Kapitel 10
Klassifikationsverfahren

Skript zur Vorlesung
Geo-Informationssysteme
Wintersemester 2013/14
Ludwig-Maximilians-Universität München

10. Klassifikationsverfahren

1. Einführung
2. Partitionierendes Clustering
3. Hierarchisches Clustering
4. Single Scan Clustering
5. Maximum-Likelihood Klassifikation
6. Nearest-Neighbor Klassifikation

10.1 Einführung (I)

Problem

- Gegeben ein Rasterbild, das durch ein Fernerkundungssystem geliefert wurde.
- Die Grauwerte stellen gemessene Amplituden in bestimmten Spektralbereichen dar und sind für einen Benutzer nicht direkt zu interpretieren.
- Gesucht ist eine Zuordnung aller Pixel (x,y) zu je einer von wenigen disjunkten *Klassen Ci, $1 \leq i \leq k$,* die für eine gegebene Anwendung Bedeutung besitzen.

10.1 Einführung (II)

Anwendung

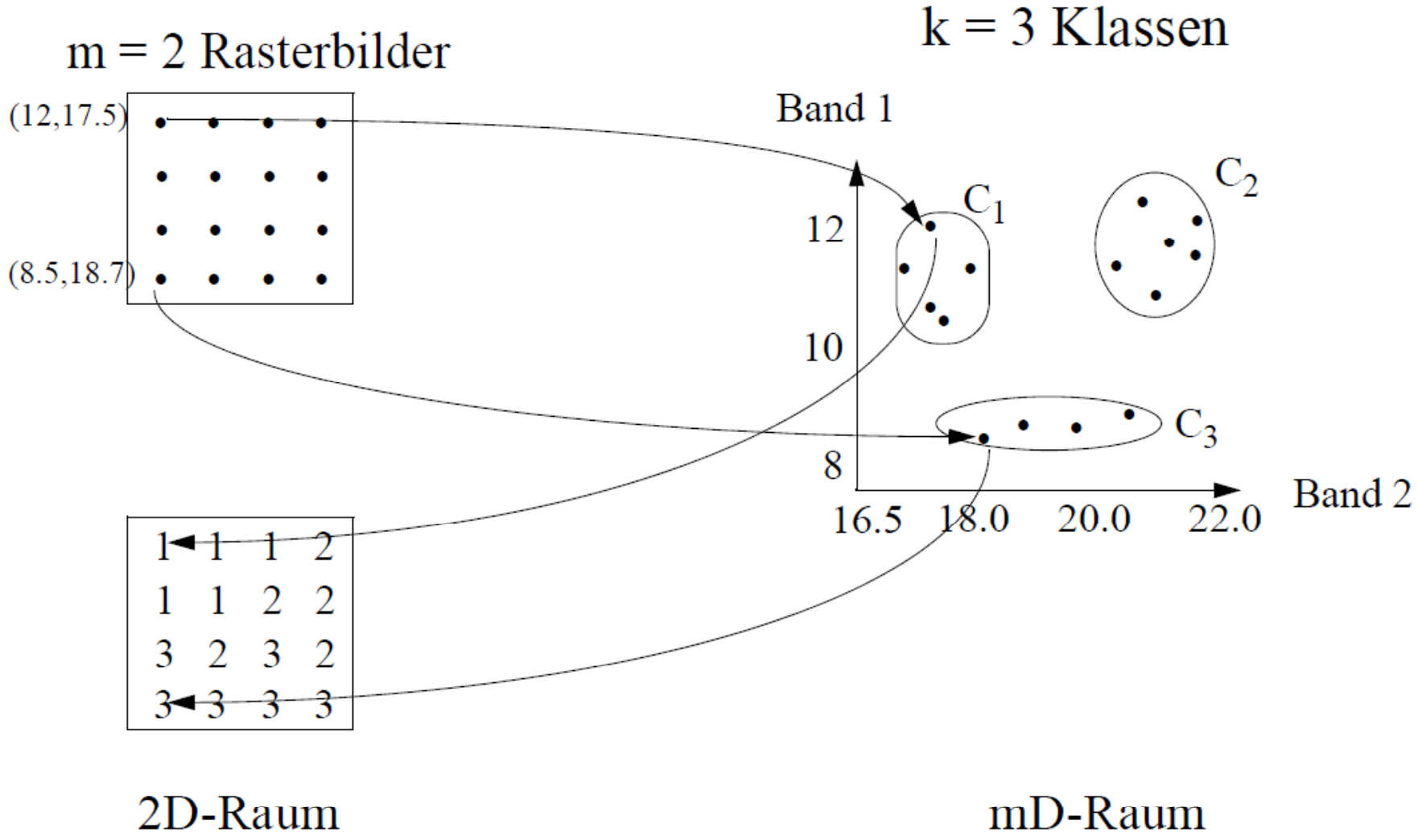
- Die Klassen können z.B. die Bodennutzung repräsentieren (*landuse).*
- Klassen z.B. water, urban, croplands, rangelands.

