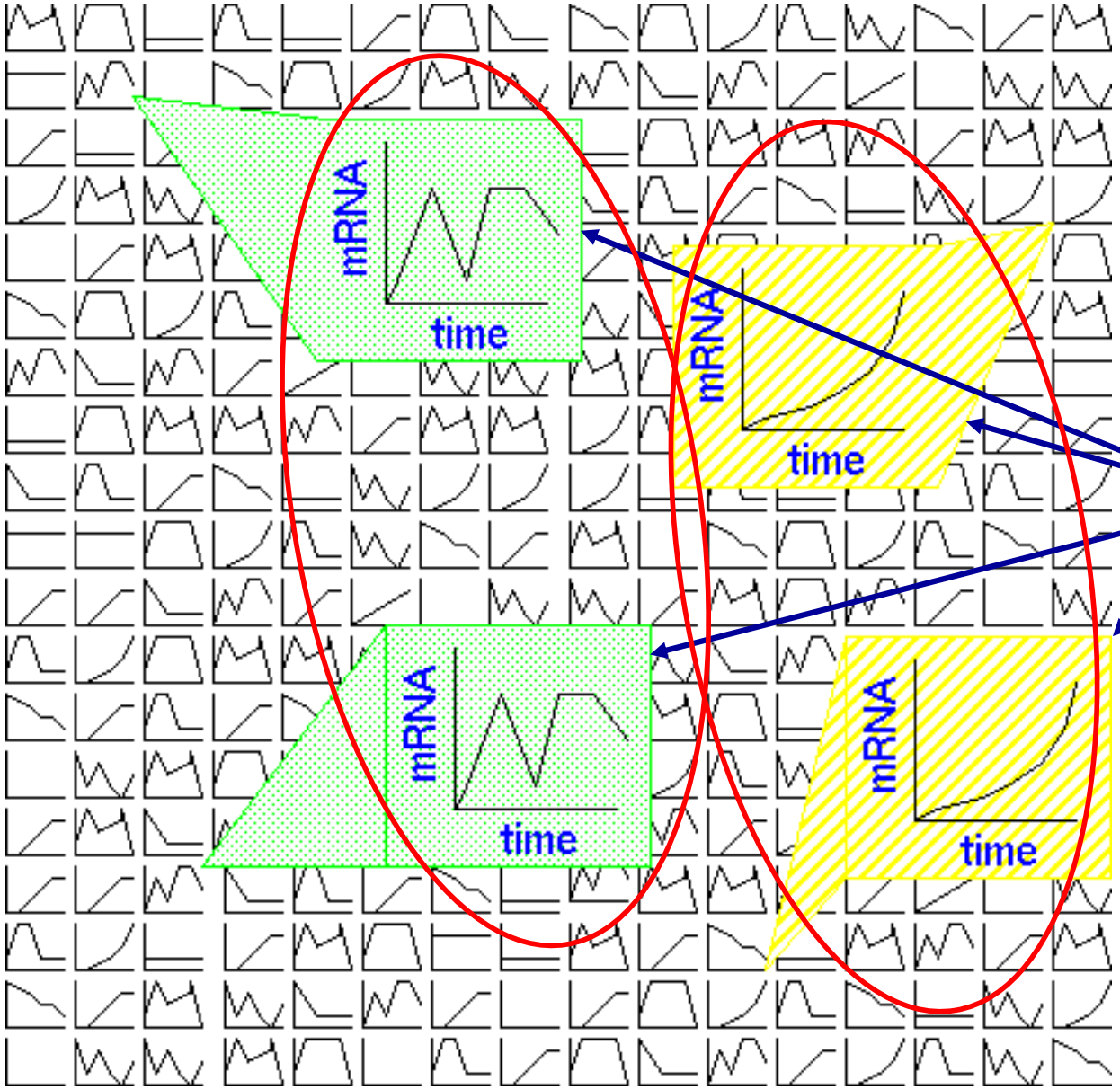
Clustern in der Biologie

- Verwandtschaft zwischen Arten
- ‘Gemeinsam’ exprimierte Gene (funktionaler Zusammenhang?)
- Definition von Genfamilien (Gene mit ähnlicher Sequenz / Funktion)
- Identifikation metabolischer Pathways
- …

- https://en.wikipedia.org/wiki/Cluster_analysis#Applications


Genexpression

Genexpressions-Zeitverläufe (time courses)

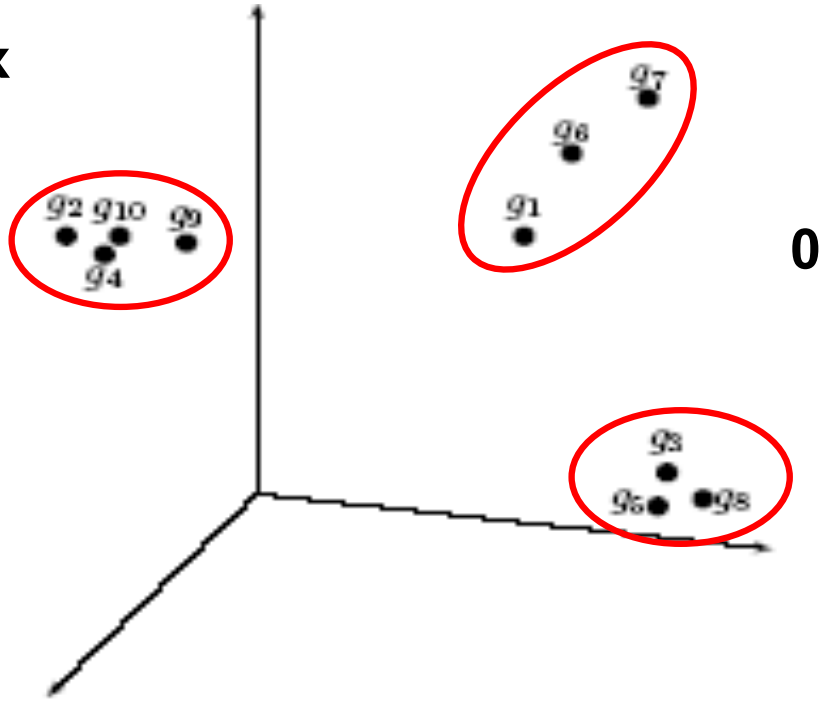


mRNA-Menge
je eines Gens
zu 6 Zeitpunkten

Clustern von Genexpressions-Zeitverläufen

Time	1 hr	2 hr	3 hr
g_1	10.0	8.0	10.0
g_2	10.0	0.0	9.0
g_3	4.0	8.5	3.0
g_4	9.5	0.5	8.5
g_5	4.5	8.5	2.5
g_6	10.5	9.0	12.0
g_7	5.0	8.5	11.0
g_8	2.7	8.7	2.0
g_9	9.7	2.0	9.0
g_{10}	10.2	1.0	9.2

Intensitäts-Matrix



0

Abstands-Matrix:

Gene mit kleinem Abstand gehören zusammen

	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8	g_9	g_{10}
g_1	0.0	8.1	9.2	7.7	9.3	2.3	5.1	10.2	6.1	7.0
g_2	8.1	0.0	12.0	0.9	12.0	9.5	10.1	12.8	2.0	1.0
g_3	9.2	12.0	0.0	11.2	0.7	11.1	8.1	1.1	10.5	11.5
g_4	7.7	0.9	11.2	0.0	11.2	9.2	9.5	12.0	1.6	1.1
g_5	9.3	12.0	0.7	11.2	0.0	11.2	8.5	1.0	10.6	11.6
g_6	2.3	9.5	11.1	9.2	11.2	0.0	5.6	12.1	7.7	8.5
g_7	5.1	10.1	8.1	9.5	8.5	5.6	0.0	9.1	8.3	9.3
g_8	10.2	12.8	1.1	12.0	1.0	12.1	9.1	0.0	11.4	12.4
g_9	6.1	2.0	10.5	1.6	10.6	7.7	8.3	11.4	0.0	1.1
g_{10}	7.0	1.0	11.5	1.1	11.6	8.5	9.3	12.4	1.1	0.0

Abstandsberechnung für Genexpressions-Daten

Einige Möglichkeiten für zwei Genexpressions-Vektoren

$v = (t_1 \ldots t_m)$, $v' = (t'_1 \ldots t'_m)$

Pearsons Korrelationskoeffizient:

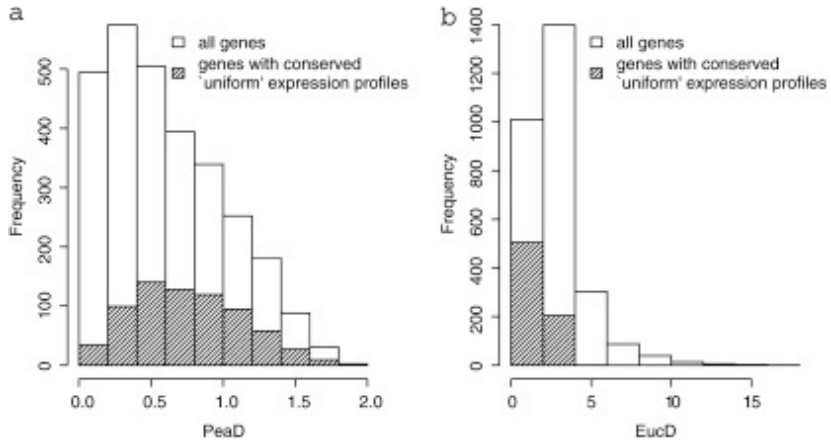
$d(v,v') = R(v,v')$

Euklidischer Abstand:

