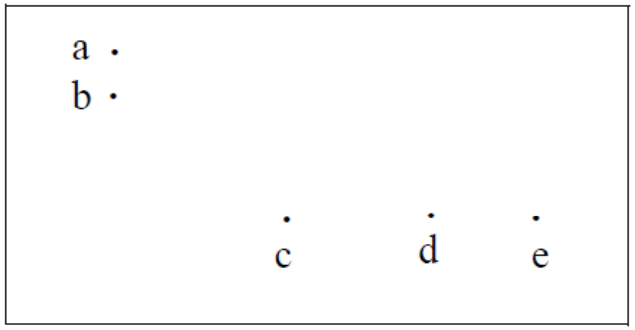
Single Link

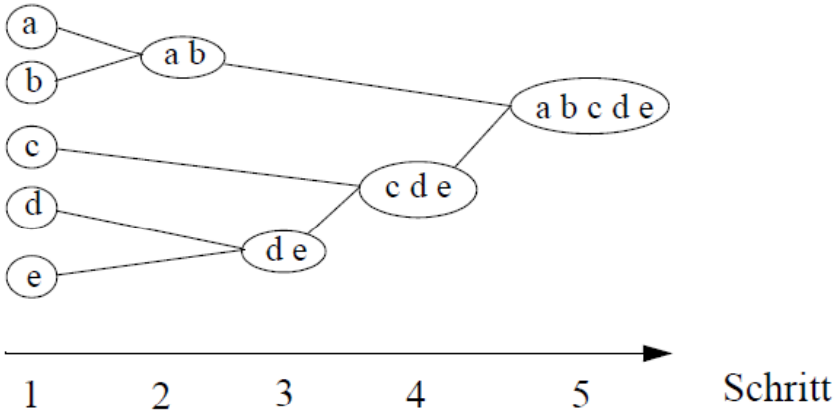
ALGORITHMUS SingleLinkClustering (Datenbank DB, Distanzfunktion dist, Abbruchbedingung Ab)

FOR i **FROM** 1 **TO** Size(DB) **DO**
    Cluster[i]:= i-tes Objekt von DB;
END ;
FOR i **FROM** 1 **TO** Size(DB) **DO**
    Cluster[i].Link:= Cluster j mit minimaler dist zu Cluster i;
    Cluster[i].LinkDist:= dist(Cluster[i],Cluster[i].Link);
END;
WHILE Bedingung Ab nicht erfüllt **DO**
    Sei Cluster [i] das Cluster mit minimalem Wert von LinkDist;
    Verschmelze Cluster[i] und Cluster[i].Link;
    Aktualisiere die Werte für Cluster[i].Link und Cluster[i].LinkDist;
    Aktualisiere die Werte für Link und LinkDist bei allen Clustern, deren Link Cluster[i].Link war;
    Dekrementiere die aktuelle Anzahl von Clustern um 1;
END WHILE;

10.3 Hierarchisches Clustering (II)

Beispiel





Nachteile des Single Link Algorithmus

- Abbruchbedingung erfordert Bereichswissen

  ⇒ interaktive Variante: Benutzer analysiert Dendrogramm und bestimmt eine Ebene, an der das Dendrogramm abgeschnitten werden soll
- Anfälligkeit gegen Rauschen: eine Linie von Rauschpunkten kann zwei Cluster zu einem verbinden
- Ineffizienz für große Datenbanken: in jedem Schritt wird die Distanz aller Paare von aktuellen Clustern betrachtet: Laufzeitkomplexität $O(n^2)$

10.4 Single Scan Clustering (I)

DBSCAN (Density Based Spatial Clustering of Applications with Noise)

- ist ein *Single Scan Clustering* Algorithmus
- Clusterbedingung: die "Dichte" in der Eps-Umgebung eines Punkts ist höher als ein vorgegebener Grenzwert MinPts.
- DBSCAN berechnet für MinPts = 2 eine Ebene eines Dendrogramm
- DBSCAN unterstützt den Benutzer beim Bestimmen geeigneter Werte für die Parameter Eps und MinPts.
- Die Dichte in einer Umgebung ändert sich durch Einfügen weniger "Ausreißer"-Punkte weniger als die Distanz der Punkte zum nächsten Nachbarn.

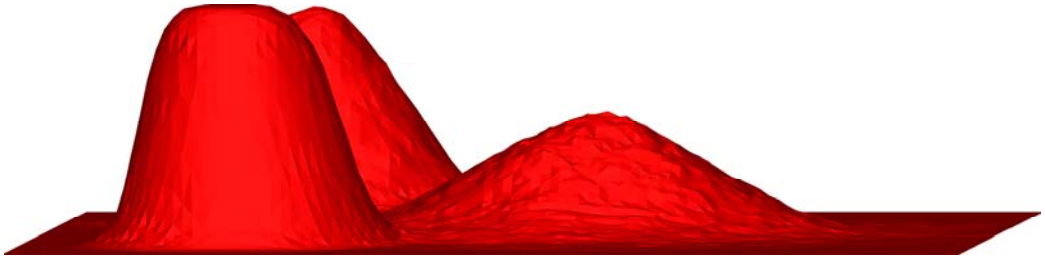
  ⇒ geringere Anfälligkeit gegen Rauschen
- Betrachte nicht die Distanz aller Paare von aktuellen Clustern, sondern betrachte nur die Distanz zum aktuellen Cluster.

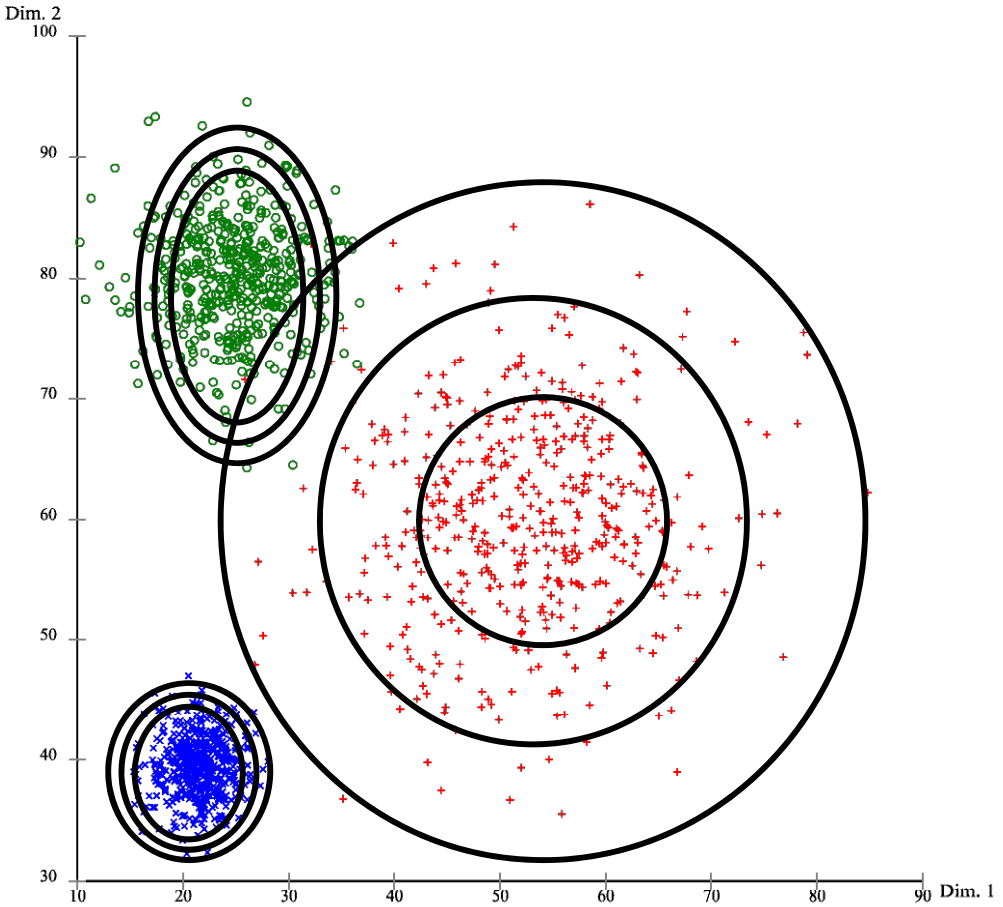
  ⇒ Effizienz auch für große Datenbanken

