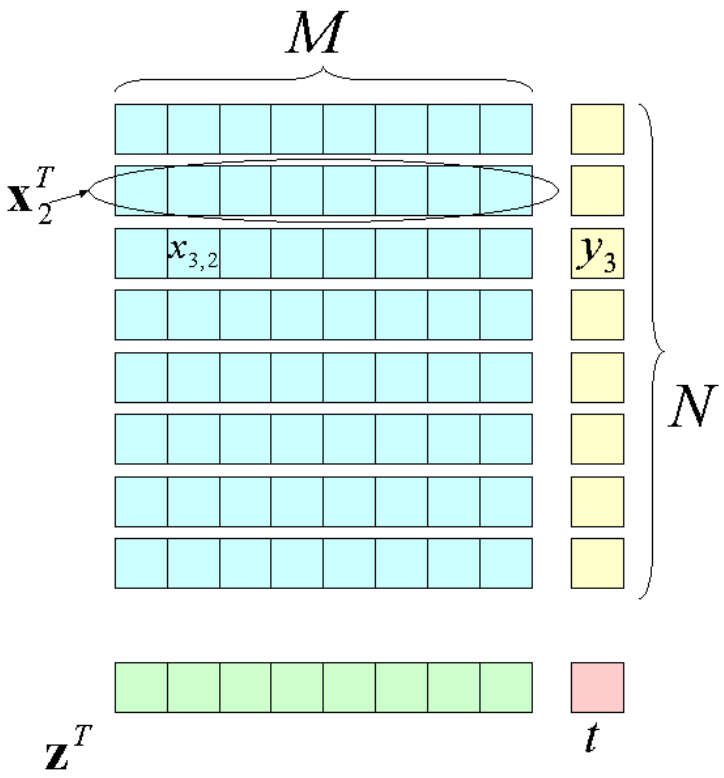
Die Datenmatrix und Memory-basiertes Lernen

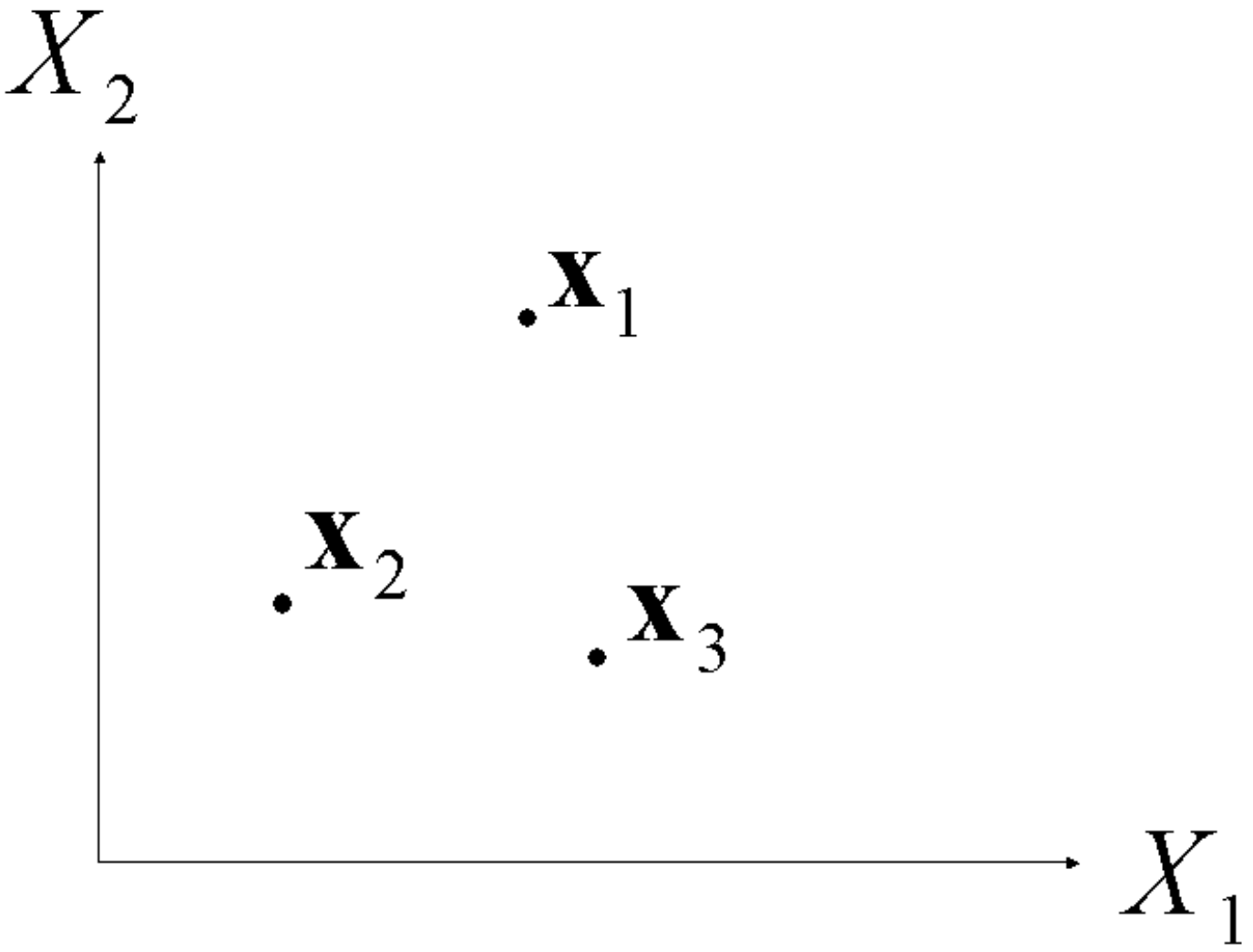
Volker Tresp

Die Datenmatrix für Überwachtes Lernen

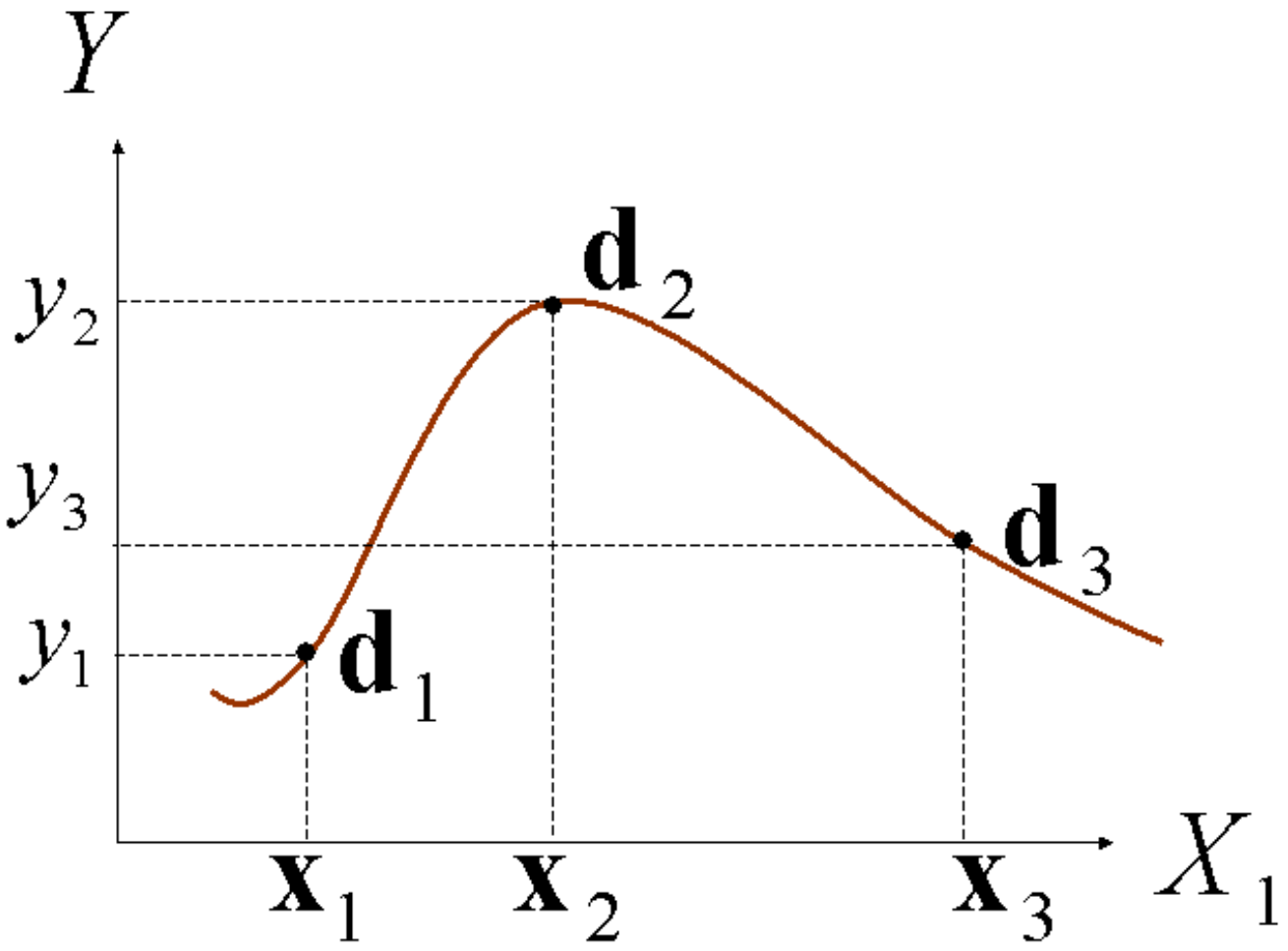


X_j	j-te Eingangsvariable
X	$= (X_1, \ldots, X_M)^T$ Vektor von Eingangsvariablen
M	Anzahl der Eingangsvariablen
N	Anzahl der Datenpunkte
Y	Ausgangsvariable
$\mathbf{x}_i$	$= (x_{i,1}, \ldots, x_{i,M})^T$ i-ter Eingangsvektor
$x_{i,j}$	j-te Komponente von $\mathbf{x}_i$
y_i	i-te Zielgröße zu $\mathbf{x}_i$
$\widehat{y}_i$	Vorhersage zu $\mathbf{x}_i$
$\mathbf{d}_i$	$= (x_{i,1}, \ldots, x_{i,M}, y_i)^T$ i-tes Muster
D	$= \{\mathbf{d}_1, \ldots, \mathbf{d}_N\}$ (Trainings-) Datensatz
$\mathbf{z}$	Testeingangsvektor
t	Unbekannte Testzielgröße zu $\mathbf{z}$

Eingangsvektoren: Datenpunkte im Eingangsraum



Datenmuster



Die Variablen

- Die Variablen beschreiben Eigenschaften (Attribute) von Objekten oder Personen, Messgrößen, Zustandsgrößen, Messwerte einer Zeitreihe, ...
- Die Domäne (der Wertebereich) einer Variablen ist in der Regel entweder
  - stetig, $x_{i,j} \in \Re$
  - binär diskret, $x_{i,j} \in \{0, 1\}$, oder $x_{i,j} \in \{-1, 1\}$
  - diskret, $x_{i,j} \in \{1, 2, \ldots, C\}$

Offensichtliche Lösung: Memorisieren

- Ziel ist die Klassifikation eines neuen Eingangsmusters $\mathbf{z}$. Wir nehmen C Klassen an, so dass $y_i \in \{1, \ldots, C\}$
- Beachte, dass Klassenzuordnung nicht eindeutig sein muss (überlappende Klassen), d.h. es kann sein, dass trotz $\mathbf{x}_i = \mathbf{x}_j$ es gilt, dass $y_i \neq y_j$. In der Regel wünschen wir, dass in diesem Fall die Klasse vorhergesagt wird, die gegeben $\mathbf{x}_i$ am wahrscheinlichsten ist
- Wenn die Anzahl unterschiedlicher Eingangsmuster begrenzt ist durch K und $N >> K$, dann wertet man die Muster mit übereinstimmendem Eingangsvektor aus:

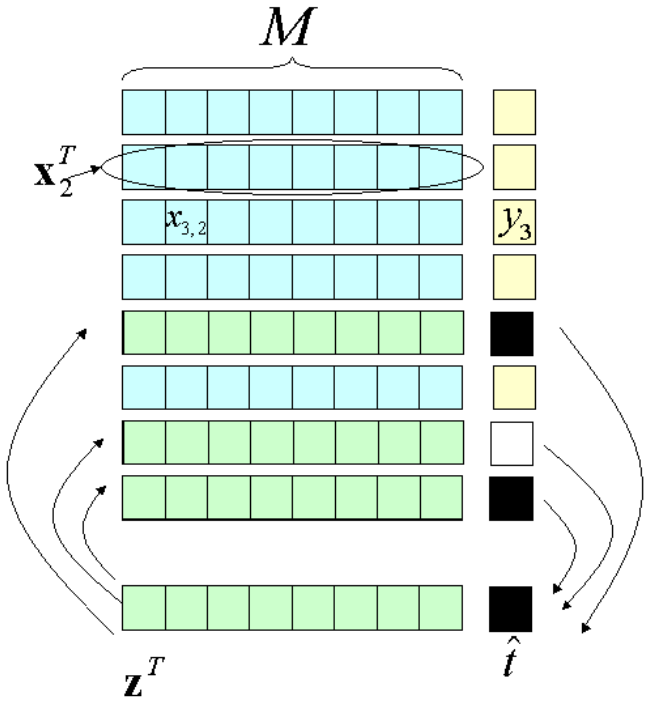
$$n_l = \sum_{i=1}^{N} I(y_i = l \wedge \mathbf{x}_i = \mathbf{z})$$

und man ordnet zu

$$\hat{t} = \arg\max_l n_l$$

mit Indikatorfunktion $I(x) = 1$ falls $x = 1$ ist und $I(x) = 0$ sonst.

Memorisieren (2)



Dieser Ansatz kann nicht häufig angewandt werden

- im Allgemeinen $K >> N$. Zum Beispiel bei einem M-dimensionalen binären Eingangsvektor gibt es 2^M mögliche Varianten.
- wenn Eingangsgrößen kontinuierliche Werte annehmen können, ist dieser Ansatz nicht anwendbar

Prinzip der Ähnlichkeit

- Betrachten wir ein Trainingsmuster und verändern die Eingangsgrößen minimal, so macht es Sinn anzunehmen, dass sich auch die Zielgröße nur minimal verändert
- Diese Annahme der Regularität oder Stetigkeit ist eine (a priori) Grundannahme im ML (Regularization Theory): Ähnliche Eingangsgrößen produzieren ähnliche Ausgänge
- Ein zentrales Problem im ML ist es, problemangepasst das richtige Ähnlichkeitsmaß zu finden
- Beispiel: Eingangsgrößen: Größe, Alter, Haarfarbe, Ausgang: Gewicht. Das Gewicht wird primär eine Abhängigkeit von Größe zeigen, eine gewisse Abhängigkeit vom Alter aber recht unabhängig von Haarfarbe zu sein. Dies bedeutet, dass das Ähnlichleitsmaß sensitiv sein sollte in Bezug auf die Größe, weniger sensitiv in Bezug auf das Alter und kaum sensitiv auf die Haarfarbe.
- Soll eine andere Zielgröße vorhergesagt werden (Einkommen) kann sich ein ganz anderes optimales Ähnlichkeitsmaß ergeben (Alter wird am wichtigsten)

Nachbarschaftsklassifikatoren

- Die Idee ist es, ein Abstandsmaß zu definieren und für die Zuordnung der Trainingsmuster den Abstand zu $\mathbf{z}$ zu bewerten.
- Nächste-Nachbarschaft Klassifikation (Nearest-Neighbor Classification) mit

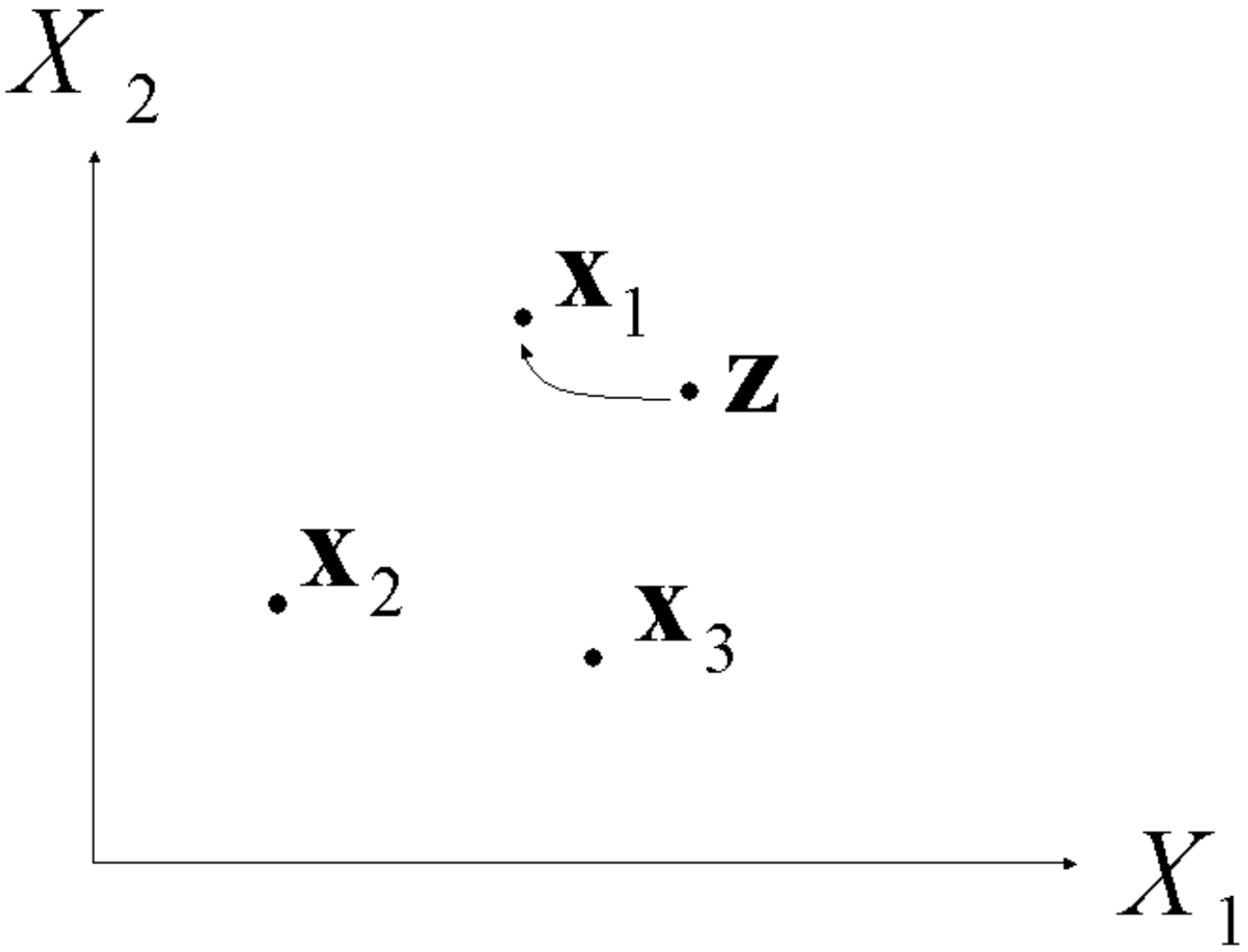
$$l = \arg\min_i \text{dist}(\mathbf{z}, \mathbf{x}_i)$$

  ordnen wir zu $\hat{t} = y_l$
- k-Nächste-Nachbarschaft Klassifikation (k-Nearest-Neighbor Classification): sei J die Menge der Indizes der k nächsten Nachbarn zu $\mathbf{z}$. Sei