$d(v,v') = ||v - v'||_2 = \sqrt{\sum_{i=1}^{m} (t_i - t'_i)^2}$

Quadrierter Euklidischer Abstand:

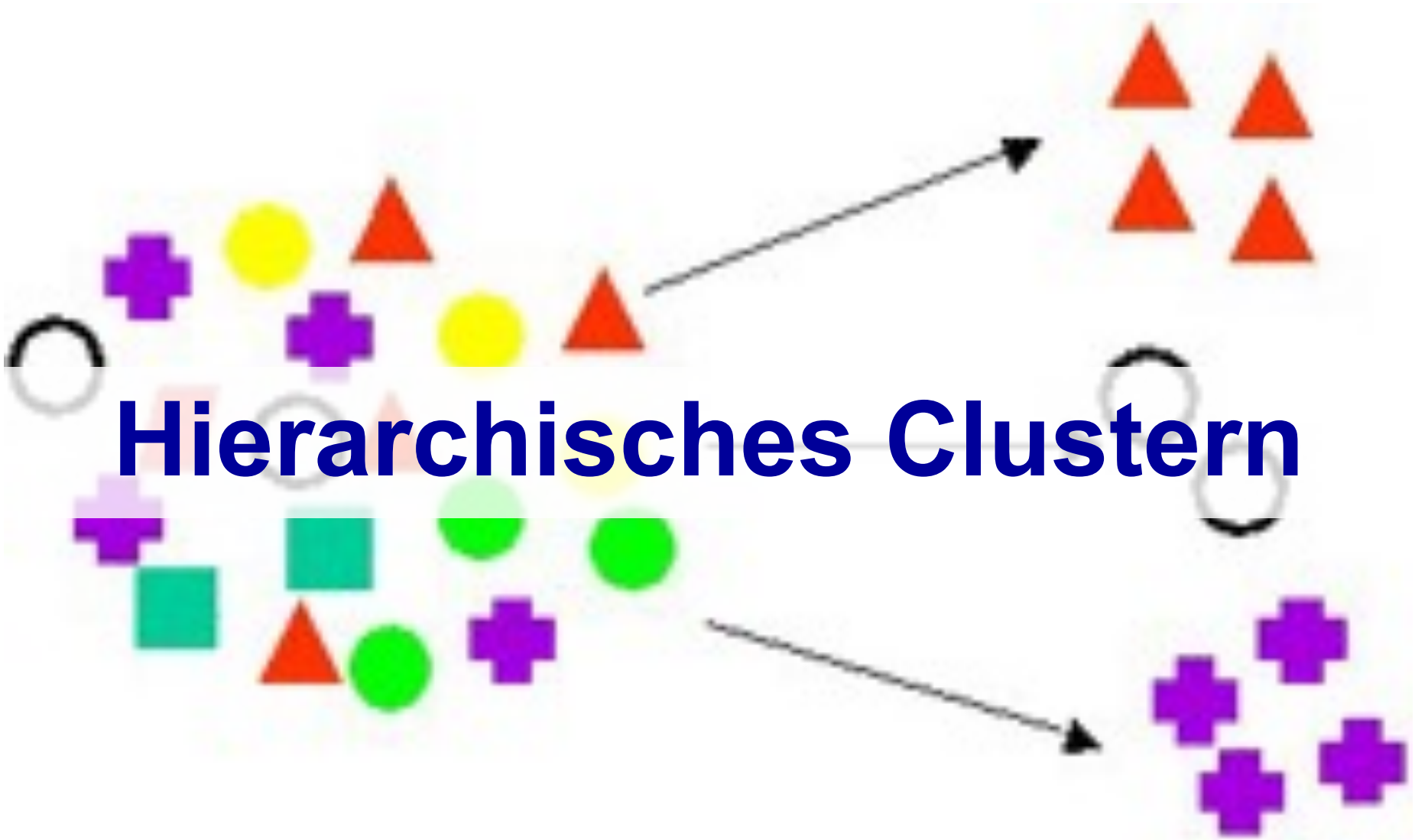
$d(v,v') = ||v - v'||_2^2$



Adam Eyre-Walker (*Genetics* 2009, **183**:1597):

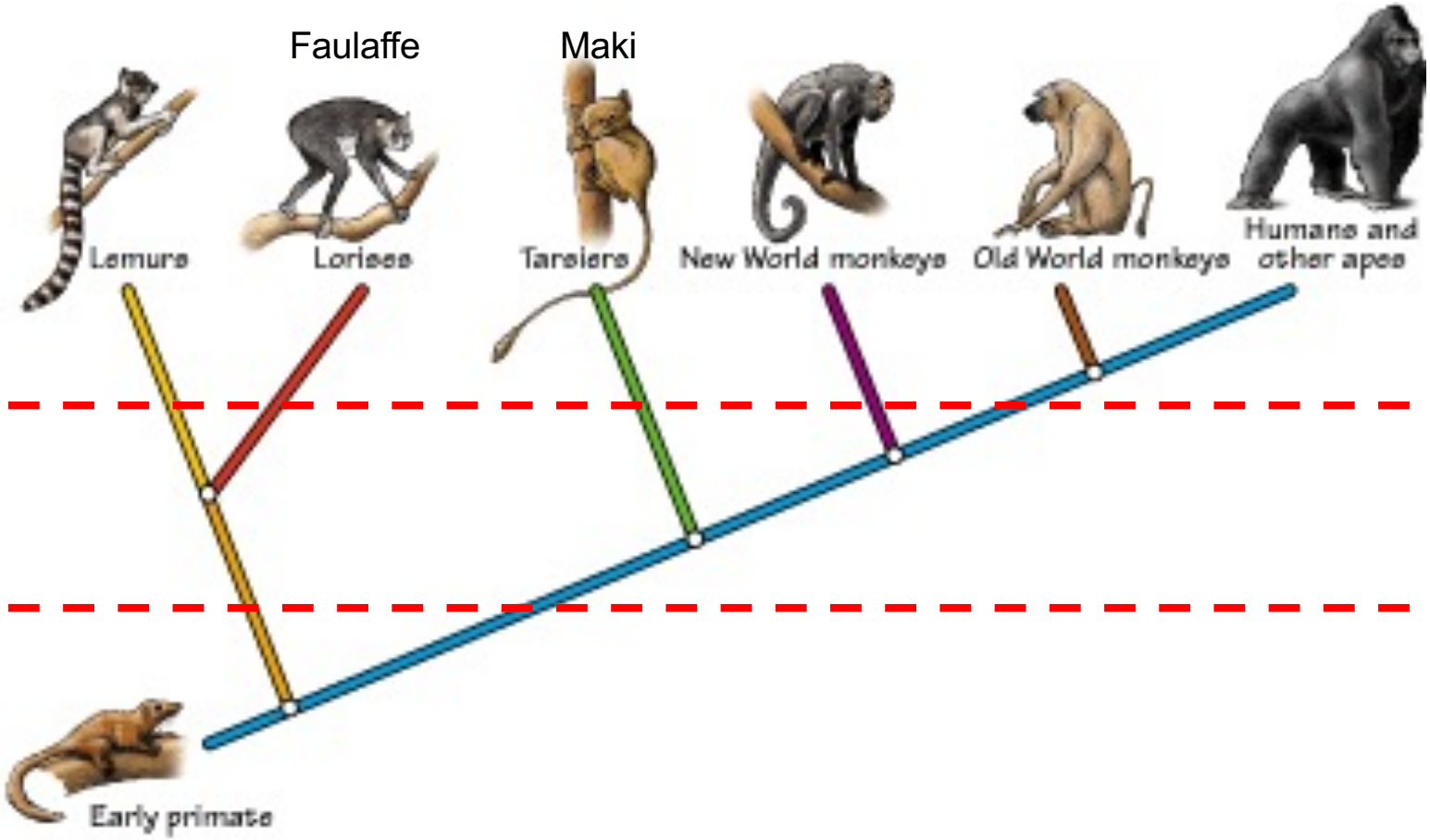
> "if measurement error is large relative to the differences in expression, the correlation coefficient will tend to show high divergence for genes that have relatively uniform levels of expression across tissues or time points… the Euclidean distance yields low estimates of expression divergence for genes with a conserved uniform pattern of expression."