10.4 Single Scan Clustering (II)

Intuition

- selektiere Bereiche, die mit Mindest-Dichte verbunden sind

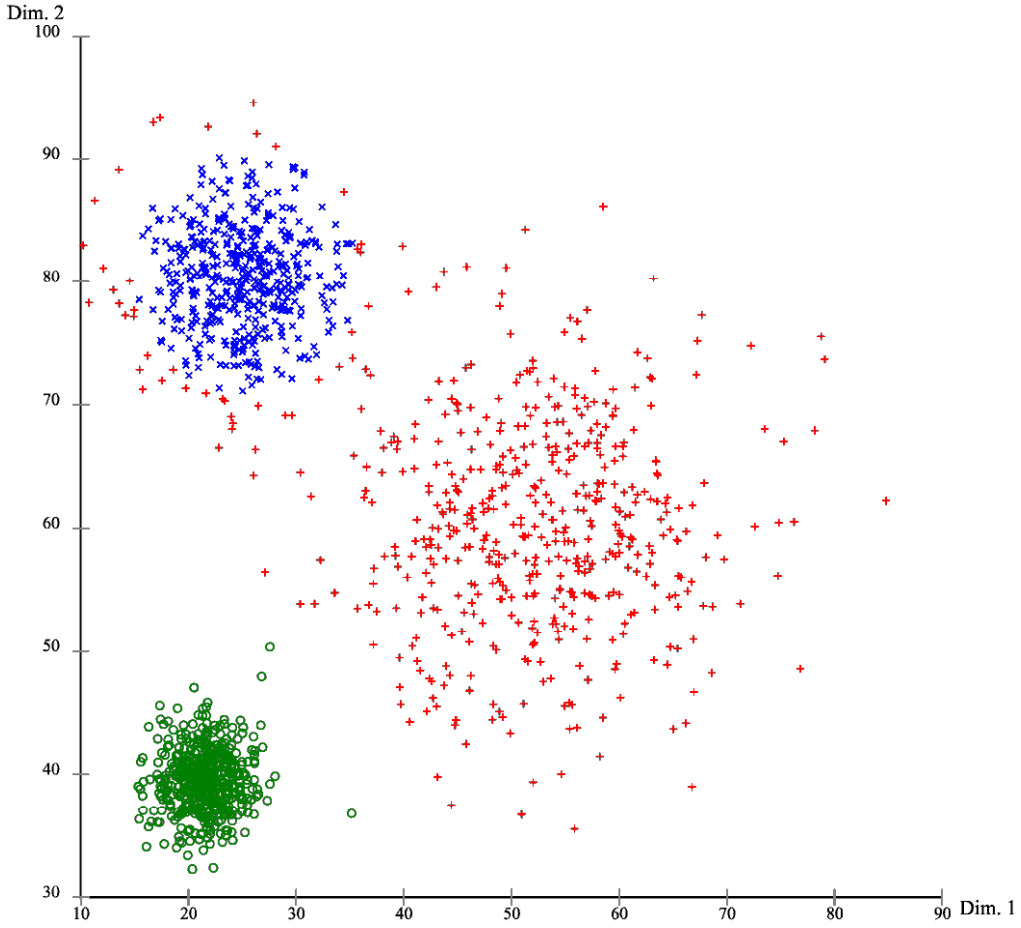


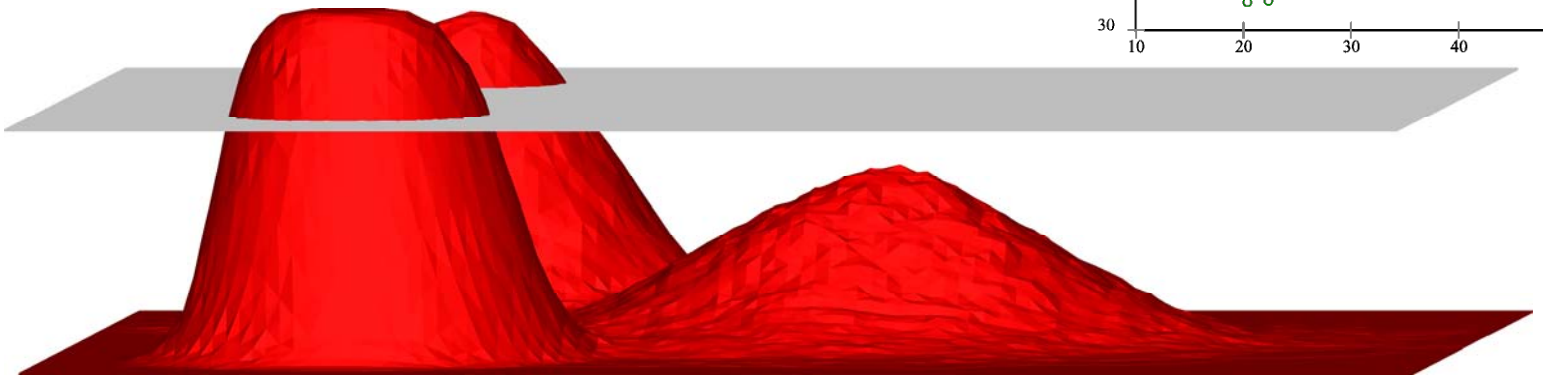


10.4 Single Scan Clustering (III)

Intuition

- hohes Dichte-Niveau: Cluster geringer Dichte gilt als Noise

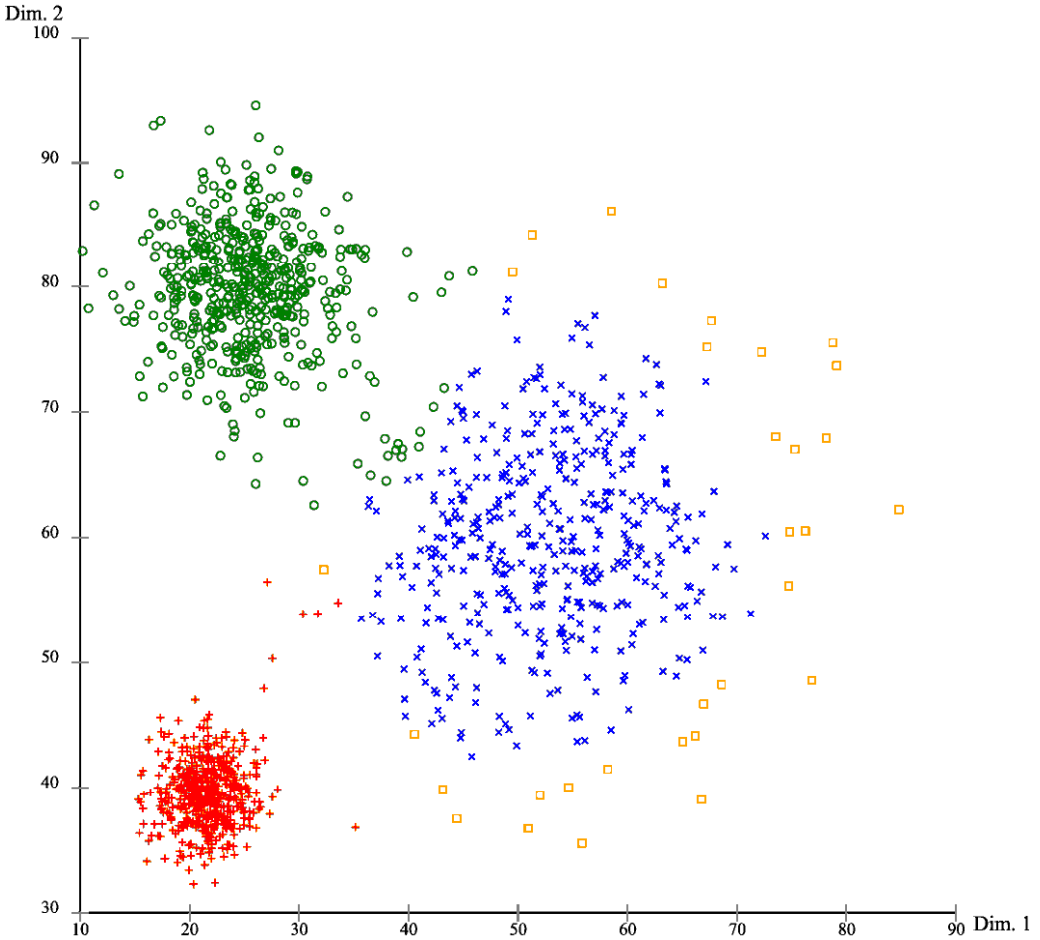


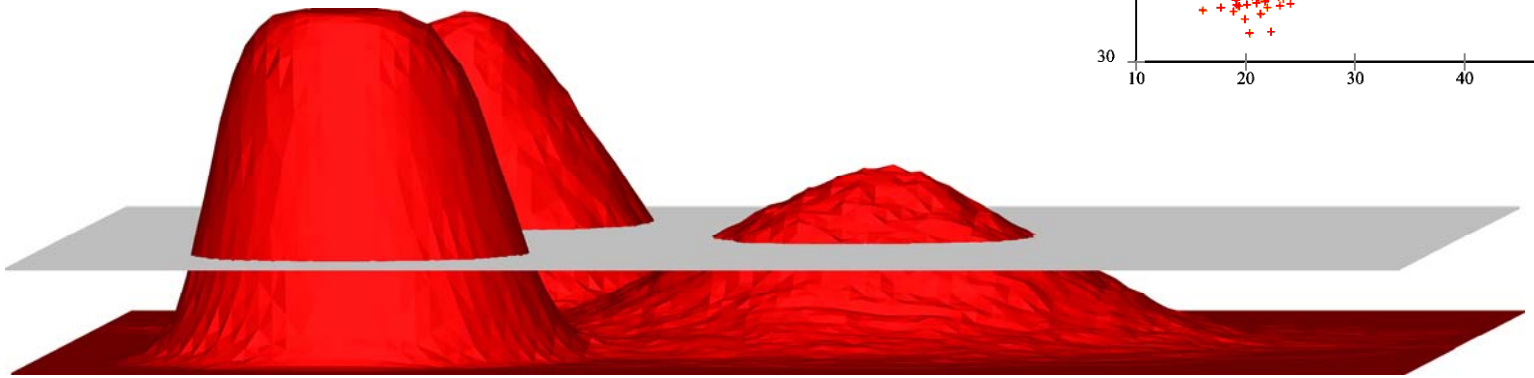


10.4 Single Scan Clustering (IV)

Intuition

- mittleres Dichte-Niveau: Identifikation von drei Clustern, einige Punkte in den Ausläufern der Verteilungen werden zu Noise