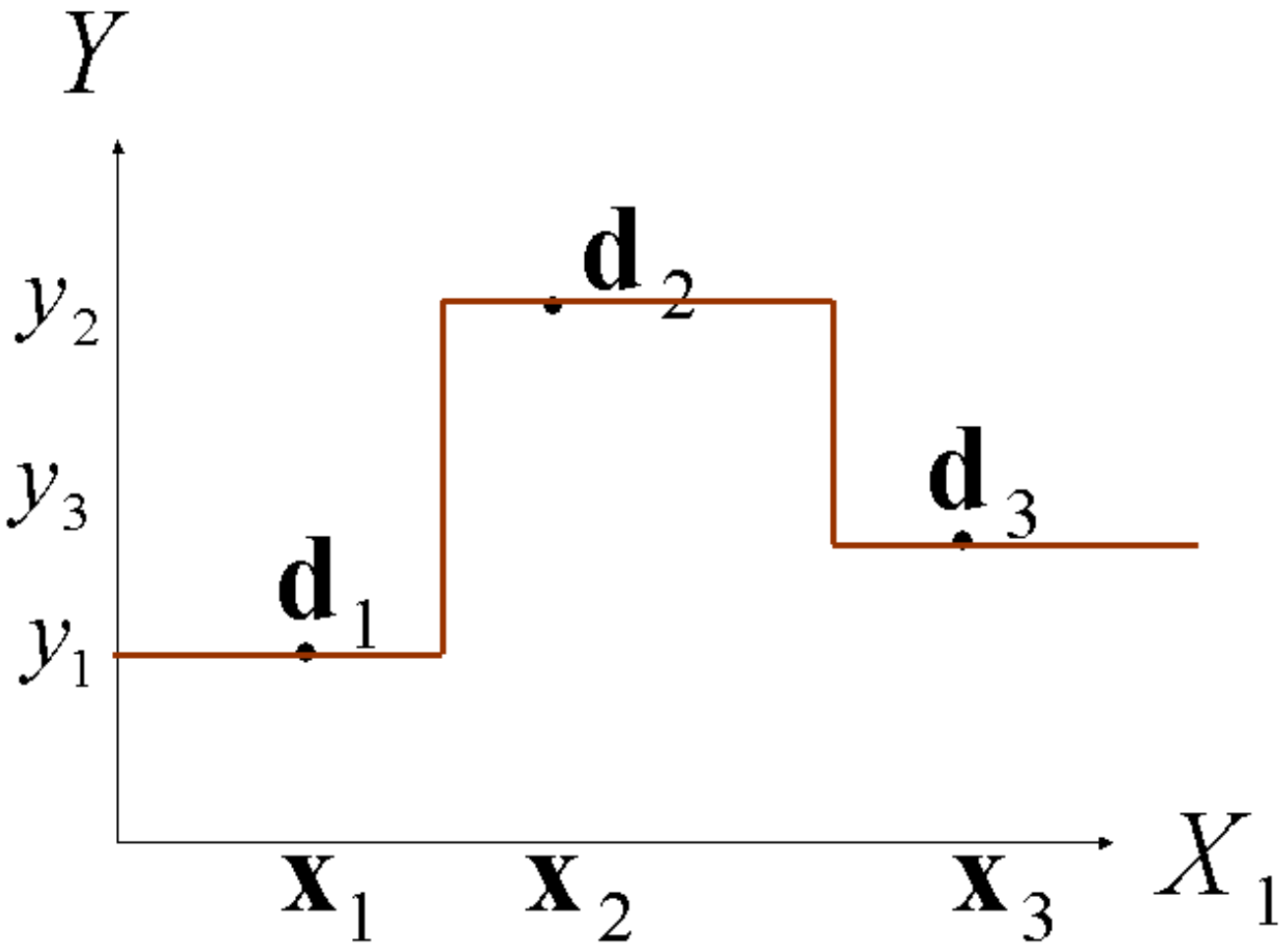
$$n_l = \sum_{i \in J} I(y_i = l)$$

  Dann klassifiziert man $\mathbf{z}$ nach $\hat{t} = \arg\max_l n_l$

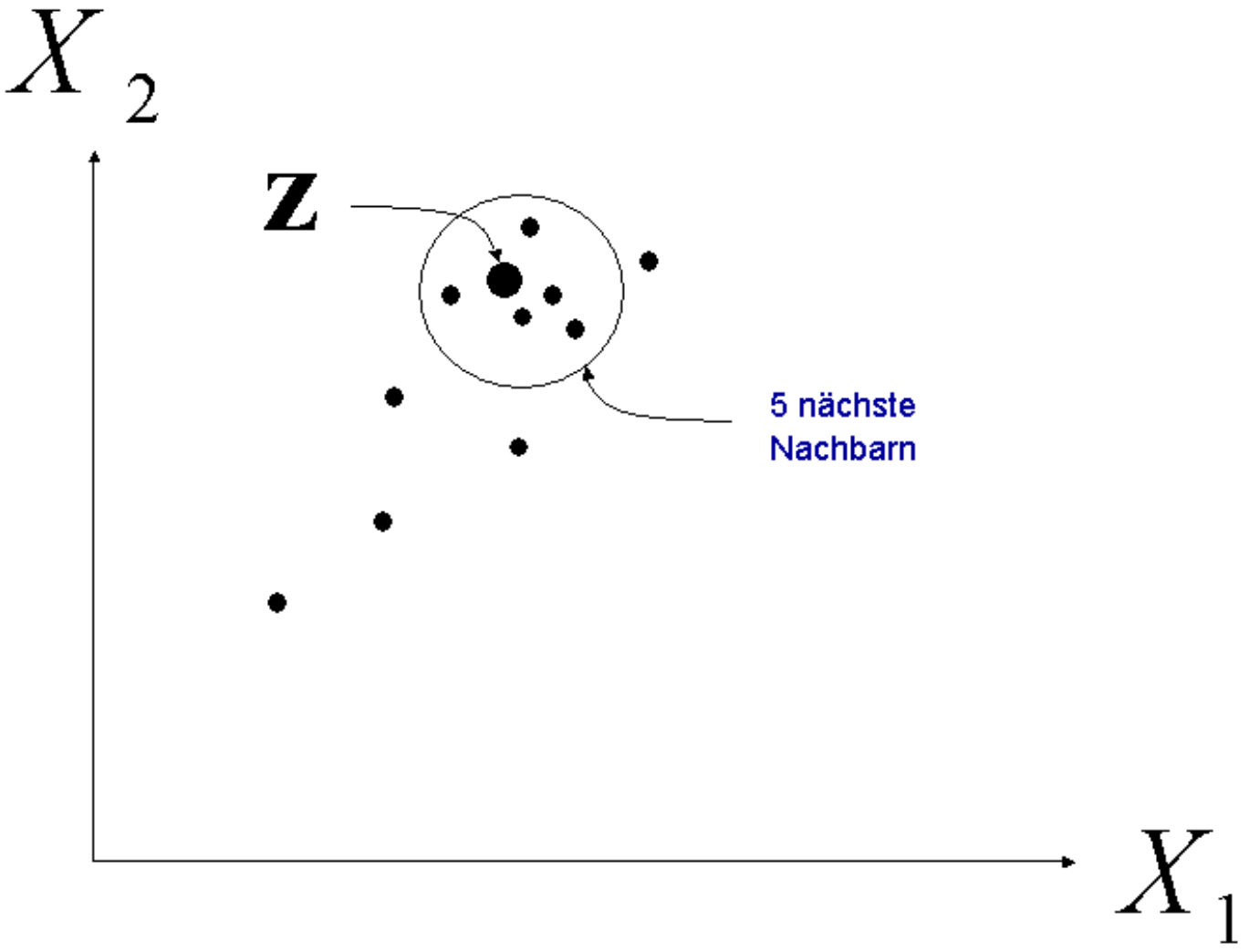
NN-Klassifikation



NN-Klassifikation



5-NN-Klassifikation



Abstandsmaße für stetige Variablen

- Euklidischer Abstand

$$\text{dist}_{euklid}(\mathbf{x}_i, \mathbf{z}) = \|\mathbf{x}_i - \mathbf{z}\| = \sqrt{\sum_{j=1}^{M} (x_{i,j} - z_j)^2}$$

Abstandsmaße für diskrete Variablen

- Für diskrete Variable benutzt man häufig die Anzahl der Unterschiede (*simple matching coefficient*):

$$\text{dist}_{simple}(\mathbf{x}_i, \mathbf{z}) = \frac{1}{M} \sum_{j=1}^{M} (1 - I(x_{i,j} = z_j))$$

  Dieses Abstandsmaßeignet sowohl für nominale diskrete Variablen, bei denen es keine natürliche Ordnung in den Zuständen gibt (z.b. Farben) als auch für ordinale diskrete Variablen, bei denen es eine solche natürliche Ordnung (z.b. Schulnoten) gibt.

Abstandsmaße für dünn besetzte diskrete Eingangsvektoren

- In manchen Anwendungen dominiert der Zustand 0; dies sollte man im Abstandsmaß berücksichtigen. Beispiel: Wenn es 1000 Objekte gibt und zwei Kunden jeweils ein unterschiedliches Objekt gekauft haben, so ist deren euklidischer Abstand $\sqrt{2}$ Haben zwei Kunden jedoch jeweils 100 Objekte gekauft von denen 95 gleich sind, ergibt sich als euklidischer Abstand $\sqrt{10}$ In diesem Fall wären entgegen der Intuition die beiden ersten Kunden ähnlicher als die beiden letzteren Kunden. Das Gleiche gilt für dist_{simple}.

- In Fällen, wo Objekte in irgendeinem Sinne generiert werden (Kaufen, Schreiben, ...) und wo typischerweise Nullen dominieren, wählt man

$$\text{dist}_{simple00}(\mathbf{x}_i, \mathbf{z}) = \frac{1}{M - F} \sum_{j=1}^{M} (1 - I(x_{i,j} = z_j))$$

  wobei F die Anzahl der Variablen ist, in denen beide Vektoren übereinstimmend gleich Null sind. Im Beispiel ergeben sich dann für die Abstände $2/(1000 - 998) = 1$ und $10/(1000 - 895) \approx 0.09$.

Abstandsmaße für dünn besetzte diskrete Eingangsvektoren (2)

- Man will vielleicht zwei Kunden als ähnlich einstufen, wenn sie verschieden häufig die gleichen Produkte gekauft haben. Hier eignet sich das Kosinusmaß. Aus der Definition des inneren Produktes ergibt sich als ein Maß der *Ähnlichkeit* für nicht-negative Größen

$$\cos(\mathbf{x}_i, \mathbf{z}) = \frac{\mathbf{x}_i^T \mathbf{z}}{\|\mathbf{x}_i\| \|\mathbf{z}\|} = \frac{\sum_{j=1}^{M} x_{i,j} z_j}{\sqrt{\sum_{j=1}^{M} x_{i,j}^2} \sqrt{\sum_{j=1}^{M} z_j^2}}$$

und als Kosinus-Abstandsmaß

$$\text{dist}_{\cos}(\mathbf{x}_i, \mathbf{z}) = 1 - \cos(\mathbf{x}_i, \mathbf{z}) \geq 0$$

In unserem Beispiel: ergibt sich als Abstand 1 und $1 - 95/100 = 0.05$.

Wie schon angedeutet gilt

$$\text{dist}_{\cos}(a\mathbf{x}_i, b\mathbf{z}) = \text{dist}_{\cos}(\mathbf{x}_i, \mathbf{z})$$

mit postitiven Skalaren a, b,

Abstandsmaße für dünn besetzte diskrete Eingangsvektoren (3)

- Pearson Korrelation. Wie das Kosinusmaß, nur dass vor Anwendung die Mittelwerte jeden *Datenvektors* abgezogen werden. Seien $\tilde{\mathbf{x}}_i = \mathbf{x}_i - mean(\mathbf{x}_i)$, $\tilde{\mathbf{z}} = \mathbf{z} - mean(\mathbf{z})$, dann ist

$$pearson(\mathbf{x}_i, \mathbf{z}) = \frac{\sum_{j=1}^{M} \tilde{x}_{i,j}\tilde{z}_j}{\sqrt{\sum_{j=1}^{M} \tilde{x}_{i,j}^2}\sqrt{\sum_{j=1}^{M} \tilde{z}_j^2}}$$

und als Abstandsmaße

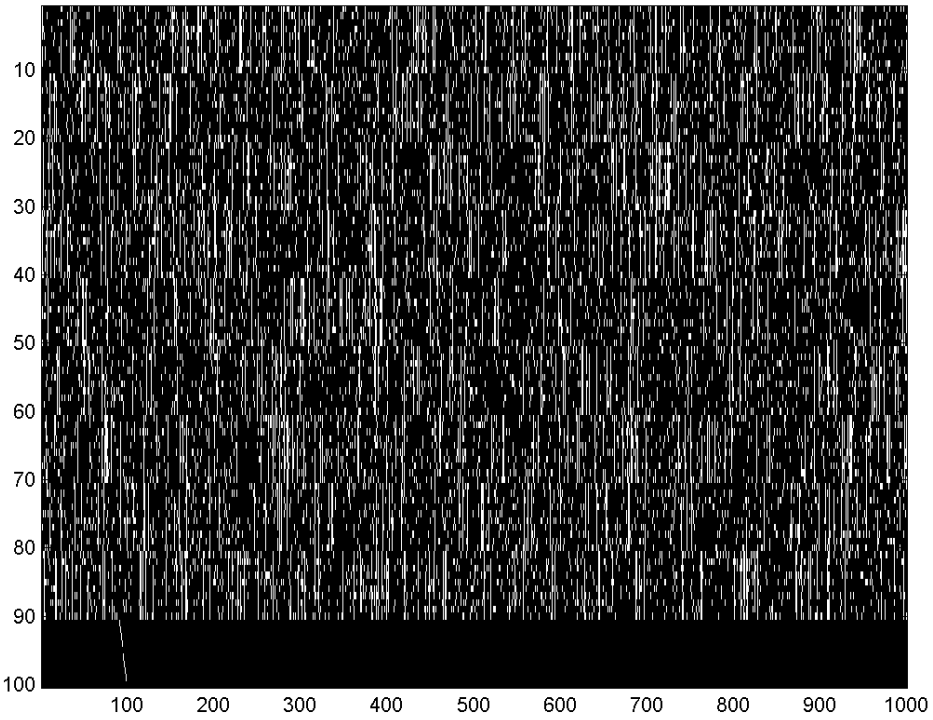
$$\text{dist}_{pearson}(\mathbf{x}_i, \mathbf{z}) = 1 - pearson(\mathbf{x}_i, \mathbf{z}) \geq 0$$

In unserem Beispiel: ergeben sich als Abstände 1.01 und 0.056. Pearson eignet sich auch für stetige Größen.