Hierarchisches Clustern



Hierarchisches Clustern

- Elemente als Blätter eines Baums
- Pfadlänge zwischen Blättern repräsentiert deren Ähnlichkeit
- Kurzer Pfad = ähnliche Elemente, langer Pfad = unähnliche Elemente
- Schnitte durch den Baum ergeben verschiedene Clusterlevel



Hierarchisches Clustern: Problem

Ziel: Erstelle ein hierarchisches Clustering (den Ähnlichkeits- oder Abstandsbaum) für eine Menge von *n* Elementen mit der Abstandsmatrix ***d***

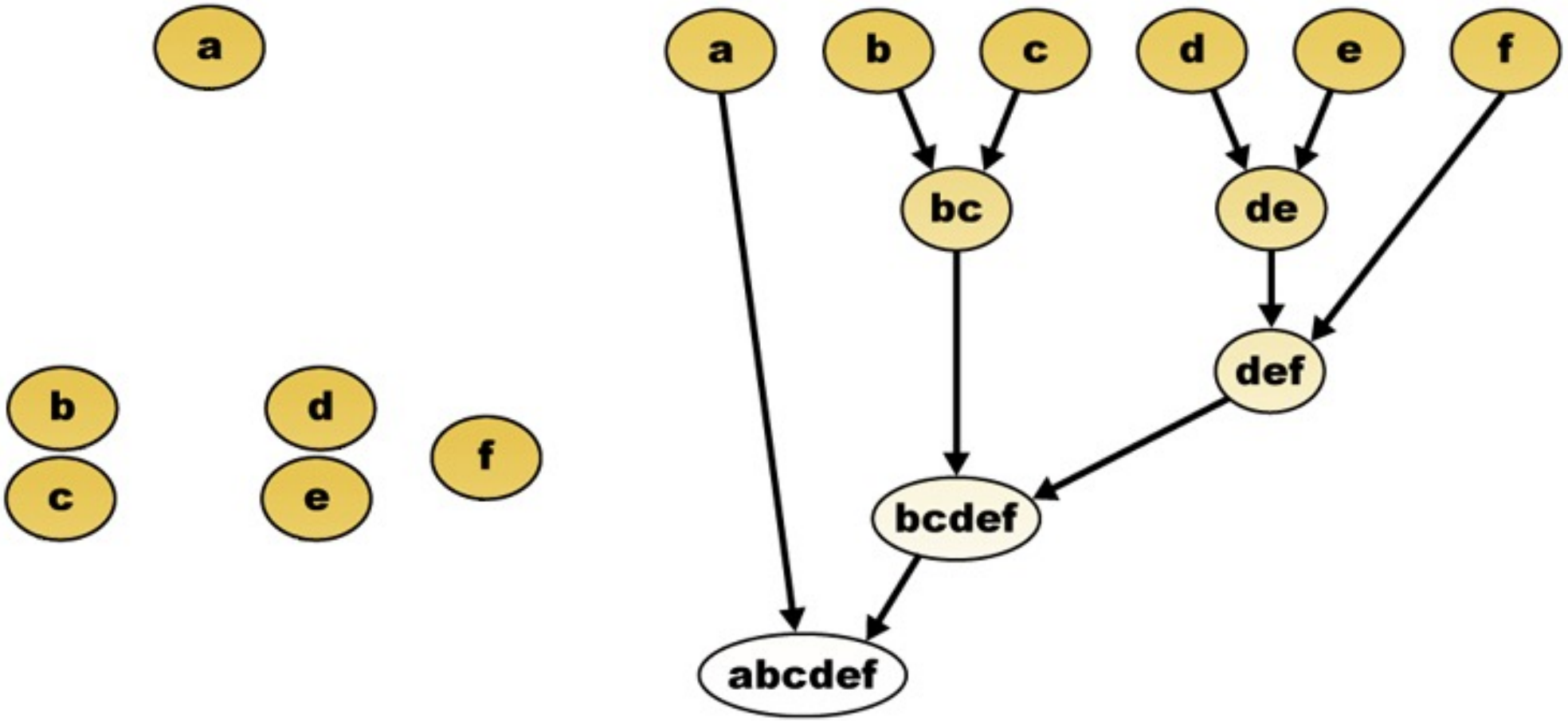
Eingabe: Anzahl der Elemente *n*, Abstandsmatrix ***d*** (mit den paarweisen Abständen zwischen allen Elementen)

Ausgabe: Ein Baum ***T*** als Repräsentation des hierarchischen Clusterings

Hierarchisches Clustern: Algorithmus

1. HierarchischesClustern(*n*, ***d***)
2. Bilde *n* Cluster mit je 1 Element
3. Konstruiere Graphen ***T***: Knoten sind Cluster
4. **while** (es gibt > 1 Cluster in ***d***)
5. Finde die Cluster C_1 und C_2 mit kleinstem $d[C_1,C_2]$
6. Verbinde C_1 und C_2 zu neuem Cluster *C*
7. Füge neuen Knoten *C* zu ***T*** hinzu
8. Füge Kanten von *C* zu C_1 und C_2 zu ***T*** hinzu
9. Entferne Zeilen und Spalten für C_1 und C_2 aus ***d***
10. Berechne den Abstand von *C* zu allen anderen Clustern
11. Füge Zeile und Spalte für *C* zu ***d*** hinzu
12. **return** ***T***

Hierarchisches Clustern: Beispiel



http://en.wikipedia.org/wiki/Data_clustering

Berechnung von Cluster-Abständen

Optimistischer Cluster-Abstand (single linkage):

Kürzester Abstand zwischen zwei beliebigen Elementen von *C* und *C**

$$d_{min}(C, C^*) = \min_{x \in C, y \in C^*} d[x,y]$$

Durchschnittlicher Cluster-Abstand (average linkage):

Mittelwert aller paarweisen Abstände zwischen Elementen aus *C* und *C**

$$d_{avg}(C, C^*) = \frac{1}{|C||C^*|} \sum_{x \in C, y \in C^*} d[x,y]$$

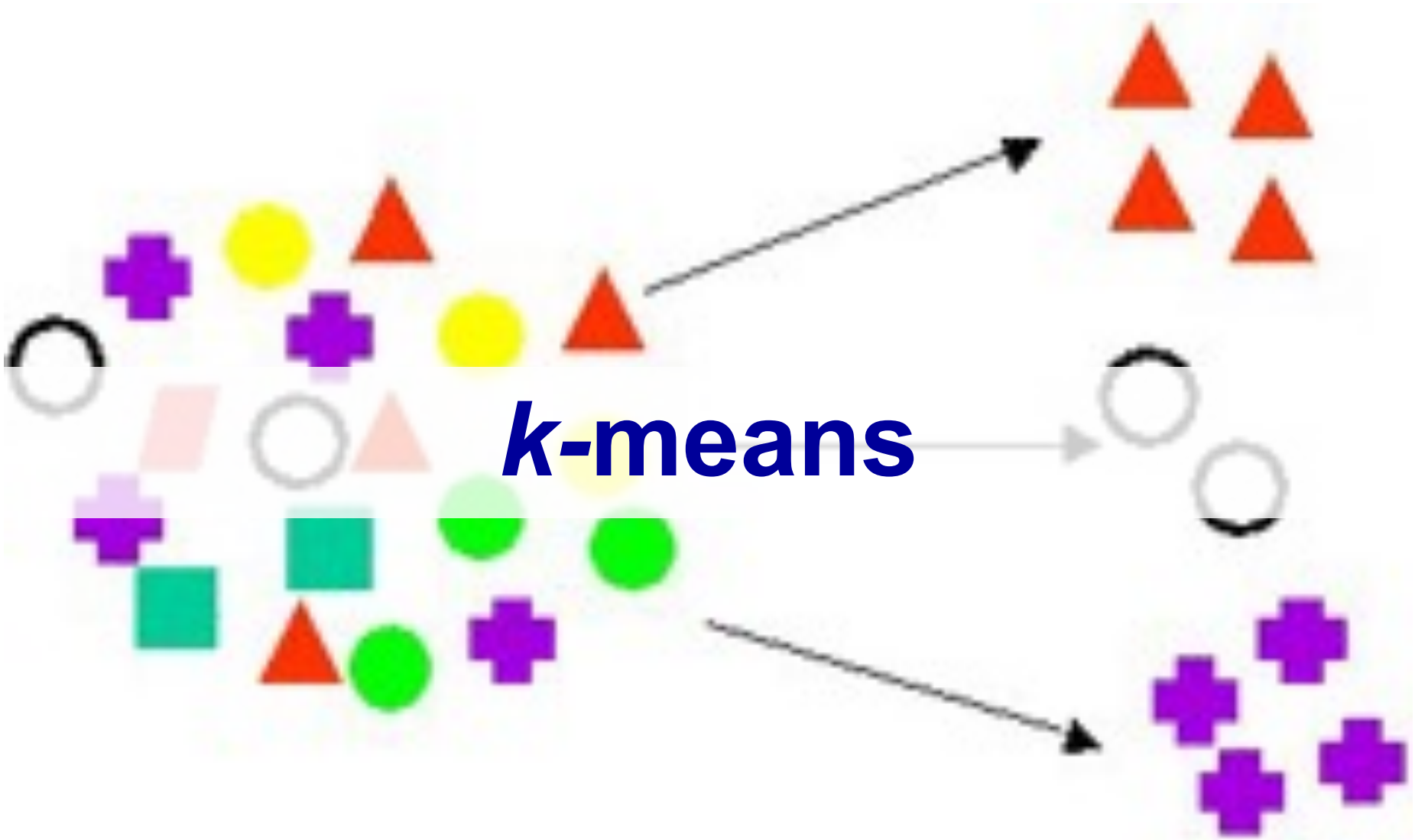
... und noch viel mehr ...