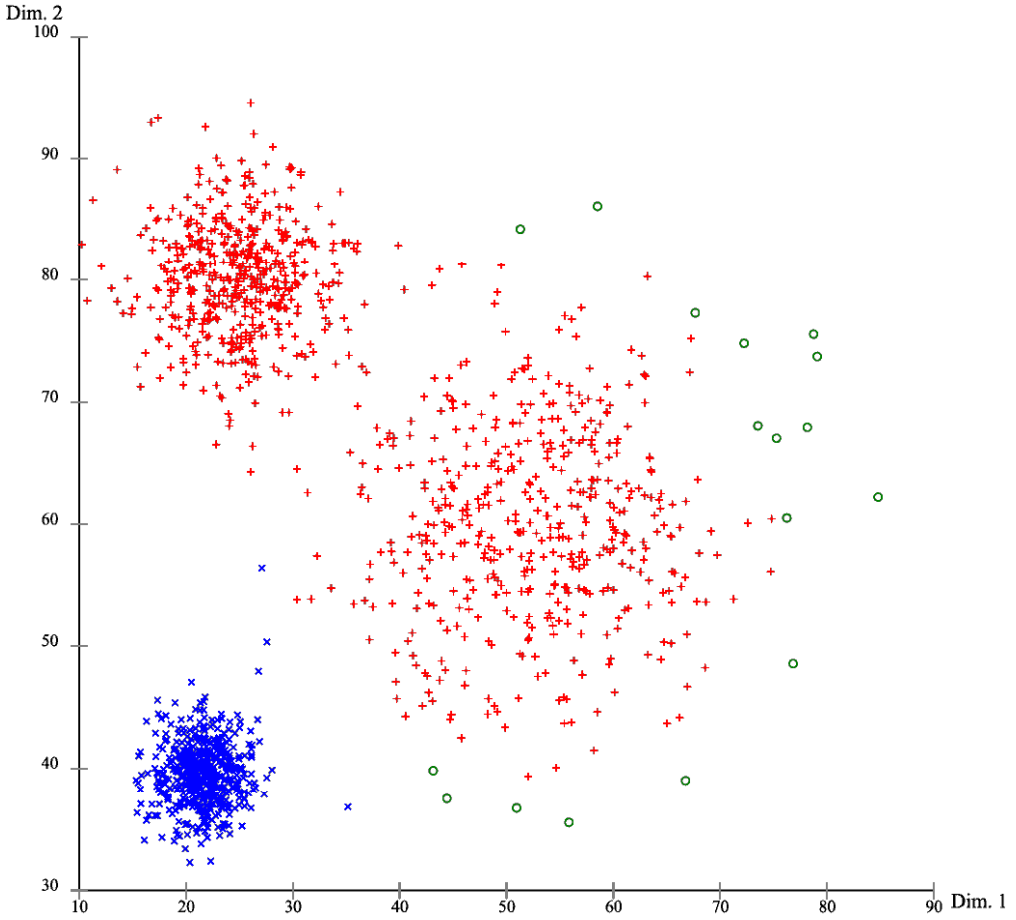


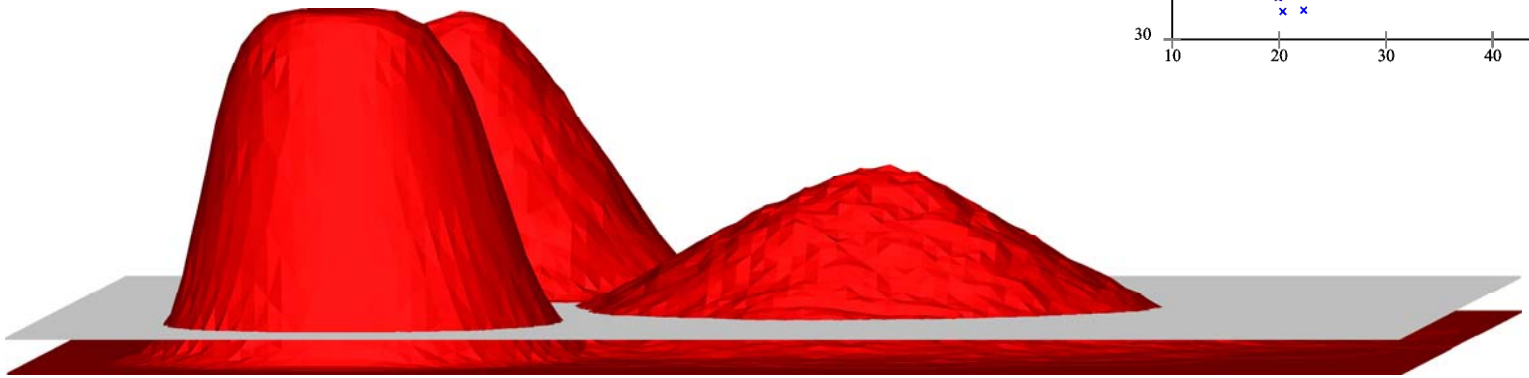


10.4 Single Scan Clustering (V)

Intuition

- geringes Dichte-Niveau: zwei Cluster verschmelzen

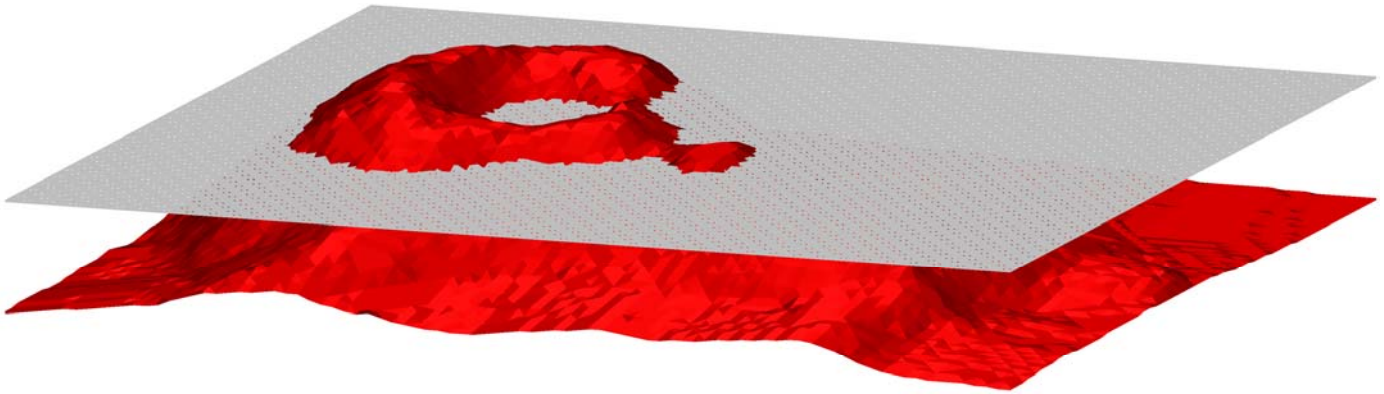




10.4 Single Scan Clustering (VI)

Intuition

- Partitionierend: Optimierungsproblem
- Dichte-basiert: „natürliche“ Strukturen (oft: geographisch/ topographisch, oder biologisch)
- Beispiel: Höhenprofil (Vulkan Mt. Eden, Auckland)



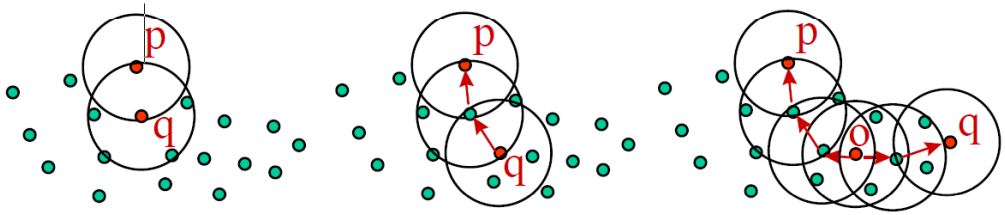
10.4 Single Scan Clustering (VII)

Notationen

- Ein Punkt *p* ist direkt dichte-erreichbar von einem Punkt *q* (bzgl. Eps und MinPts), wenn
  - 1) $|U_{Eps}(q)| \geq$ MinPts (*q* ist Kernpunkt)
  - 2) $p \in U_{Eps}(q)$

    ⇒ die direkte Dichte-Erreichbarkeit ist i.a. nicht symmetrisch
- Ein Punkt *p ist dichte-erreichbar von einem Punkt q,* wenn es eine Kette von Punkten p1, ..., pn mit p1 = q, pn = p gibt, so dass pi+1 direkt dichte-erreichbar von pi ist.
- Ein Punkt *p ist dichte-verbunden mit einem Punkt q,* wenn es einen Punkt o gibt, so dass sowohl p als auch q von o dichte-erreichbar sind.

  ⇒ die Dichte-Verbundenheit ist eine symmetrische Relation



10.4 Single Scan Clustering (VIII)

Notationen (Forts.)

- Ein *Cluster C* ist eine nichtleere Teilmenge einer Datenbank *D* mit folgenden Eigenschaften:
  - 1) $\forall\ p, q \in D: p \in C \wedge q$ dichte-erreichbar von $p \rightarrow q \in C$ (Maximalität)
  - 2) $\forall\ p, q \in C: p$ *ist* dichte-verbunden mit q (Verbundenheit)
- Seien $C_1, \ldots, C_k$ die Cluster der Datenbank D. Dann definieren wir das *Rauschen* als die Teilmenge aller Punkte von *D*, die zu keinem Cluster C_i gehören.

10.4 Single Scan Clustering (IX)

Beobachtungen

- Sei p ein Kernpunkt von *D*. Dann ist die Menge aller Punkte aus *D*, die von *p* dichte-erreichbar sind, ein Cluster.

  ⇒wenn man von einem Punkt ausgehend alle Punkte sammelt, die von ihm dichte-erreichbar sind, dann findet man einen Cluster

- Sei *C* ein Cluster und p ein beliebiger Kernpunkt dieses Clusters. Dann ist *C* gleich der Menge aller Punkte aus *D*, die von *p* dichte-erreichbar sind.

  ⇒von jedem seiner Kernpunkte ausgehend findet man den selben Cluster

10.4 Single Scan Clustering (X)

Algorithmus DBSCAN

DBSCAN (Datenbank DB, Eps real, MinPts int)

```
FOR i FROM 1 TO Size(DB) DO
  Punkt:= i-tes Objekt von DB;
  IF ClusterId von Punkt = UNCLASSIFIED THEN
        IF ExpandiereCluster(DB,Punkt,ClusterId,Eps,MinPts) THEN
       ClusterId:= nächste ClusterId;
        END IF;
  END IF;
END FOR;
```

10.4 Single Scan Clustering (XI)

Algorithmus DBSCAN (Forts.)

```
METHOD ExpandiereCluster (Datenbank DB, Punkt p, ClusterId int, Eps real, MinPts int): boolean
Seed:= RegionQuery in DB für Eps-Umgebung von p;
IF Size(Seed) < MinPts THEN  setze die ClusterId von p auf NOISE; RETURN FALSE;
ELSE
   Setze die ClusterId aller Punkte aus Seed auf ClusterId;  Lösche p aus Seed;
   WHILE Seed nichtleer DO
      AktuellerPunkt := erstes Element aus Seed;
      Ergebnis:= RegionQuery in DB für Eps-Umgebung von AktuellerPunkt;
      IF Size(Ergebnis) >= MinPts THEN
         FOR each Punkt IN Ergebnis DO
               IF ClusterId von Punkt IN {UNCLASSIFIED, NOISE} THEN
              setze ClusterId von Punkt auf ClusterId; füge Punkt in Seed ein;
         END FOR;
      END IF;
      Lösche AktuellerPunkt aus Seed;
   END WHILE;
   RETURN True;
END IF;
```

10.4 Single Scan Clustering (XII)

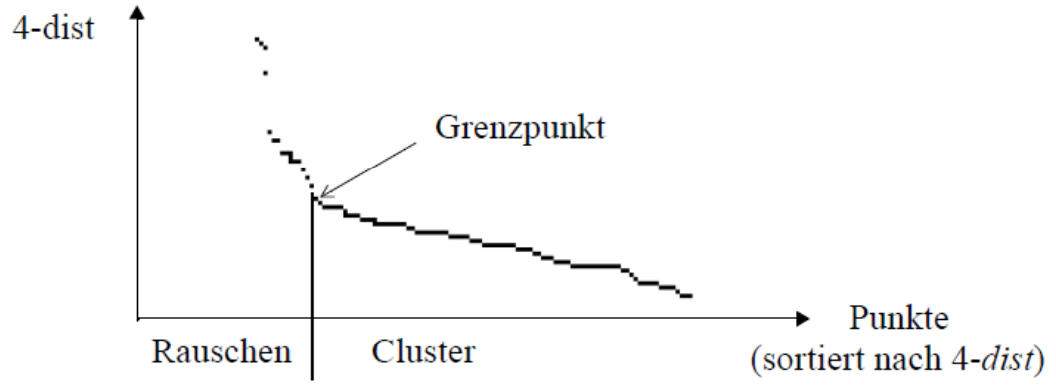
Bestimmung der Parameter

- Gesucht sind die Werte Eps und MinPts des "dünnsten" Clusters.
- Wir definieren die Funktion *k-dist,* die jedem Punkt die Distanz zu seinem k-nächsten Nachbarn zuordnet. Wenn wir die Punkte aus D nach absteigendem *k-dist* Wert sortieren erhalten wir den *sortierten k-dist Graphen.*

  [Beobachtung: die sortierten *k-dist* Graphen für $k > 4$ unterscheiden sich nicht signifikant von demjenigen für $k = 4$]
- Suche das erste "Tal" im sortierten 4-*dist* Graphen *(Grenzpunkt).*
  - Punkt liegt "links" vom Grenzpunkt: Rauschen
  - Punkt liegt "rechts" vom Grenzpunkt: Cluster

  ⇒ MinPts = 4 und

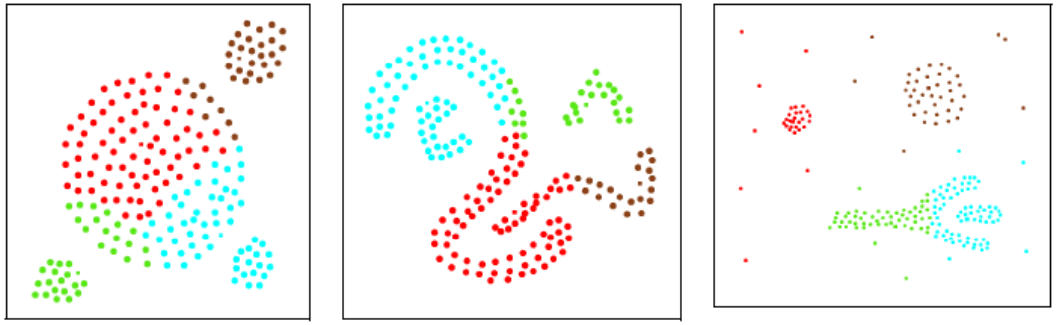
  Eps = 4-*dist des Grenzpunkts*



10.4 Single Scan Clustering (XIII)

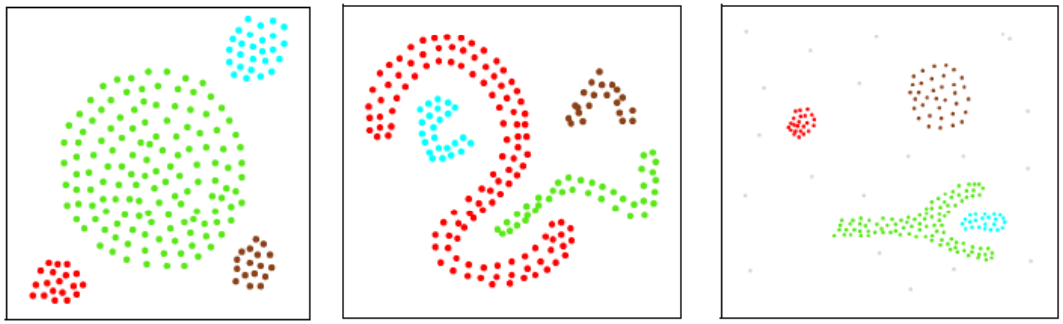
Leistungsuntersuchung

Gefundene Clusterings



Datenbank 1 | Datenbank 2 | Datenbank 3

Clusterings, die CLARANS findet



Clusterings, die DBSCAN findet

Laufzeit in sec

Anzahl der Punkte	1252	2503	3910	5213	6256	7820	8937	10426	12512
DBSCAN	3.1	6.7	11.3	16.0	17.8	24.5	28.2	32.7	41.7
CLARANS	758	3026	6845	11745	18029	29826	39265	60540	80638

10.5 Maximum-Likelihood Klassifikation (I)

Idee

- Für die Klassifikation sind wir interessiert an den bedingten Wahrscheinlichkeiten $p(C_i(x,y)|D(x,y))$.
- Wenn man diese bedingten Wahrscheinlichkeiten kennt, dann ordnet man einem Pixel (x,y) mit Grauwertvektor D(x,y) die Klasse C_j mit dem maximalen Wert für diese Wahrscheinlichkeit zu:

  Bayes'sche Entscheidungsregel

$$(1)\ \forall i \neq j [p(C_j(x,y)|D(x,y)) > p(C_i(x,y)|D(x,y))] \Rightarrow C_j(x,y)$$

Maximum Likelihood Entscheidungsregel

- Die obigen bedingten Wahrscheinlichkeiten sind meist a priori unbekannt.
- Die umgekehrten bedingten Wahrscheinlichkeiten $p(D(x,y)|C_i(x,y))$. lassen sich jedoch aus Trainingsdaten schätzen (überwachte Klassifikation).

10.5 Maximum-Likelihood Klassifikation (II)

Maximum Likelihood Entscheidungsregel (Fortsetzung)

- Die gesuchten Wahrscheinlichkeiten lassen sich daraus mit Hilfe des *Bayes'schen Theorems* folgendermassen berechnen:

  (2) $p\,(C_i(x,y)|D\,(x,y)) = p\,(D\,(x,y)|C_i(x,y)|C_i(x,y) \cdot p\,(C_i(x,y))/p\,(D\,(x,y))$

  wobei $p(C_i(x,y))$ die unbedingte Wahrscheinlichkeit bzw. die Häufigkeit der Klasse C_i im Bild ist und $p(D(x,y))$ die Häufigkeit des Grauwertvektors $D(x,y)$.
- Einsetzen von (2) in (1) liefert die *Maximum Likelihood Entscheidungsregel:*

  (3) $\forall i \neq j\ [p(D(x,y)|C_j(x,y)) \cdot p(C_j(x,y)) > p(D(x,y)|C_i/x,y)) \cdot p\,(C_i\,(x,y))] \Rightarrow C_j(x,y)$
- Die $p(D(x,y))$, die schwer zu bestimmen sind, kürzen sich dabei heraus.
- (3) lässt sich mit der Definition

$$g_{i(D(x,y))} = \ln(p(D(x,y)|C_j(x,y))) + \ln(p(C_j(x,y)))$$ so umschreiben:

$$(4)\ \forall i \neq j[g_{j(D(x,y))} > g_{i(D(x,y))}] \Rightarrow C_j(x,y)$$

10.5 Normalverteilungen (I)

Schätzung der bedingten Wahrscheinlichkeiten

- Frage: Wie kann man aus den beobachteten $p(D(x,y)|C_i(x,y))$ die bedingten Wahrscheinlichkeiten für alle $D(x,y)$ und für alle C_i bestimmen?
- Methode: nehme an, dass die bedingten Wahrscheinlichkeiten $p(D(x,y)|C_i(x,y))$ *normalverteilt* sind.

10.5 Normalverteilungen (II)

Eindimensionaler Fall

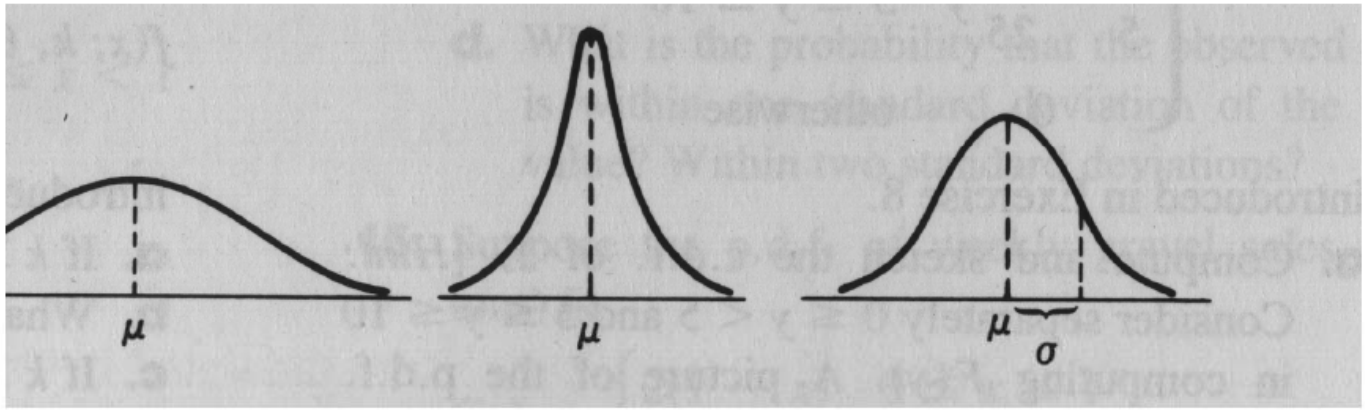
- Eindimensionaler Fall: m = 1, d.h. nur ein Grauwert pro Pixel. Dann gilt

$$p(D(x,y)|C_i(x,y)) = \frac{1}{\sqrt{2\pi}\sigma_i} \cdot e^{-(D(x,y)-m_i)^2/2\sigma_i^2}$$

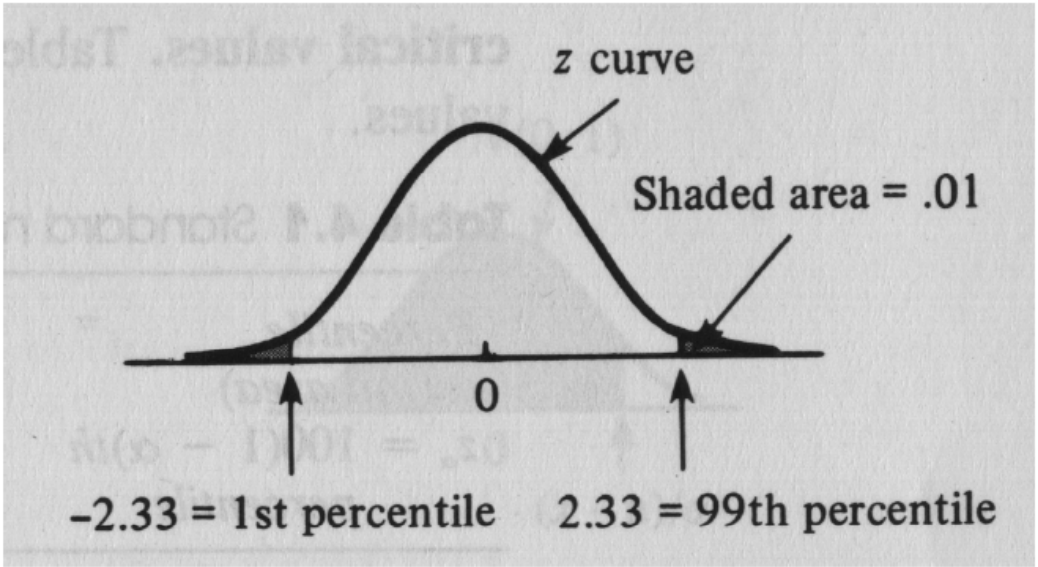
  mit m_i als Mittelwert und σ_i als Standardabweichung von $D(x,y)$ für die Klasse C_i
- Annahme: es ist eine hinreichend große Menge von D(x,y)-Werten gegeben.
- Dann wird m_i geschätzt als Mittelwert der $D(x,y)$, σ_i^2 als Durchschnitt der Quadrate der Differenz der D(x,y) zu m_i.

10.5 Normalverteilungen (III)

Beispiel



μ = m: Mittelwert, σ: Standard-Abweichung



10.5 Normalverteilungen (IV)

Mehrdimensionaler Fall

- m > 1, d.h. mehrere Grauwerte pro Pixel.
- Dann ist *mi ein m-dimensionaler Vektor und σi 2 eine mxm Matrix Σi .*
- Die *Kovarianz-Matrix Σi für die Klasse Ci beschreibt die Korrelation zwischen den verschiedenen Spektralbereichen und ist definiert als:*

$$\Sigma^{jk}{}_i = E[((D_j(x,y) - m_{ij}) \cdot (D_k(x,y) - m_{ik}))]$$

- Sie wird geschätzt als:

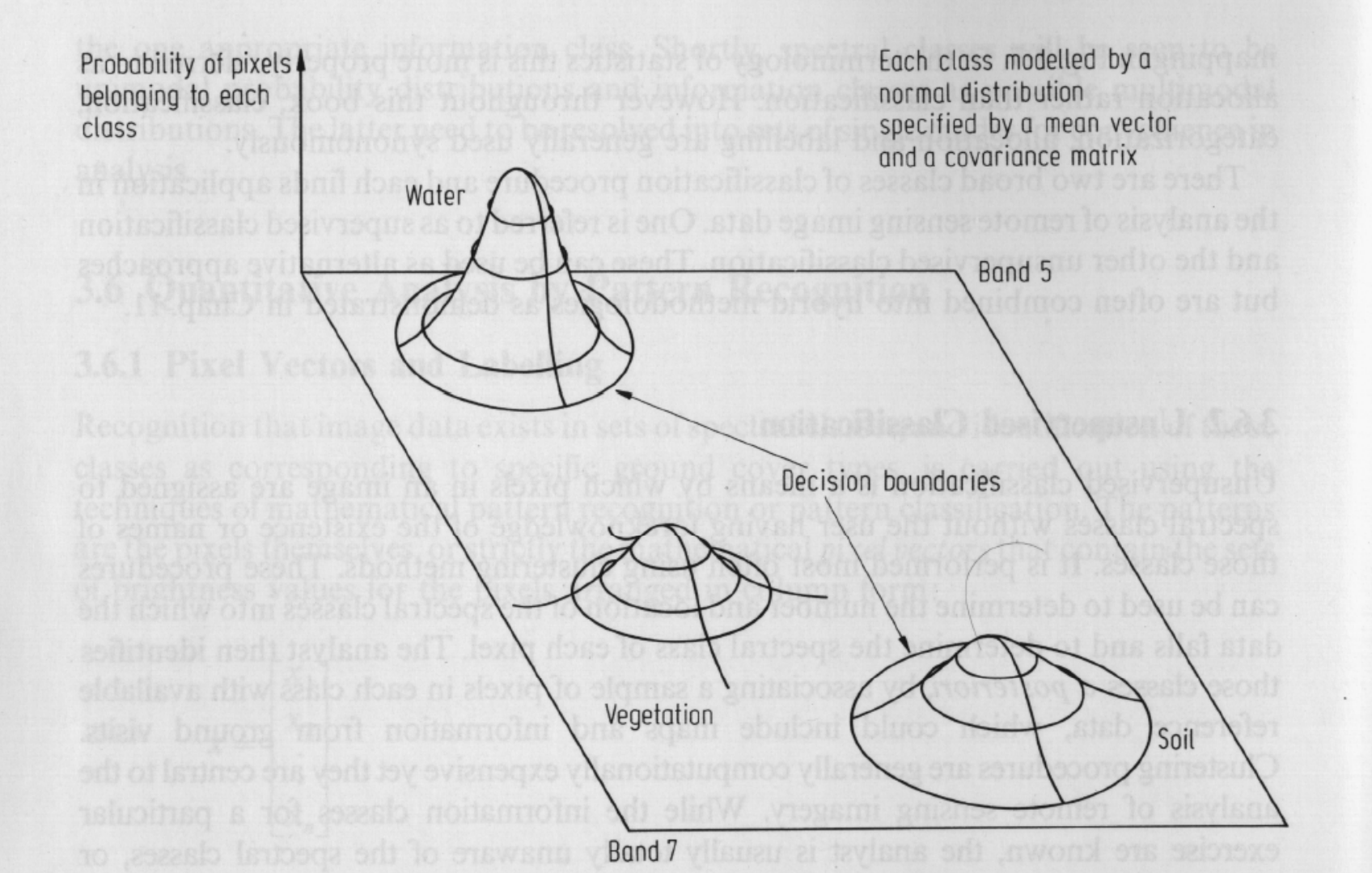
$$\Sigma^{jk}{}_i = \frac{1}{card(C_i) - 1} \cdot \sum_{(x,y) \in C_i} [((D_j(x,y) - m_{ij}) \cdot (D_k(x,y) - m_{ik}))]$$

- Es gilt:

$$(5)\ p(D(x,y)|C_i(x,y)) = \frac{1}{\sqrt{(2\pi)^m}\sqrt{|\Sigma_i|}} \cdot e^{\left\{-\frac{1}{2} \cdot ((D(x,y) - m_i)^T \cdot \Sigma_i^{-1} \cdot (D(x,y) - m_i))\right\}}$$

10.5 Normalverteilungen (V)

Beispiel



10.5 Maximum-Likelihood Klassifikation (III)

Endgültige Maximum-Likelihood Entscheidungsregel

- Durch Einsetzen von (5) erhalten wir

$$g_{i(D(x,y))} = \ln(p(C_i(x,y))) + \ln(p(D(x,y)|C_i(x,y)))$$

$$g_{i(D(x,y))} = \ln(p(C_i(x,y))) - \frac{1}{2} \cdot \ln|\Sigma i| - \frac{1}{2} \cdot ((D(x,y) - m_i)^T \cdot \Sigma i^{-1} \cdot (D(x,y) - m_i))$$

- Falls die $p(C_j(x,y))$ unbekannt sind, wird die Gleichverteilung aller Klassen C_i angenommen. Dann vereinfacht sich die Funktion g_i zu:

$$g_{i(D(x,y))} = -\ln|\Sigma i| - ((D(x,y) - m_i)^T \cdot \Sigma i^{-1} \cdot (D(x,y) - m_i))$$

Anzahl der Trainingspixel

- Es werden *mindestens m + 1* Trainingpixel pro Klasse C_i benötigt, damit die Kovarianz-Matrix nicht singulär wird.
- Experimentelle Untersuchungen zeigen jedoch, dass in praktischen Anwendungen mindestens *10 m* und für gute Ergebnisse etwa *100 m* Trainingspixel pro Klasse nötig sind.

10.5 Maximum-Likelihood Klassifikation (IV)

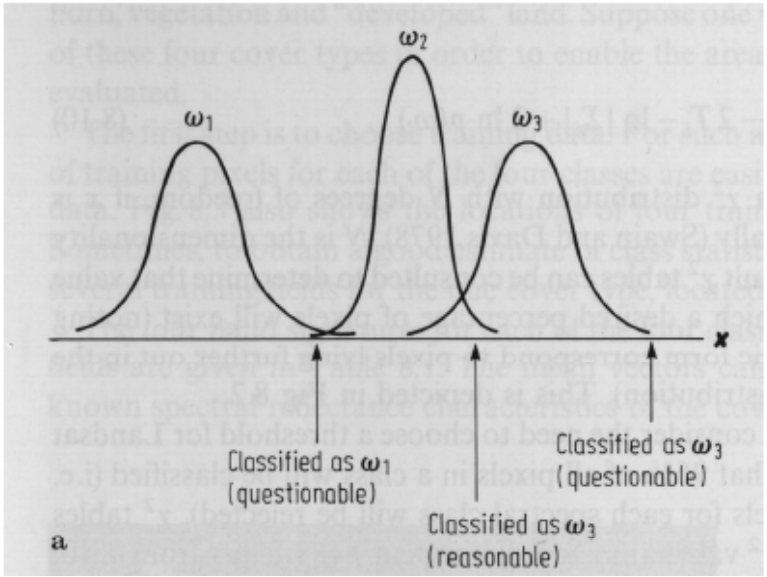
Problem

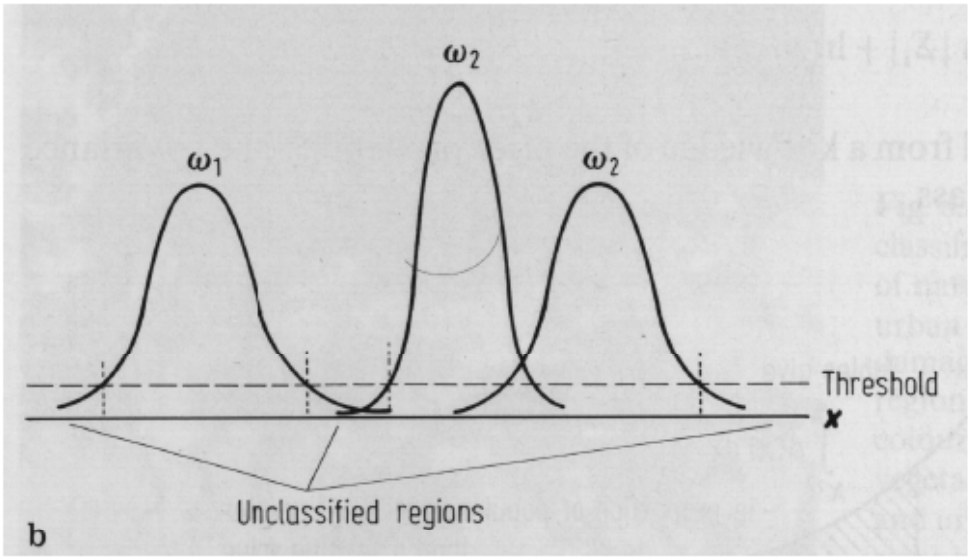
- Nach der bisherigen Entscheidungsregel wird jedes Pixel einer Klasse zugeordnet, auch wenn die bedingte Wahrscheinlichkeit für diese Klasse sehr klein ist.
- Dieser Fall kann z.B. auftreten, wenn für eine Klasse nicht genügend Trainingsdaten vorhanden sind. Er führt leicht zu Fehlern in der Klassifikation (a).

10.5 Maximum-Likelihood Klassifikation (IV)

Lösung: Einführung von Grenzwerten

- Wähle einen *Grenzwert (Threshold)* p_{min}.
- Pixel, deren Maximum-Likelihood-Wert nicht grösser als p_{min} ist, werden keiner Klasse zugeordnet (b).

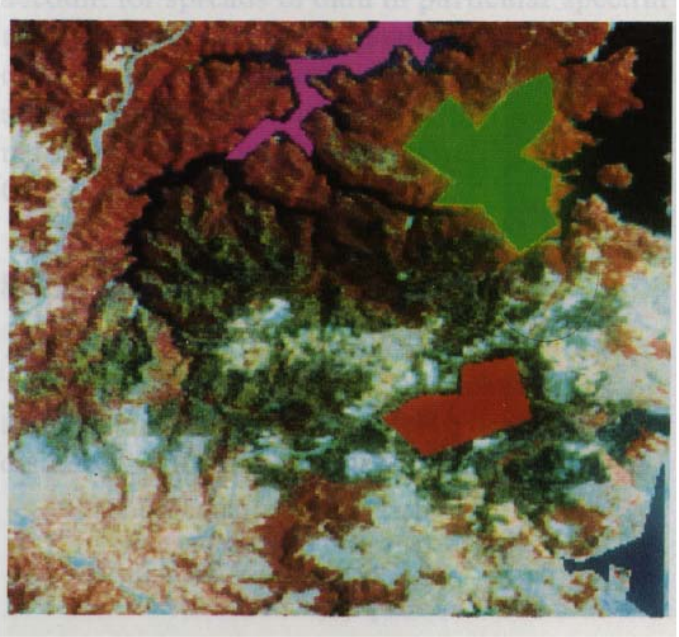




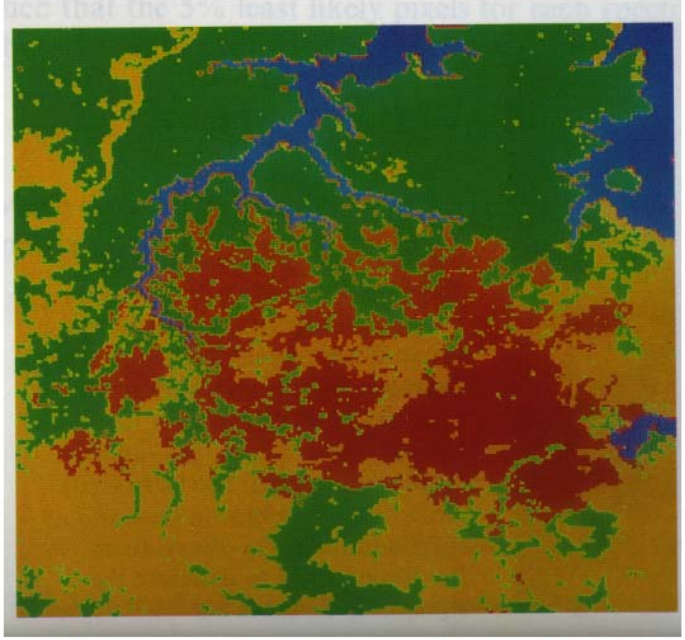
10.5 Maximum-Likelihood Klassifikation (V)

Beispiel

- Landsat Rasterbild 256 x 276 Pixel
- Vier Klassen: water, fire burn, vegetation, urban
- für jede Klasse ein Trainingsfeld (Menge räumlich benachbarter Trainingspixel



Eingangsbild mit Trainingsfeldern



Ausgangsbild (thematische Karte)

10.5 Maximum-Likelihood Klassifikation (VI)

Beispiel (Fortsetzung)

- Mittelwerte und Kovarianz-Matrizen für die vier Klassen

Class	Mean vector	Covariance matrix			
Water	44.27	14.36	9.55	4.49	1.19
	28.82	9.55	10.51	3.71	1.11
	22.77	4.49	3.71	6.95	4.05
	13.89	1.19	1.11	4.05	7.65
Fire burn	42.85	9.38	10.51	12.30	11.00
	35.02	10.51	20.29	22.10	20.62
	35.96	12.30	22.10	32.68	27.78
	29.04	11.00	20.62	27.78	30.23
Vegetation	40.46	5.56	3.91	2.04	1.43
	30.92	3.91	7.46	1.96	0.56
	57.50	2.04	1.96	19.75	19.71
	57.68	1.43	0.56	19.71	29.27
Developed (urban)	63.14	43.58	46.42	7.99	−14.86
	60.44	46.42	60.57	17.38	−9.09
	81.84	7.99	17.38	67.41	67.57
	72.25	−14.86	−9.09	67.57	94.27

- Tabellarisches Ergebnis der Klassifikation

Class	No. of pixels	Area (ha
Water	4830	2137
Fireburn	14182	6274
Vegetation	28853	12765
Developed (urban)	22791	10083

10.6 Nearest-Neighbor Klassifikation (I)

Motivation

- Die Maximum-Likelihood Klassifikation benötigt eine zuverlässige Schätzung der Mittelwerte und der Kovarianz-Matrizen für jede Klasse.
- Bei gleicher Menge von Trainingspixeln können die Mittelwerte zuverlässiger geschätzt werden als die Kovarianz-Matrizen.

  ⇒ entwickle Klassifikator, der nur Mittelwerte benutzt

Idee

- Bestimme die Mittelwerte m_i *für jede Klasse* C_i .
- Ordne jedes Pixel *D(x,y)* der Klasse C_j mit dem nächstgelegen m_j (*Nearest Neighbor*) zu.

10.6 Nearest-Neighbor Klassifikation (II)

Nearest-Neighbor Entscheidungsregel

- Wir definieren

$$d(D(x,y), m_i) = (D(x,y) - m_i)^T \cdot (D(x,y) - m_i)$$

$$d(D(x,y), m_i) = D(x,y) \cdot D(x,y) - 2 \cdot m_i \cdot D(x,y) + m_i \cdot m_i$$

- Die *Nearest-Neighbor Entscheidungsregel* lautet wie folgt:

$$\forall i \neq j [g_{j(D(x,y))} > g_{i(D(x,y))}] \Rightarrow C_j(x,y)$$

mit

$$g_{i(D(x,y))} = 2 \cdot m_i \cdot D(x,y) - m_i \cdot m_i$$

10.6 Nearest-Neighbor Klassifikation (III)

Vergleich

- Nearest-Neighbor Klassifikation benötigt weniger Trainingspixel
- Die Trennflächen zwischen den *D(x,y)* verschiedener Klassen sind linear (Nearest-Neighbor) bzw. quadratisch (Maximum-Likelihood), so dass die Maximum-Likelihood Klassifikation bei Klassen mit konkaver Form eine wesentlich bessere Klassifikationsgüte liefert.
- Effizienz
  - Maximum-Likelihood Klassifikation:
    - $\ln(p(C_i(x,y))) - \frac{1}{2} \cdot \ln|\Sigma i|$ wird für jede Klasse vorberechnet
    - für jedes Pixel und für jede Klasse bleiben *m² + m Multiplikationen und m² + 2 m + 1* Additionen auszuführen
  - Nearest-Neighbor Klassifikation
    - *2* m_i und $m_i * m_i$ werden für jede Klasse vorberechnet
    - für jedes Pixel und für jede Klasse *m* Multiplikationen und *m* Additionen

⇒ Nearest-Neighbor Klassifikation ist wesentlich effizienter