Bemerkungen zur Pearson Korrelation

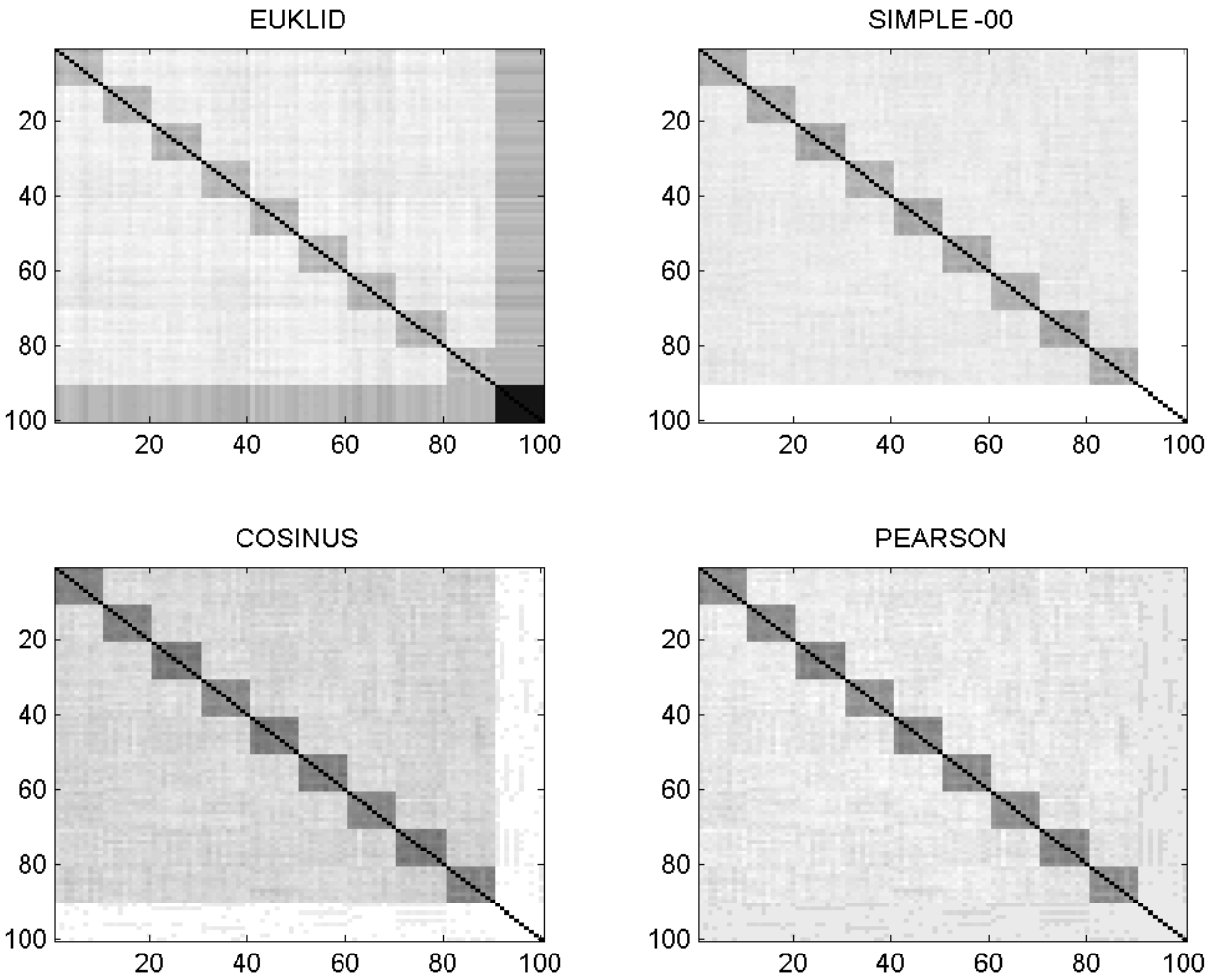
- Wir berechnen hier die Pearson Korrelation zwischen zwei Objekten (hier: Benutzern) also Zeilen der Datenmatrix
- Häufiger wir die Pearson Korrelation zwischen Spalten berechnet; sie entspricht dann einer Schätzung der *Covariance* zweier Variablen normiert über das Produkt der Varianzen

Hohe Dimensionen mit dünn besetzten gruppierten Daten: Datenmatrix mit $M = 1000$, Histogramme über Abstände



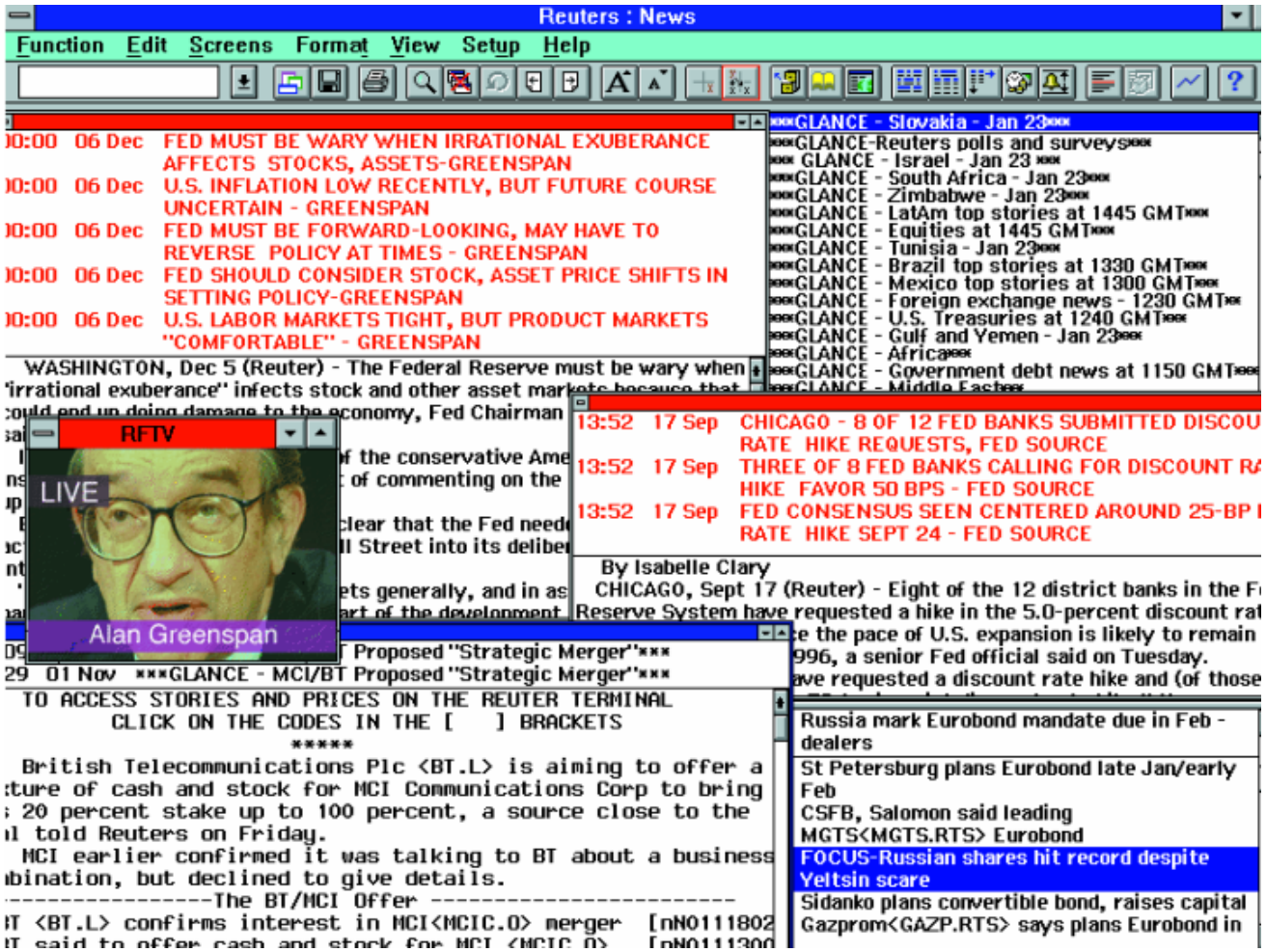
Dünn-besetzte Daten in 10 Clustern. Im zehnten Cluster gibt es in jedem Datenvektor jeweils nur einen Eintrag ungleich Null. Die Dimension jedes Datenvektors ist $M = 1000$

Hohe Dimensionen mit dünn besetzten gruppierten Daten: Datenmatrix mit $M = 1000$, heller Eintrag: hoher Abstand

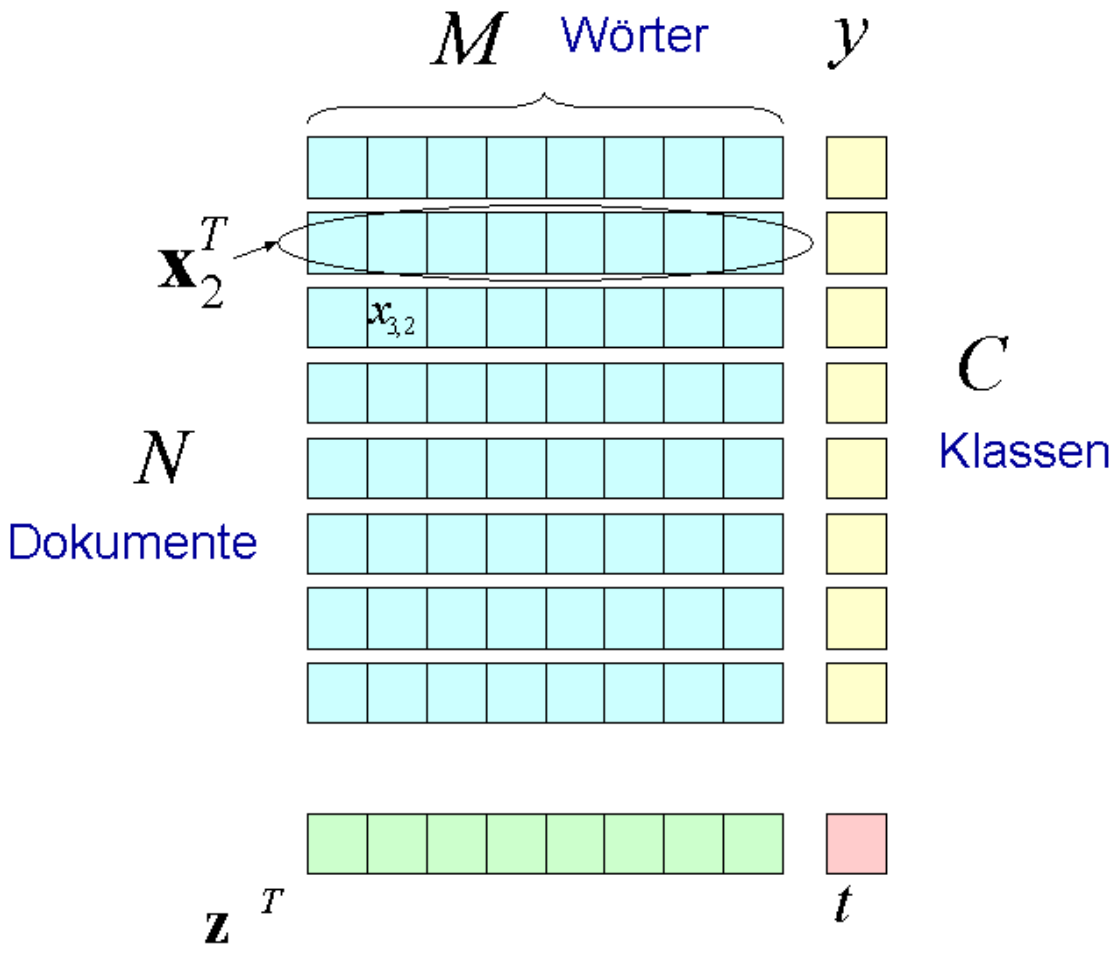


x

Beispiel: Klassifikation von Dokumenten



Vektorraummodell eines Dokuments


M Wörter
y
$\mathbf{x}_2^T$
$x_{3,2}$
C
Klassen
N
Dokumente
$\mathbf{z}^T$
t

Vektorraummodel eines Dokuments

- Je nach Vorverarbeitung kann $x_{i,j}$ verschiedene Größen darstellen
- (A) $x_{i,j} \in \{0, 1\}$ ist gleich 1, wenn Wort j in Dokument i vor, ansonsten 0
- (B) $x_{i,j} \in \{0, 1, 2, \ldots\}$ stellt dar, wie häufig Wort j in Dokument i vorkommt (*term frequency* tf)
- (C) (B) $x_{i,j} \in \Re$ wird durch Gewichtung von tf erzielt. Am beliebtesten ist sie *inverse document frequency*:

$$\text{idf}_j = \log\left(\frac{N}{n_j}\right)$$

  so dass, $x_{i,j} = \text{tfidf}_{i,j} = \text{tf}_{i,j} \times \text{idf}_j$. Hierbei ist n_j die Anzahl der Dokumente, in denen das Wort j mindestens einmal erscheint. Dadurch werden häufige Wörter weniger einflußreich. N is die Anzahl der Dokumente. Wenn ein Wort in allen Dokumenten vorkommt ist die Gewichtung gleich Null!

- Vorverarbeitungsschritte: Zurückführung auf Stammform (*stemming*), Eliminierung von *Stoppwords*; Kosinusmaß ist beliebt

Text categorization using weight-adjusted nearest neighbor classification: Han, Karyois, Kumar

	Source	# train	# test	# class	# words used
west-1	West Group	500	1500	10	977
west-2	West Group	300	900	10	1078
west-3	West Group	488	245	10	1035
west-4	West Group	559	280	10	887
west-5	West Group	621	311	10	1156
west-6	West Group	732	367	10	789
west-7	West Group	885	433	10	779
fbis	TREC-5	2463	1232	17	2000
trec6	TREC-5	1173	587	14	2000
reuters	Reuters-21578	6552	2581	59	2000

Table 1: Summary of data sets used.

Text categorization using weight-adjusted nearest neighbor classification: Han, Karyois, Kumar (2)

- tf
- Cosinusmaß
- C4.5, Ripper: Entscheidungsbäume, PEBLS, VSM, WAKNN: verschiedene k-NN mit Wortgewichtungen, Rainbow: naive-Bayes
- Reuters: Nachrichtenmeldungen, fbis: Foreign Broadcast Information Service, trec6L: LA-Times Zeitungsartiklel, west: juristische Dokumente

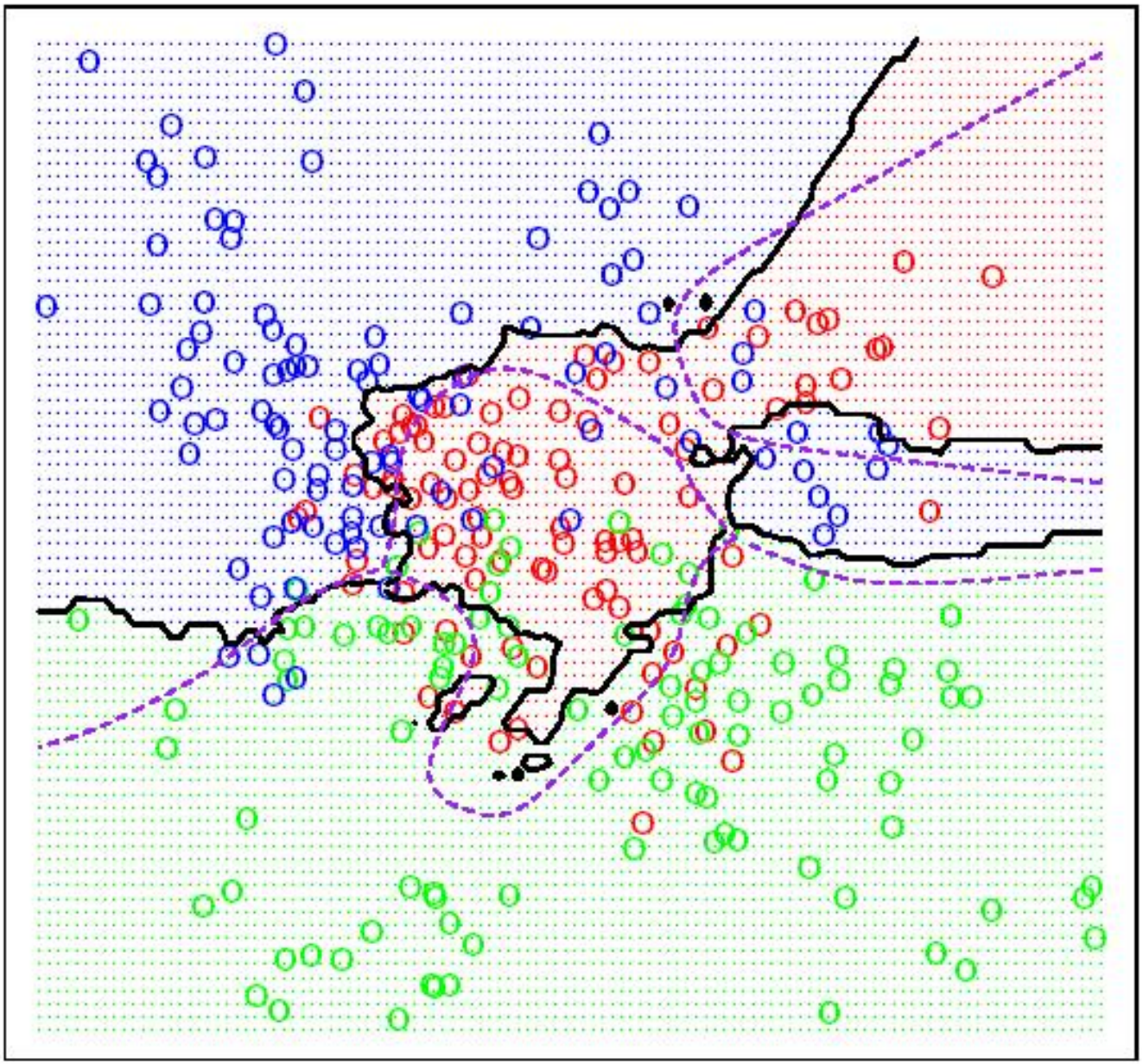
Text categorization using weight-adjusted nearest neighbor classification: Han, Karyois, Kumar (3)

Table 1: Summary of data sets used.