Hierarchisches Clustern: Algorithmus

	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8	g_9	g_{10}	g_{35}
g_1	0.0	8.1	9.2	7.7	9.3	2.3	5.1	10.2	6.1	7.0	9.2
g_2	8.1	0.0	12.0	0.9	12.0	9.5	10.1	12.8	2.0	1.0	12.0
g_3	9.2	12.0	0.0	11.2	0.7	11.1	8.1	1.1	10.5	11.5	
g_4	7.7	0.9	11.2	0.0	11.2	9.2	9.5	12.0	1.6	1.1	11.2
g_5	9.3	12.0	0.7	11.2	0.0	11.2	8.5	1.0	10.6	11.6	
g_6	2.3	9.5	11.1	9.2	11.2	0.0	5.6	12.1	7.7	8.5	11.1
g_7	5.1	10.1	8.1	9.5	8.5	5.6	0.0	9.1	8.3	9.3	8.1
g_8	10.2	12.8	1.1	12.0	1.0	12.1	9.1	0.0	11.4	12.4	1.0
g_9	6.1	2.0	10.5	1.6	10.6	7.7	8.3	11.4	0.0	1.1	10.5
g_{10}	7.0	1.0	11.5	1.1	11.6	8.5	9.3	12.4	1.1	0.0	11.5
g_{35}	9.2	12.0		11.2		11.1	8.1	1.0	10.5	11.5	

Hierarchisches Clustern

- Laufzeit $t = O(a \cdot b)$, wobei
  - a = Anzahl Aufrufe der while-Schleife
  - b = Zeit, um zwei nähste Cluster zu finden
  - $a =$ (Anzahl interne Knoten eines vollen Binärbaums)
  - $b =$
- Insgesamt: $t =$
  - Mit priority queue/sortierter Liste: $O(n^2 \log n)$


k-means

Squared Error Distortion

= Mittlerer quadratischer Abstand aller Punkte von “ihrem” Clusterzentrum

Für einen einzelnen (Daten-)Punkt *v* und eine Menge von (Clusterzentrum)-Punkten ***X***, definiere den Abstand von *v* zu ***X*** als:

$$d(v, \mathbf{X}) = \min_{x \in \mathbf{X}} \|v - x\|_2$$

(Euklidischer Abstand von *v* zum nächsten x)

Für eine Menge von *n* (Daten-)Punkten $\mathbf{V}=\{v_1 \ldots v_n\}$ und eine Menge von (Clusterzentrum-)Punkten ***X***, definiere die Squared Error Distortion als:

$$d(\mathbf{V},\mathbf{X}) = \frac{1}{n}\sum_{i=1}^{n}\left(d[v_i,\mathbf{X}]\right)^2$$

k-means Cluster: Problem

Ziel: Optimale Aufteilung von n Elementen
in eine vorgegebene Anzahl (k) von Clustern

Eingabe: Eine Menge $\boldsymbol{V}$ mit n Datenpunkten
und die gewünschte Clusteranzahl k

Ausgabe: Eine Menge $\boldsymbol{X}$ mit k Punkten (Clusterzentren)
mit minimaler Squared Error Distortion $d[\boldsymbol{V},\boldsymbol{X}]$
(verglichen mit allen anderen Mengen $\boldsymbol{X'}$)

Bsp.: 1-means Cluster

k-means Clustern ist NP-schwer für $k > 1$.

k-means Cluster: Lloyd Algorithmus

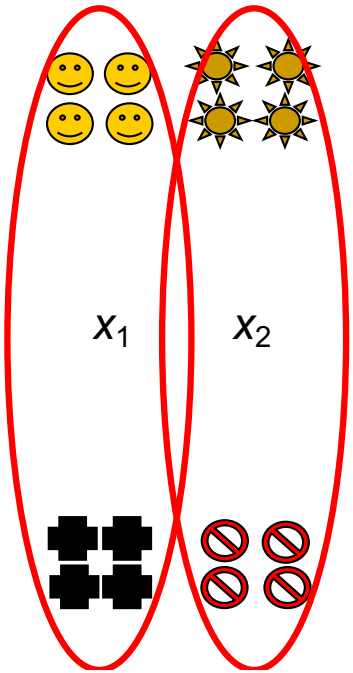
1. kmeansLloyd(***V***,*k*) **{**
2. Wähle *k* beliebige Clusterzentren $\boldsymbol{X} = \{x_1, \ldots x_k\}$
3. **while** (Clusterzentren verändern sich) {
4. Ordne jeden Datenpunkt $v \in \boldsymbol{V}$ dem Cluster C_i des nächstliegenden Zentrum $x_i \in \boldsymbol{X}$ zu
5. Berechne neue Clusterzentren $\boldsymbol{X} = \{x_1, \ldots x_k\}$ als den Schwerpunkt der zugehörigen Datenpunkte: $x_i = \frac{1}{|C_i|} \sum_{v \in C_i} v$

