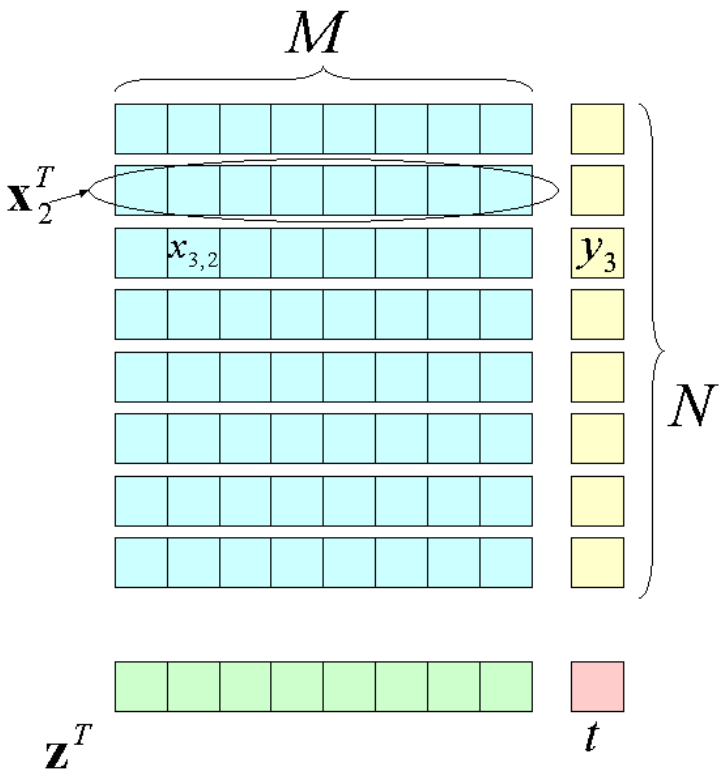
Instance-based Learning

Volker Tresp

The Data Matrix for Supervised Learning



X_j	j-th input variable (columns)
X	$= (X_1, \ldots, X_M)^T$ vector of input variables
M	number of of input variables
N	number of of training data points
Y	output variable
$\mathbf{x}_i$	$= (x_{i,1}, \ldots, x_{i,M})^T$ i-th input pattern
$x_{i,j}$	j-th component of $\mathbf{x}_i$
y_i	target to $\mathbf{x}_i$
$\widehat{y}_i$	prediction to $\mathbf{x}_i$
$\mathbf{d}_i$	$= (x_{i,1}, \ldots, x_{i,M}, y_i)^T$ i-th trainings pattern
D	$= \{\mathbf{d}_1, \ldots, \mathbf{d}_N\}$ (training-) data set
$\mathbf{z}$	test input
t	(unknown) target to $\mathbf{z}$