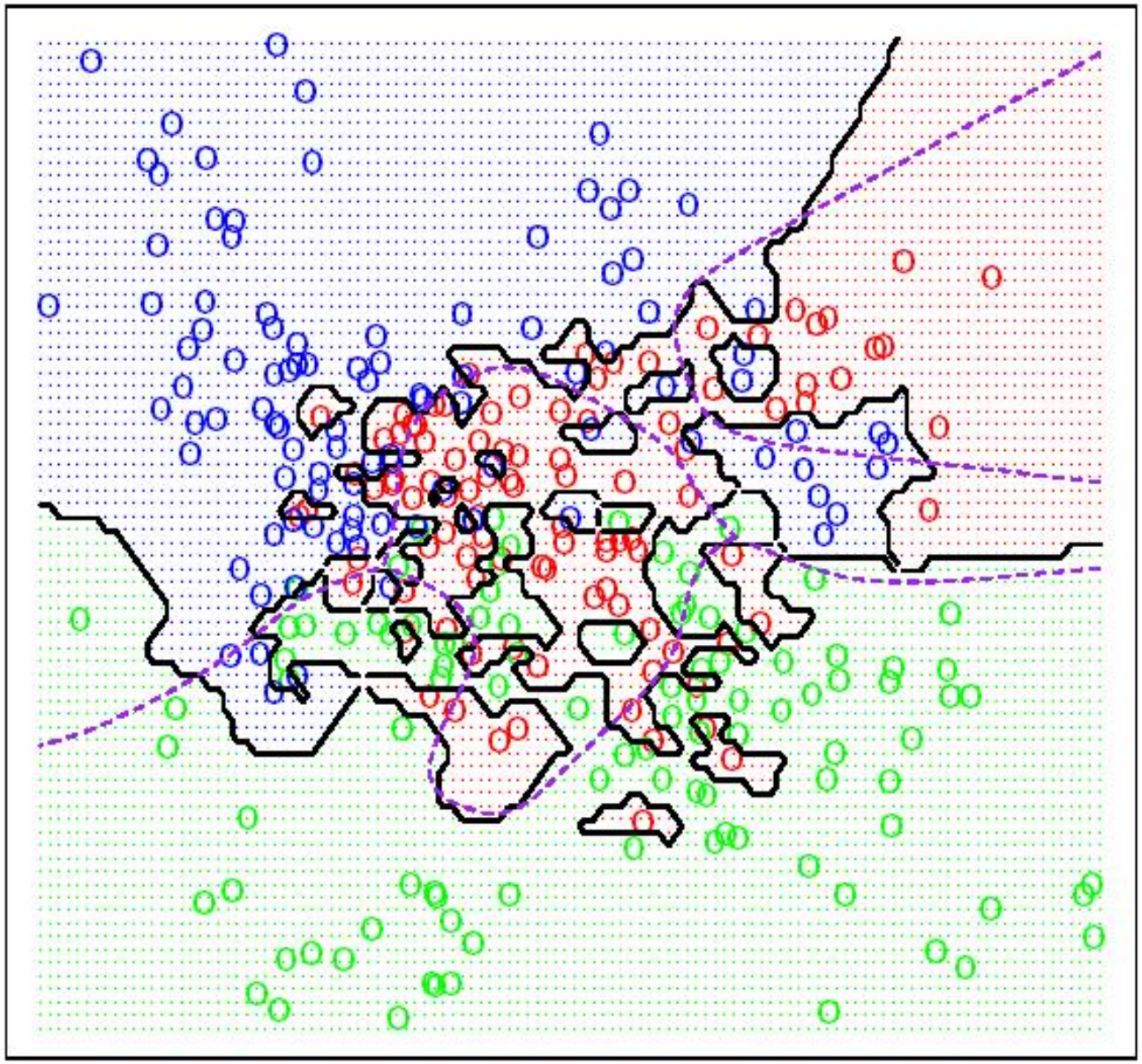
	C4.5	RIPPER	PEBLS	VSM	Rainbow	*k*-NN	WAKNN
west-1	85.50	84.47	78.50	85.20	84.40	76.73	**89.60**
west-2	71.30	68.33	67.80	77.44	72.11	68.33	**80.44**
west-3	79.60	75.92	72.70	86.53	80.00	70.61	**88.16**
west-4	81.80	77.14	78.60	87.86	**88.57**	73.93	85.00
west-5	84.60	89.71	86.80	89.71	85.21	84.57	**95.18**
west-6	83.70	83.38	79.80	87.19	85.29	73.57	**88.92**
west-7	80.10	80.14	71.80	83.52	81.26	74.94	**84.42**
fbis	57.10	73.94	69.80	76.14	76.38	78.49	**81.09**
trec-6	67.50	80.58	84.30	87.56	92.16	91.99	**92.67**
reuters	84.50	85.59	84.60	87.68	**91.04**	90.62	90.04

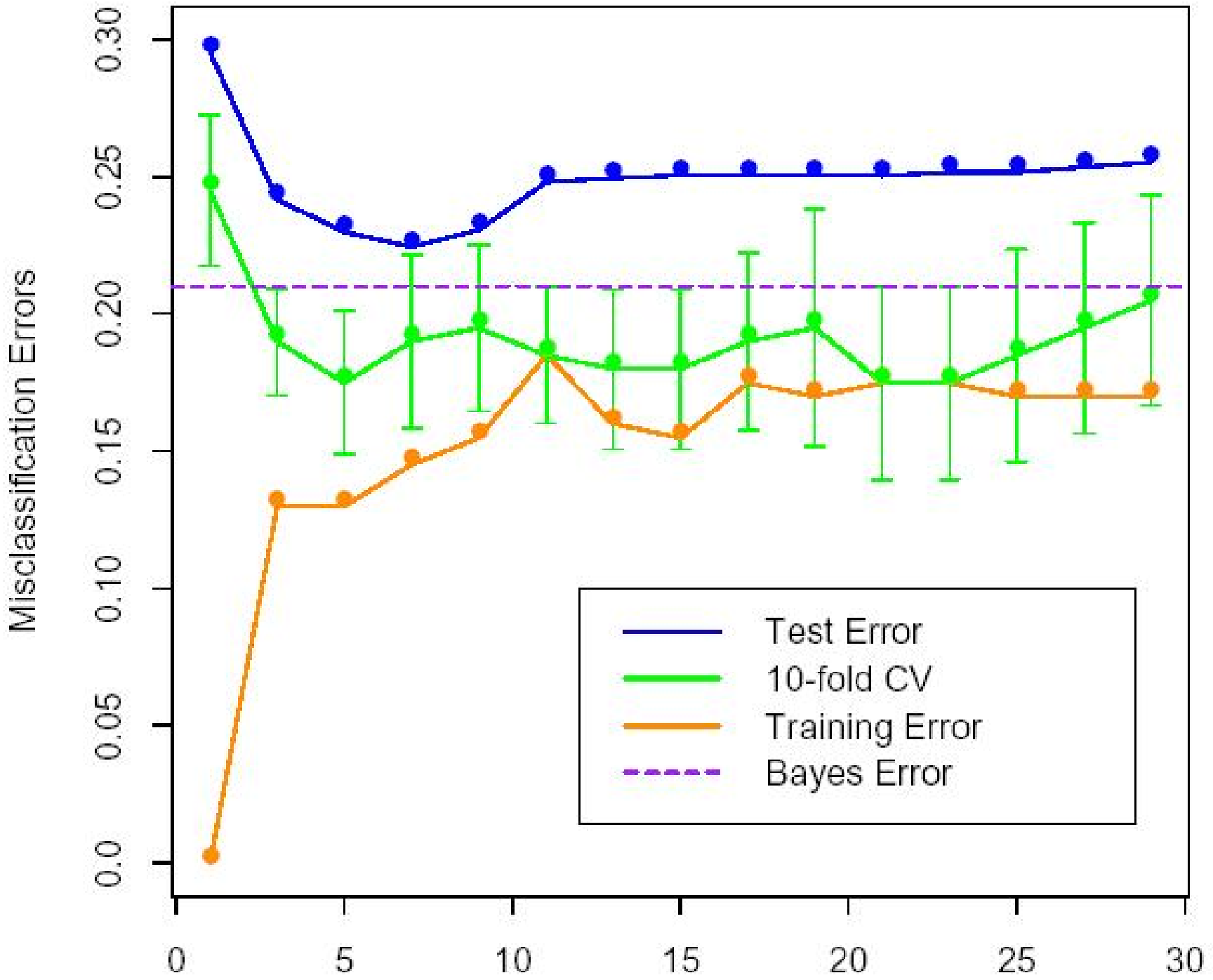
Optimierung von K im K-NN-Klassifikator

15-Nearest Neighbors

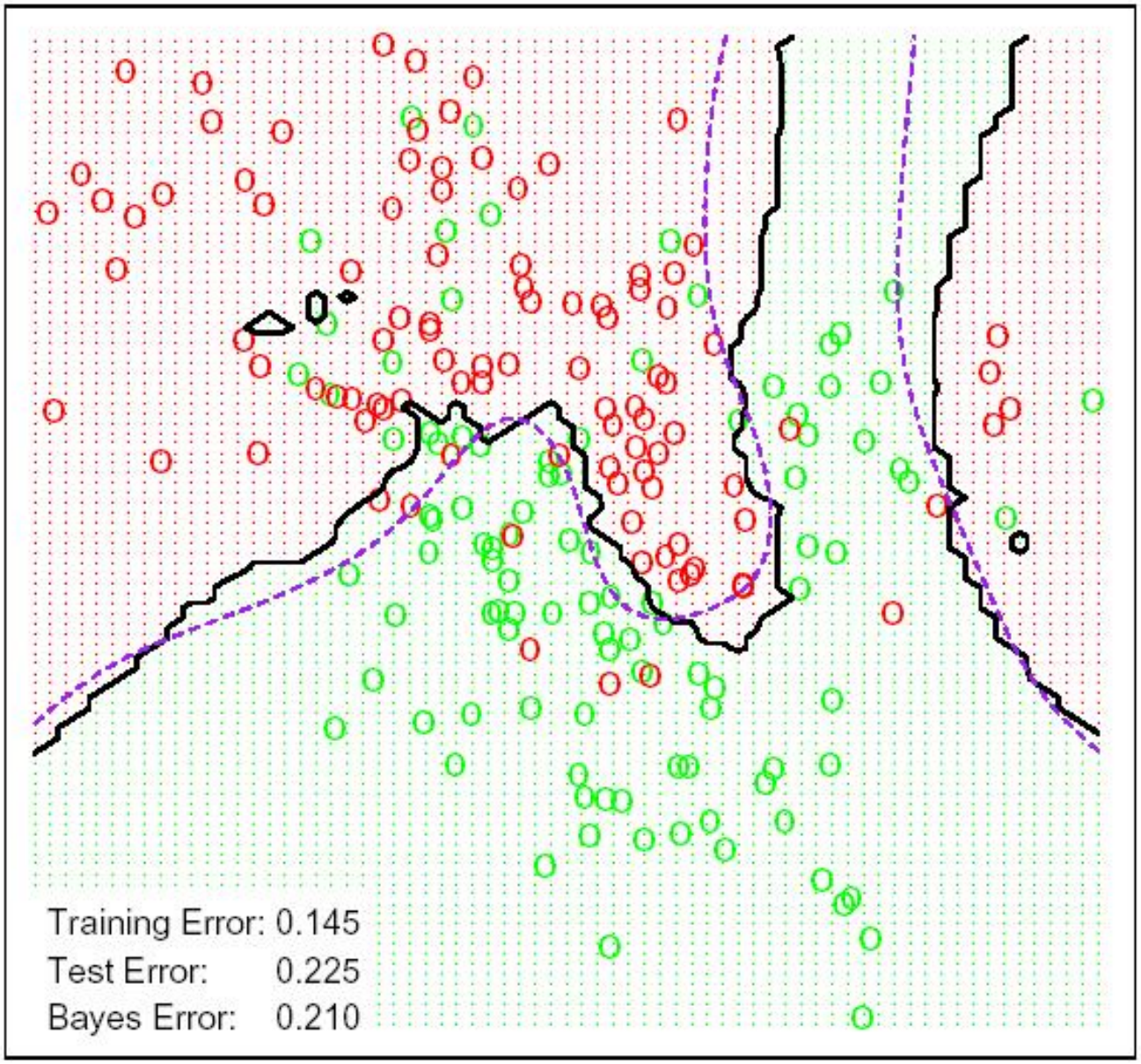


1-Nearest Neighbor




Misclassification Errors
0.0
0.05
0.10
0.15
0.20
0.25
0.30
0
5
10
15
20
25
30
Test Error
10-fold CV
Training Error
Bayes Error

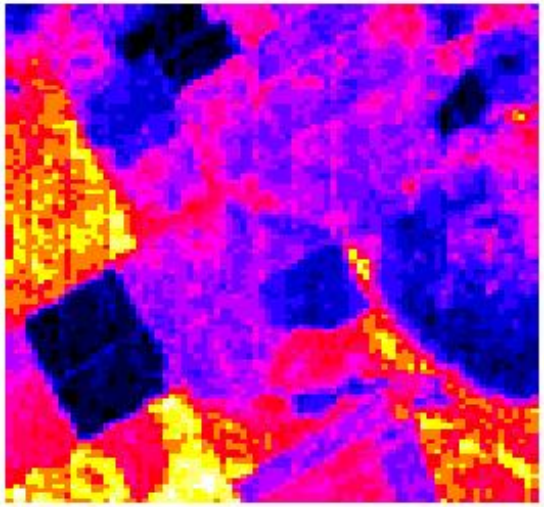
7-Nearest Neighbors



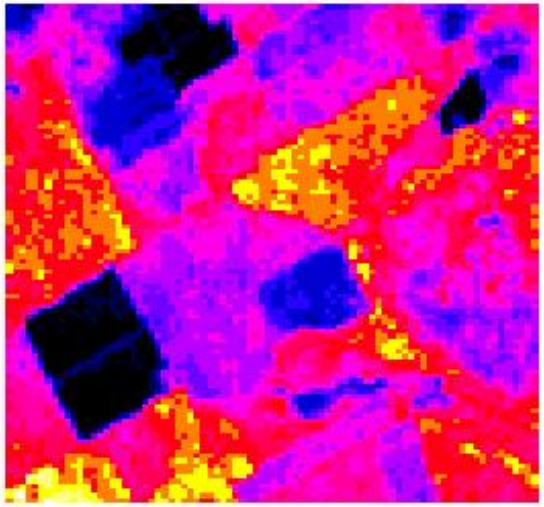
K-nächste Nachbarn für LANDSAT

- Daten: 4 Spektralbänder (Infrarot)
- Ein Pixel wird klassifizert in eine von 7 Klassen: Baumwolle, roter Boden, grauer Boden, ...
- Eingang: Spektrale Bänder des Pixels und der 8 Nachbar-Pixel: $4 \times (8 + 1) = 36$ Eingangsvariable
- 5-NN gaben beste Resultate im Vergleich zu anderen Methoden
- Entweder sind die Klassengrenzen komplex oder die Daten sind klar ge-clustert