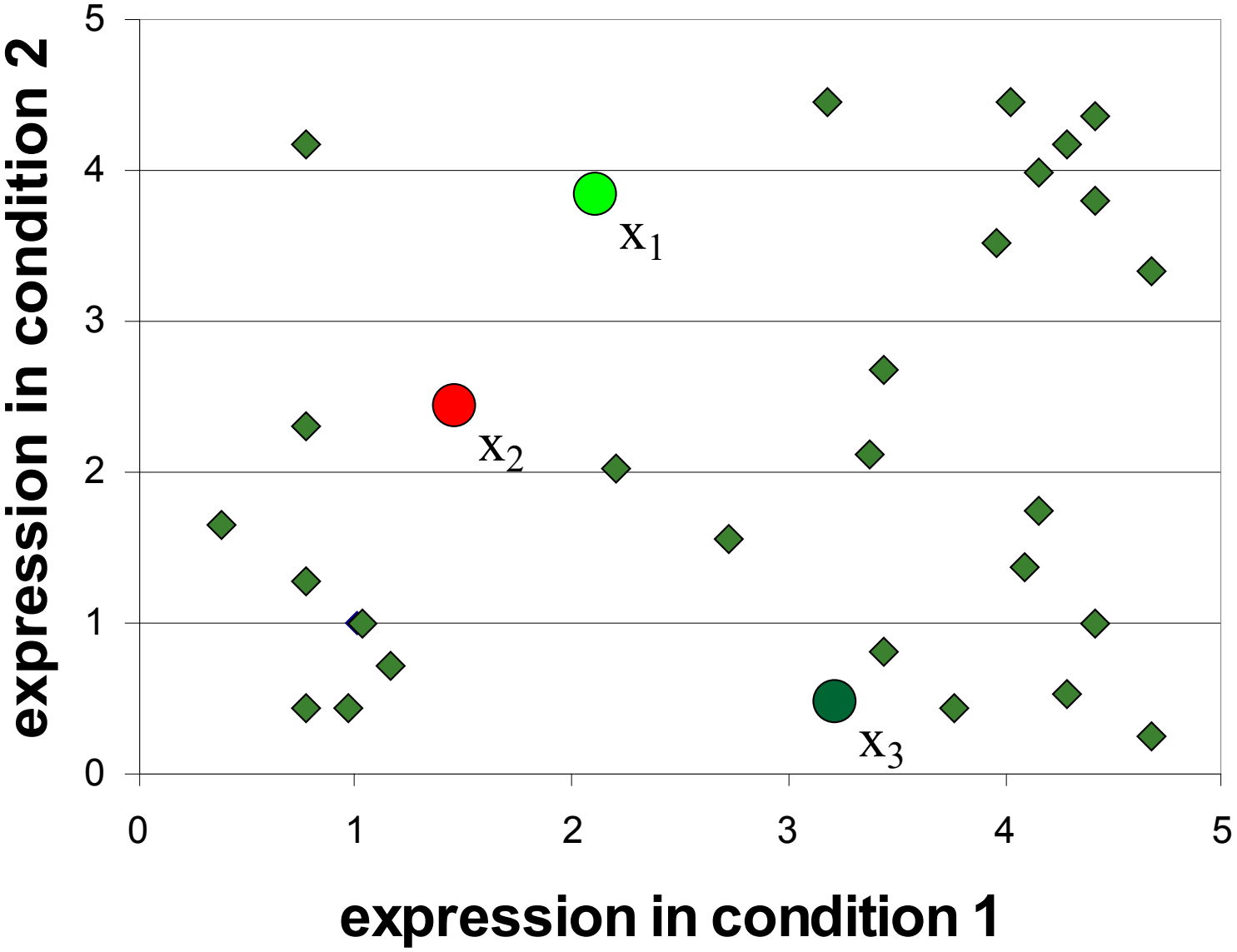
   }
6. **return** ***X***
7. *}*

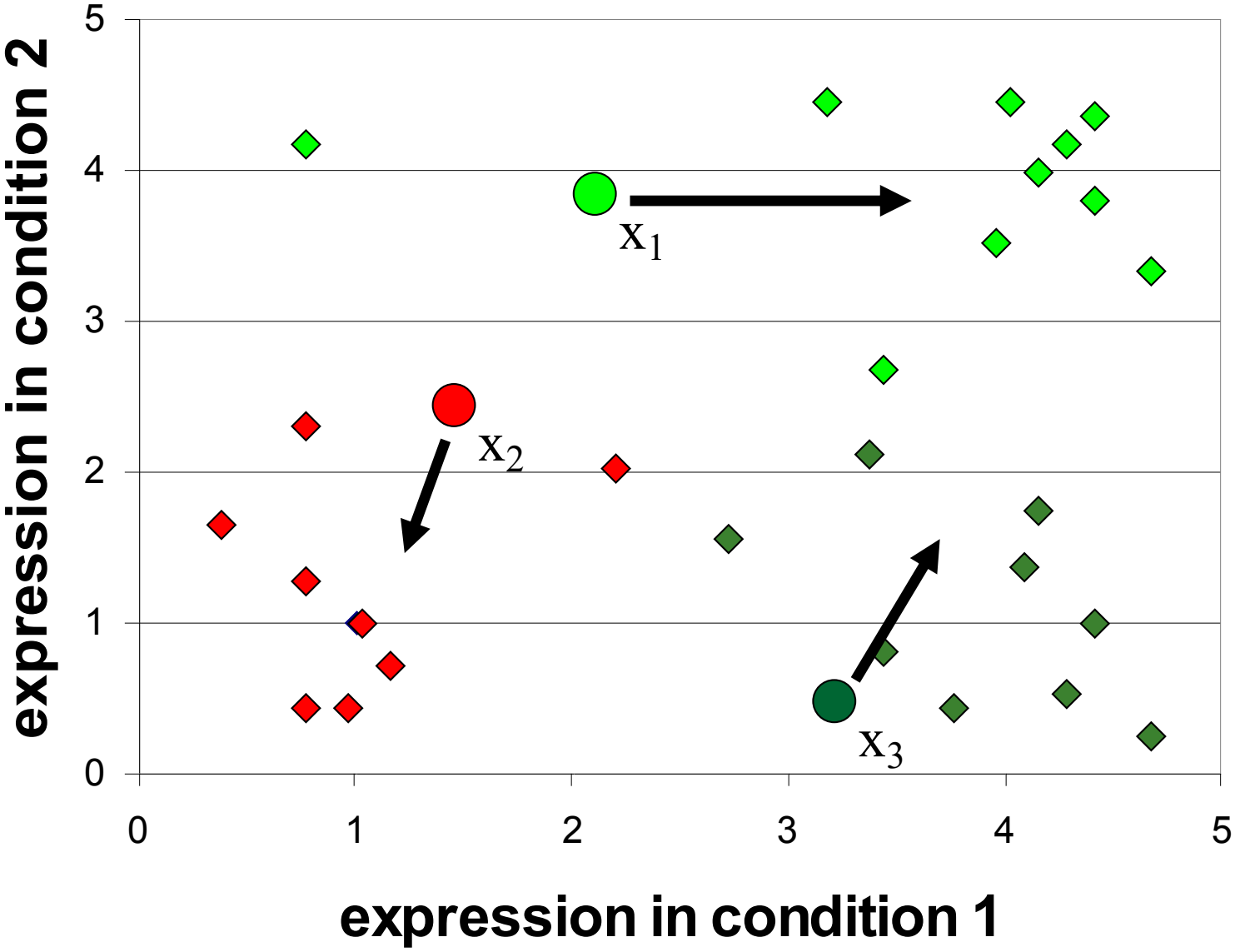
Ist der Algorithmus exakt?

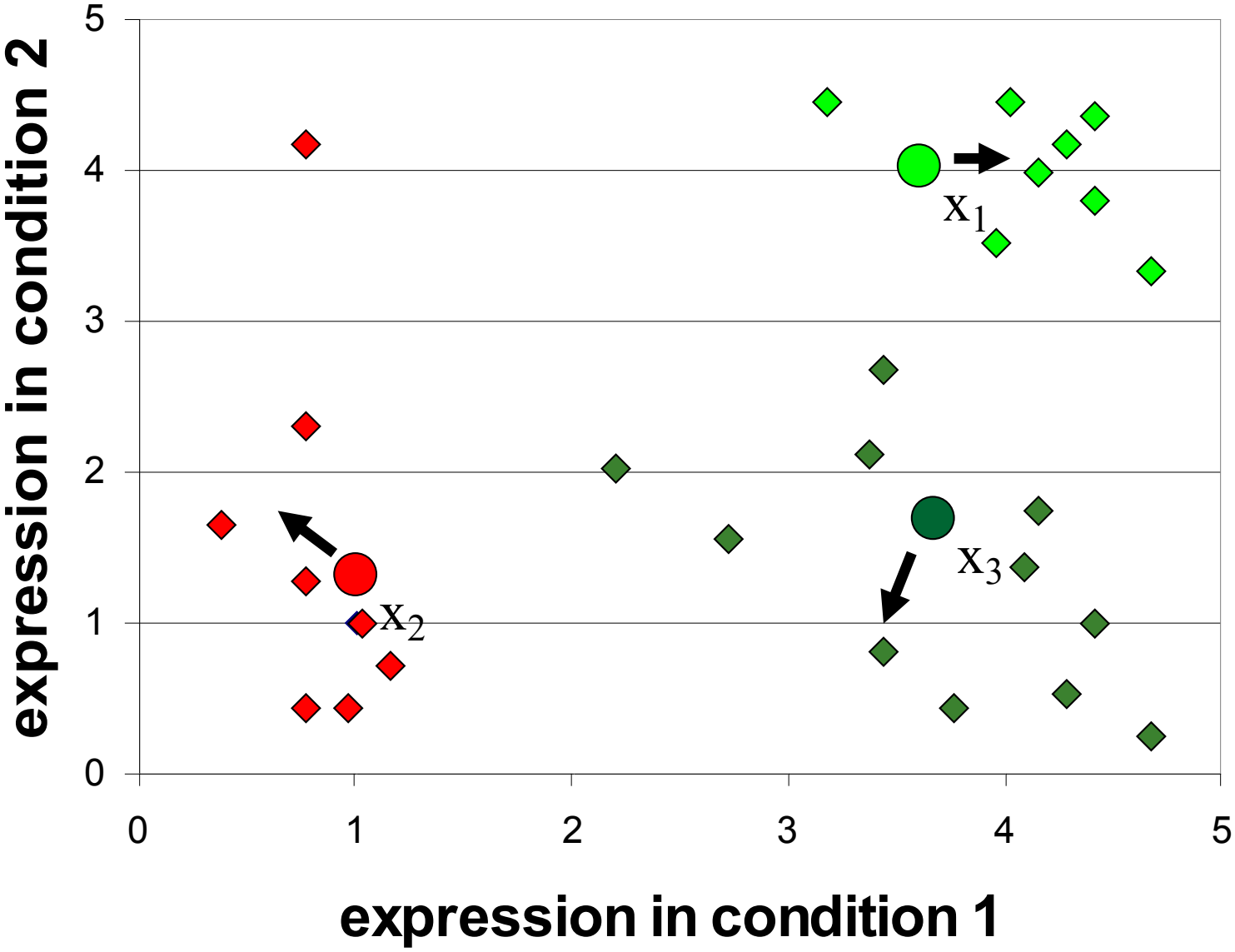
Kann die Schleife ewig laufen?

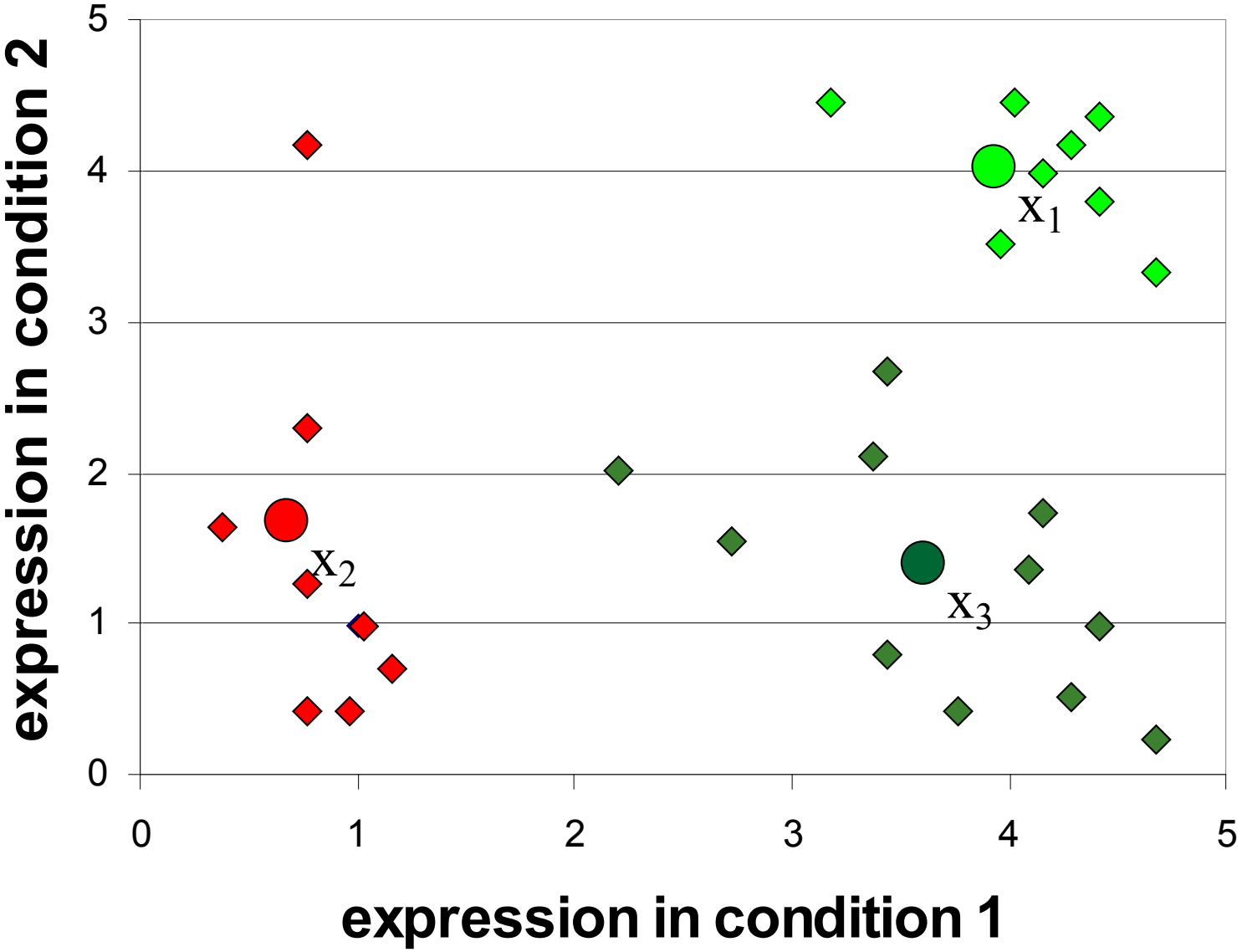
Jede Veränderung führt zu einer Verbesserung der SED – kann das unendlich oft passieren?