Lösungsansätze

- nicht überwachte Klassifikation (*unsupervised classification)*
  - die Klassen C_i sind noch nicht bekannt, insbesondere ist auch die geeignete Anzahl k von zu unterscheidenden Klassen nicht vorgegeben
- überwachte Klassifikation (*supervised classification)*
  die Klassen C_i sind bereits bekannt

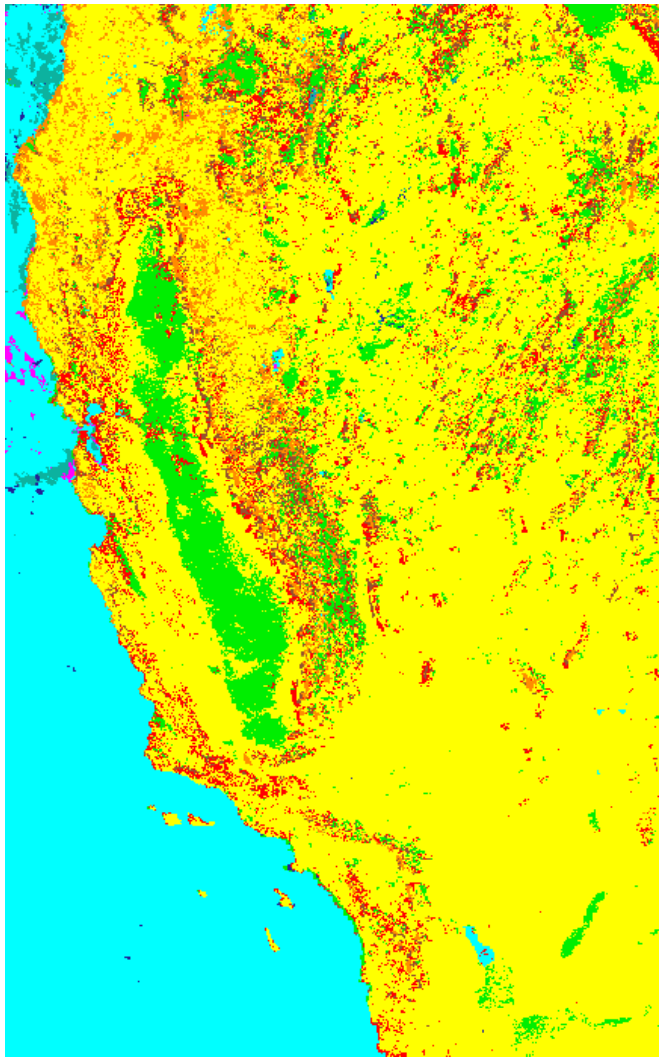
10.1 Einführung (III)

Schema einer Klassifikation



10.1 Einführung (IV)

Beispiel (Landnutzung von Kalifornien)



Überwachte Klassifikation

- Die Klassen C_i *, 1 ≤ i ≤ k, sind gegeben.*
- Gesucht ist eine Funktion f, die jedem Pixel (x,y) mit Grauwerten $d_1(x,y),\ldots, d_m(x,y)$ genau ein i, 1 ≤ i ≤ k, zuordnet, d.h. eine *Klassifikationsfunktion.*
- Wir bezeichnen im folgenden mit D(x,y) den Grauwertvektor $(d_1(x,y),\ldots,d_m(x,y))$. C_i *(x,y)* sei das Prädikat, das genau dann TRUE liefert, wenn $(x,y) \in C_i$.