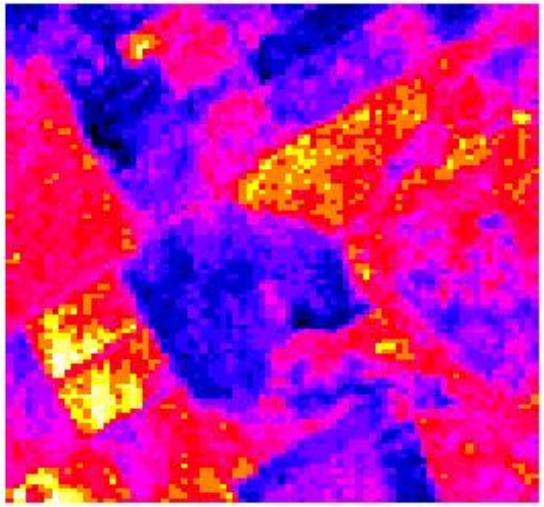
Spectral Band 1



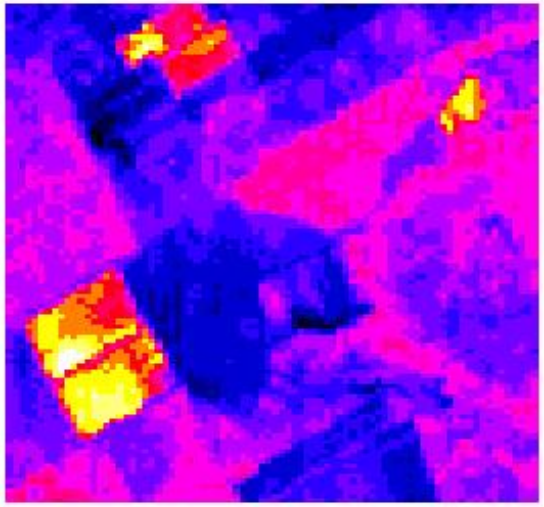
Spectral Band 2



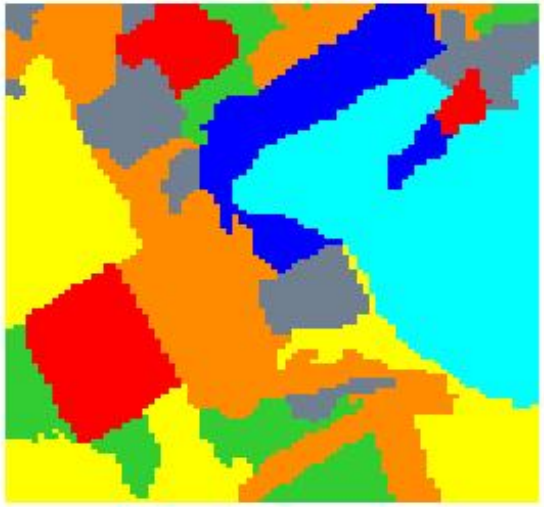
Spectral Band 3



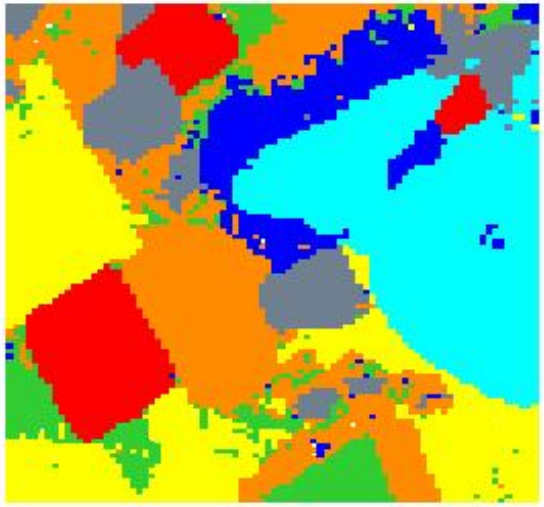
Spectral Band 4

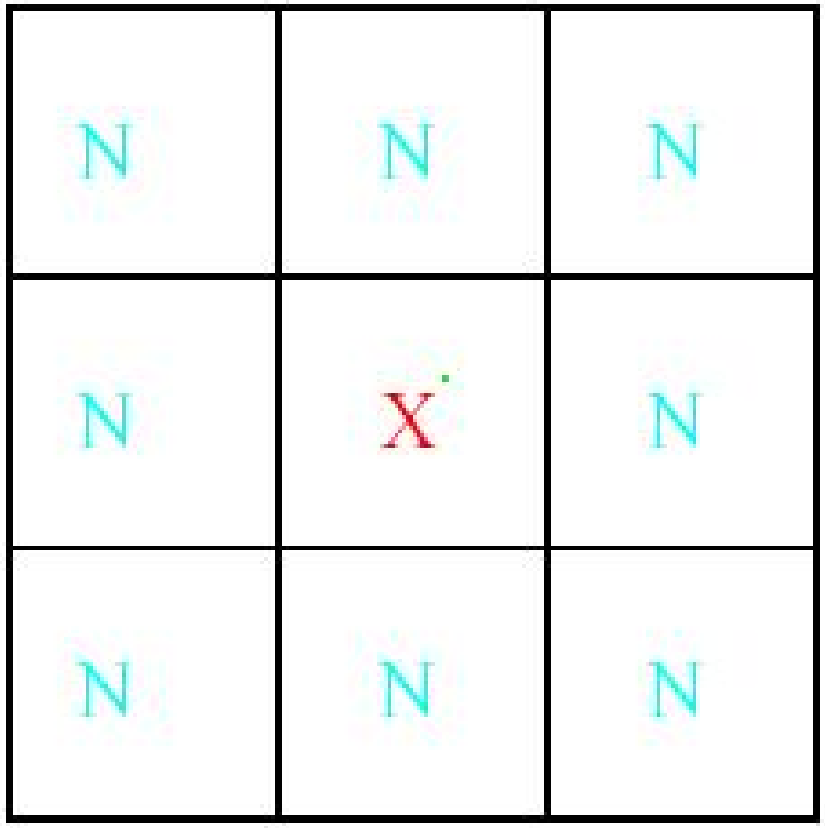


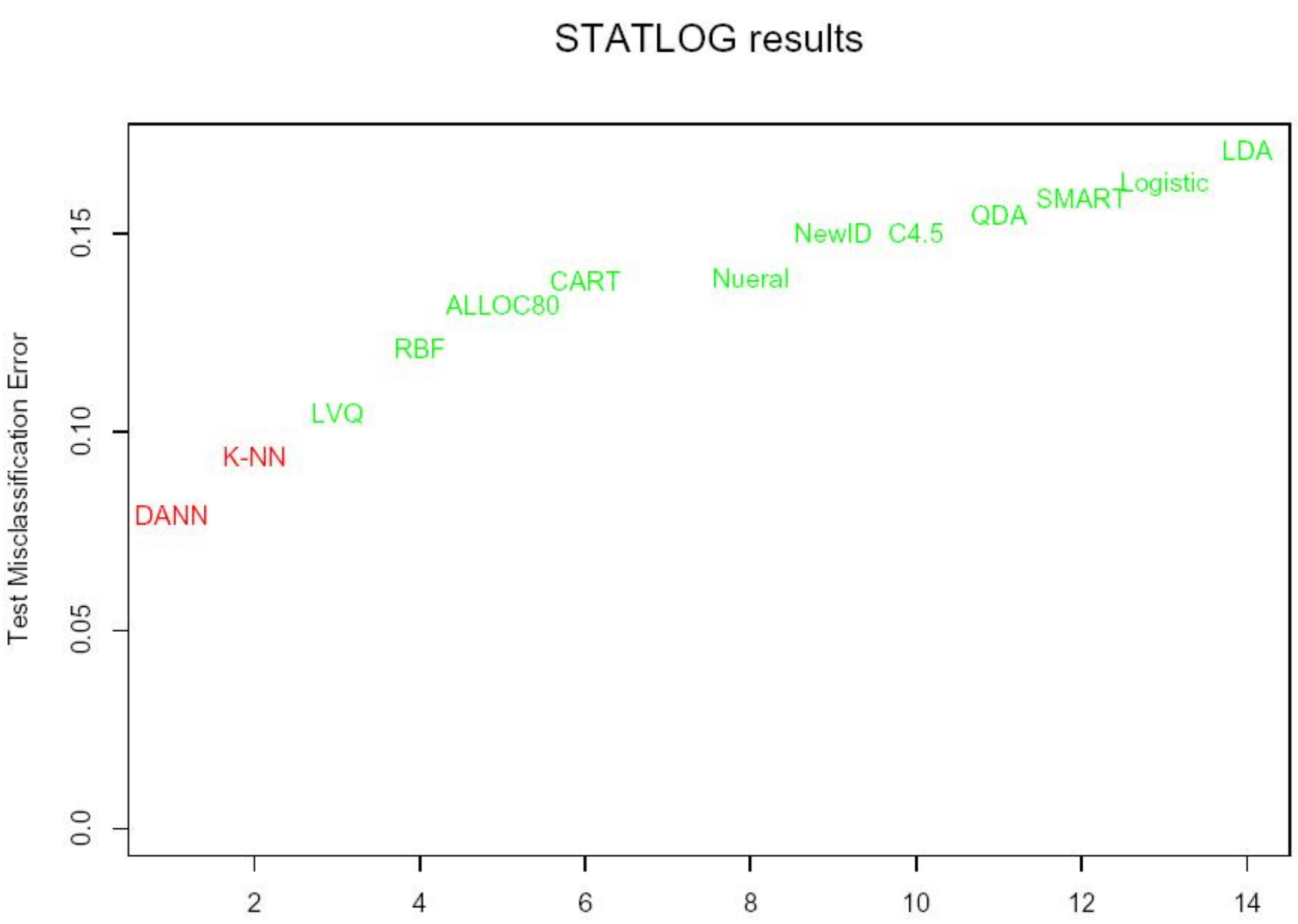
Land Usage



Predicted Land Usage




N
N
N
N
X
N
N
N
N


STATLOG results
Test Misclassification Error
0.0
0.05
0.10
0.15
2
4
6
8
10
12
14
DANN
K-NN
LVQ
RBF
ALLOC80
CART
Nueral
NewID
C4.5
QDA
SMART
Logistic
LDA

Großmutter-Zellen (Grandmother cells)

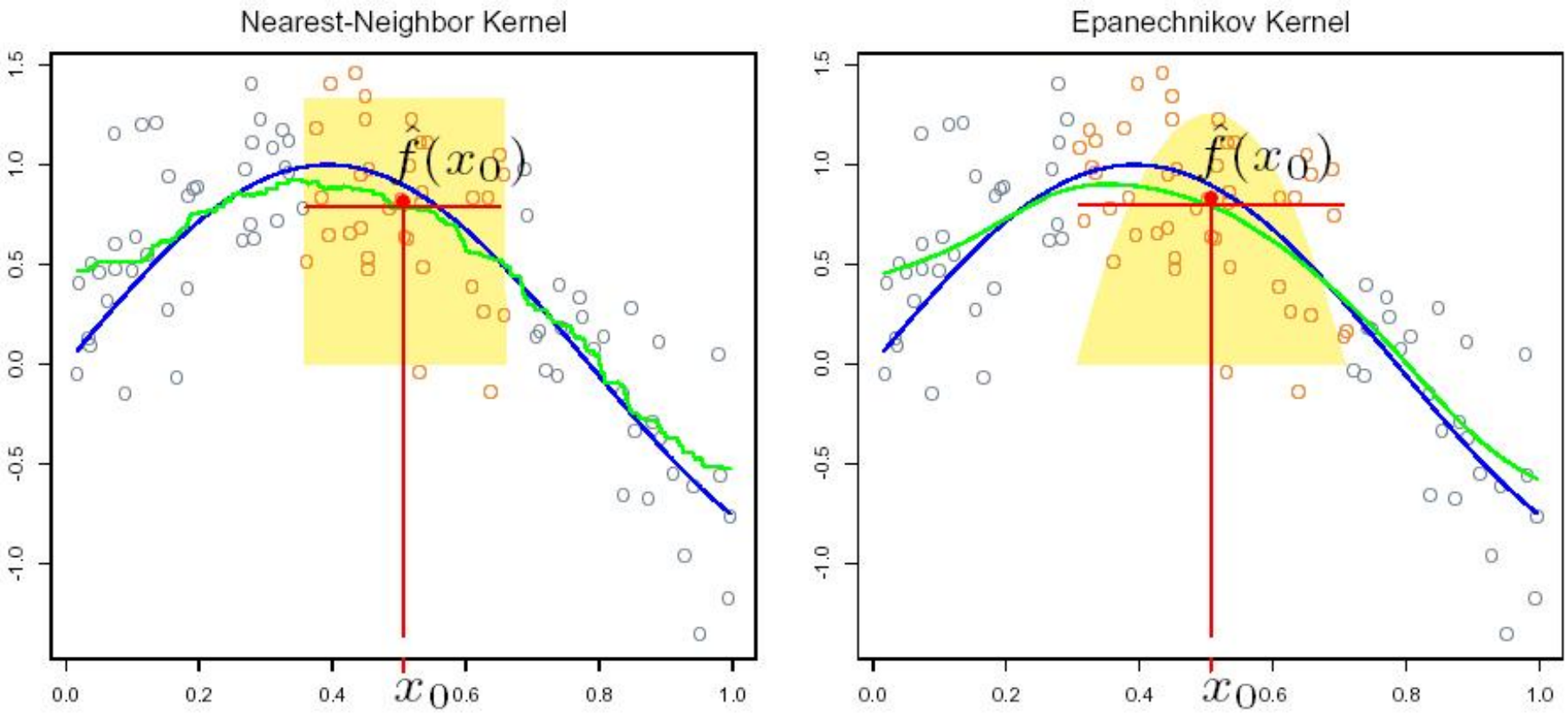
- Hypothese 1: das Gehirn speichert ein 3-D Modell eines Objektes und vergleicht in der Objekterkennung, ob ein Objekt diesem 3-D gespeicherten Objekt gleich sein kann. Dieses Verfahren ist speichereffizient, jedoch rechenaufwendig zum Zeitpunkt der Erkennung
- Hypothese 2: Das Gehirn speichert verschiedene 2-D Sichten eines 3-D Objektes und vergleicht in der Erkennung das unbekannte Objekt mit diesen 2-D Ansichten. Dieses Verfahren ist speicheraufwendig, jedoch biologisch recheneffizient (da parallelisierbar) zum Zeitpunkt der Erkennung. Für jedes abzuspeichernde Objekt gibt es für jede Sicht eine neuronale Einheit (Großmutterzelle)

Regression und Memory-basiertes Lernen

- Regression: $Y \in \Re$
- Das k-NN Verfahren liefert für die Regression in der Regel keine befriedigenden Ergebnisse
- Seien $J(\mathbf{z})$ die Indizes der k-nächsten Nachbarn zu $\mathbf{z}$
- $k-$ nächste Nachbarn Glätter:

$$\widehat{t}(\mathbf{z}) = \frac{1}{k} \sum_{i \in J(\mathbf{z})} y_i$$

- Unschöne Unstetigkeiten


Nearest-Neighbor Kernel
$\hat{f}(x_0)$
x_0
Epanechnikov Kernel
$\hat{f}(x_0)$
x_0

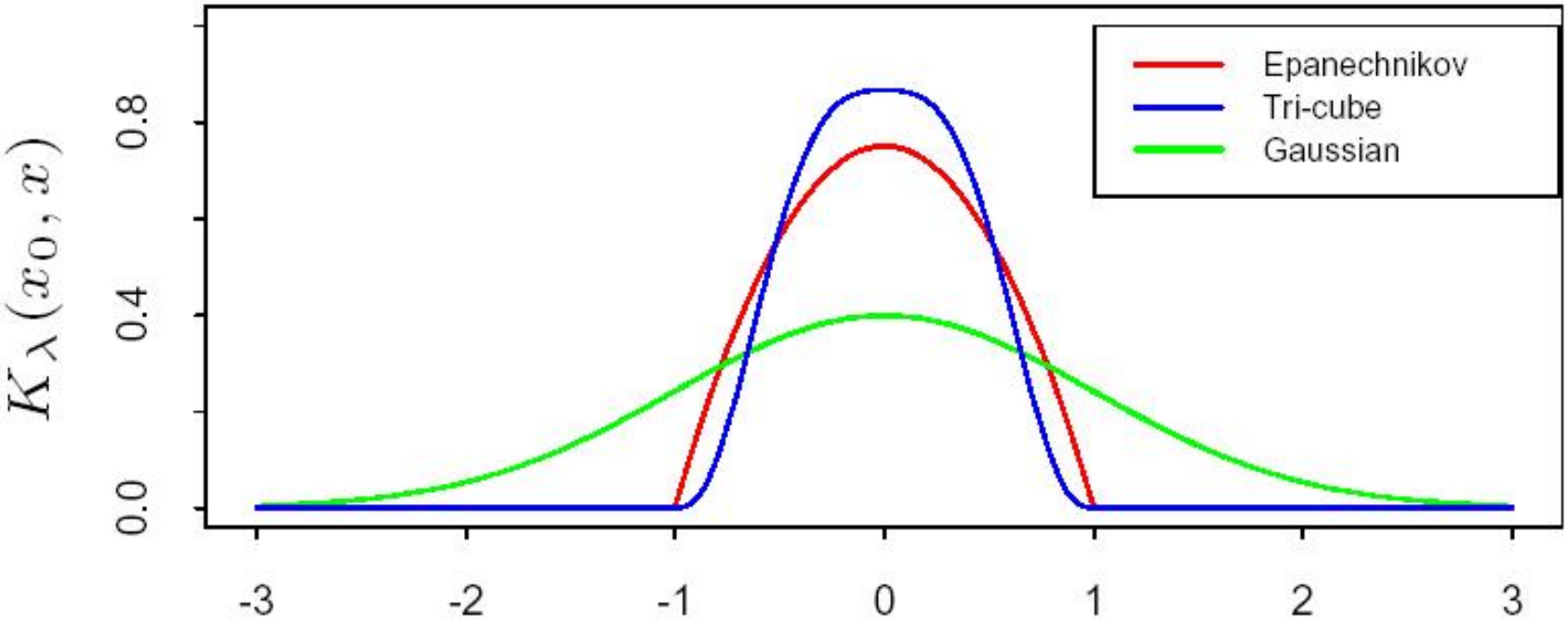
Kernel (Kern) Glätter:

- Nadaraya-Watson Glätter

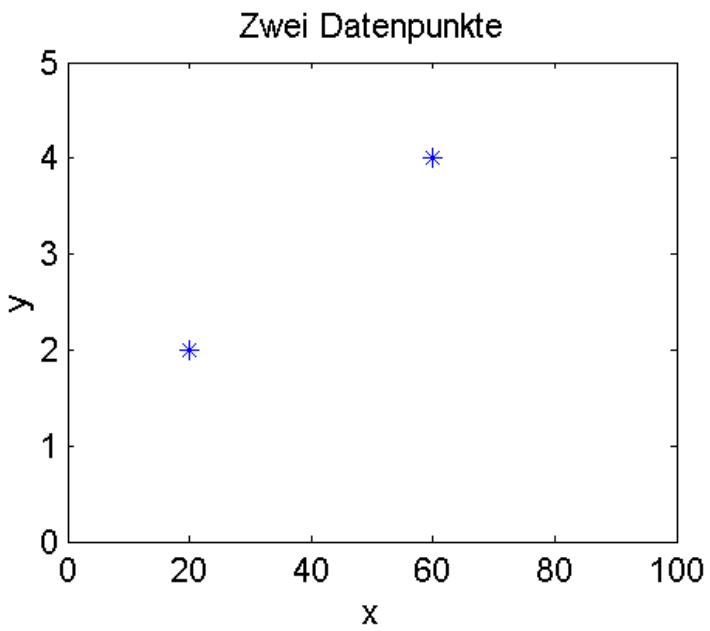
$$\hat{t} = \frac{1}{\sum_{i=1}^{N} K_\lambda(\mathbf{z}, \mathbf{x}_i)} \sum_{i=1}^{N} K_\lambda(\mathbf{z}, \mathbf{x}_i) y_i$$