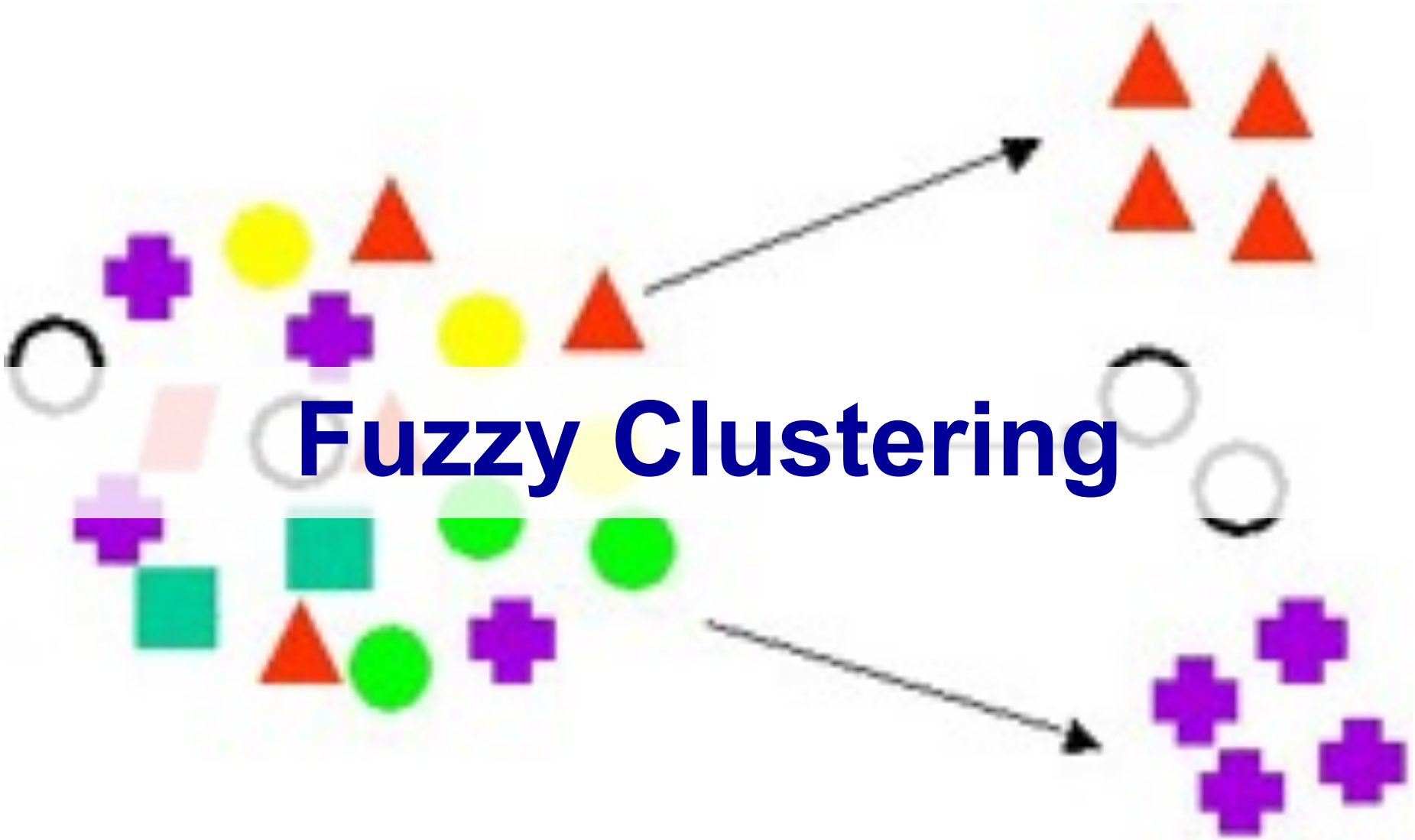

expression in condition 2
expression in condition 1
0
1
2
3
4
5
x1
x2
x3


expression in condition 2
expression in condition 1
0
1
2
3
4
5
x1
x2
x3


expression in condition 2
expression in condition 1
0
1
2
3
4
5
x1
x2
x3


expression in condition 2
expression in condition 1
0
1
2
3
4
5
x1
x2
x3

Fuzzy Clustering



Fuzzy *c*-means Cluster

Fuzzy bedeutet hier: zu welchem Cluster ein Punkt gehört, ist nicht eindeutig festgelegt - sondern nur als Wahrscheinlichkeit $u_k(x)$, mit

$$\forall x \quad \sum_{k=1}^{\text{num. clusters}} u_k(x) = 1.$$

Das Clusterzentrum ist dann gewichtet (mit Fuzzyness m>1):

$$\text{center}_k = \frac{\sum_x u_k(x)^m x}{\sum_x u_k(x)^m}.$$

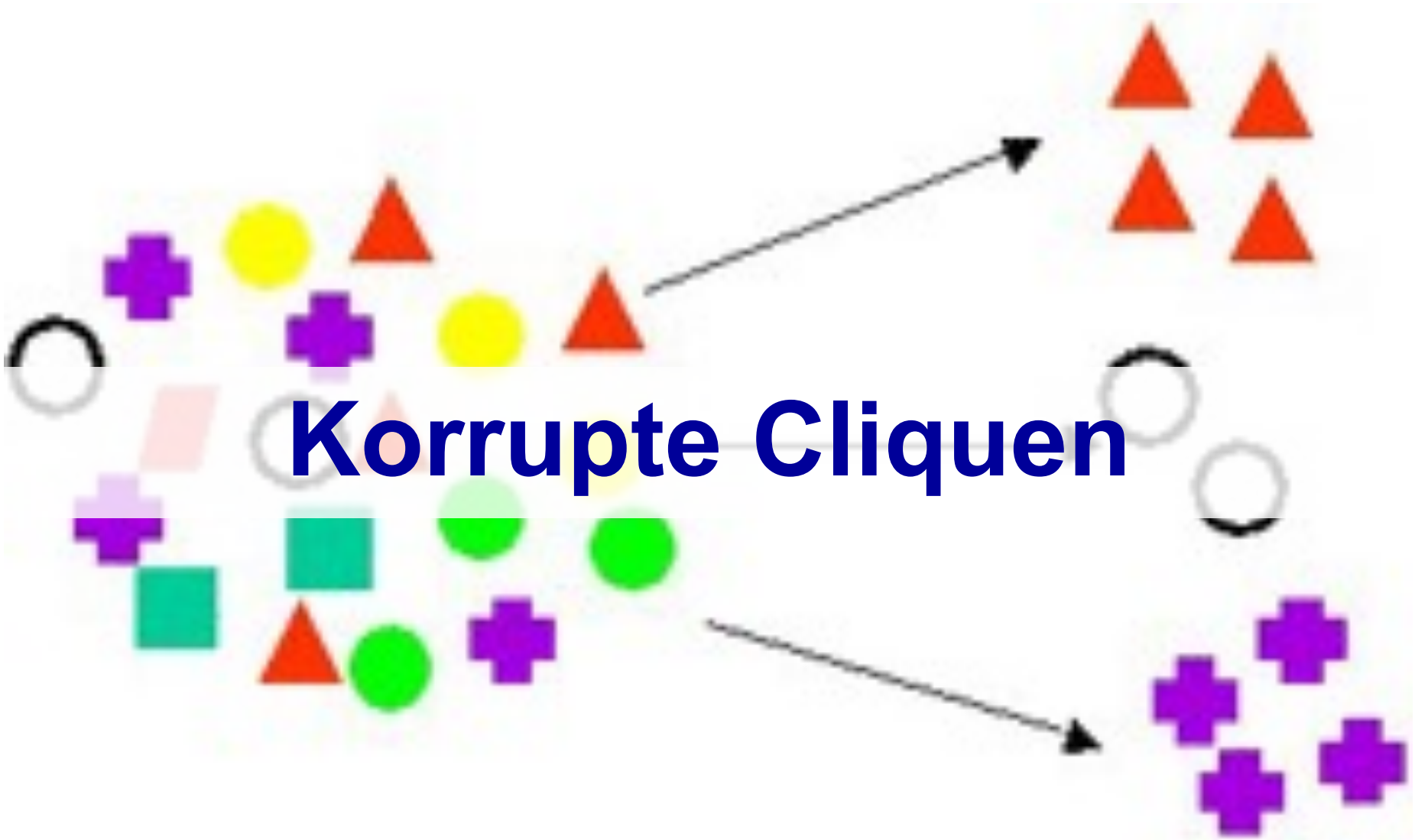
Die Wahrscheinlichkeiten können relativ zum Abstand des Punktes x vom Clusterzentrum definiert werden (normalisiert & mit Fuzzyness):

$$u_k(x) = \frac{1}{\sum_j \left(\frac{d(\text{center}_k, x)}{d(\text{center}_j, x)} \right)^{2/(m-1)}}.$$

m~1: starkes Gewicht für nächstes Clusterzentrum.