10.1 Einführung (VI)

Überwachte Klassifikation: Vorgehen

- Festlegen der Klassen $C_i, 1 \leq i \leq k$.
- Manuelle Zuordnung der Klasse für eine kleine Zahl repräsentativer Pixel. Diese Menge T von Pixeln (x,y) mit bekannter Klasse c(x,y) bildet die *Trainingsdaten/Trainingspixel*. T enthält z.B. 1% aller Pixel des Eingangsbildes.
- Suche einer Klassifikationsfunktion f, die die Trainingsdaten möglichst korrekt klassifiziert, d.h. für möglichst viele Elemente von T soll gelten $f(x,y) = c(x,y)$.
- Automatische Zuordnung der Klasse für die restlichen Pixel mit Hilfe der gefundenen Klassifikationsfunktion *f*. Restliche Pixel z.B. *99%*.

10.1 Einführung (VII)

Nicht Überwachte Klassifikation (*Clustering)*

- Weder die Klassen (*Cluster) C_i* noch ihre Anzahl *k* sind gegeben.
- Gesucht sind sowohl die Klassen $C_i, 1 \leq i \leq$ k,als auch eine geeignete Klassifikationsfunktion.

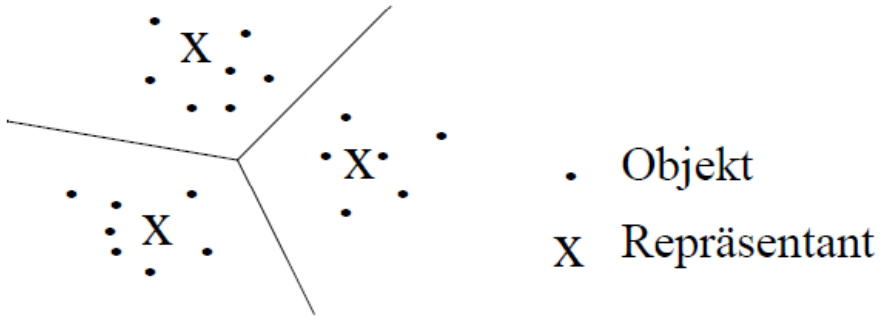
Vorgehen

- Definition der Ähnlichkeit zweier Pixel durch Definition einer Distanzfunktion *dist* basierend auf den *m* Grauwerten beider Pixel.
- Bestimmung einer Partitionierung aller Pixel in Klassen C_i so dass die Pixel einer Klasse möglichst ähnlich (kleiner Wert für dist) und die Pixel untersch. Klassen möglichst unähnlich (grosser Wert für dist) sind.
- Manuelle Interpretation der Klassen C_i mit Hilfe einer kleinen Zahl repräsentativer Pixel, für die die gesuchte Klasse bekannt ist. Z.B. stellt man fest, dass die Pixel, die C_1 bzw. C_2 zugeordnet wurden, Wasser bzw. bebautes Gebiet darstellen, d.h. C_1 = water C_2 = urban.

10.1 Typen von Clusteringalgorithmen (I)

Partitionierende Algorithmen

- Gegeben ist eine Datenbank *DB* (Menge der Pixel repräsentiert als *m*-dimensionale Vektoren), eine Distanzfunktion dist und ein Integer k.
- Gesucht ist eine Partitionierung (*Clustering)* in k Cluster mit minimalen Kosten.
- Zweistufiges Verfahren
  - Bestimme *k* Repräsentanten
  - ordne jedes Objekt aus *DB* dem nächstgelegenen Repräsentanten zu



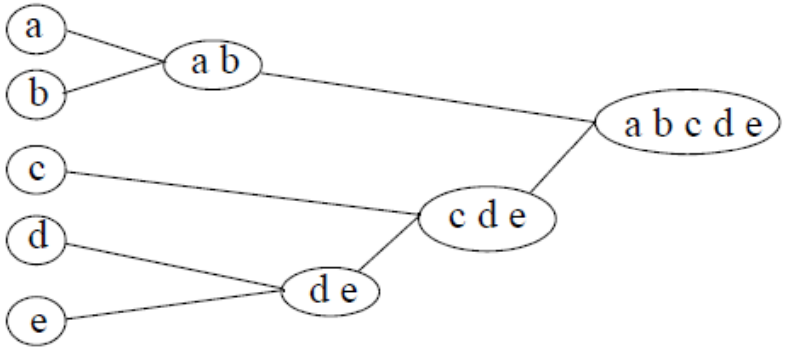
- Zwei Arten von Repräsentanten:
  - Schwerpunkt des Clusters (*k-means Algorithmen)*
  - ein Objekt des Clusters (*k-medoid Algorithmen)*

⇒Kosten einer Partitionierung = Σ *dist(o, repräsentant (o))*

10.1 Typen von Clusteringalgorithmen (II)

Hierarchische Algorithmen

- Gegeben: Datenbank *DB*, Distanzfunktion *dist* , Abbruchbedingung *Ab*
- Gesucht: hierarchische Aufteilung in Cluster, die *Ab* erfüllt.
- Das hierarchische Clustering wird als Dendrogramm dargestellt. Ein *Dendrogramm* ist ein Baum, dessen Knoten je einen Cluster repräsentieren und die folgende Eigenschaften besitzen:
  - die Wurzel repräsentiert die ganze *DB*
  - die Blätter repräsentieren die einzelnen Objekte (Pixel) aus *DB*
  - ein innerer Knoten repräsentiert die Vereinigung aller Objekte (Pixel) seiner Sohnknoten.



Dendrogramm
für DB = {a, b, c, d, e}

10.1 Typen von Clusteringalgorithmen (III)

Single Scan Algorithmen

- führen einen einzigen Durchlauf der Datenbank aus
- lokale Clusterbedingung wie z.B. minimale Dichte, passende Distanz zum Nearest Neighbor, . . .
- algorithmisches Schema wie folgt:

```
ALGORITHM SingleScanClustering (database DB, cluster condition COND)
        FOR each object o of DB DO
                IF o is not yet member of some cluster THEN
                        create a new (empty) cluster C;
                        WHILE neighboring objects of o satisfy COND DO
                                add o and ist neighbors to C
                        ENDWHILE
                ENDIF
        ENDFOR
END SingleScanClustering
```

10.2 Partitionierendes Clustering (I)

Grundlagen

- Ziel: Partitionierung in k Cluster, so dass eine Kostenfunktion minimiert wird (Gütekriterium: Kompaktheit)
- Zentrale Annahmen:
  - Anzahl k der Cluster bekannt (Eingabeparameter)
  - Clustercharakteristik: Kompaktheit
  - Kompaktheit: Abweichung aller Objekte im Cluster von einem ausgezeichneten Cluster-Repräsentanten ist minimal
  - Kompaktheitskriterium führt meistens zu sphärisch geformten Clustern
- Erschöpfende Suche ist zu ineffizient (WARUM?)
- Daher: Lokal optimierende Verfahren
  - wähle k initiale Cluster-Repräsentanten
  - optimiere diese Repräsentanten iterativ
  - ordne jedes Objekt seinem ähnlichsten Repräsentanten zu

PAM (Partitioning Around Medoids)

ALGORITHM PAM (Datenbank DB, Distanzfunktion dist, int k)

Medoide:={}; Nichtmedoide := DB; Wähle das zentralste Objekt aus DB als ersten Medoid;

FOR i FROM 2 TO k DO

wähle den bisherigen Nichtmedoid als weiteren Medoid, der die Kosten der Partitionierung am stärksten reduziert;

END FOR;

LOOP

FOR each pair (Medoid, Nichtmedoid) DO

berechne die Änderung der Kosten der Partitionierung, die durch das Ersetzen des Medoids durch den Nichtmedoid bewirkt würde;

END FOR;

Sei (M,N) das Paar mit minimaler Kosten K;

IF K ist Verbesserung zu vorher **THEN**

Ersetze den Medoid M durch den Nichtmedoid N;

ELSE exit **LOOP;**

END IF;

END LOOP;

10.2 Partitionierendes Clustering (III)

Komplexitätsanalyse von PAM

- Auswahl der initialen k Medoide: $O(n^2)$ für jeden Medoid, d.h. Zeitkomplexität $O(k * n^2)$
- Zahl der Durchläufe der LOOP: ???
- Zeitkomplexität eines Durchlaufs:
  - (n - k) * k Berechnungen der Änderung der Kosten
  - eine Berechnung ist $O(n) \Rightarrow O(k * n^2)$

Verbesserung 1

Nicht das Paar (Medoid, Nichtmedoid) mit minimaler Änderung (< 0) suchen sondern für das erste Paar, das die Kosten reduziert, die Ersetzung durchführen ⇒ Algorithmus CLARANS

Verbesserung 2

Die Medoide nicht mit allen Objekten der DB bestimmen sondern nur auf einer Stichprobe von DB (reduziere n) ⇒ Sampling

10.2 Partitionierendes Clustering (IV)

CLARANS (Clustering Large Applications based on RANdomized Search)

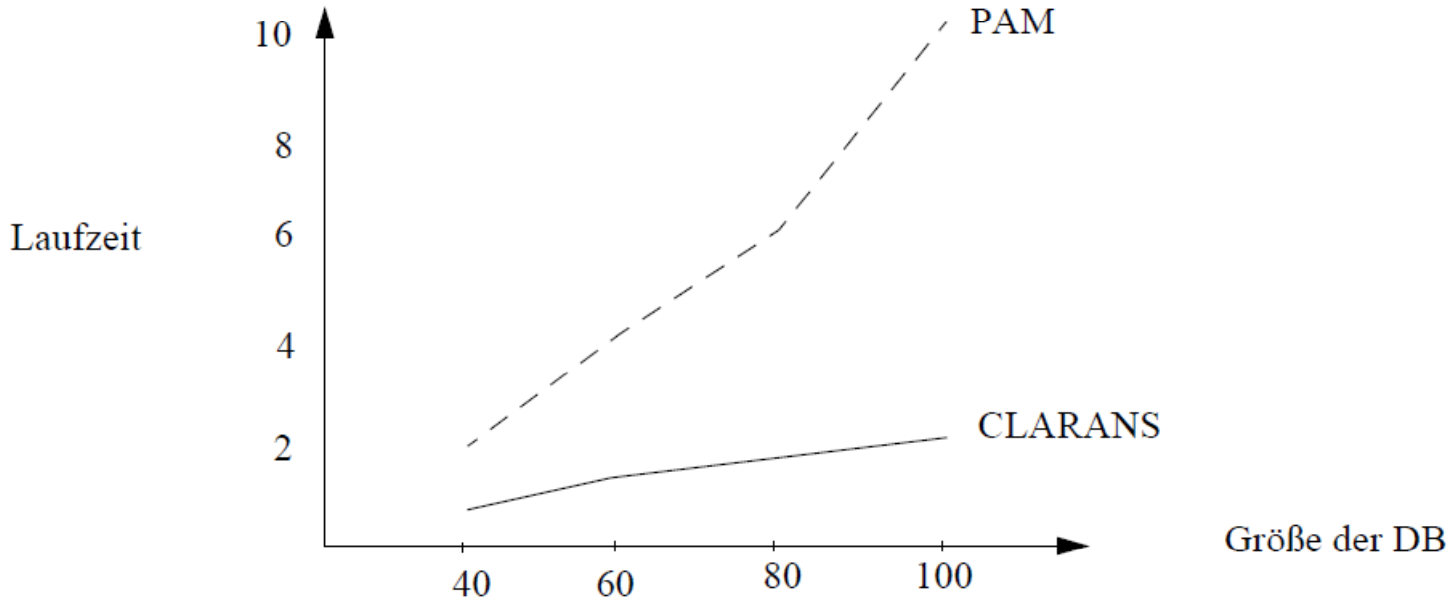
ALGORITHM CLARANS (Datenbank DB, int k, Distanzfunktion dist, int numlocal, int maxneighbor)
Medoide := {}; Nichtmedoide := DB; Kmin := MAXFLOAT; BestMedoide := {};
FOR i FROM 1 TO numlocal DO
wähle zufällig k der Objekte aus DB als Medoide;
WHILE NOT maxneighbor Paare (Medoid, Nichtmedoid) betrachtet DO
wähle zufällig einen der Medoide und einen der Nichtmedoide;
berechne die Änderung K der Kosten der Partitionierung, die durch das
Ersetzen des Medoids durch den Nichtmedoid bewirkt würde;
IF K < 0 THEN
Ersetze den Medoid durch den Nichtmedoid;
Beginne bei der Zählung der Paare wieder mit 0;
END IF;
END WHILE;// Innere Schleife
berechne die Kosten K der aktuellen Partitionierung;
IF K < Kmin THEN
BestMedoide := Medoide;
Kmin := K;
END IF; END FOR;// Äußere Schleife

10.2 Partitionierendes Clustering (V)

Laufzeitkomplexität von CLARANS

- Auswahl des Paars (Medoid,Nichtmedoid): O(1)
- Berechnung der potentiellen Änderung der Kosten: O(n)
- Ersetzung des Medoids durch den Nichtmedoid: O(1)
- Anzahl der Durchläufe der Inneren Schleife: ???

Leistungsuntersuchung von CLARANS



10.2 Partitionierendes Clustering (VI)

Sampling

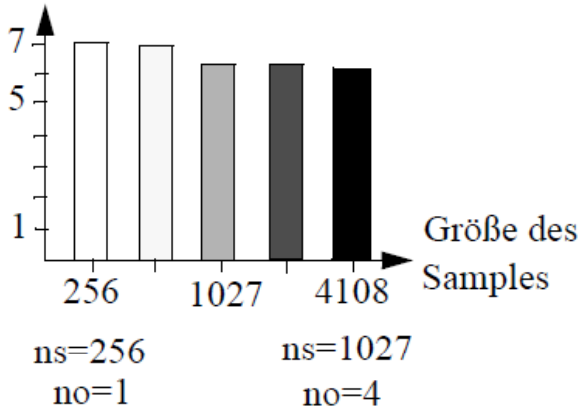
- Ein *Sample* ist eine repräsentative Stichprobe einer Menge von Daten.
- R*-Baum erhält räumliche Nachbarschaft so gut wie möglich.

⇒ wähle ein gleichverteiltes Sample von den Datenseiten des R*-Baums

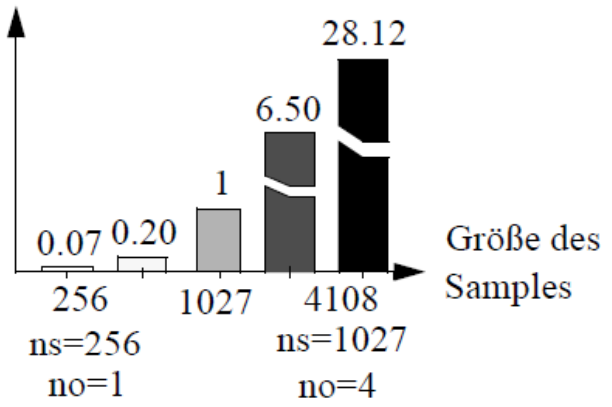
Größe des Samples

- Zwei Parameter: ns = Anzahl auszuwählender Datenseiten, no = Anzahl auszuwählender Objekte pro ausgewählter Datenseite
- experimentelle Untersuchung (mit DB von 55.000 Punkten)