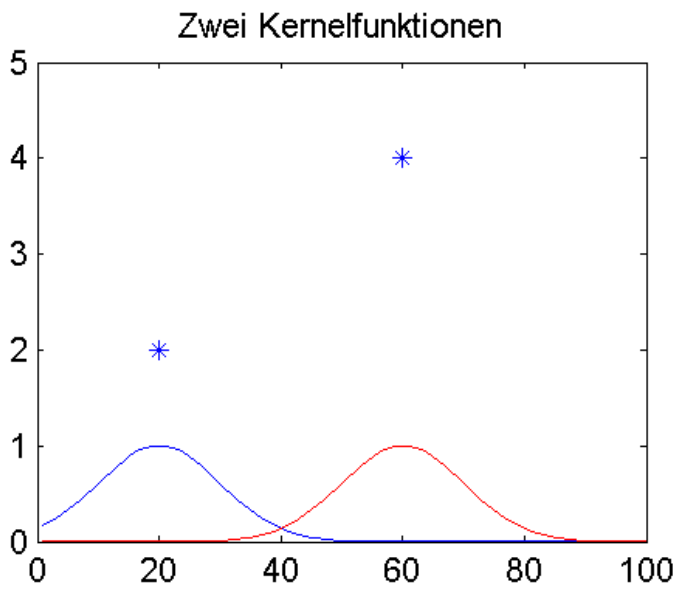
- Beispiel: Gauss-Kernel mit

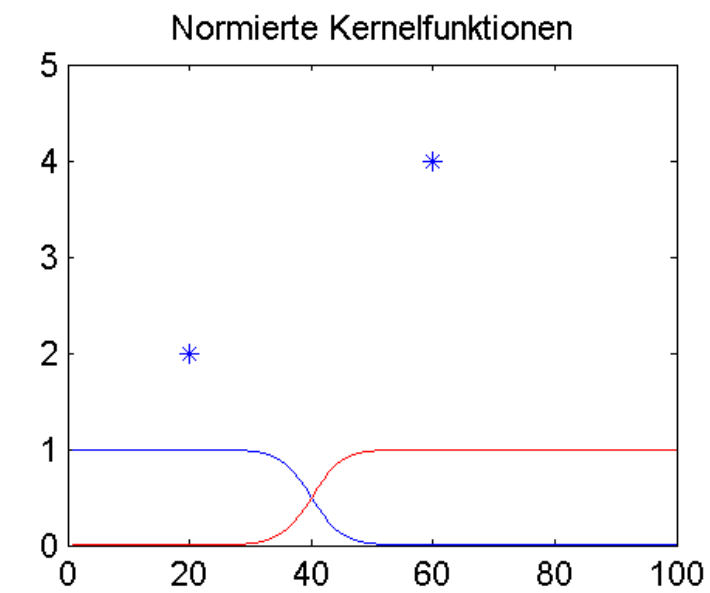
$$K_\lambda(\mathbf{z}, \mathbf{x}_i) = \exp\left(-\frac{1}{2\lambda^2}|\mathbf{z} - \mathbf{x}_i|^2\right)$$

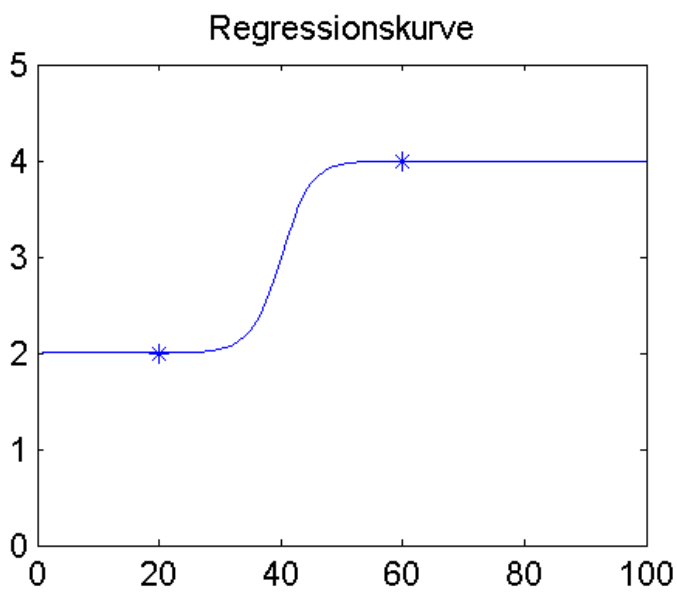

$K_\lambda(x_0, x)$
0.0
0.4
0.8
-3
-2
-1
0
1
2
3
Epanechnikov
Tri-cube
Gaussian

Kernel Glätter: Illustration


Zwei Datenpunkte
y
x
0
1
2
3
4
5
0
20
40
60
80
100


Zwei Kernelfunktionen
0
1
2
3
4
5
0
20
40
60
80
100


Normierte Kernelfunktionen
0
1
2
3
4
5
0
20
40
60
80
100


Regressionskurve
0
1
2
3
4
5
0
20
40
60
80
100

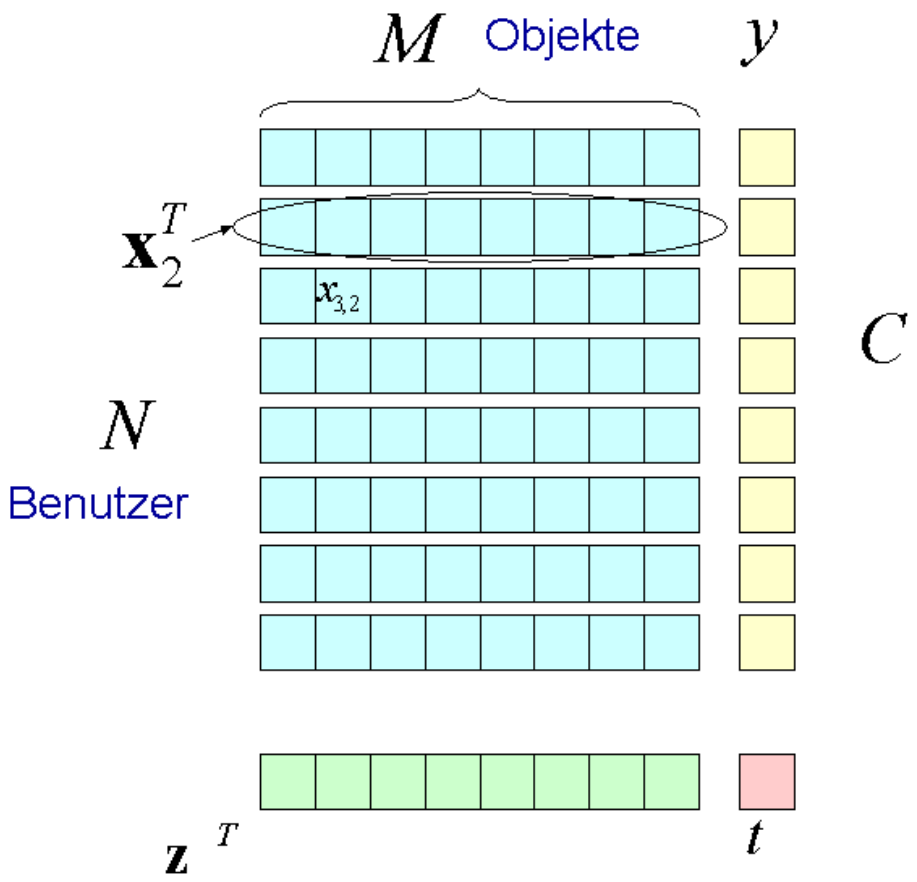
- (1) Zwei Datenpunkte $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2)$
- (2) $K_\lambda(\mathbf{z}, \mathbf{x}_1)$, $K_\lambda(\mathbf{z}, \mathbf{x}_2)$
- (3)

$$\frac{K_\lambda(\mathbf{z}, \mathbf{x}_1)}{K_\lambda(\mathbf{z}, \mathbf{x}_1) + K_\lambda(\mathbf{z}, \mathbf{x}_2)}, \quad \frac{K_\lambda(\mathbf{z}, \mathbf{x}_2)}{K_\lambda(\mathbf{z}, \mathbf{x}_1) + K_\lambda(\mathbf{z}, \mathbf{x}_2)}$$

- (4)

$$\hat{t} = \frac{1}{K_\lambda(\mathbf{z}, \mathbf{x}_1) + K_\lambda(\mathbf{z}, \mathbf{x}_2)} \left(K_\lambda(\mathbf{z}, \mathbf{x}_1) y_1 + K_\lambda(\mathbf{z}, \mathbf{x}_2) y_2\right)$$

Vektorraummodell für Collaboratives Filtern (CF)



- CF: Empfehlungssysteme, die ohne die Merkmale der Objekte auskommen, sondern nur die Bewertung anderer Benutzer verwenden
- $x_{i,j}$ ist die Bewertung des i-ten Benutzers für das j-te Objekt; die meisten Bewertungen fehlen!
- Die Zielgröße ist die (unbekannte) Bewertung eines (beliebigen) Benutzers für ein (beliebiges) Objekt
- Im Beispiel sind $\mathbf{z}$ die Bewertungen des aktives Benutzers

Empirical Analysis of Predictve Algorithms for Collaborative Filtering, Heckerman et al.

	Dataset		
	MSWEB	Neilsen	Eachmovie
Total users	3453	1463	4119
Total titles	294	203	1623
Mean votes per user	3.95	9.55	46.4
Median votes per user	3	8	26

Table 1: Number of users, titles, and votes for the datasets used in testing the algorithms. Only users with 2 or more votes are considered.

- MS Web: Web-site besucht oder nicht besucht
- TeleVision: Show angeschaut oder nicht
- EachMovie: Filmbewertung: $1, \ldots, 5$ Punkte

CF+: Memory-basiertes System für CF

- Bewertungsvorhersage (t ist die Bewertung des aktiven Benutzers für das Objekt von Interesse; y_i ist die Bewertung des i-ten Benutzers für das gleiche Objekt) :

$$\hat{t} = \overline{\mathbf{z}} + \frac{1}{\sum_{i:i \text{ rated } Y} \|w(\mathbf{x}_i, \mathbf{z})\|} \sum_{i:i \text{ rated } Y} w(\mathbf{x}_i, \mathbf{z})(y_i - \overline{\mathbf{x}}_i)$$

  wobei $\overline{\mathbf{z}}$ der Mittelwert der Bewertungen des aktiven Benutzers und $\overline{\mathbf{x}}_i$ der Mittelwert der Bewertungen des i-ten Benutzers sind. Das Gewicht $w(\mathbf{x}_i, \mathbf{z})$ ist die Pearson-Korrelation welche über Objekte berechnet wird, für die Benutzer i und der aktive Benutzer eine Beurteilung abgegeben haben

Empfehlungssystem

- Man empfiehlt die K best-bewerteten Objekte in der Reihenfolge der Bewertungen dem aktiven Benutzer
- BN:Bayes net, CR+: vorgestellter Algorithmus , VSIM: ähnlich mit Cosinus-Maß, BC: Clustering, POP: immer das populärste Objekt vorschlagen
- RD (required difference): Mindestabstand für signifikanten Unterschied
- Given 5: wieviele Objekte der aktive Benutzer bewertet hat

	MS Web, Rank Scoring			
Algorithm	Given2	Given5	Given10	AllBut1
BN	59.95	**59.84**	53.92	**66.69**
CR+	**60.64**	57.89	51.47	**63.59**
VSIM	**59.22**	56.13	49.33	**61.70**
BC	**57.03**	**54.83**	**47.83**	**59.42**
POP	49.14	46.91	41.14	49.77
RD	*0.91*	*1.82*	*4.49*	*0.93*

Table 2: Ranked scoring results for the MS Web dataset. Higher scores indicate better performance.

Ergebnis: Nielsen

	Neilsen, Rank Scoring			
Algorithm	Given2	Given5	Given10	AllBut1
BN	**34.90**	42.24	**47.39**	**44.92**
CR+	39.44	**43.23**	**43.47**	**39.49**
VSIM	**39.20**	**40.89**	**39.12**	**36.23**
BC	19.55	18.85	**22.51**	**16.48**
POP	20.17	19.53	19.04	13.91
RD	*1.53*	*1.78*	*2.42*	*2.40*

Table 3: Ranked scoring results for the Neilsen dataset. Higher scores indicate better performance.

Ergebnis: EachMovie

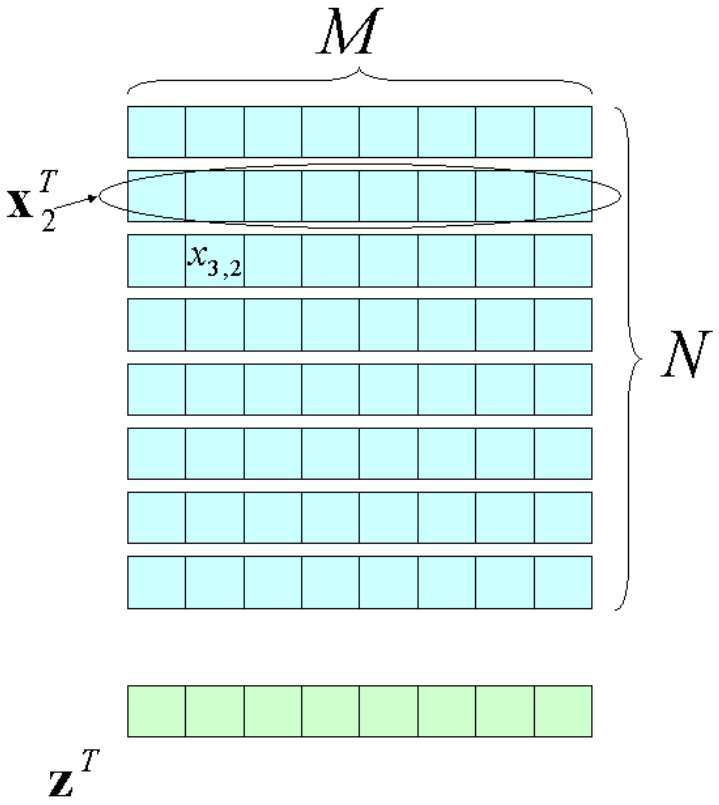
	EachMovie, Rank Scoring			
Algorithm	Given2	Given5	Given10	AllBut1
CR+	**41.60**	42.33	**41.46**	**23.16**
VSIM	**42.45**	**42.12**	**40.15**	22.07
BC	**38.06**	**36.68**	**34.98**	**21.38**
BN	**28.64**	**30.50**	**33.16**	**23.49**
POP	30.80	28.90	28.01	13.94
RD	*0.75*	*0.75*	*0.78*	*0.78*

Table 4: Ranked scoring results for the EachMovie dataset. Higher scores indicate better performance.

Memory-basiert versus Modell-basiert

- In einem **modellbasierten Ansatz** wird anhand der Trainingsdaten eine Modell erstellt
- Zur Interpretation der Domäne und zur Vorhersage wird dann dieses Modell herangezogen
- Der größte Anteil der Berechnungen geschieht zum Zeitpunkt der Modellbildung
- In einem **Memory-basierten Ansatz** werden die eigentlichen Berechnungen zum Zeitpunkt der Vorhersage durchgeführt
- Diese Berechnungen werden u.U. durch Erstellung geeigneter Datenstrukturen vorbereitet (k-d Trees)

Die Datenmatrix für unüberwachtes Lernen



X_j	j-te Variable
X	$= (X_1, \ldots, X_M)^T$ Vektor von Variablen
M	Anzahl der Variablen
N	Anzahl der Datenpunkte
$\mathbf{x}_i$	$= (x_{i,1}, \ldots, x_{i,M})^T$ i-ter Datenvektor
$x_{i,j}$	j-te Komponente von $\mathbf{x}_i$
D	$= \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ (Trainings-) Datensatz
$\mathbf{z}$	Testvektor

Kernel-Dichte Schätzung

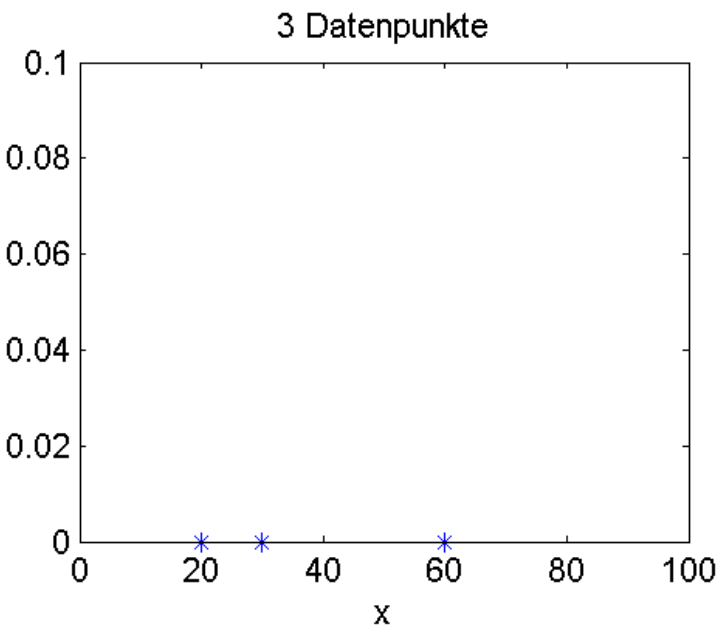
- In vielen Anwendungen hätte man gerne eine Abschätzung von $P(\mathbf{z})$, der Wahrscheinlichkeitsdichte
- z.B.: Zur Modellierung von $P(\mathbf{z}|Klasse)$
- Ebenso: Anomaly Detection