Ansonsten ist alles wie bei *k*-means Clustern!

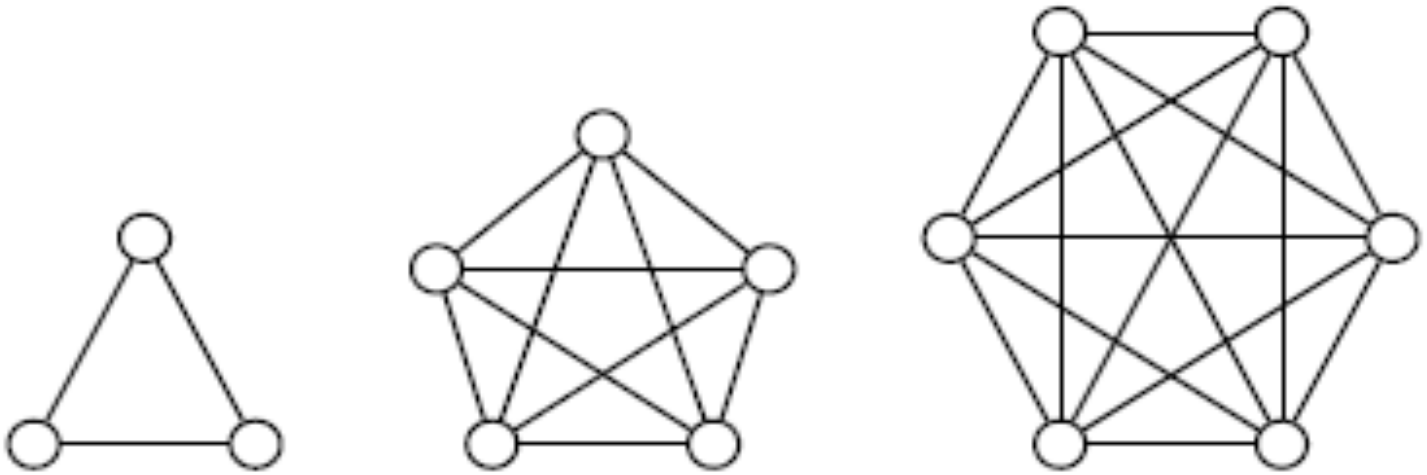
http://en.wikipedia.org/wiki/Data_clustering


Korrupte Cliquen

Cliquen-Graphen

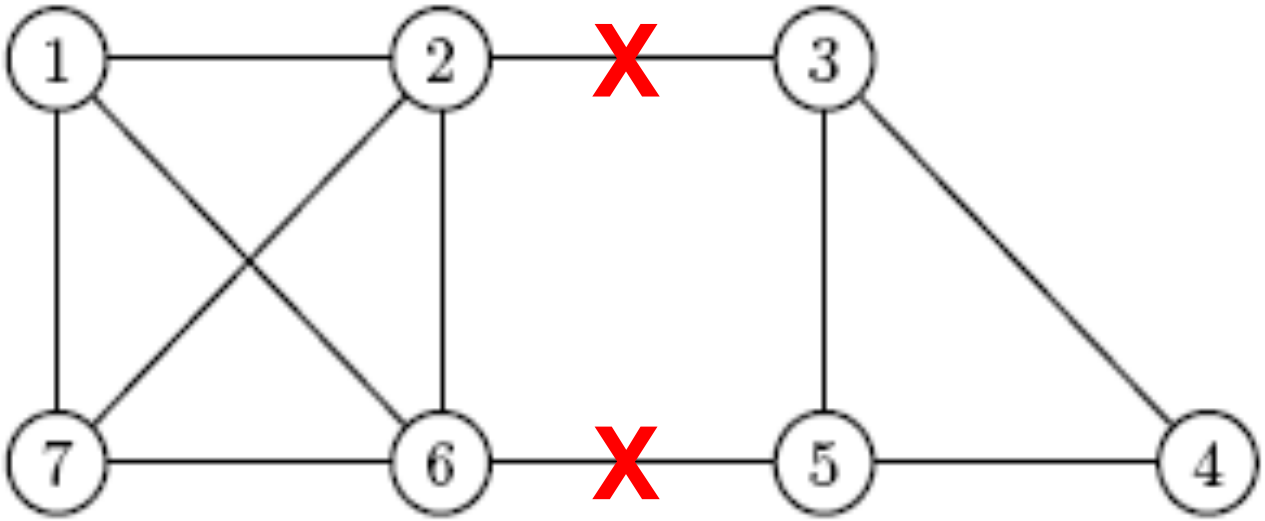
Def.: Eine **Clique** ist ein Teilgraph bei dem jeder Knoten mit jedem anderen verbunden ist.

Def.: Ein **Cliquen-Graph** ist ein Graph, bei dem jede Zusammenhangskomponente eine Clique ist.



Transformation in einen Cliquen-Graph

...durch die Hinzufügung oder Entfernung (möglichst weniger) Kanten



Der Ursprungsgraph hat '**korrupte**' Cliquen.

Cliquen-Graphen und Clustern

Verwandele die Abstandsmatrix ***d*** in einen Abstandsgraphen:

- Wähle einen Abstands-Cutoff θ
- Datenpunkte (Gene) sind die Knoten
- Wenn der Abstand zweier Knoten $d[v,v'] \leq \theta$ ist, füge eine Kante hinzu

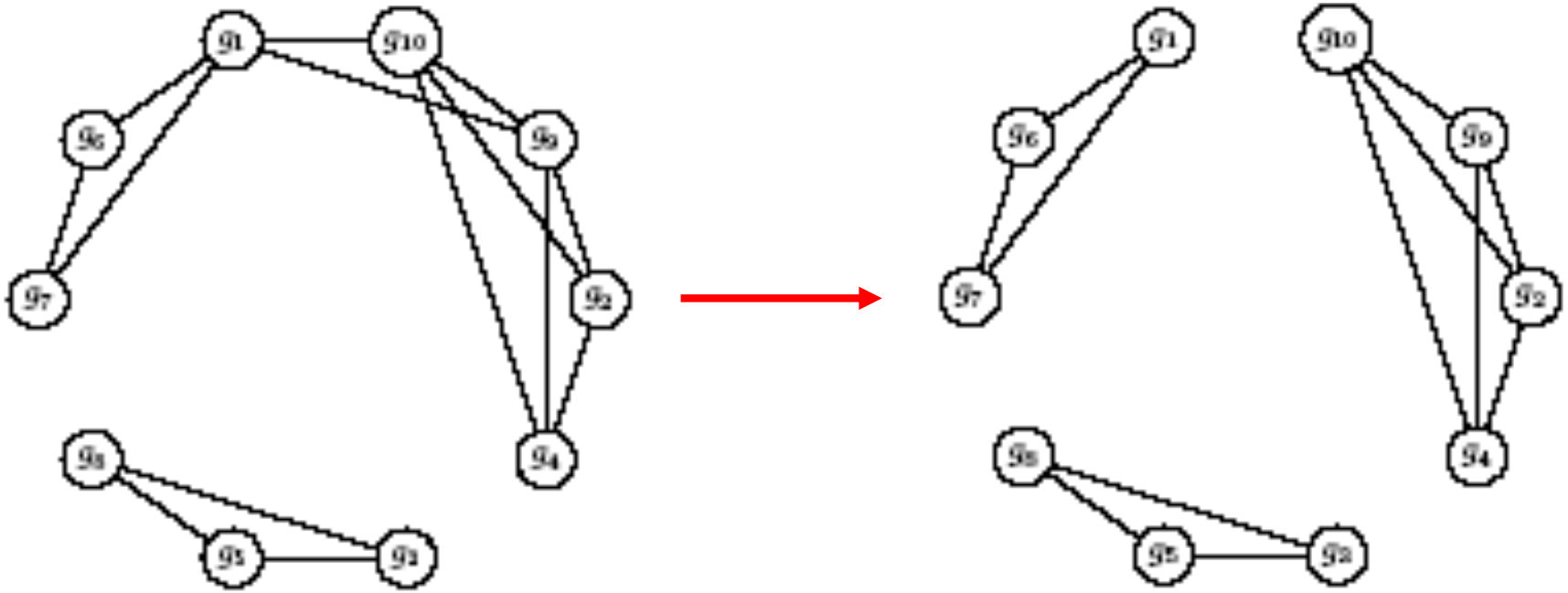
Der resultierende Graph enthält (wahrscheinlich korrupte) Cliquen:
diese repräsentieren ‘ähnliche’ Datenpunkte!