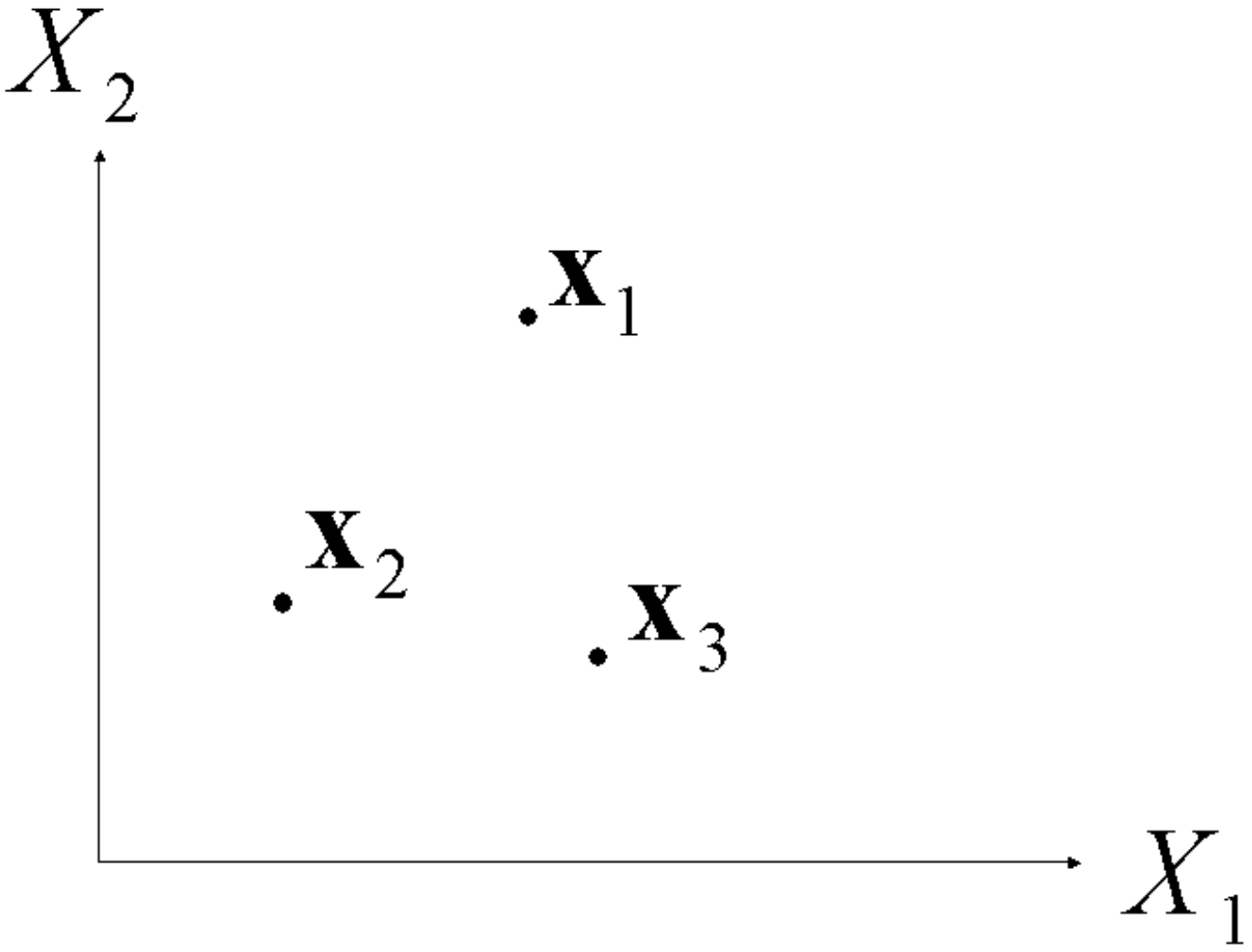
Definition

- In instance based approaches the training data are available at the time of prediction and the computational load is at the time of prediction
- Thus "training time" is close to zero but the computational load at prediction time can be significant. This is the reason why instance-based approaches are sometimes referred to as "lazy learning"
- To achieve fast prediction, it can be important to support instance-based approaches using appropriate index structures

Overview

- Instance-based Approaches for Classification
- Instance-based Approaches for Regression
- Instance-based Approaches for Density Estimation (One-Class Classification)

Data Points in Input Space



Variables

- The variables (columns) can describe properties of objects, measurements, state variables, time series measurements, ...
- The domain of a variable can be
  - continuous, $x_{i,j} \in \Re$
  - binary discrete, $x_{i,j} \in \{0, 1\}$, oder $x_{i,j} \in \{-1, 1\}$
  - discrete, $x_{i,j} \in \{1, 2, \ldots, C\}$

