LMU LUDWIG-MAXIMILIANS-UNIVERSITY MUNICH
DEPARTMENT INSTITUTE FOR INFORMATICS
DATABASE SYSTEMS GROUP

Kapitel 10
Klassifikationsverfahren

Skript zur Vorlesung
Geo-Informationssysteme
Wintersemester 2015/16
Ludwig-Maximilians-Universität München
Vorlesung: PD Dr. Peer Kröger

10. Klassifikationsverfahren

1. Einführung
2. Partitionierendes Clustering
3. Maximum-Likelihood Klassifikation
4. 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)

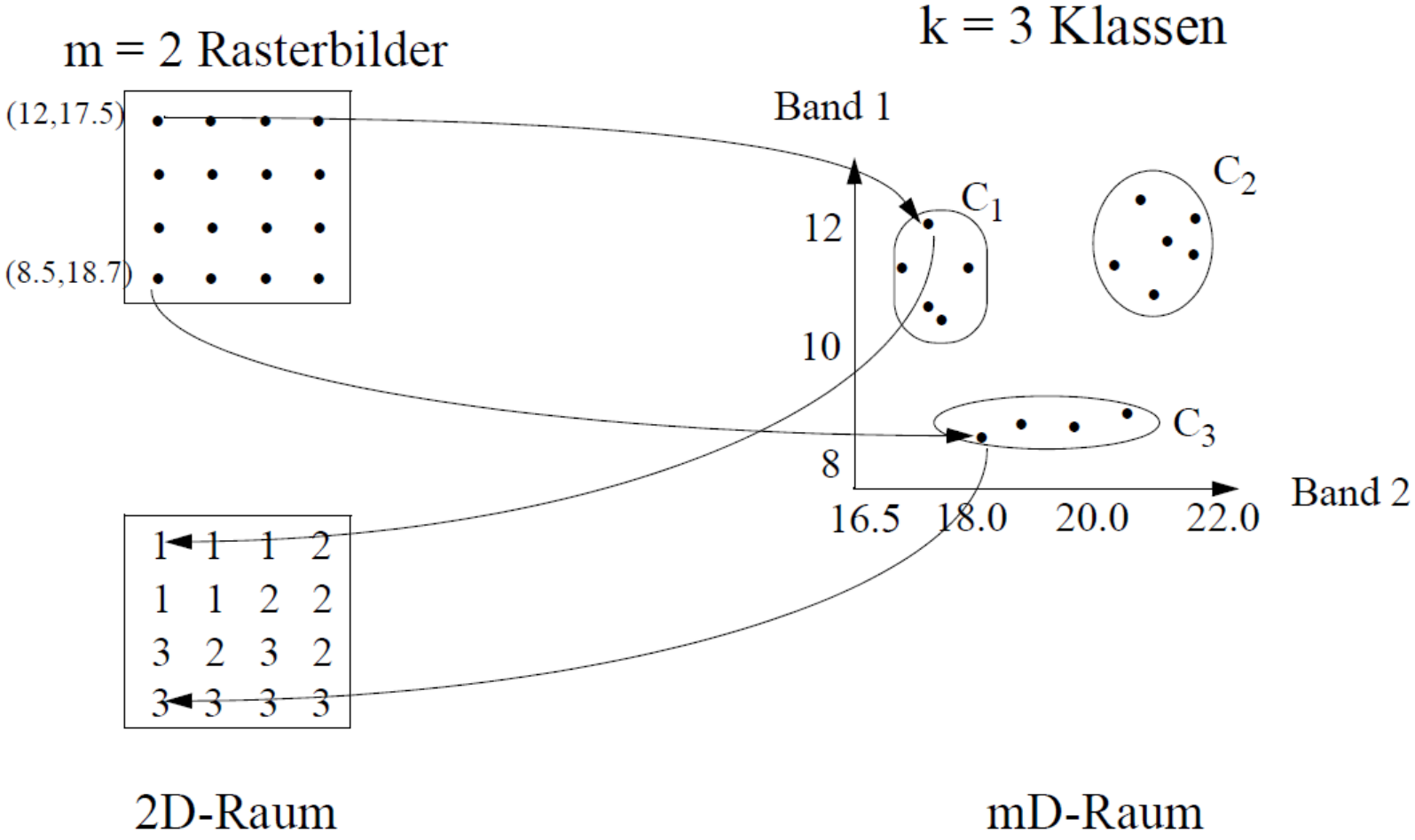
Anwendung

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

Lösungsansätze

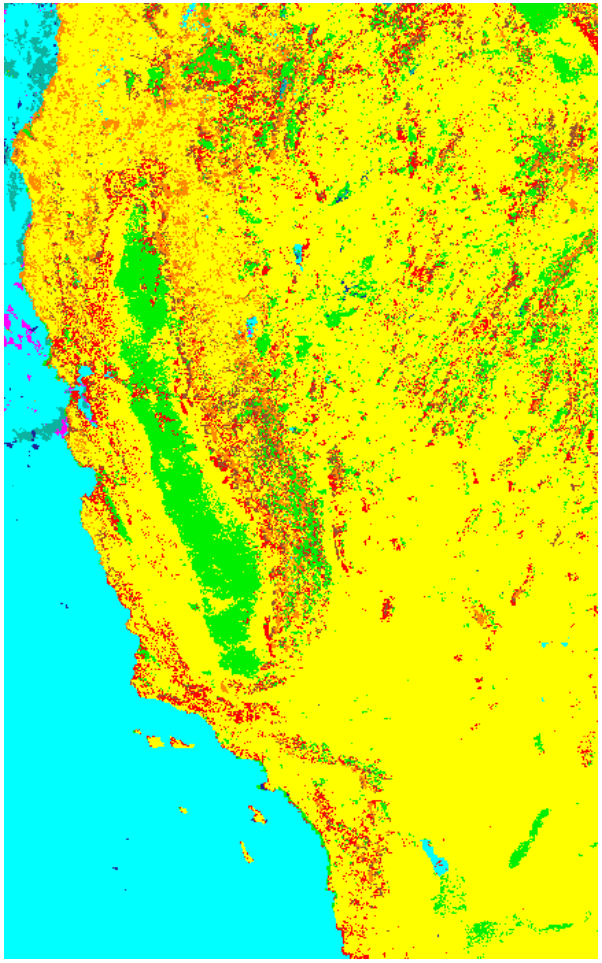
- nicht überwachte Klassifikation (*unsupervised classification)*
  - die Klassen C_i (und typischerweise auch deren Anzahl k) sind noch nicht bekannt