Kosten der resultierenden Partitionierung



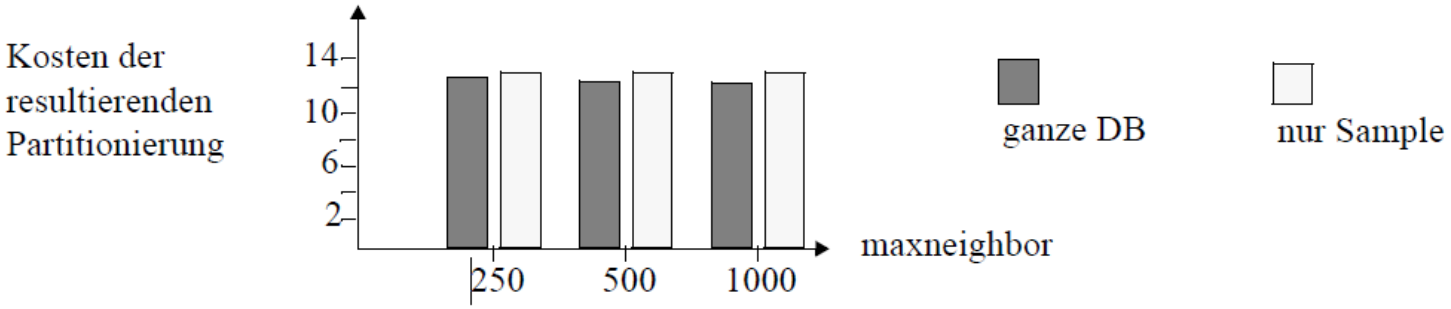
Relative Laufzeit von CLARANS



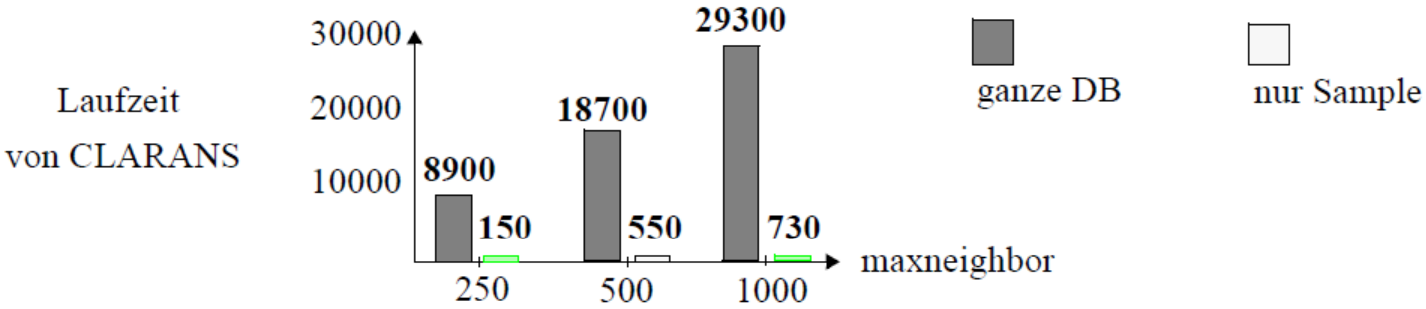
10.2 Partitionierendes Clustering (VII)

Leistungsuntersuchung des Sampling

- Vergleich der Effektivität



- Vergleich der Effizienz



=> kleiner Verlust an Effektivität und großer Gewinn an Effizienz