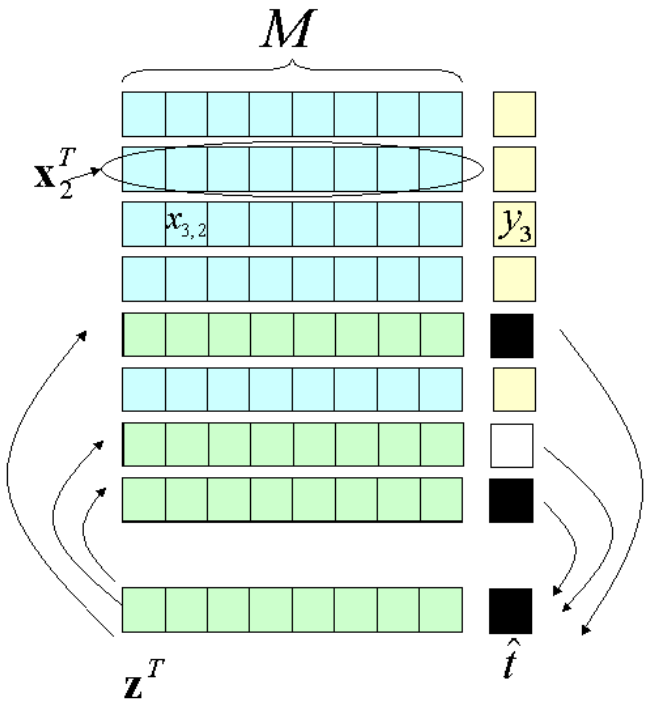
Table Representation

- Let's assume C classes, such that $y_i \in \{1, \ldots, C\}$
- Now we assume that the input variables can only assume K states $p_1, \ldots, p_K$. Let $N_{k,l}$ be the number of times that in the training data $\mathbf{x}_i = p_k$ when, and output variable was $y_i = l$
- Then when $\mathbf{z} = p_k$ we classify

$$\hat{t} = \arg\max_l N_{k,l}$$

- Recall that table representations are often used on discrete Bayes nets

Table Representation (con'd)



- When $K >> N$ this approach is not applicable. For example with M binary inputs we get $K = 2^M$
- Similarly, when the inputs are continuous this approach cannot be applied in general

Similarity and Distance

- If we consider a training pattern with a very similar input then it makes sense that the target also stays constant of also is very similar
- On many ways this is the basic assumption in machine learning