- überwachte Klassifikation (*supervised classification)*
  - die Klassen C_i sind bereits bekannt und für jede Klasse gibt es bereits einige „gelabelte“ Pixel

Schema einer Klassifikation



10.1 Einführung (IV)

Beispiel (Landnutzung von Kalifornien)



10.1 Einführung (V)

Überwachte Klassifikation

- Die Klassen C_i *, $1 \leq i \leq k$, 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, $1 \leq i \leq k$, 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.

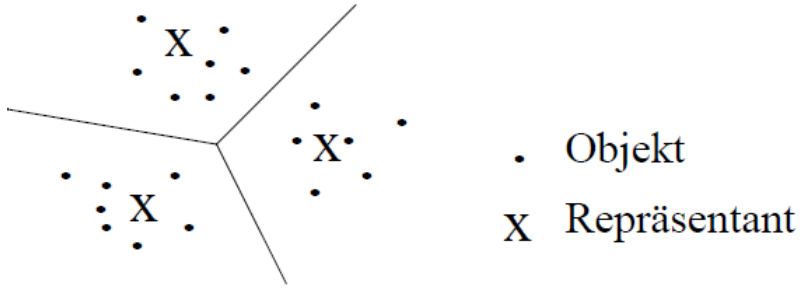
Vorgehen

- 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



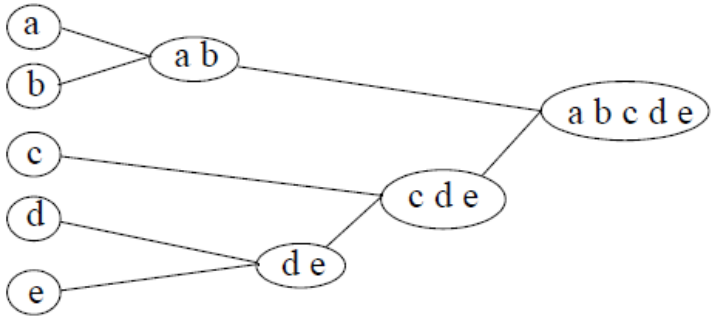
- Zwei Arten von Repräsentanten:
  - Schwerpunkt des Clusters (*k-means Algorithmen)*
  - ein Objekt des Clusters (*k-medoid Algorithmen)*