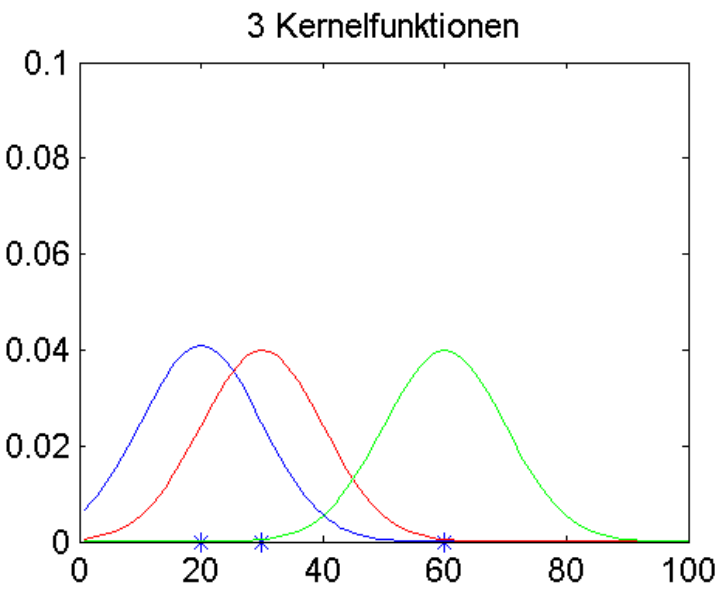
Kernel-Dichte Schätzung

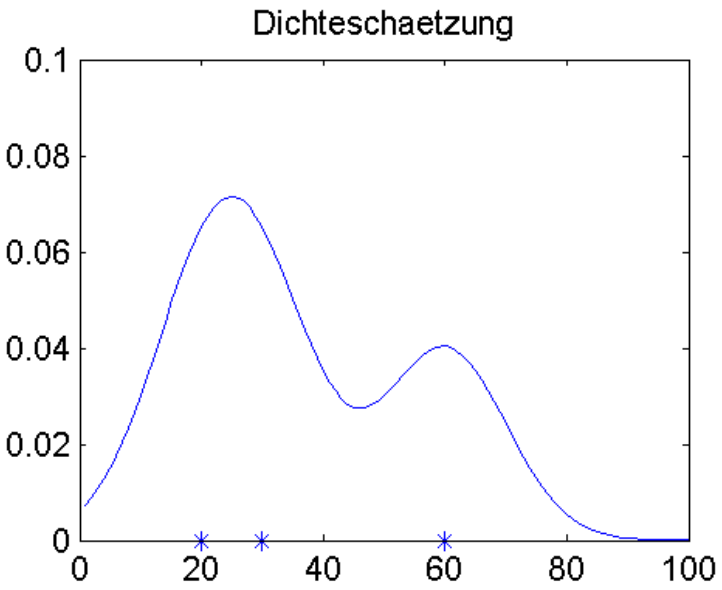
- Parzen Schätzer

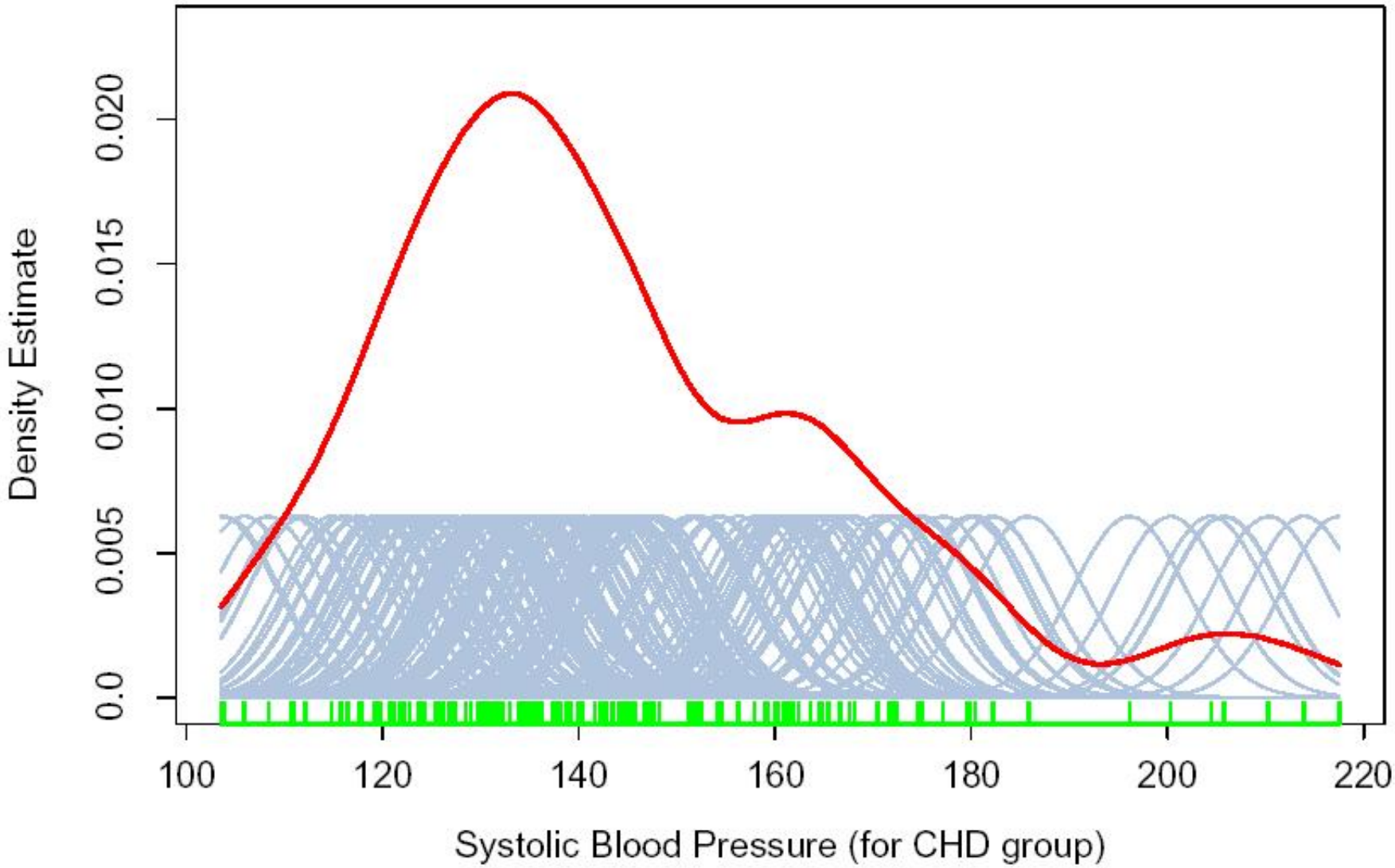
$$\hat{P}(\mathbf{z}) = \frac{1}{c \times N} \sum_{i=1}^{N} K_\lambda(\mathbf{z}, \mathbf{x}_i)$$

wobei $c = \int K_\lambda(\mathbf{z}, \mathbf{x}_i) d\mathbf{z}$.

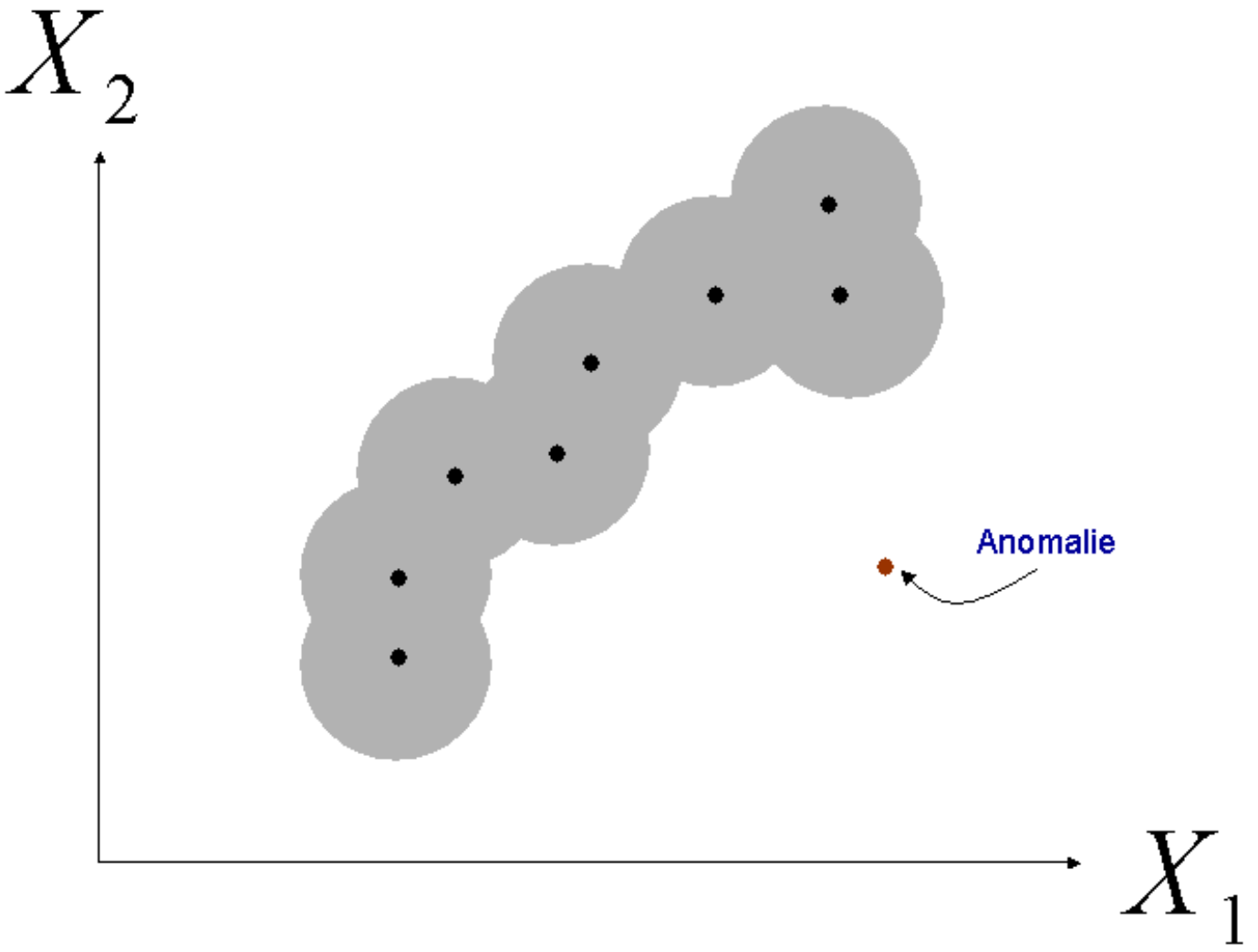

3 Datenpunkte
0.1
0.08
0.06
0.04
0.02
0
0
20
40
60
80
100
x


3 Kernelfunktionen
0.1
0.08
0.06
0.04
0.02
0
0
20
40
60
80
100


Dichteschaetzung
0.1
0.08
0.06
0.04
0.02
0
0
20
40
60
80
100

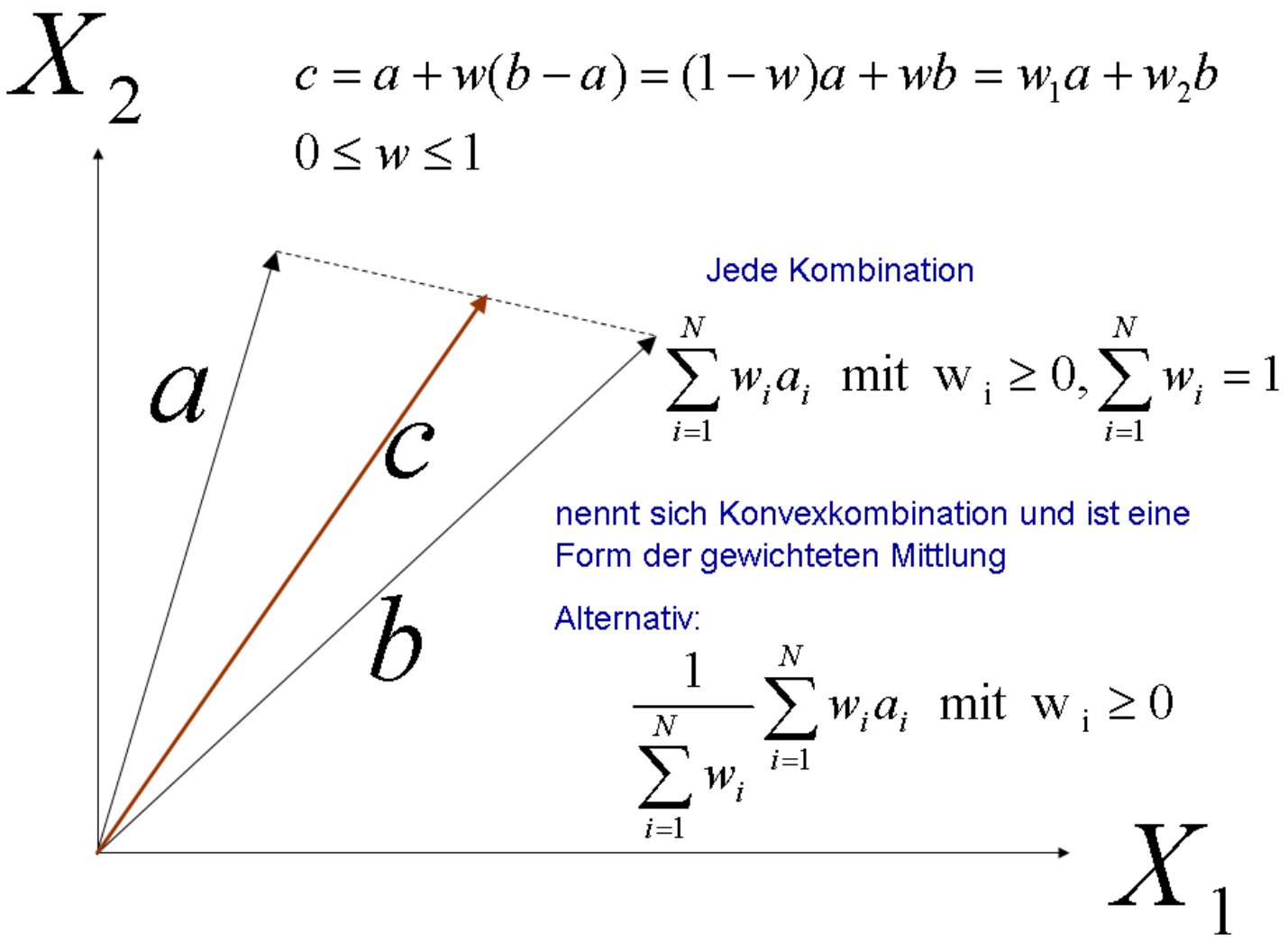

Density Estimate
0.0
0.005
0.010
0.015
0.020
100
120
140
160
180
200
220
Systolic Blood Pressure (for CHD group)

Kernel Glätter: Anomalie Detektion



APPENDIX:

APPENDIX I: Konvexe Kombination



$$c = a + w(b-a) = (1-w)a + wb = w_1 a + w_2 b$$

$$0 \le w \le 1$$

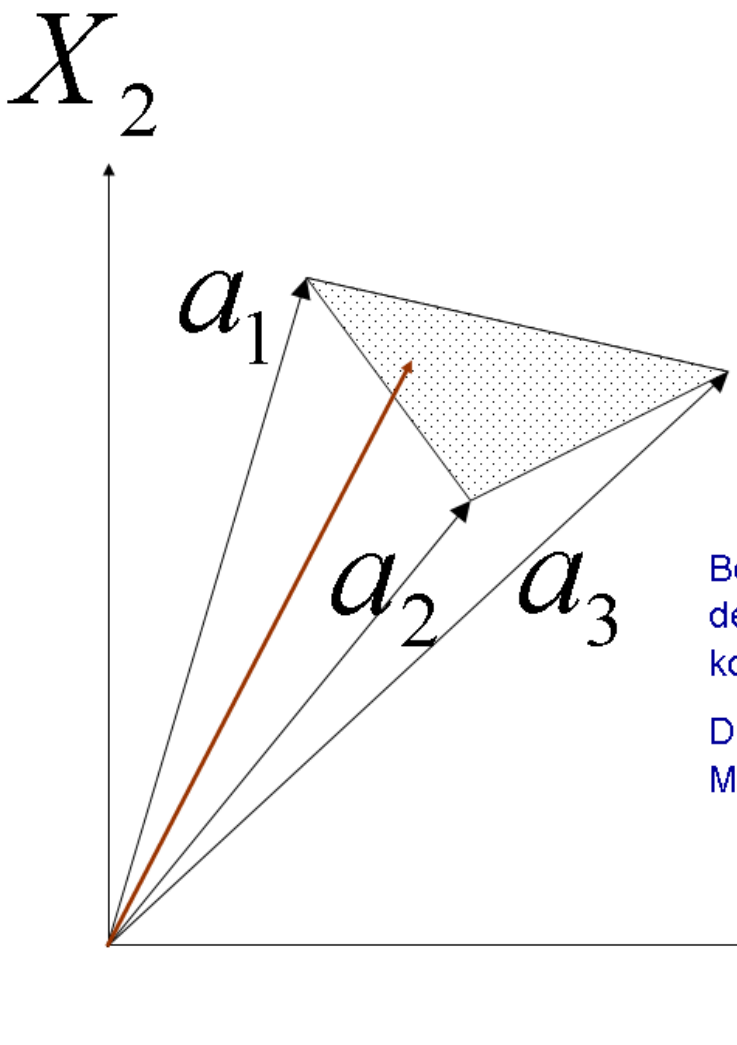
Jede Kombination

$$\sum_{i=1}^{N} w_i a_i \quad \text{mit} \quad w_i \ge 0, \sum_{i=1}^{N} w_i = 1$$

nennt sich Konvexkombination und ist eine Form der gewichteten Mittlung

Alternativ:

$$\frac{1}{\sum_{i=1}^{N} w_i} \sum_{i=1}^{N} w_i a_i \quad \text{mit} \quad w_i \ge 0$$



$$\sum_{i=1}^{N} w_i a_i \quad \text{mit} \quad w_i \geq 0, \sum_{i=1}^{N} w_i = 1$$