- But the central question: what does it mean to be similar? Are some inputs more important then others? Are there preferred directions? A central problem in machine learning is to find the application specific right similarity measure
- Example: inputs are height, age, hair color. The output is weight. The weight will have some correlation with height, a certain dependency on age but might be less dependent on hair color. Similarity might change quickly with height but slower with age and hair color
- If the target is income, then the similarity might be most sensitive towards age!

Nearest Neighbor Classification

- The ideas is to define an appropriate distance measure and assign the test pattern to the closest trainings pattern
- Nearest-Neighbor Classification. Let

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

then

$$\hat{t} = y_l$$

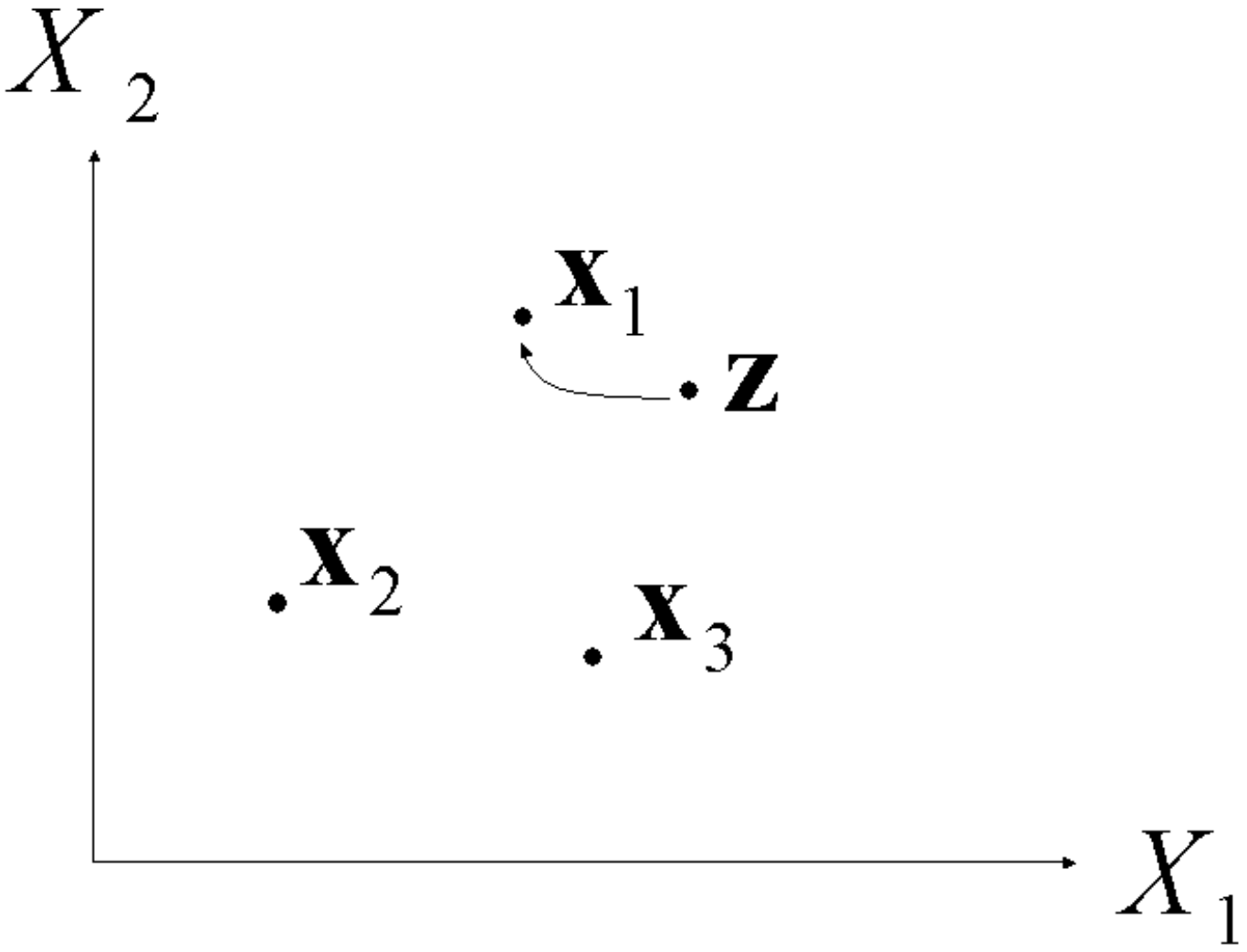
- k-Nearest-Neighbor Classification: let J be the set of indices indicating the k nearest neighbors to $\mathbf{z}$. Then,

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

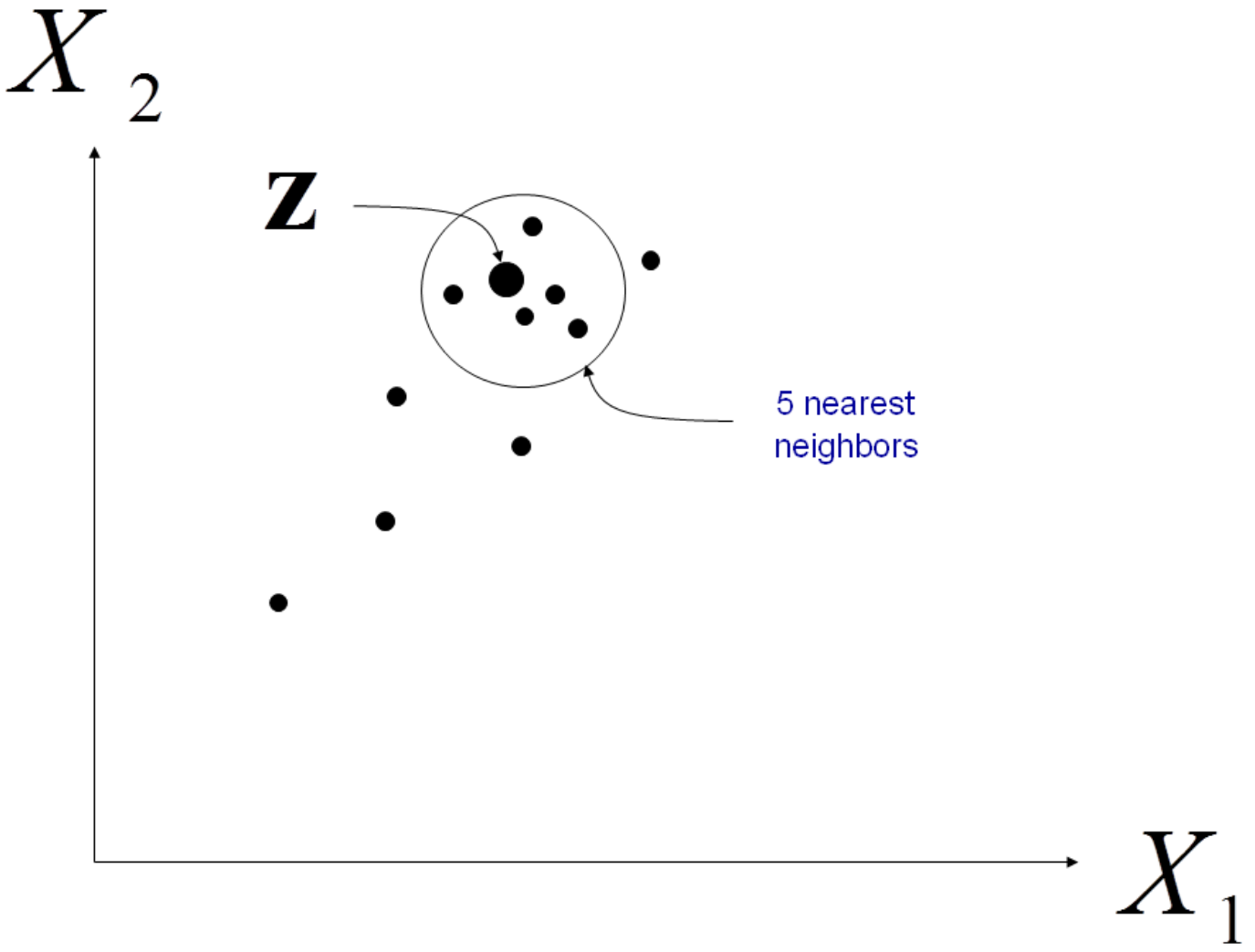
Then one classifies $\mathbf{z}$ as

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

Nearest Neighbor Classification



k=5-Nearest Neighbor Classification



Distance Function for Continuous Variables

- Euclidean distance

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

- Manhatten distance

$$d_{Manhatten}(\mathbf{x}_i, \mathbf{z}) = \|\mathbf{x}_i - \mathbf{z}\|_1 = \sum_{j=1}^{M} |x_{i,j} - z_j)|$$

Distance Metrics

- $d(x, z)$ is a metric (or distance metric), if (non-negativity, or separation axiom)

$$d(x, z) \geq 0$$

if (identity of indiscernibles, or coincidence axiom)

$$d(x, z) = 0 \quad \text{if and only if} \quad x = z$$

if (symmetry)

$$d(x, z) = d(z, x)$$

and if (subadditivity / triangle inequality)

$$d(x, z) \leq d(x, y) + d(y, z)$$

- Conditions 1 and 2 together define a positive-definite function. The first condition is implied by the others.
- The distance functions we are using are nor all proper metric; in particular the triangle inequality might not be obeyed

Distance Function for Discrete Variables

- For discrete variables one often uses the number of differences (*simple matching coefficient*):

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

  This distance function can be used both for nominal discreet variables with not natural order in its states (e.g., colors) and for ordinal discrete variables, where such an order exists (e.g., school grades)

- For binary variables with $z_j \in \{0, 1\}$, and $x_{i,j} \in \{0, 1\}$, this is

$$d_{simple}(\mathbf{x}_i, \mathbf{z}) = \frac{1}{M} \|\mathbf{x}_i - \mathbf{z}\|^2$$

Distance Functions for Sparse Data

- In some applications one state dominates, typically the state 0.
- Case 1: if there are 1000 items. Customer 1 and Customer 2 each bought one item but these two items are different. Then the Euclidian distance divided my M is $\sqrt{2}$ ($d_{simple} = 2/1000$)
- Case 2: If two customers bought 100 items each and 95 of those are identical, we get an Euclidean distance of $\sqrt{10}$ ($d_{simple} = 10/1000$)
- Thus, against intuition, the customers in Case 2 are mutually less similar than in Case 1
- A similar result we get for d_{simple}

Correction for the Simple Matching Coefficient

- In cases, where state state zero dominates, one can use

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

  where F is the number of components for which both vectors have both zeros.

- In the example, we get for Case 1 $d_{simple00}(\mathbf{x}_i, \mathbf{z}) = 2/(1000 - 998) = 1$ and for Case 2, $d_{simple00}(\mathbf{x}_i, \mathbf{z}) = 10/(1000 - 895) \approx 0.09$

Cosine Distance Function

- The cosine similarity is

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

  and the cosine distance (not a proper distance metric as it does not have the triangle inequality property)

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

- In our example, we get for Case 1 $d_{\cos} = 1$ and in Case 2 $d_{\cos} = 1 - 95/100 = 0.05$.