⇒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

10.2 Partitionierendes Clustering (II)

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)

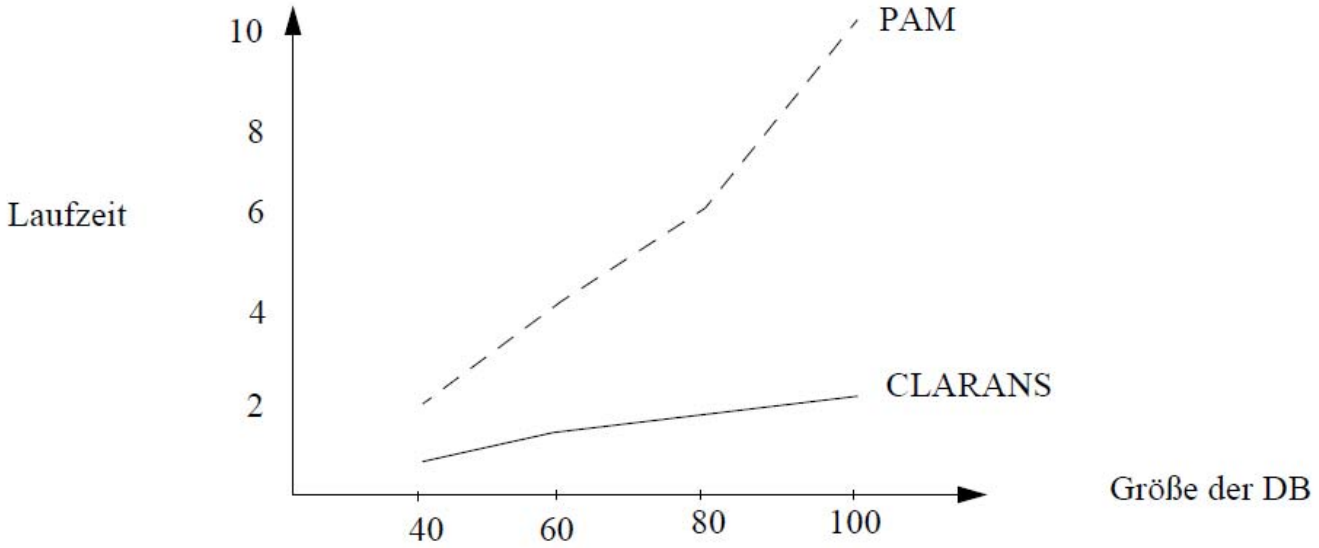
ALGORITHM CLARANS (Datenbank DB, int k, Distanzfunktion dist, int numlocal, int maxneighbor)
Medoide := {}; Nichtmedoide := DB; Kmin := MAXFLOAT; BestMedoide := {};
FOR i FROM 1 TO numlocal DO
wähle zufällig k der Objekte aus DB als Medoide;
WHILE NOT maxneighbor Paare (Medoid, Nichtmedoid) betrachtet 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

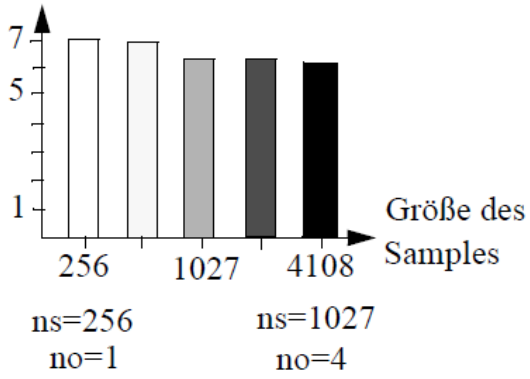
- Ein *Sample* ist eine repräsentative Stichprobe einer Menge von Daten.
- R*-Baum erhält räumliche Nachbarschaft so gut wie möglich.

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

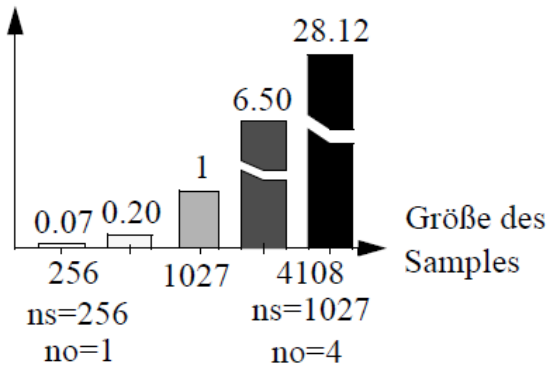
Größe des Samples

- 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



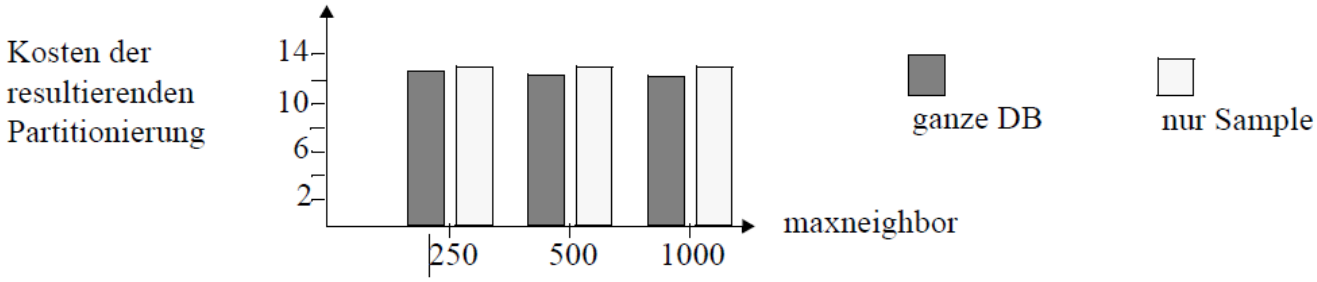
Relative Laufzeit von CLARANS



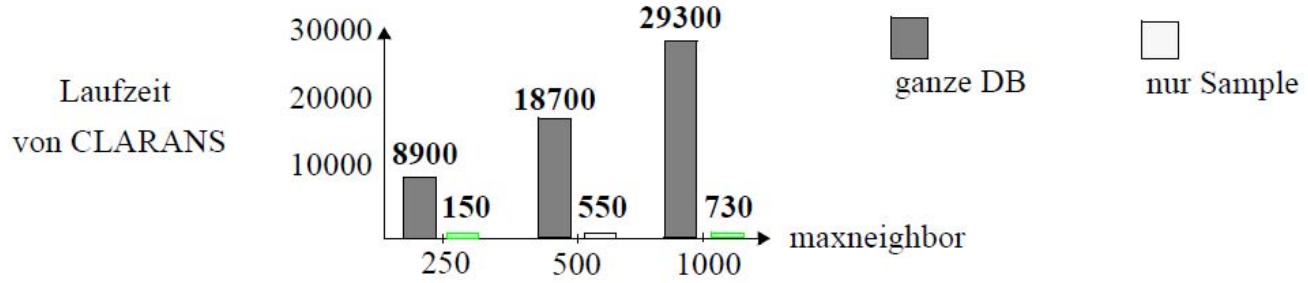
10.2 Partitionierendes Clustering (VII)

Leistungsuntersuchung des Sampling

- Vergleich der Effektivität



- Vergleich der Effizienz



=> kleiner Verlust an Effektivität und großer Gewinn an Effizienz

10.3 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).

10.3 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:

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

10.3 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.

10.3 Normalverteilungen (II)

Eindimensionaler Fall

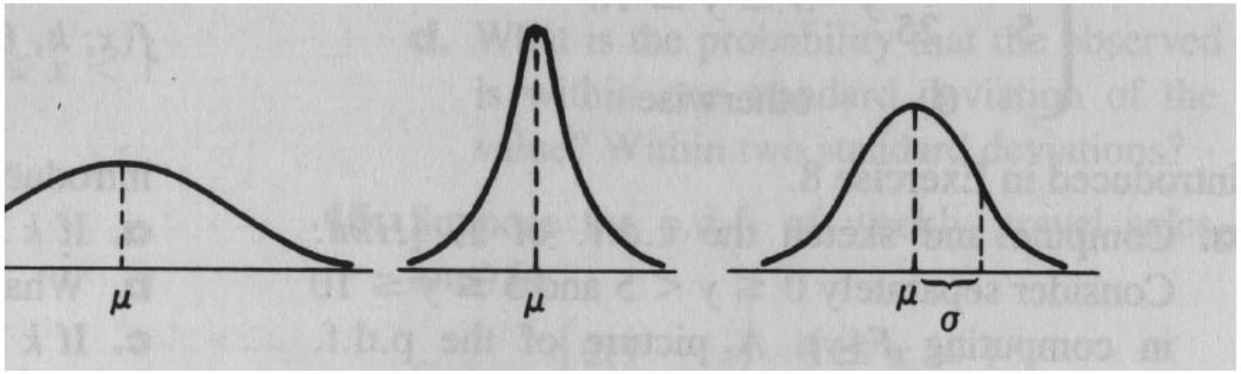
- Eindimensionaler Fall: m = 1, d.h. nur ein Grauwert pro Pixel. Dann gilt

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

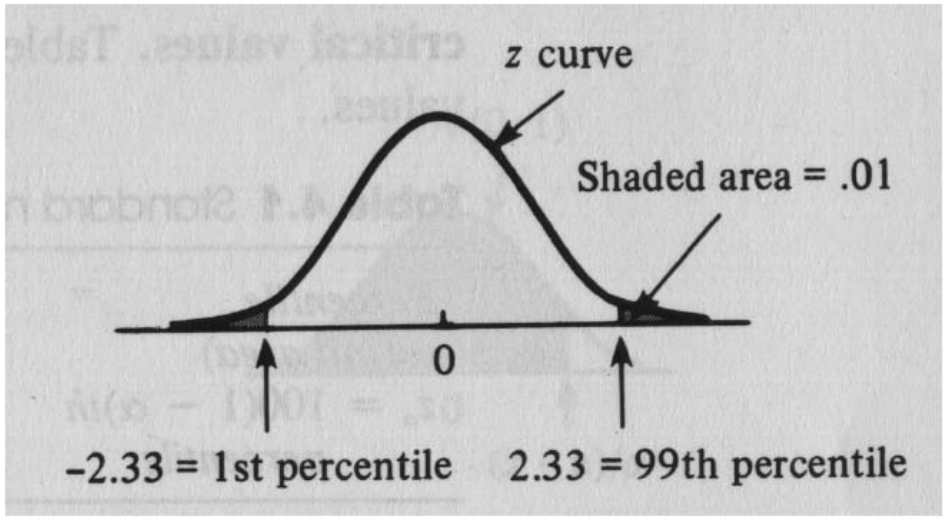
  mit m_i als Mittelwert und σ_i als Standardabweichung von $D(x,y)$ für die Klasse C_i
- Annahme: es ist eine hinreichend große Menge von D(x,y)-Werten gegeben.
- Dann wird m_i geschätzt als Mittelwert der $D(x,y)$, σ_i^2 als Durchschnitt der Quadrate der Differenz der D(x,y) zu m_i.

10.3 Normalverteilungen (III)

Beispiel



μ = m: Mittelwert, σ: Standard-Abweichung



10.3 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:

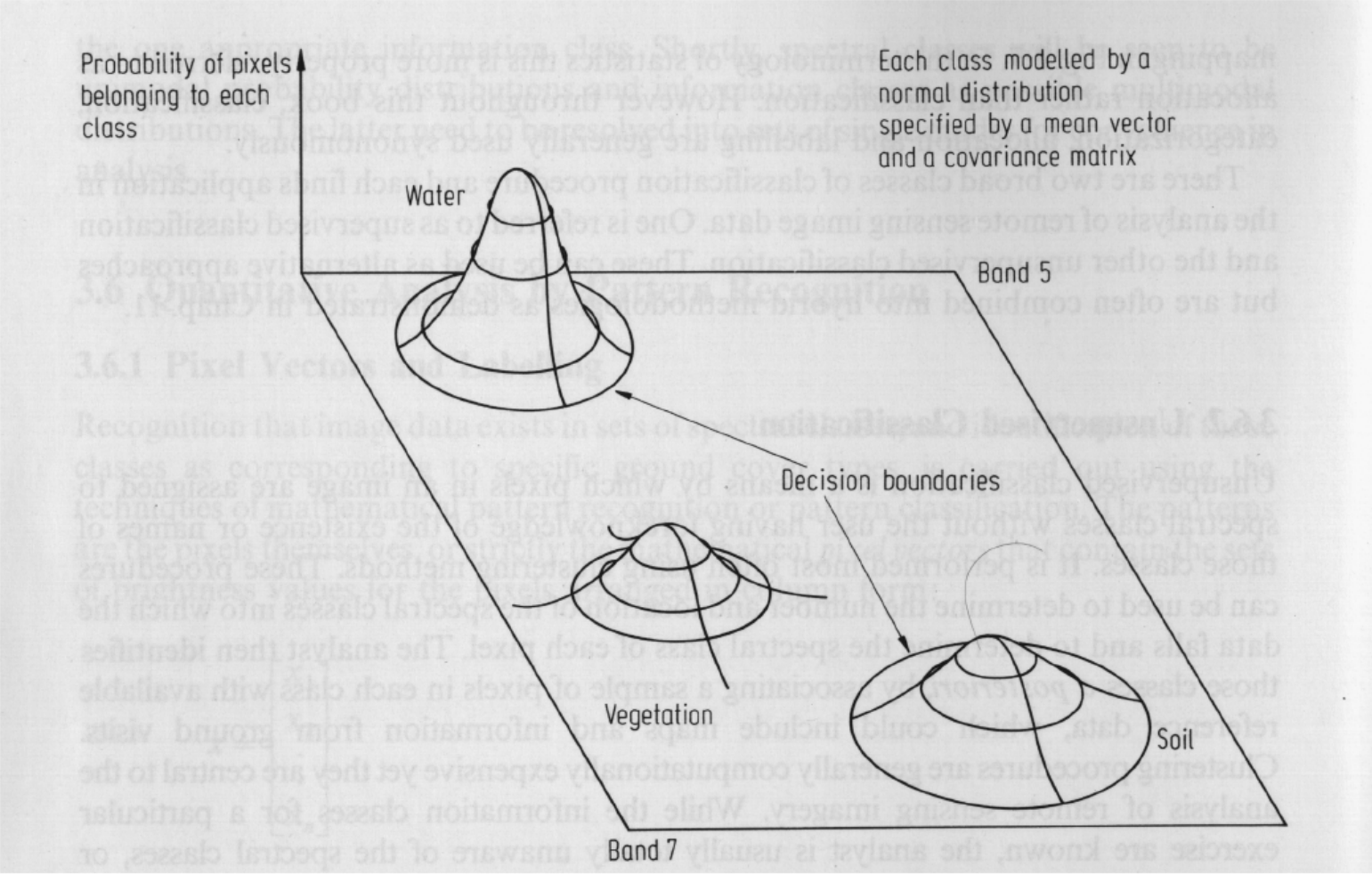
$$\Sigma^{jk}{}_i = \frac{1}{card(C_i) - 1} \cdot \sum_{(x,y) \in C_i} [((D_j(x,y) - m_{ij}) \cdot (D_k(x,y) - m_{ik}))]$$

- Es gilt:

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

10.3 Normalverteilungen (V)

Beispiel



10.3 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.

10.3 Maximum-Likelihood Klassifikation (IV)

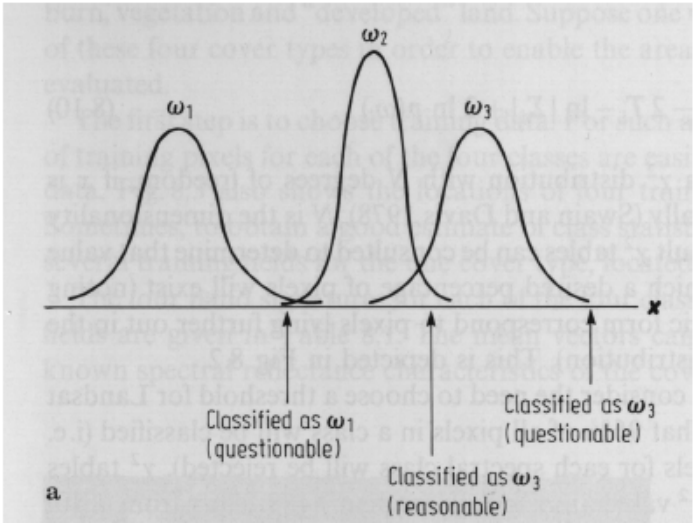
Problem

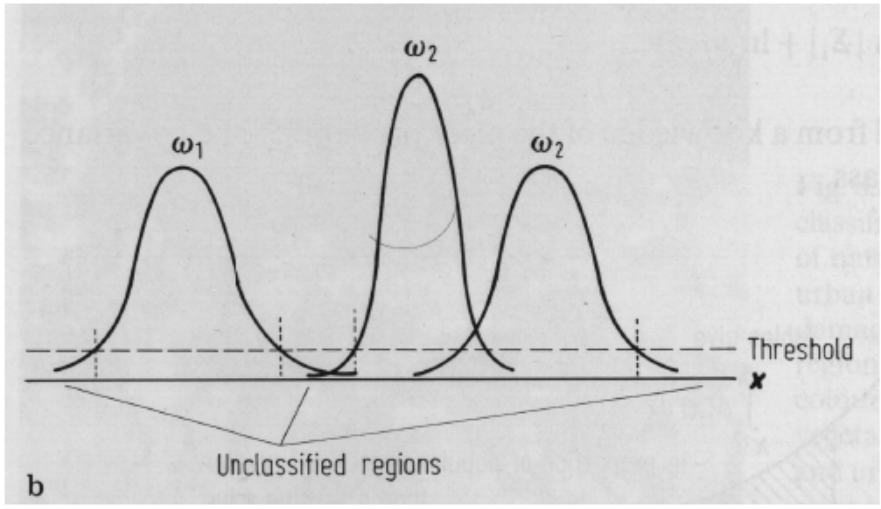
- Nach der bisherigen Entscheidungsregel wird jedes Pixel einer Klasse zugeordnet, auch wenn die bedingte Wahrscheinlichkeit für diese Klasse sehr klein ist.
- Dieser Fall kann z.B. auftreten, wenn für eine Klasse nicht genügend Trainingsdaten vorhanden sind. Er führt leicht zu Fehlern in der Klassifikation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

10.3 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).

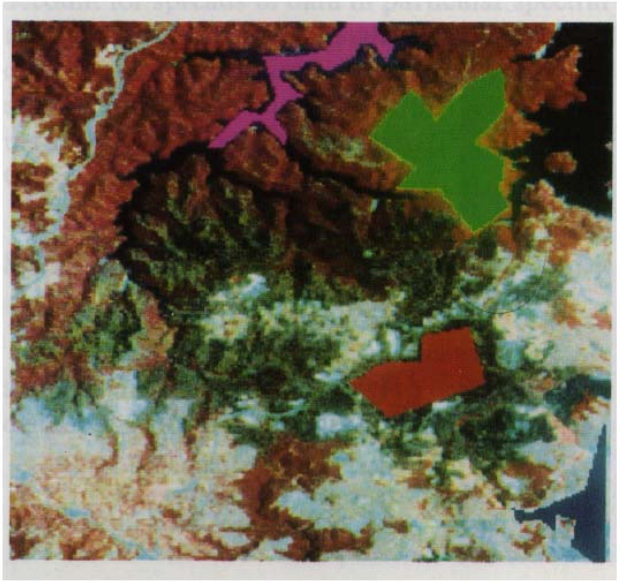




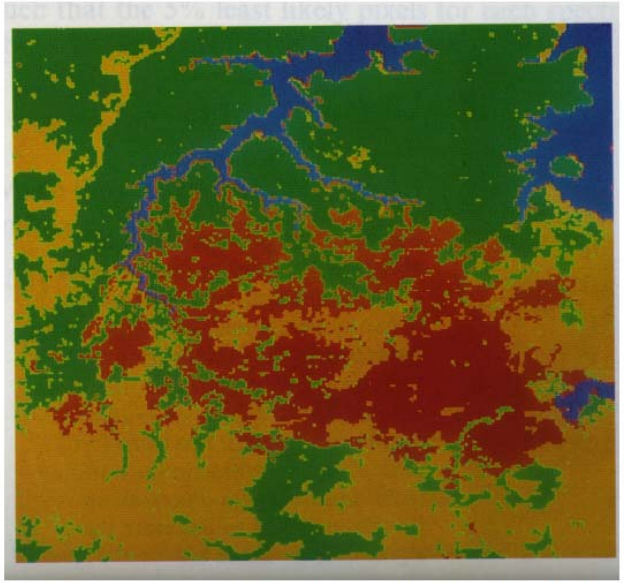
10.3 Maximum-Likelihood Klassifikation (V)

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)

10.3 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

10.4 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.

10.4 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

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

10.4 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 $m^2 + m$ *Multiplikationen und* $m^2 + 2\ m + 1$ 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