$\Rightarrow$ Transformiere den Abstandsgraphen in einen Cliquengraphen

Abstandsgraph → Cliquen-Graph

Abstands-Cutoff $\theta = 7$

	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8	g_9	g_{10}
g_1	0.0	8.1	9.2	7.7	9.3	2.3	5.1	10.2	6.1	7.0
g_2	8.1	0.0	12.0	0.9	12.0	9.5	10.1	12.8	2.0	1.0
g_3	9.2	12.0	0.0	11.2	0.7	11.1	8.1	1.1	10.5	11.5
g_4	7.7	0.9	11.2	0.0	11.2	9.2	9.5	12.0	1.6	1.1
g_5	9.3	12.0	0.7	11.2	0.0	11.2	8.5	1.0	10.6	11.6
g_6	2.3	9.5	11.1	9.2	11.2	0.0	5.6	12.1	7.7	8.5
g_7	5.1	10.1	8.1	9.5	8.5	5.6	0.0	9.1	8.3	9.3
g_8	10.2	12.8	1.1	12.0	1.0	12.1	9.1	0.0	11.4	12.4
g_9	6.1	2.0	10.5	1.6	10.6	7.7	8.3	11.4	0.0	1.1
g_{10}	7.0	1.0	11.5	1.1	11.6	8.5	9.3	12.4	1.1	0.0



Korrupte Cliquen: Problem

Ziel: Transformiere einen beliebigen (korrupten) Graphen mit möglichst wenigen Operationen in einen Cliquen-Graphen.

Eingabe: Ein Graph ***G***

Ausgabe: n_H Hinzufügungen und n_E Entfernungen von Kanten, die gemeinsam ***G*** in einen Cliquen-Graphen überführen, so dass n_H+n_E minimal ist.

NP-schwer!

Korrupte Cliquen: Brachial

complete_enumeration_CE(G):

    $c = \infty$

    für alle möglichen Partitionen P von V

        if (cost(P) < c):

            c = cost(P)

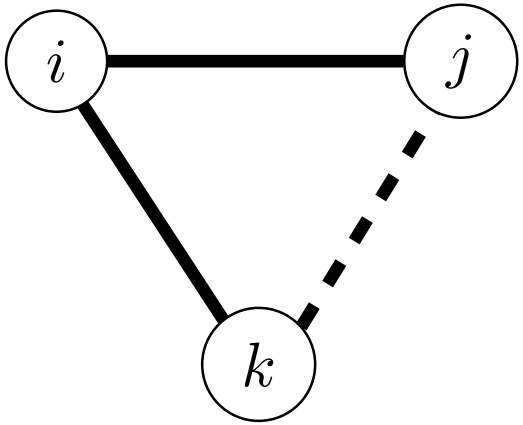
            $P_{out} = P$;

    return P_{out}

Sieht harmlos aus, ist aber furchtbar langsam. Es gibt viel viel mehr als $2^{n-1} - 1$ Partitionen von V .

Korrupte Cliquen: clevere Enumeration

- Ein **Konflikttripel** {*i*, *j*, *k*} ist ein Knotentripel mit (*i*,*j*) und (*i*,*k*) in *E* aber (*j*,*k*) nicht in *E*



- **Lemma:** *G* ist Cliquen-Graph genau dann wenn *G* kein Konflikttripel enthält.

Korrupte Cliquen: clevere Enumeration

// Transformiere $G = (V, E)$ in einen Cliquengraphen mit Kosten ≤ k

smarter_enum(G, k):

if G Cliquengraph **return** G

if $k = 0$ **return** {}

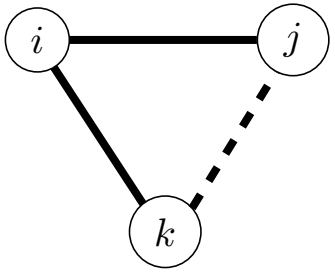
Finde Konfliktripel $\{i, j, k\}$ in G

smarter_enum(G – (i,j), k-1);

smarter_enum(G – (i,k), k-1);

smarter_enum(G + (k,j), k-1);

Laufzeit
$O(3^k n^3)$



→ Laufzeit gar nicht mehr so schlimm:
$O(3^{k^*} n^3)$, wobei k^* die minimale Anzahl der Modifikationen ist

Fixed-parameter tractability

- Ziel: exakter Algorithmus für NP-vollständiges Problem
- Problemgröße n, **Parameter k** (viel kleiner als n)

- Idee: akzeptiere kombinatorische Explosion, aber begrenze diese auf einen eher kleinen Parameter k.
- Formell: Algorithm hat Laufzeit $O(f(k)n^{O(1)})$

Intermezzo: Integer Linear Programming (ILP)

- Model problem as ILP → solve ILP → exact algorithm

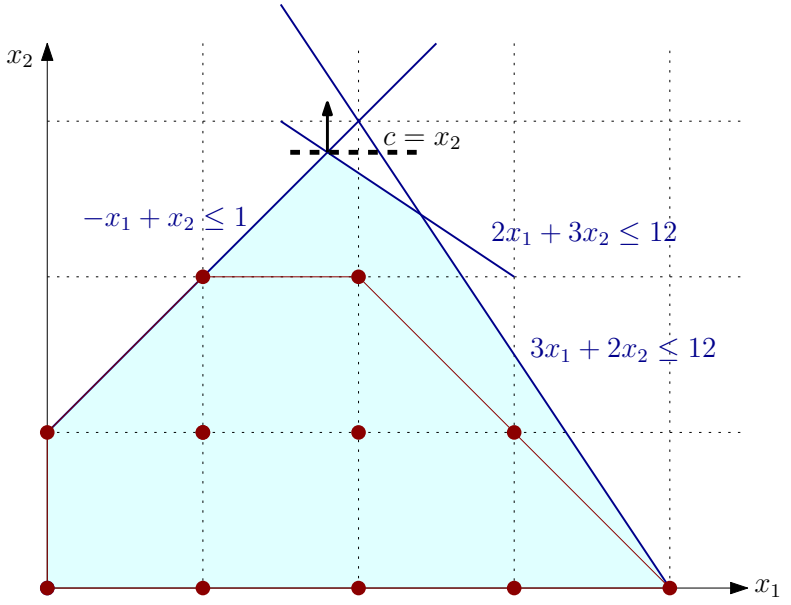
standard form

$$\max\ c^T x \quad \text{(objective function)}$$
$$\text{subject to}\ Ax \leq b \quad \text{(constraints)}$$
$$x \text{ integer}$$

- solve using branch-and-bound, use, e.g., LP bound

example

$$\max\ x_2$$
$$\text{s.t.}\ -x_1 + x_2 \leq 1$$
$$3x_1 + 2x_2 \leq 12$$
$$2x_1 + 3x_2 \leq 12$$
$$x_1, x_2 \geq 0,\ \text{integer}$$

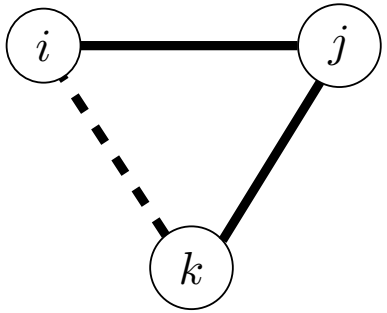


Cluster editing: Integer Linear Program

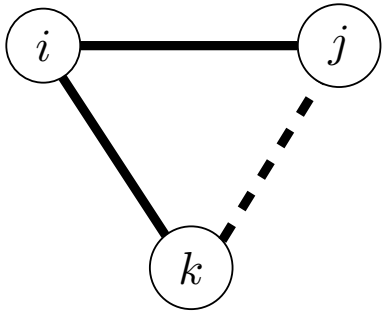
- Für jedes Knotenpaar $\{i, j\}$: Binärvariable x_{ij} mit der Interpretation

$$x_{ij} = \begin{cases} 1 & (i,j) \text{ in solution} \\ 0 & (i,j) \text{ not in solution} \end{cases}$$

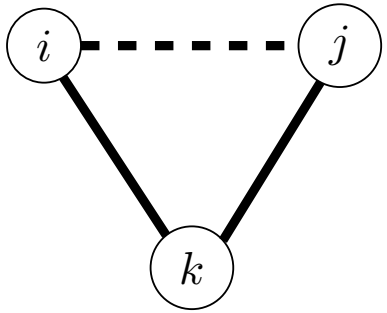
- Schließe Konflikttripel aus: Für alle $\{i, j, k\} \subset V$



$$x_{ij} + x_{jk} - x_{ik} \leq 1$$



$$x_{ij} - x_{jk} + x_{ik} \leq 1$$



$$-x_{ij} + x_{jk} + x_{ik} \leq 1$$

- Zielfunktion:

$$\min \sum_{ij \in E} \qquad + \sum_{ij \notin E}$$

Cluster statistics (Randomization; Lukzsa)

- Jede Anwendung von k-means wird Cluster liefern – auch wenn die Daten rein zufällig verteilt sind
- Vergleiche die Squared Error Distortion (SED) der Daten mit der SED von zufälligen Daten.
- Erzeuge zufällige Daten durch Randomisierung: …
- Sehr aufwändiges Verfahren bei großen Datensätzen

Alternative: parametrischer statistischer Test (Lukzsa, Berg, Lässig)

- Voraussetzung: Daten müssen unabhängig verteilt sein (für biologische Daten meist nicht gegeben)
- Vorteil: schnell zu berechnen

Caveat

Erst anschauen, dann clustern!