Bei mehr als zwei Basisvektoren befindet sich der resultierende Vektor innerhalb der konvexen Hülle der Basisvektoren

Die konvexe Hülle ist die kleinste konvexe Menge, die die Basispunkte umschließt

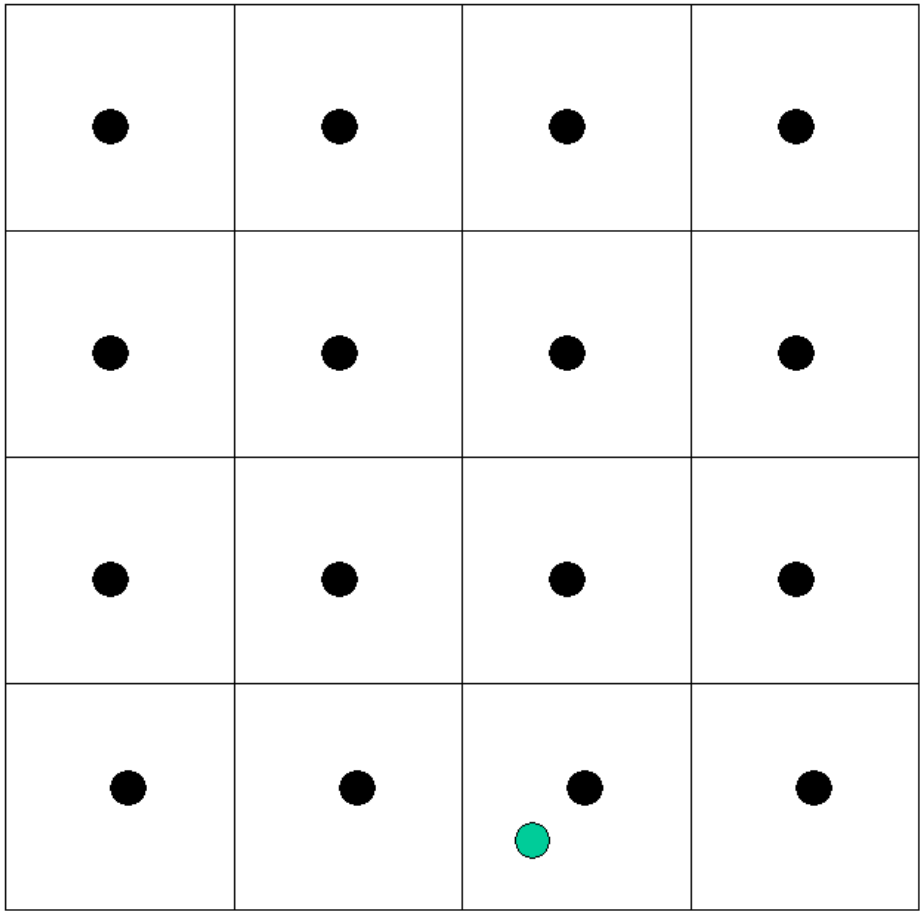
APPENDIX II: Alternatives memory-basiertes CF+

- Betrachten wir noch einmal den CF+ Ansatz; hier muss zum Zeitpunkt der Bewertung eine Summation über aller Benutzer durchgeführt werden
- Dies ist ungünstig, wenn es sehr viel mehr Benutzer als Objekte gibt
- Man kann ein alternativen CF+ Ansatz definieren, der im Prinzip darin besteht, dass man in der Datenmatrix Zeilen und Spalten vertauscht
- Dann entsprechen den gewichten die Pearson-Korrelation zwischen Objekten, und die Summation geht über alle Objekte, die der aktive Benutzer bewertet hat
- Der grosse Vorteil ist, dass zum Zeitpunkt der Auswertung nur eine Summe über die Objekte ausgewertet werden muss

APPENDIX III: Rechnerische Komplexität: NN-Berechnung

- Mit N Trainingsdaten benötigt man $\mathcal{O}(N)$ Operationen für nächste Nachbar-Klassifikatoren
- Vorverarbeitung der Daten in Form eines Entscheidungsbaums (k-d tree) : der Aufwand reduziert sich unter idealen Bedingungen zu $\mathcal{O}(\log_2 N)$;
- Reduktion ist dramatisch für große N: $N = 10^9$, $\log_2(N) \approx 30$

Einfaches Beispiel

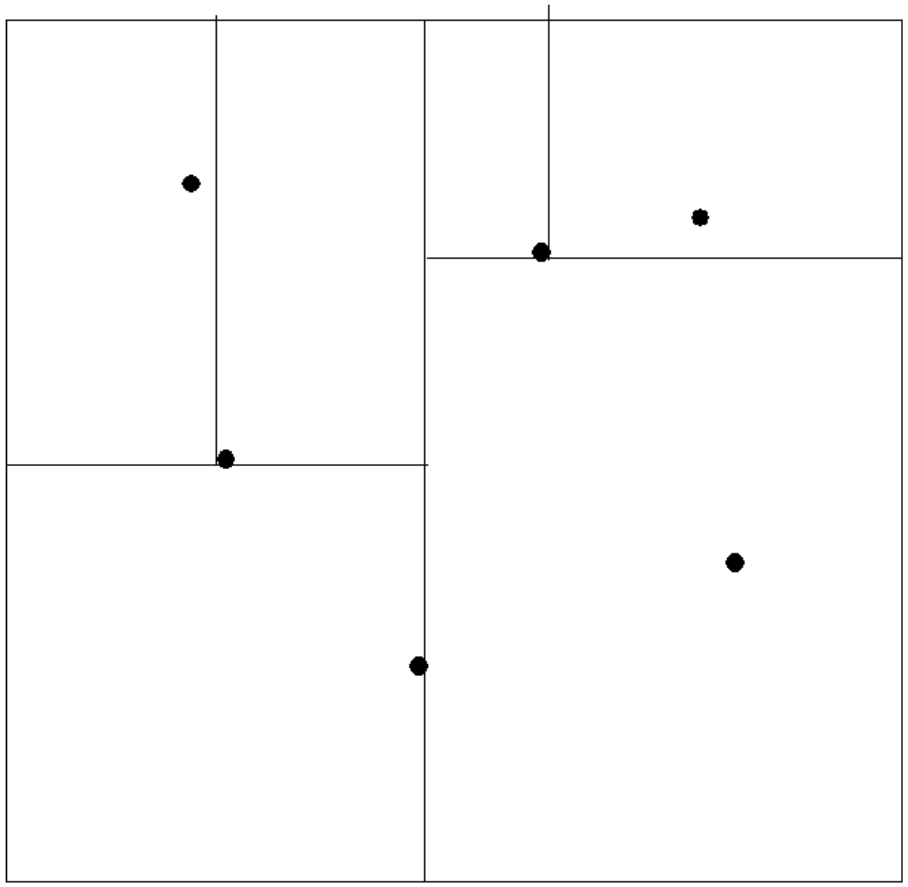


- Im Beispiel gibt es 16 Datenpunkte, die regelmäßig angeordnet sind
- Um den nächsten Nachbarn zum grünen neuen Datenpunkt zu finden, muss ich $\log_2 16 = 4$ Abstände berechnen

Aufbau eines k-dimensionalen Baumes (k-d tree)

- k ist die Eingangsdimensionalität (bei uns: M)
- Gehe der Reihe nach wiederholt durch alle Dimensionen, bis sich in jedem Blatt nur genau ein Datenpunkt befindet:
- Ein Split teilt die Daten des Knotens in zwei gleichgroße Hälften, je nachdem ob der Wert des Datenpunktes in der betreffenden Dimension größer oder kleiner dem Median-Wert ist
- Wegen des Verzweigungsfaktors 2 enthält ein Baum der Tiefe d dann 2^d Blätter. D.h. die Tiefe des Baumes ist bestimmt durch $2^d \geq N$, oder $d \approx \log_2 N$

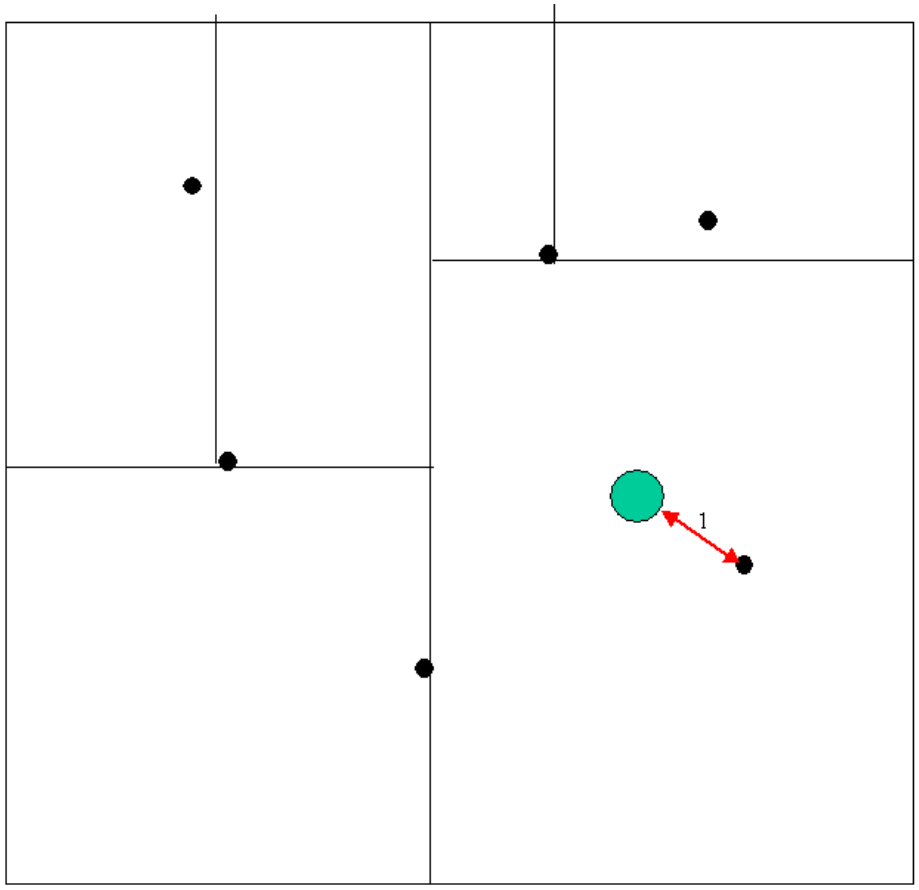
Aufbau eines k-d Baumes



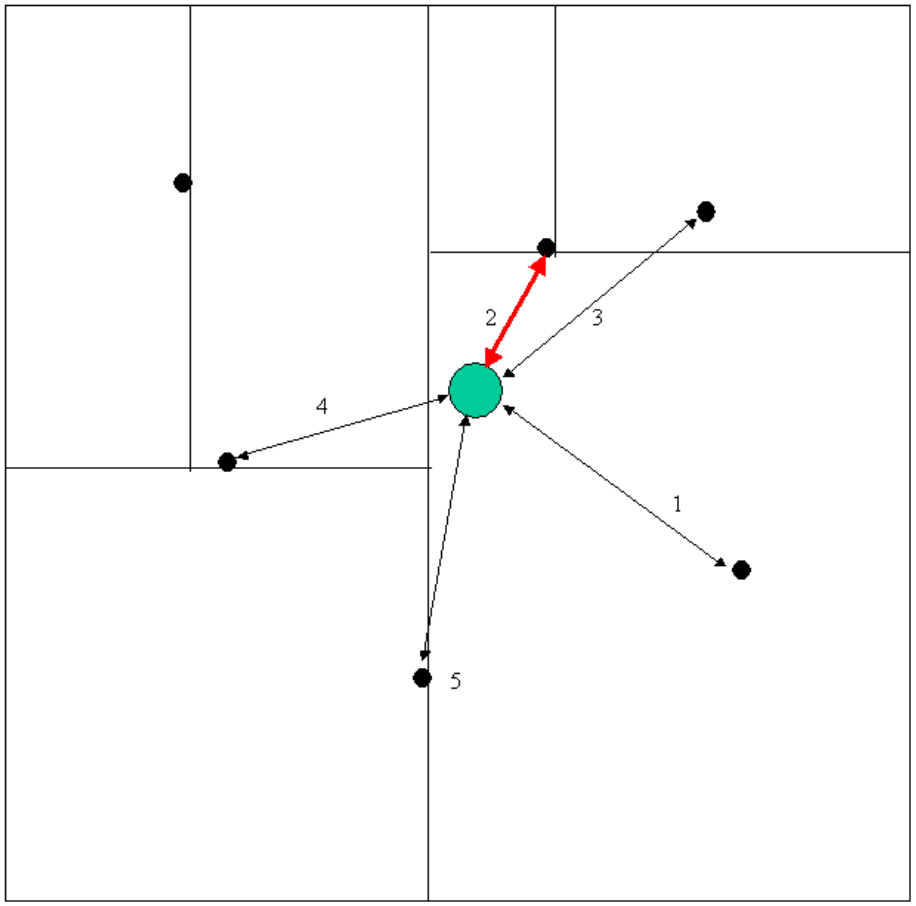
Bestimmung des nächsten Nachbarn mit Hilfe des k-d Baums

- Bestimme den Split des Datenpunktes $\mathbf{x}$ entsprechend der Dimension und des Schwellwertes eines Knotens und Teile die Datenmengen ein in wahrscheinliche und unwahrscheinliche Mengen, je nachdem in welchen Zweig $\mathbf{x}$ gehört
- Finde den nächsten Nachbarn in der wahrscheinlichen Menge (mit dieser Prozedur)
- Falls der Abstand zum nächsten Nachbarn in der wahrscheinlichen Menge kleiner ist als der Abstand zum Schwellwert, dann berichte (nach oben) den nächsten Nachbarn in der wahrscheinlichen Menge
- Anderenfalls
  - berechne den nächsten Nachbarn ebenfalls in der unwahrscheinlichen Menge (mit dieser Prozedur)
  - Vergleiche den nächsten Nachbarn der wahrscheinlichen Menge und der unwahrscheinlichen Menge und berichte den näheren der beiden Datenpunkte

Effizientes Finden des NN



Ineffizientes Finden des NN



Kommentare zum k-d Baum

- Es müssen mindestens $\mathcal{O}(\log_2 N)$ Operationen (Vergleiche) durchgeführt werden
- Im günstigsten Fall muss nur der Abstand zu einem Prototypen berechnet werden
- In niedrigen Dimensionen ist die Einsparung dramatisch
- Die Dimensionalität des Problems ist eine kritische Größe: in hohen Eingangsdimensionen müssen häufig die Abstände zu allen Trainingsdatenpunkten berechnet werden! Dies liegt daran, dass die Tendenz besteht, dass $\mathbf{x}$ zu allen Trainingsdatenpunkten annähernd den gleichen Abstand besitzt
- Weiterentwicklungen: R-Baum (Guttman et al.), R^*-Baum (Kriegel et al.)

APPENDIX VI: Fall-basiertes Schließen

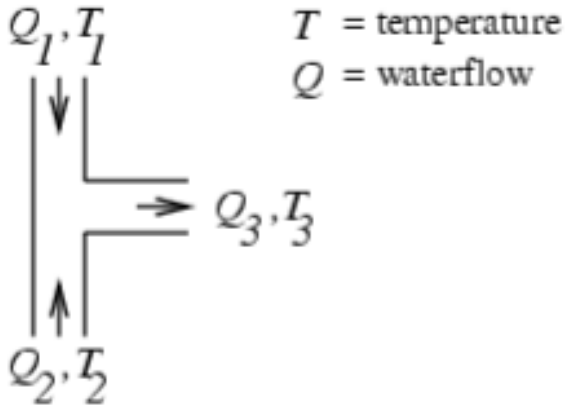
- KI-Version von Memory-basiertem Lernen
- Symbolisches Schließen basierend auf Nachbarschaftsbeziehungen: KI-Version von Memory-basiertem Lernen
- Beispiel: technische oder medizinische Diagnose; es wurde eine Datenbank mit bekannten Symptomen und Diagnosen aufgebaut. Die Diagnose zu einem neuen Problem geschieht auf Basis von ähnlichen Problemfällen aus der Datenbank
- Man kann entsprechendes Vorwissen zur Bewertung der Ähnlichkeit verwenden (Diagnose): Zur Inferenz kann ein ganzes Regelwerk aufgebaut werden
- Beispiele werden mit einer reicheren symbolischen Representation beschrieben
- Beispiel: über einen neueren juristischen Fall eine Schlussfolgerung abgeben, basierend auf ähnlichen vergangenen Fällen

CADET: Entwurf mechanischer Gegenstände

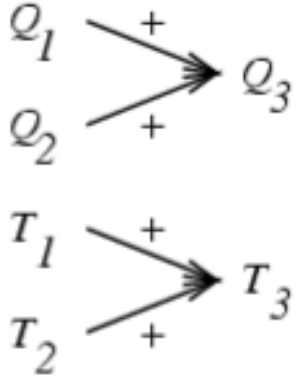
- Entwurf einfacher mechanischer Gegenstände (Wasserhahn)
- Datenbank von 75 vorhandenen Gegenständen präsentiert durch Struktur und qualitativer Funktion
- Ziel: Realisierung aufgrund einer neuen Spezifikation
- CADET vergleicht zunächst, ob eine Realisierung zu einer Problemstellung in der Datenbank existiert
- Falls nicht, versucht CADET Subgraphisomorphismen zu finden und diese zu passenden Lösungen zu integrieren unter Zuhilfenahme einfacher physikalischer Regeln

A stored case: T–junction pipe

Structure:



Function:



A problem specification: Water faucet

Structure:

?

Function: