Grundlagen der Künstlichen Intelligenz

18 Unüberwachtes Lernen

Clusteranalyse: partitionierende,
hierarchische und dichtebasierte Ansätze

Volker Steinhage

Inhalt

- Unüberwachtes Lernen ⇝ Clusteranalyse
- Partitionsverfahren ⇝ k-Means
- Hierarchisches Clustering ⇝ Agglomeratives Clustering
- Dichtebasiertes Clustering ⇝ DBSCAN & OPTICS

Unüberwachtes Lernen (1)

- *Unüberwachtes* Lernen (*Unsupervised* Learning): die Trainingsmenge enthält nur Eingabewerte.

↝ keine explizite Rückkopplung in Form korrespondierender richtiger Ausgabewerte!

↝ Keine implizite Rückkopplung in Form korrespondierender Verstärkungssignale (Gewinne/Kosten)!

↝ Der Agent kann nur Modelle für das Auftreten von *Mustern* bzw. *Regelmäßigkeiten* in seinen Beobachtungen lernen, aber *nicht*, was er richtigerweise tun müsste.

Unüberwachtes Lernen (2)

Kaufverhalten

- Jedes Eingabetupel besteht aus Verkaufszahlen von verschiedenen Produkten sowie kaufsituationsbeschreibenden Größen (Wetter, Wochentag, Tageszeit, …).

- Der Agent sucht nach Mustern, die Zusammenhänge zwischen Produktkäufen und Kaufsituationen aufdecken.

Ansätze für das unüberwachte Lernen

Unüberwachtes Lernen kann auf zwei Arten umgesetzt werden

In dieser Vorlesung

- Clusteranalyse (auch kurz Clustering oder Ballungsanalyse): durch Gruppenzuordnung (engl. *Clustering*) werden die Datensätze derart aufgeteilt, dass Gruppen (Anhäufungen, engl. Cluster) von „*ähnlichen*“ Datensätzen entstehen

- Dimensionsreduktion reduziert die Zahl der die Datensätze beschreibenden Attribute durch die

Nicht in dieser Vorlesung

  – Auswahl von relevanten Attributen aus der Gesamtmenge aller Attribute (engl. *Feature Selection*)

  oder

  – Erzeugung einer kleineren Menge beschreibender Attribute (engl. *Feature Extraction*).

Clusteranalyse

- Ziel der unüberwachten Clusteranalyse ist also:
  - Die Aufteilung einer bzgl. der Werte ihrer Attribute heterogenen Gesamtmenge von Datensätzen in Teilgruppen (Cluster) derart, dass die Datensätze jeder Teilgruppe in sich möglichst homogen hinsichtlich ihrer Attributwerte sind.

- Es gibt verschiedene Ansätze der Clusteranalyse.
  - Beginnen wir mit einem der einfachsten Ansätze der Clusteranalyse, dem k-Means-Algorithmus.

k-Means (1)

- Der k-Means-Algorithmus

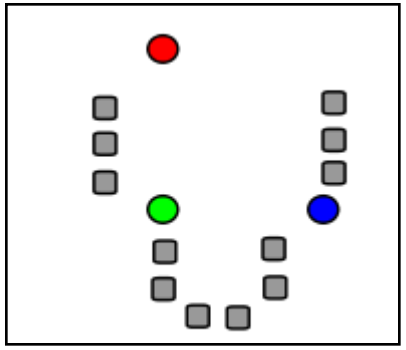
  – erzeugt eine a priori vorgegebene Anzahl von *k* Clustern aus einer Menge von Datensätzen;

  – ist eine der meist verwendeten Techniken zur Clusteranalyse, da er die Zentren der Cluster schnell findet;

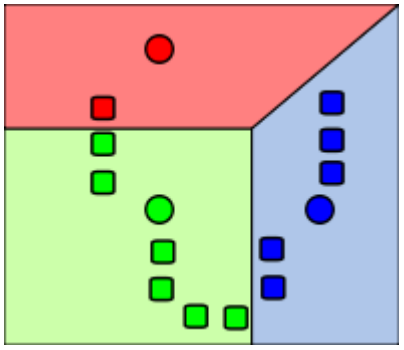
  – zeichnet sich durch große Einfachheit aus.

k-Means (2)

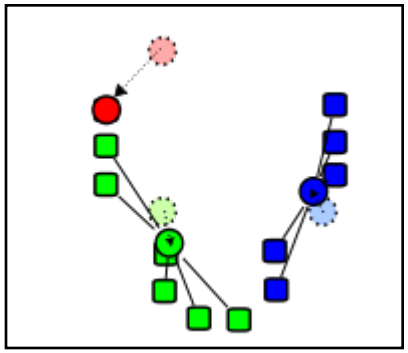
Beispiel für Datensätze mit zwei beschreibenden Attributen:



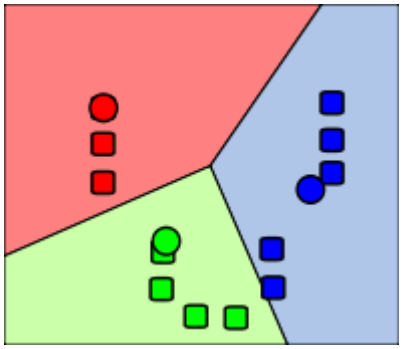
k = 3 initiale Zentren zufällig gewählt



k = 3 Cluster mit Zuordnung der Datenpunkte zu den nächsten Zentren



Neuberechnung der Zentren



Wiederholte Neuberechnungen von Zentren und Cluster-Zuordnungen

Quelle:http://de.wikipedia.org/wiki/K-Means-Algorithmus (18.06.12)

k-Means (3)

Die Schritte vom ***k-Means-Algorithmus***:

(0) Vor Ausführung ist die Anzahl *k* der zu ermittelnden Cluster festzulegen.

(1) Die *k Cluster-Schwerpunkte* werden *zufällig* im Datenraum verteilt.

(2) Datenzuordnung: Jeder Datensatz wird demjenigen Cluster zugeordnet, dessen Schwerpunkt ihm am nächsten liegt.*

(3) Schwerpunkteberechnung: Nach der Neuzuordnung der Datensätze werden die Schwerpunkte aller Cluster neu berechnet.

(4) Gehe zu Schritt (2), bis

- eine festgelegte maximale Zahl von Iterationen erreicht wird … oder
- die Positionen der Schwerpunkte stabil bleiben (d.h. keine Neuverteilung der Datensätze erfolgt).

* Unter Verwendung einer Distanzfunktion wie z.B. der Euklid. Distanz.

k-Means (4)

Die zentralen Schritte des ***k*-Means-Algorithmus** konkreter:

(1) Initialisierung: zufällige Vorgabe von k Schwerpunkten $\mathbf{m}_1^{(1)}, \ldots, \mathbf{m}_k^{(1)}$.

(2) Neuzuordnung der Datensätze $\mathbf{x}_j$ zu dem Cluster $\mathbf{S}_i^{(t)}$ in Iteration t:

$$\mathbf{S}_i^{(t)} = \{ \mathbf{x}_j \text{ mit } \|\mathbf{x}_j - \mathbf{m}_i^{(t)}\| \leq \|\mathbf{x}_j - \mathbf{m}_{i'}^{(t)}\| \forall\, i' \in \{1,\ldots,k\} \setminus \{i\} \}$$

(3) Neuberechnung der Schwerpunkte:

$$\boldsymbol{m}_i^{(t+1)} = \frac{1}{\left|\boldsymbol{S}_i^{(t)}\right|} \cdot \sum_{\boldsymbol{x}_j \in \boldsymbol{S}_i^{(t)}} \boldsymbol{x}_j \ .$$

(4) Terminierung, wenn keine Neuzuordnungen in (2)

Der Algorithmus versucht also, die *Kompaktheit* aller Cluster $S = \{S_1,\ldots,S_k\}$ zu *maximieren*:

$$\arg\min_{\boldsymbol{S}} = \sum_{i=1}^{k} \sum_{\boldsymbol{x}_j \in \boldsymbol{S}_i} \left\| \boldsymbol{x}_j - m_i \right\|.$$

k-Means (5)

Bewertung:

- Heuristischer Ansatz:
  - das Ergebnis kann von der Initialisierung abhängig sein,
  - das globale Optimum wird nicht garantiert erreicht.
- Zeitkomplexität:
  - im Worst Case exponentiell in der Zahl der Datensätze,
  - im Average Case polynomiell in der Zahl der Datensätze.
- Cluster-Modell:
  - geht von sphärischen Clustern ähnlicher Größe aus, da die Zuordnung der Datensätze nach *minimaler Distanz* zu den Clusterzentren erfolgt.
- Anzahl der Cluster muss vorgegeben werden!

Ansätze der Clusteranalyse

Der k-Means-Algorithmus ist nur eine von vielen Methoden der Clusteranalyse.

Die Verfahren der Clusteranalyse lassen sich einordnen in

- partitionierende Verfahren,
- hierarchische Verfahren,
- dichtebasierte Verfahren,
- graphentheoretische Verfahren,
- andere Verfahren.

In dieser Vorlesung werden exemplarisch Ansätze für partitionierende Verfahren, hierarchische Verfahren und dichtebasierte Verfahren vorgestellt.

Partitionierende Clusterverfahren

Partitionierende Verfahren

- verwenden eine initiale Partitionierung aller Datensätze,
- ordnen die Datensätze durch Austauschfunktionen solange um, bis die verwendete Zielfunktion ein Optimum erreicht.
- Zusätzliche Cluster können nicht gebildet werden, da die Anzahl der Cluster bereits am Anfang festgelegt wird.

⇝ Der vorgestellte *k*-Means-Algorithmus ist ein partitionierendes Verfahren. ✔

Hierarchische Clusterverfahren

Hierarchische Verfahren zeigen zwei Varianten

- Agglomerative Verfahren
  - Start mit feinster Partition: jeder Datensatz bildet ein eigenes Cluster.
  - Prozess: schrittweise Bildung größere Cluster durch Zusammenfassung von Clustern ähnlicher Datensätze.
- Divisive Verfahren
  - Start mit gröbster Partition = Gesamtheit aller Elemente
  - Prozess: schrittweise disjunktive Unterteilung in Cluster mit Daten größerer Ähnlichkeit.
- Agglomerative Verfahren kommen in der Praxis häufiger vor.

Terminierung hierarchischer Clusterverfahren

Kriterien zur Terminierung agglomerativer Verfahren:

- Maximale Variation innerhalb der Cluster: alle Cluster zeigen eine maximale Distanz zwischen ihren Elementen, die nicht mehr überschritten werden soll. Weitere Elemente aus anderen Clustern würden zu große Unähnlichkeiten innerhalb der Cluster erzeugen.
- Minimale Distanz zwischen Clustern: alle Cluster zeigen eine minimale Distanz untereinander, die nicht überschritten werden soll, sonst würden Agglomerationsschritte mit diesen Clustern zu Fusionen von zu unähnlichen Elementen führen.
- Zahl von Clustern: Eine hinreichend kleine Zahl von Clustern ist ermittelt worden.

Analog angepasst für divisive Verfahren.

Agglomeratives Clustering durch Single-Linkage Clustering (1)

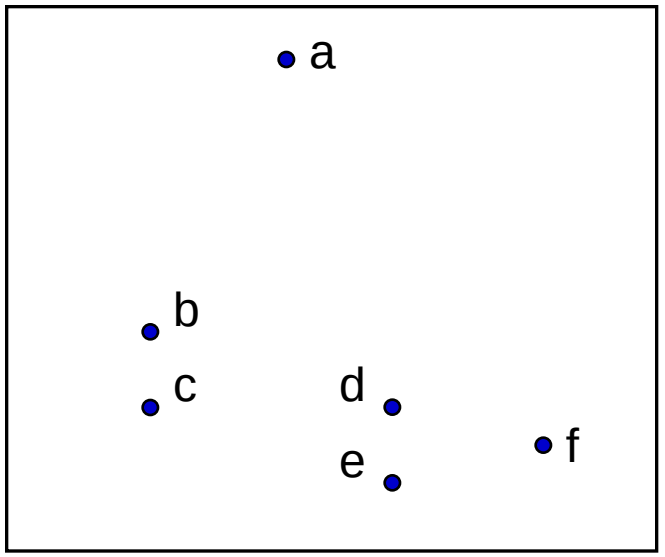
Beispiel: gegeben seien 6 Datensätze *a, ... ,f* in einem 2-dim. Datenraum. Die Euklidische Distanz sei als Distanzmaß anwendbar.

Start: jeder Datensatz bildet ein eigenes Cluster, also Cluster {*a*} {*b*} {*c*} {*d*} {*e*} und {*f*}.

Zusammenfassung von ähnlichen Clustern erfolgt im *Single-linkage Clustering* nach der *minimalen Distanz* zwischen Elementen verschiedener Cluster C_1, C_2 :

$$\min \{ d(x,y) \mid x \in C_1, y \in C_2 \}.$$

Eine 6×6 *Distanzmatrix* kodiert im Eintrag (*i,j*) die min. Distanz zwischen *i*-tem und *j*-tem Cluster. Die Zusammenfassung von Clustern entspricht der Zusammenfassung von Spalten und Zeilen der Matrix.

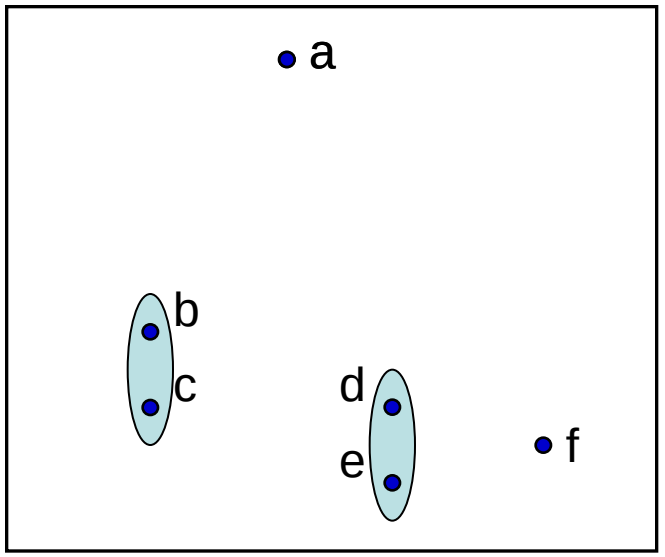


	a	*b*	*c*	*d*	*e*	*f*
a	0					
b	4.0	0				
c	5.0	1.2	0			
d	5.0	3.3	3.1	0		
e	5.8	3.6	3.2	1.2	0	
f	6.1	5.4	5.2	2.3	2.3	0

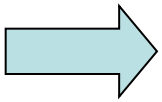
Agglomeratives Clustering durch Single-Linkage Clustering (2)

Die Cluster {*b*} und {*c*} sowie {*d*} und {*e*} zeigen den min. Abstand von 1.2 und werden zu neuen Clustern {*b*,*c*} bzw. {*d*,*e*} zusammengeführt.

Die Zeilen und Spalten für die Cluster {*b*}, {*c*}, {*d*} und {*e*} werden gelöscht und ersetzt durch neue Spalten und Zeilen für die neuen Cluster {*b*,*c*} und {*d*,*e*}:



	a	*b*	*c*	*d*	*e*	*f*
a	0					
b	4.0	0				
c	5.0	1.2	0			
d	5.0	3.3	3.1	0		
e	5.8	3.6	3.2	1.2	0	
f	6.1	5.4	5.2	2.3	2.3	0

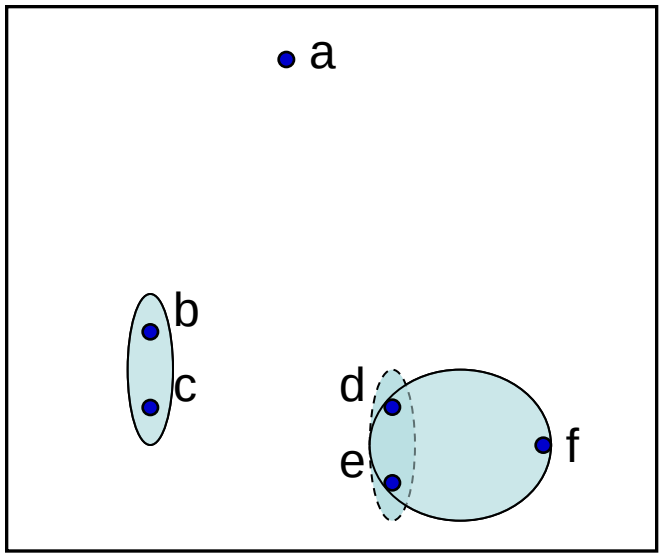


	a	*b* *c*	*d* *e*	*f*
a	0			
b,c	4.0	0		
d,e	5.0	3.1	0	
f	6.1	5.2	2.3	0

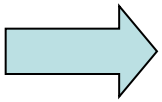
Agglomeratives Clustering durch Single-Linkage Clustering (3)

Jetzt zeigen die Elemente *d* und *e* des Clusters {*d*,*e*} den minimalen Abstand zu *f* von Cluster {*f*}.

Also wird das neue Cluster {*d*,*e*,*f*} gebildet.



	a	*b* *c*	*d* *e*	*f*
a	0			
b,c	4.0	0		
d,e	5.0	3.1	0	
f	6.1	5.2	2.3	0

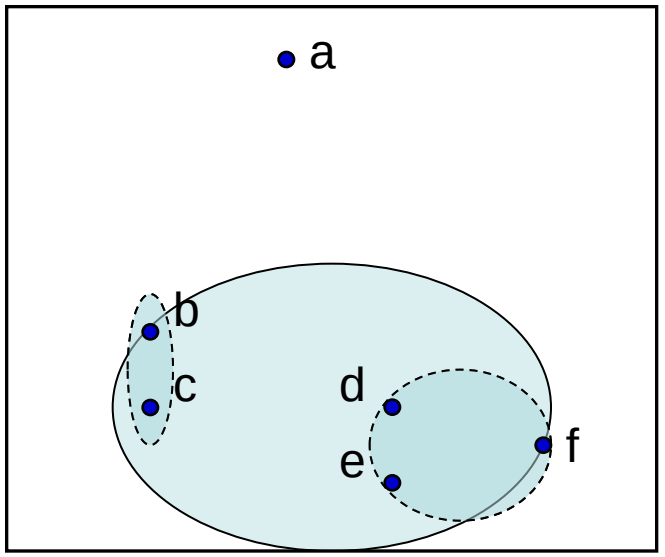


	a	*b* *c*	*d* *e* *f*
a	0		
b,c	4.0	0	
d,e,f	5.0	3.1	0

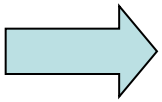
Agglomeratives Clustering durch Single-Linkage Clustering (4)

Jetzt zeigen die Elemente *c* und *d* der Cluster {*b*,*c*} und {*d*,*e*,*f*} den minimalen Abstand. Also wird das neue Cluster {*b*,*c*,*d*,*e*,*f*} gebildet.

Das Agglomerieren mag im Bspl. terminieren, wenn mit 4.0 eine Mindestdistanz zwischen den ähnlichsten Elementen zwei verschiedener Cluster überschritten ist. Ansonsten würde ein einziges Cluster {*a*,*b*,*c*,*d*,*e*,*f*} als Ergebnis resultieren.



	a	*b* *c*	*d* *e* *f*
a	0		
b,c	4.0	0	
d,e,f	5.0	3.1	0



	a	*b,c,* *d,e,* *f*
a	0	
b,c,d,e,f	4.0	0

Agglomeratives Clustering durch Single-Linkage Clustering (5)

Der Algorithmus:

Geg.: Distanzmatrix D mit Distanzen $d(C_i,C_j)$ zwischen nächsten Elementen aus den Clustern C_i und C_j .

1) Start mit ein-elementigen Clustern.

2) Suche ähnlichstes Paar* x und y über alle Clusterpaare C_i und C_j über

$$\min \{ d(x,y) \mid x \in C_i, y \in C_j \}.$$

3) Fasse die ermittelten Cluster C_i und C_j zu neuem Cluster $C_{i,j}$ zusammen.

4) Ersetze die Reihen und Spalten in D mit Bezug zu den Clustern C_i und C_j durch eine neue Reihe und Spalte für $C_{i,j}$, wobei $d(C_{i,j},C_k)$ für neues Cluster $C_{i,j}$ und bisherige Cluster C_k so, dass $d(C_{i,j},C_k) = \min \{ d(C_i,C_k), d(C_j,C_k) \}$.

5) Terminierung, wenn alle Cluster eine bestimmte Ähnlichkeit ihrer Elemente erzielt haben *oder* eine bestimmte Distanz zueinander überschreiten *oder* eine genügend kleine Zahl von Clustern ermittelt worden ist.

---

* ggf. auch mehrere Paare bei mehrfachem Auftreten der minimalen Distanz

Single-Linkage, Complete-Linkage und Average Linkage Clustering

Das Single-Linkage-Clustering (SLC) zeigt einen methodischen Aspekt, der bei bestimmten Datenmengen nachteilig sein kann und als Kettungsbildung (*chaining phenomenon*) bezeichnet wird:

Zwei Cluster werden auch dann fusioniert, wenn nur zwei einzelne Elemente aus beiden Clustern ähnlich sind, obwohl alle restlichen Elemente beider Cluster sehr verschieden von einander sind. SLC kann also zu heterogenen Clustern führen.

Als Alternative zum SLC (1) gibt es daher Varianten, die die Clusterfusion nicht über deren ähnlichste Elemente steuern, sondern z.B. über die unähnlichsten Elemente (2)* bzw. über die Mittelungen der Distanzen (3):

(1) Single-Linkage-Clustering: $\min\{ d(x,y) \mid x \in C_1, y \in C_2 \}$,

(2) Complete-Linkage-Clustering: $\max\{ d(x,y) \mid x \in C_1, y \in C_2 \}$,

(3) Average-Linkage-Clustering: $(|C_1| \cdot |C_2|)^{-1} \sum_{x \in C_1} \sum_{y \in C_2} d(x,y)$.

* beim Complete-Linkage-Clustering kann es wiederum zur Bildung kleiner Cluster kommen.

Agglomeratives Clustering über Zentroiddistanz und Intraclustervarianz

Einige weitere Bewertungsmaße für das agglomerative Clustering sind auch

- die Zentroiddistanz $d_{centroids}(C_1,C_2) = d(\underline{x},\underline{y})$

  für Mittelwerte $\underline{x}$, $\underline{y}$ von C_1 bzw. C_2 ,

- die Varianzzunahme *nach* Fusion von C_1 und C_2 (Ward-Kriterium):

$$d_{Ward}(C_1,C_2) = \sum_{z\in C_1\cup C_2} d(z,\underline{z})^2 - \sum_{x\in C_1} d(x,\underline{x})^2 - \sum_{y\in C_2} d(y,\underline{y})^2 = \frac{|C_1||C_2|}{C_1+C_2} d(\underline{x},\underline{y})^2.$$

  für Mittelwerte $\underline{x}$, $\underline{y}$, $\underline{z}$ von C_1, C_2 bzw. $C_1 \cup C_2$.

Zur Umsetzung der genannten alternativen Distanzmaße sind im Algorithmus *Single-Linkage-Clustering* in Schritten 2 und 4 nicht die Abstände der ähnlichsten Elemente zweier Cluster, sondern die maximalen Distanzen (Complete-Linkage-Clustering) bzw. durchschnittlichen Distanzen (Average-Linkage-Clustering) bzw. die Zentroiddistanzen bzw. die entstehenden Intraclustervarianzen zu minimieren.

Dichtebasierte Clusterverfahren

Dichtebasierte Verfahren modellieren Cluster als dicht beieinander liegende Datensätze in einem d-dimensionalen Raum. Diese Cluster sind wiederum durch Gebiete mit geringerer Dichte getrennt.

Ein bekannter dichtebasierter Algorithmus ist *DBSCAN* (für *Density-Based Spatial Clustering of Applications with Noise).*

Eine Erweiterung von *DBSCAN* ist der Algorithmus *OPTICS,* der im Gegensatz zu *DBSCAN*

- mit Clustern unterschiedlicher Dichte arbeiten kann,
- ein hierarchisches Ergebnis liefert, und
- eine visuelle Evaluierung erlaubt.

Bemerkung: Die folg. Notationen entsprechen denen der originalen Publikationen.

DBSCAN (1)

Grundlegende Begriffe (1):

Objektmenge O

Parameter *m* und ε

- Ein Datensatz (Objekt) $o \in O$ heißt ***dicht*** bzw. ***Kernobjekt***, wenn es eine Mindestzahl *m* von Nachbarobjekten $o' \in O$ in einer ε-Nachbarschaft von o gibt:

$$|N_\varepsilon(o)| \geq m \text{ mit } N_\varepsilon(o) = \{o' \in O \backslash \{o\} \mid dist(o,o') \leq \varepsilon\}.$$

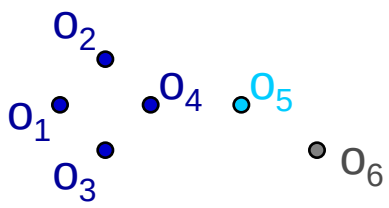
- Jedes Nachbarobjekt $o' \in O$ *in der ε-Nachbarschaft* eines Kernobjektes $o \in O$ heißt ***direkt dichte-erreichbar*** vom Kernobjekt $o \in O$ bzgl. *m* und ε.
- Jedes Objekt $o' \in O$ heißt ***dichte-erreichbar*** von einem Kernobjekt $o \in O$, wenn es eine verbindende Kette $o_1,\ldots,o_n$ von Objekten aus O für o und o' derart gibt, dass $o_1 = o$ und $o_n = o'$ und o_{i+1} *direkt dichte-erreichbar* ist von o_i für alle i.

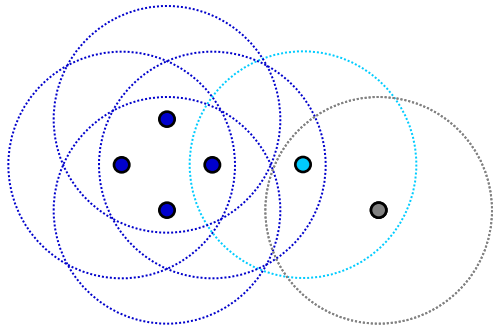
Aber: die Relation *dichte-erreichbar*(o,o') ist nicht symmetrisch, da o' selbst ggf. nicht Kernobjekt ist. Daher die folg. Definition von *Dichte-Verbundenheit*:

- Zwei Objekte $o_1, o_2 \in O$ heißen ***dichte-verbunden***, wenn sie beide von einem dritten Objekt $o_3 \in O$ *dichte-erreichbar* sind.
- Jedes Objekt $r \in O$ heißt ***Rauschobjekt***, wenn es weder *dicht* noch *dichte-erreichbar* ist.

DBSCAN (2)

Beispiel: $m = 3$, Objekte o_1, o_2, o_3, o_4, o_5, o_6 mit ε-Radien rechts





- o_1 hat o_2, o_3 und o_4 in der ε-Nachbarschaft $\rightsquigarrow$ o_1 ist Kernobjekt
- o_2 hat o_1, o_3 und o_4 in der ε-Nachbarschaft $\rightsquigarrow$ o_2 ist Kernobjekt
- o_3 hat o_1, o_2 und o_4 in der ε-Nachbarschaft $\rightsquigarrow$ o_3 ist Kernobjekt
- o_4 hat o_1, o_2, o_3 und o_5 in der ε-Nachbarschaft $\rightsquigarrow$ o_4 ist Kernobjekt
- o_5 hat o_4 und o_6 in der ε-Nachbarschaft $\rightsquigarrow$ o_5 ist direkt dichte-erreichbar von o_4
- o_5 ist dichte-erreichbar von o_1; o_1 ist nicht dichte-erreichbar von o_5
- o_5 und o_1 sind dichte-verbunden über o_4
- o_6 ist Rauschobjekt

o_5 ist kein Kernobjekt

DBSCAN (3)

Grundlegende Begriffe (2):

Aufgrund der bisherigen Definitionen gibt es drei Klassen von Objekten:

- Kernobjekte, die selbst als dicht bezeichnet werden, weil in ihrer ε-Umgebung die Mindestzahl von *m* Nachbarobjekten zu finden ist.
- *Dichte-erreichbare* Objekte, die zwar von einem Kernobjekt des Clusters erreichbar sind, selbst aber nicht dicht sind. Anschaulich werden diese den Rand eines Clusters bilden.
- Rauschobjekte, die weder dicht noch dichte-erreichbar sind und daher keinem Cluster zugeordnet werden.

Entsprechend wird ein ***Cluster*** wie folgt definiert:

- Ein Cluster C bzgl. der Parameter *m*, ε ist eine nicht-leere Teilmenge von O, für die folgende Bedingungen gelten:
  - Maximalität: $\forall\ o_1, o_2 \in O$: wenn $o_1 \in C$ und o_2 dichte-erreichbar von o_1 ist, dann ist auch $o_2 \in C$
  - Verbundenheit: $\forall\ o_1, o_2 \in C$: o_1 und o_2 sind dichte-verbunden.

DBSCAN (4): der Algorithmus

```
DBSCAN (D, eps, MinNeighbors)
  C = 0
  for each unvisited object O in dataset D
        mark O as visited
        N ← getNeighbors(O, eps)
        if sizeof(N) < MinNeighbors then mark O as NOISE
        else
              C ← next cluster
              expandCluster(O, N, C, eps, MinNeighbors)

expandCluster(O, N, C, eps, MinNeighbors)
  add O to cluster C
  for each object O' in N
        if O' is not visited then
              mark O' as visited
              N' ← getNeighbors (O', eps)
              if sizeof(N') ≥ MinNeighbors then N ← N joined with N'
        if O' is not yet member of any cluster add O' to cluster C
```

kann später (letzter Befehl von *expand Cluster*) noch in ein Cluster kommen.

DBSCAN (5)

- DBSCAN ist exakt bzgl. der Definitionen von *dichte-verbunden* und *Rauschen*: alle Objekte im selben Cluster sind garantiert dichte-verbundene Objekte, während *Rauschobjekte* sicher außerhalb von Clustern sind. Nicht exakt ist DBSCAN bei nur *dichte-erreichbaren* Objekten, diese werden nur einem Cluster zugeordnet, nicht allen möglichen.
- Die Zahl der Cluster muss nicht a priori festgelegt werden (wie z.B. bei vielen Partitionsverfahren wie k-Means).
- DBSCAN kann Cluster beliebiger Form (z.B. nicht nur kugelförmige) erkennen.
- DBSCAN ist deterministisch und reihenfolgeunabhängig: unabhängig von der Verarbeitungsreihenfolge der Objekte entstehen die selben Cluster (mit der Ausnahme der nur dichte-erreichbaren Nicht-Kern-Objekte und der Cluster-Nummerierung).
- DBSCAN ist von quadratischer Zeitkomplexität in der Zahl der Datenobjekte.

DBSCAN (6)

1) DBSCAN benötigt die Festlegung von *zwei Parametern*: ε und die Mindestzahl *MinNeighbors* von Nachbarobjekten für ein Kernobjekt in dessen ε-Umgebung.

   Als *Daumenregel* wird vorgeschlagen:

   – *MinNeighbors* = k $\geq$ dim(D)+1 (empirisch k=4 für 2D Daten vorgeschlagen),

   – ε aus einer Abschätzung über den k-Distanzen (Distanz zum k-nächsten Nachbarn) der Datenobjekte.

2) DBSCAN arbeitet nicht gut auf solchen Datenmengen, deren Cluster unterschiedliche Dichten zeigen, da das Parameterpaar (Epsilon, MinNeighbors) für alle Cluster vorgegeben wird.

➢ Beide Aspekte führten zur Entwicklung von *OPTICS* (*Ordering Points To Identify the Clustering Structure*). *OPTICS* kann Cluster unterschiedlicher Dichte erkennen und eliminiert (weitgehend) den ε-Parameter.

OPTICS (1)

- *OPTICS* basiert auf *DBSCAN,* weist aber zwei Verbesserungen auf:
  - *OPTICS* kann im Gegensatz zu *DBSCAN Cluster unterschiedlicher Dichte* erkennen.
  - Gleichzeitig *eliminiert OPTICS* (weitgehend) den *ε-Parameter* von DBSCAN.

- Hierzu ordnet OPTICS die Punkte der Datenmenge so, dass ähnliche bzw. benachbarte Punkte in dieser Ordnung nahe aufeinander folgen.

- Gleichzeitig wird die sog. *Erreichbarkeitsdistanz* notiert. Zeichnet man diese Erreichbarkeitsdistanzen in ein Diagramm, so bilden Cluster „Täler“ und können so identifiziert werden.

OPTICS (2)

- Auch *OPTICS* verwendet zwei Parameter *minNeighbors* und ε. ε steht hier aber für eine Maximaldistanz, bis zu der man überhaupt noch von einer für das Clustering relevanten Dichte sprechen kann. ε dient so der Komplexitäts-begrenzung von *OPTICS*.

- In *DBSCAN* ist ein Objekt ein *Kernobjekt*, wenn seine ε-Umgebung mindestens *minNeighbors* Objekte enthält.

Dadurch sind Kerndistanzen hier verschieden, weil abhängig von der Dichte

- Dies wird in *OPTICS* umgedreht: die *Kerndistanz* eines Objekts wird als der Abstand zum *minNeighbors*-nächsten Nachbarn definiert.
  - Die Kerndistanz wäre *in DBSCAN* derjenige ε-Wert, ab dem ein Objekt ein Kernobjekt wäre.
  - Hat ein Objekt *in OPTICS* in seiner ε-Umgebung keine *minNeighbors* Nachbarn, so ist seine Kerndistanz unendlich oder „undefiniert“.

OPTICS (3)

- Die *Erreichbarkeitsdistanz* eines Objekts *o* von einem zweiten Objekt *o'* ist definiert als max(*kerndistanz*(*o*),*dist*(*o*,*o'*)), also als das Maximum vom der Kern-distanz des verweisenden Punktes und des wahren Abstandes.
- OPTICS ordnet jetzt alle Objekte.
  - Begonnen wird mit einem beliebigen unbearbeiteten Objekt o.
    - Die Nachbarn der ε-Umgebung von *o* werden ermittelt und nach ihrer Erreichbarkeitsdistanz zu *o* in einer Vorrangwarteliste *OrderedList* gemerkt.
    - Nun wird immer der Nachbar mit minimaler Erreichbarkeitsdistanz als nächster in die Ordnung aufgenommen. Durch das Verarbeiten eines neuen Nachbarn können sich die Erreichbarkeitsdistanzen der unver-arbeiteten Nachbarn verbessern. Durch die Sortierung dieser Vorrang-warteschlange sucht OPTICS die Mitte eines Clusters und verarbeitet diesen vollständig, bevor er beim nächsten Cluster weitermacht.

OPTICS (4)

- OPTICS liest in der Hauptschleife zunächst alle Objekte der Datenmenge D ein.
- Für jedes Objekt O werden
  - alle Objekte O' aus der ε-Nachbarschaft gelesen
  - die Erreichbarkeitsdistanz von O auf *undefiniert* gesetzt
  - seine Kerndistanz core-distance(O, eps, MinNeighbors) zum *MinNeighbors*-Nachbarn ermittelt
  - Die if-Anweisung überprüft, ob O ein Kernobjekt ist.
    - Wenn nicht, wird das nächste Objekt in der Hauptschleife eingelesen.
    - Wenn ja, werden über *OrderSeedsUpdate(N,O,Seeds,eps,MinNeighbors)* iterativ alle direkt dichte-erreichbaren Nachbarn von O bzgl. ε und *Min-Neighbors* in die *Seeds*-Liste zur weiteren Cluster-Expansion geordnet nach ihrer Erreichbarkeitsdistanz eingefügt.
    - In der innersten Schleife werden die Objekte der Seeds-Liste gelesen, ihre Kerndistanz bestimmt und sie werden dann mit ihrer Erreichbarkeitsdistanz in *OrderedList* geschrieben. Wenn auch sie Kernobjekte sind, werden weitere Objekte ihrer ε-Nachbarschaft in *Seeds* eingelesen usw.

OPTICS (5): der Algorithmus

```
OPTICS (D, eps, MinNeighbors, OrderedList)
  for each object O in dataset D
      O.reachability-distance ← undefined
  for each unvisited object O in dataset D
      N = getNeighbors (O, eps)  // set of eps neighbors
      mark O as visited
      output O to OrderedList
      Seeds ← empty priority queue
      if  core-distance(O, eps, MinNeighbors) ≠ undefined then
          OrderSeedsUpdate(N, O, Seeds, eps, MinNeighbors)
          for each next O' in Seeds
              N' ← getNeighbors (O', eps)
              mark O' as visited
              output O' to OrderedList
              if  core-distance(O', eps, MinNeighbors) ≠ undefined then
                  OrderSeedsUpdate (N', O', Seeds, eps, MinNeighbors)
```

Ermittlung der *core-distance* = Kerndistanz = Distanz zum *MinNeighbors*-Nachbarn oder *undefined*.

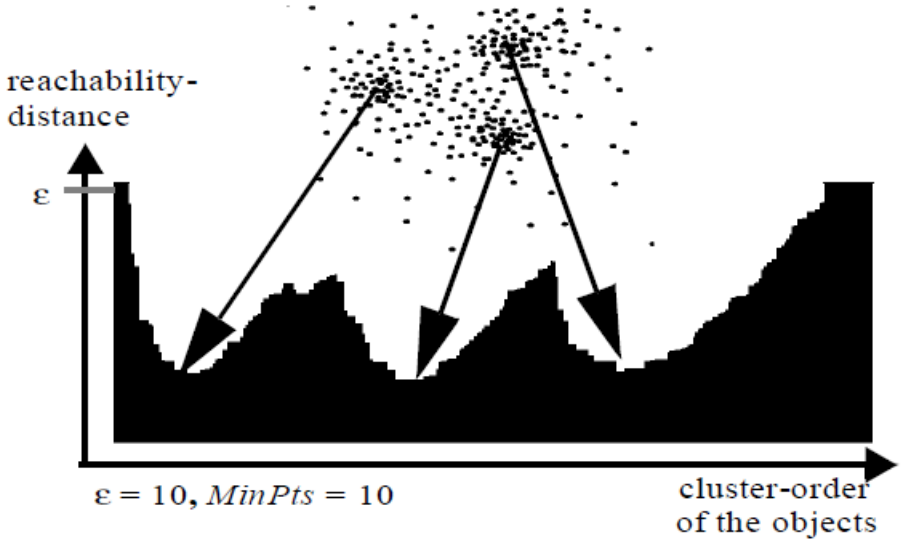
In *OrderSeedsUpdate* wird die Vorrangliste mit den neuen ε-Nachbarn des Kernobjekts O aktualisiert.

OPTICS (6): der Algorithmus von *OrderSeedsUpdate*

```
OrderSeedsUpdate(N, O, Seeds, eps, MinNeighbors)
  core-dist ← core-distance(O, eps, MinNeighbors)
  for each O' in N
      if  O' is not visited
          new-reach-dist ← max(core-dist,dist(O,O')
          if  O'.reachability-distance = undefined then // O' not in Seeds
              O'.reachability-distance ← new-reach-dist
              Seeds.insert (O', new-reach-dist )
          else  // O' in Seeds, check for improvement
              if  new-reach-dist < O'.reachability-distance
                  O'.reachability-distance ← new-reach-dist
                  Seeds.move-up (O', new-reach-dist )
```

OPTICS (7)

- Das Ergebnis des Algorithmus OPTICS ist eine Liste *OrderedList* von *n* Objekten o_i, die nach ihren Erreichbarkeitsdistanzen $r(o_i) \geq 0$ geordnet sind.
- Diese kann man sich grafisch in Form einer Häufigkeitsverteilung, eines sog. *Erreichbarkeitsdiagramms* vorstellen. „Täler“ in diesem Diagramm entsprechen erkannten Clustern im Datensatz, die Tiefe des Tales zeigt die Dichte des Clusters an:



- Über einen zweiten Wert $\varepsilon' \leq \varepsilon$ sind nun Cluster über einen einfachen Scan-Algoríthmus *ExtractClustering* aus der Verteilung der Erreichbarkeitsdistanzen $r(o_i) \geq 0$ ableitbar.

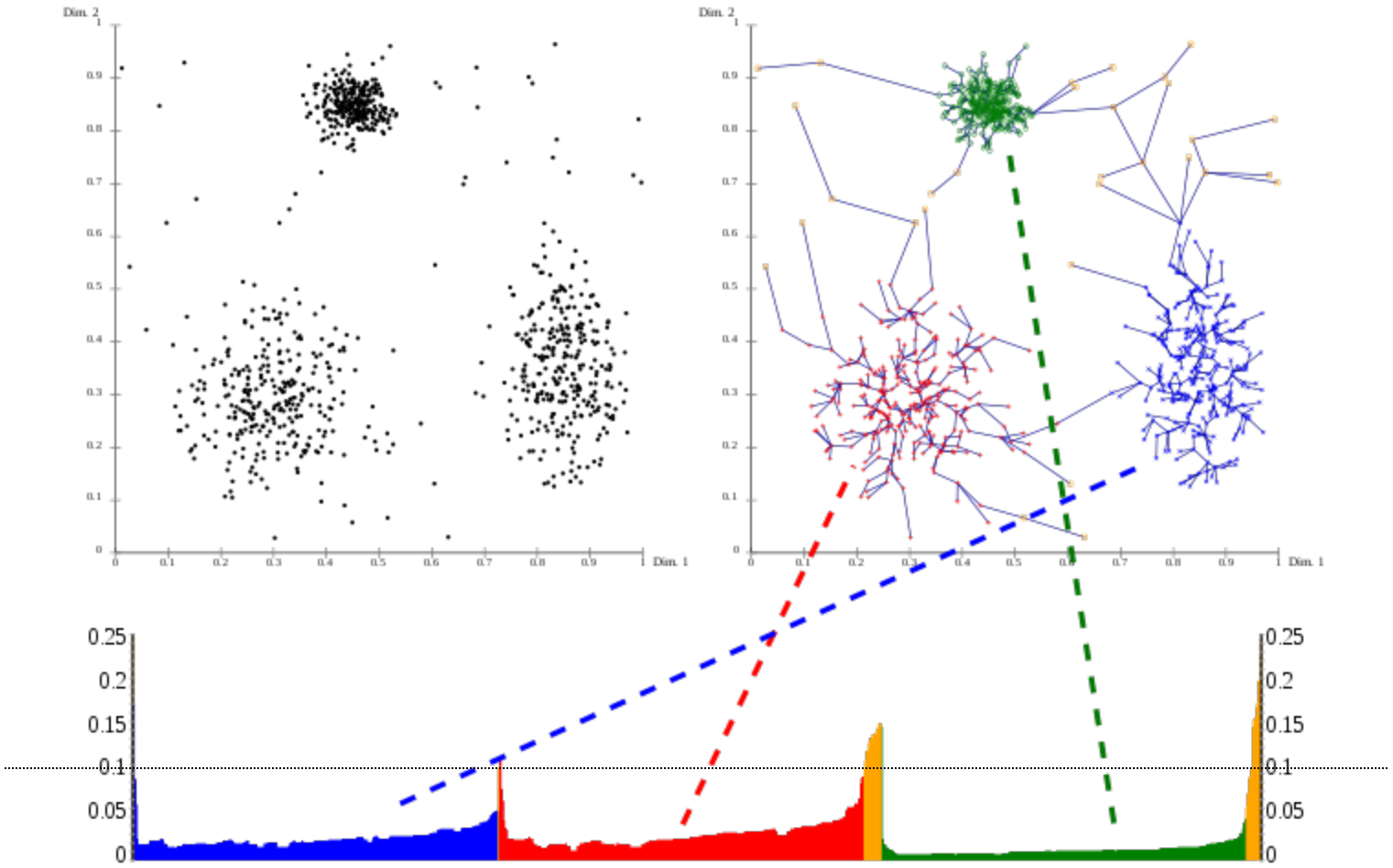
OPTICS (8): Algorithmus *ExtractClustering*

ExtractClustering(OrderedList, eps', MinNeighbors)

```
// precondition: clustering distance eps' ≤ generating eps
for each O in OrderedList
    if  O.reachability-distance > eps' then
        // remind that undefined > eps
        if  O.reachability-distance ≤ eps then
            ClusterId ← nextId(ClusterId)
            O.clusterId ← ClusterId
        else
            O.clusterId ← Noise
    else // O.reachability-distance ≤ eps'
        O.clusterId ← ClusterId
```

OPTICS (9): Beispiel

Hier ist die Dichteverbundenheit der Objekte verdeutlicht, indem *jeder Objektpunkt mit seinem Erreichbarkeitsvorgänger verbunden* ist. Hier sind $\varepsilon \geq 0{,}5$, MinNeighbors = 10 und $\varepsilon' = 0{,}1$.

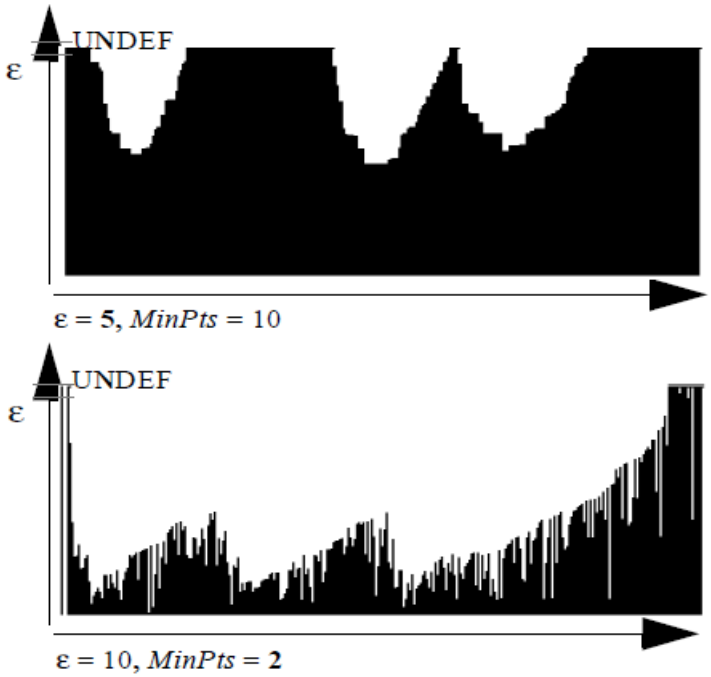


OPTICS (10): Wahl der Parameter und Komplexität

- Der generierende ε-Wert sollte der kleinste Distanzwert sein, der alle Objekte der Datenmenge umfasst. Dieser kann als halber Wert der Distanz (etwa Radius einer d-dimensionalen Hypersphäre) zwischen den unähnlichsten Objekten der Datenmenge aufgefasst werden.

- Verschiedene Werte von *MinNeighbors* zeigen im Wesentlichen ähnliche Verläufe der Distanzverteilungen. Größere Werte führen zu einer „Glättung“ der Verteilungen und verhindern Verkettungseffekte aufgrund singulärer Verbindungen.

- Quadratische Zeitkomplexität im Worst Case wie DBSCAN



Zusammenfassung

- Aufgabe des unüberwachtes Lernen ist die Erkennung von mehreren Kategorien in einer Sammlung von Datensätzen, für die aber keine Kategoriebeschriftungen vorliegen.
- Das unüberwachte Lernen stellt damit die schwierigste Form des Lernens dar.
- Unüberwachtes Lernen von Kategorien kann durch Clusteranalyse umgesetzt werden. Durch Gruppenzuordnung (engl. *Clustering*) werden die Datensätze derart aufgeteilt, dass Gruppen (Anhäufungen, engl. Cluster) von „ähnlichen“ Datensätzen entstehen.
- Es gibt verschiedene Ansätze für die Clusteranalyse. Für die Auswahl müssen die den Clusteralgorithmen zugrunde liegenden Annahmen mit den möglichen Eigenschaften der Datenverteilungen verglichen werden: haben alle Cluster ähnliche Dichte?, zeigen alle Cluster eine (ähnliche) Normalverteilung?, kann die Zahl der Cluster a priori festgelegt werden?, ….

Quellen

- Stuart Russel, Peter Norvig: *Artificial Intelligence – A Modern Approach* (2nd Ed.) Prentice Hall, 2003.

- Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu: *"A density-based algorithm for discovering clusters in large spatial databases with noise".* In Evangelos Simoudis, Jiawei Han, Usama M. Fayyad. Proc. 2nd Intern. Conf. on Knowledge Discovery and Data Mining (KDD-96). AAAI Press. pp. 226–231, 1996.

- Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, Jörg Sander: *OPTICS: Ordering Points To Identify the Clustering Structure.* Proc. ACM SIGMOD Intern. Conf. on Management of Data. ACM Press, 1999, pp. 49–60, 1999.

- Für diverse Abbildungen und Ergänzungen die Seiten Cluster_analysis, K-means, DBSCAN und OPTICS in http://en.wikipedia.org/wiki/ (alle 18.06.2012).