- Note that if we normalize the length of the vectors to one in a preprocessing step then the cosince similarity is identical to the inner product (linear kernel)

- We have the property

$$d_{\cos}(a\mathbf{x}_i, b\mathbf{z}) = d_{\cos}(\mathbf{x}_i, \mathbf{z})$$

with $a, b > 0$

Pearson Correlation

- Pearson correlation can be used to evaluate the correlation between attributes (columns) or entities (rows). Here we ar interested in the latter
- The Pearson correlation is then identical to the cosine similarity, only that the *mean of each entity (row)* is first subtracted.
- Let $\tilde{\mathbf{x}}_i = \mathbf{x}_i - mean(\mathbf{x}_i)$, $\tilde{\mathbf{z}} = \mathbf{z} - mean(\mathbf{z})$, then

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

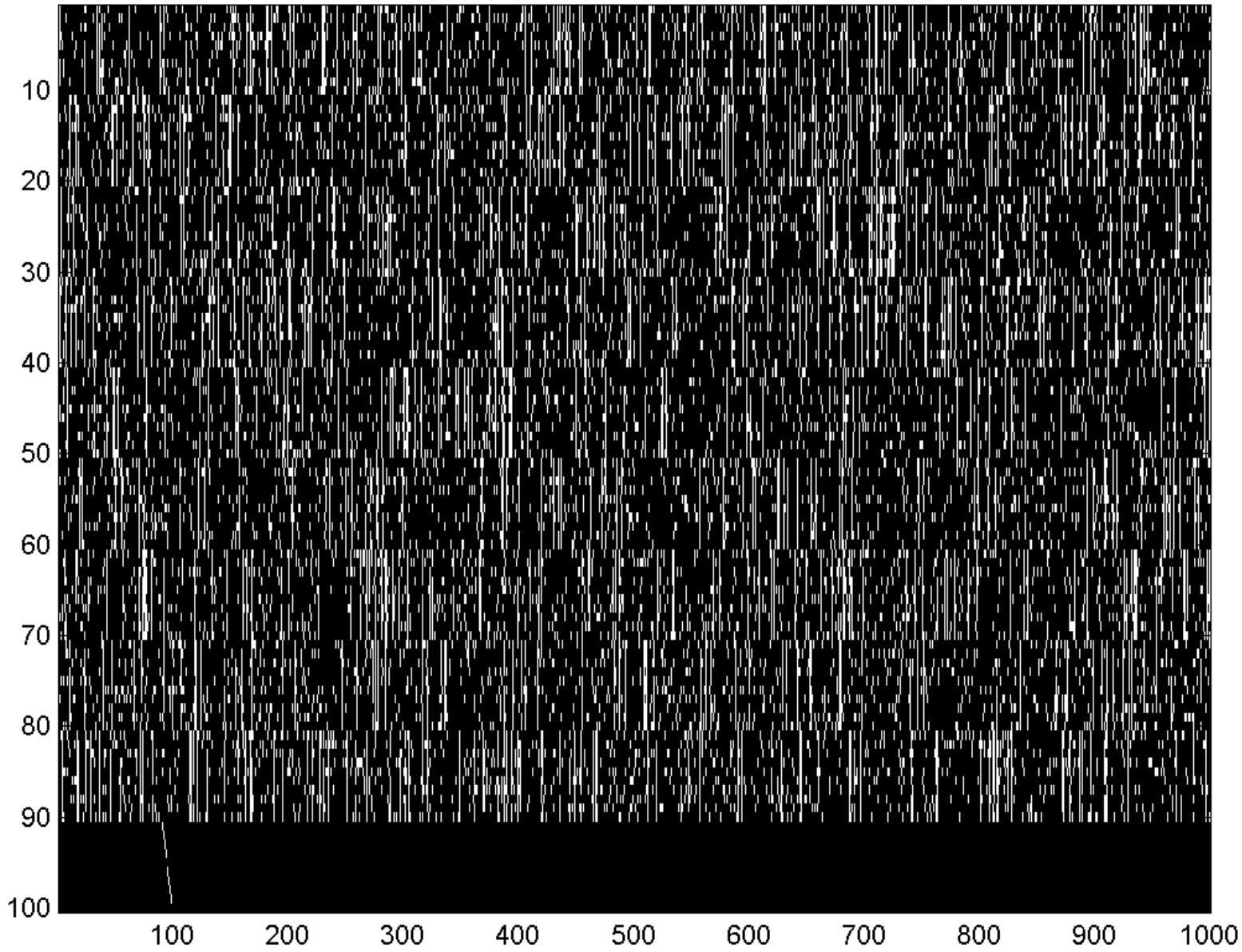
and as a distance function

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

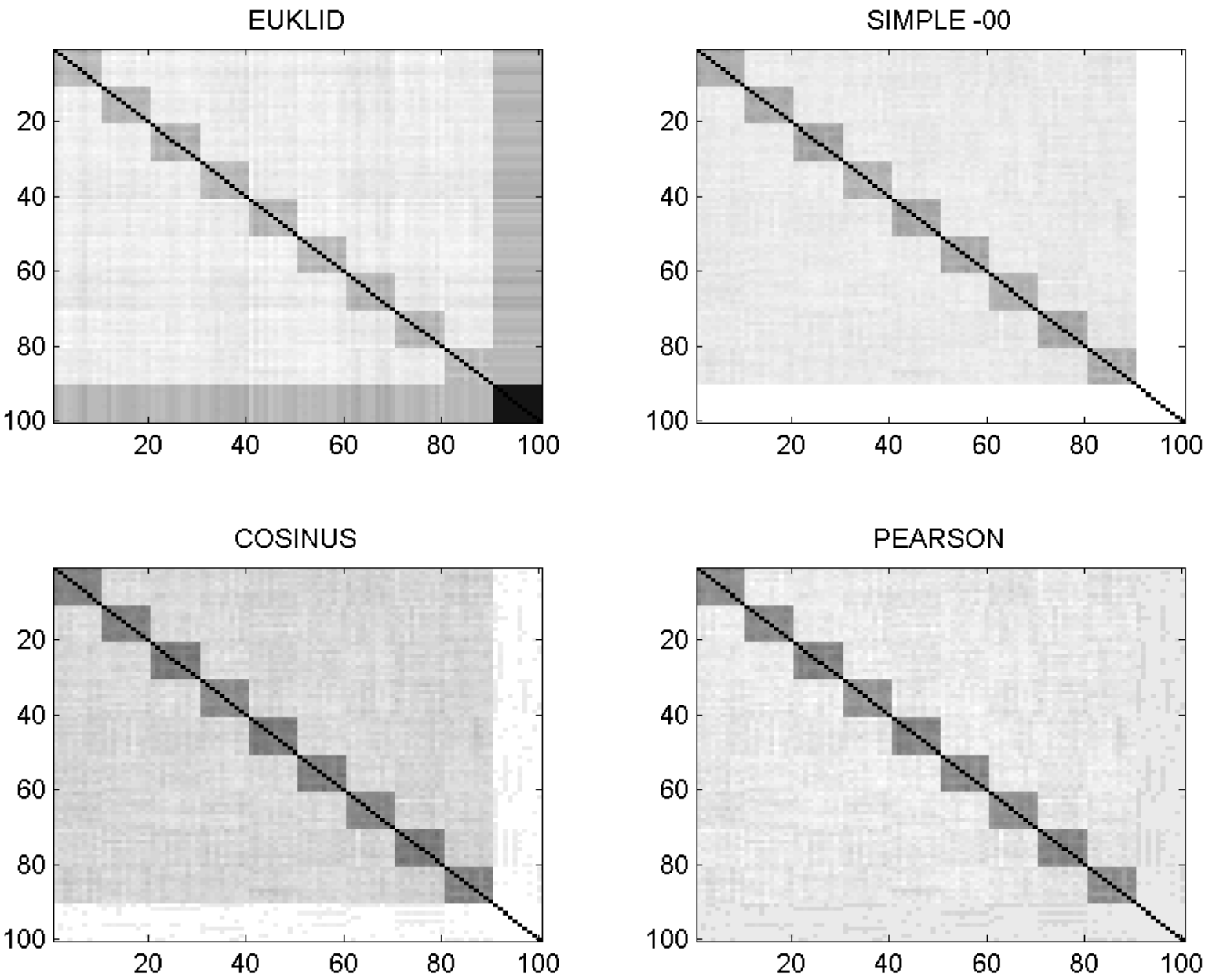
We get for Case 1 $d_{pearson} = 1.01$ and for case 2, $d_{pearson} = 0.056$.

Sparse and Clustered Data

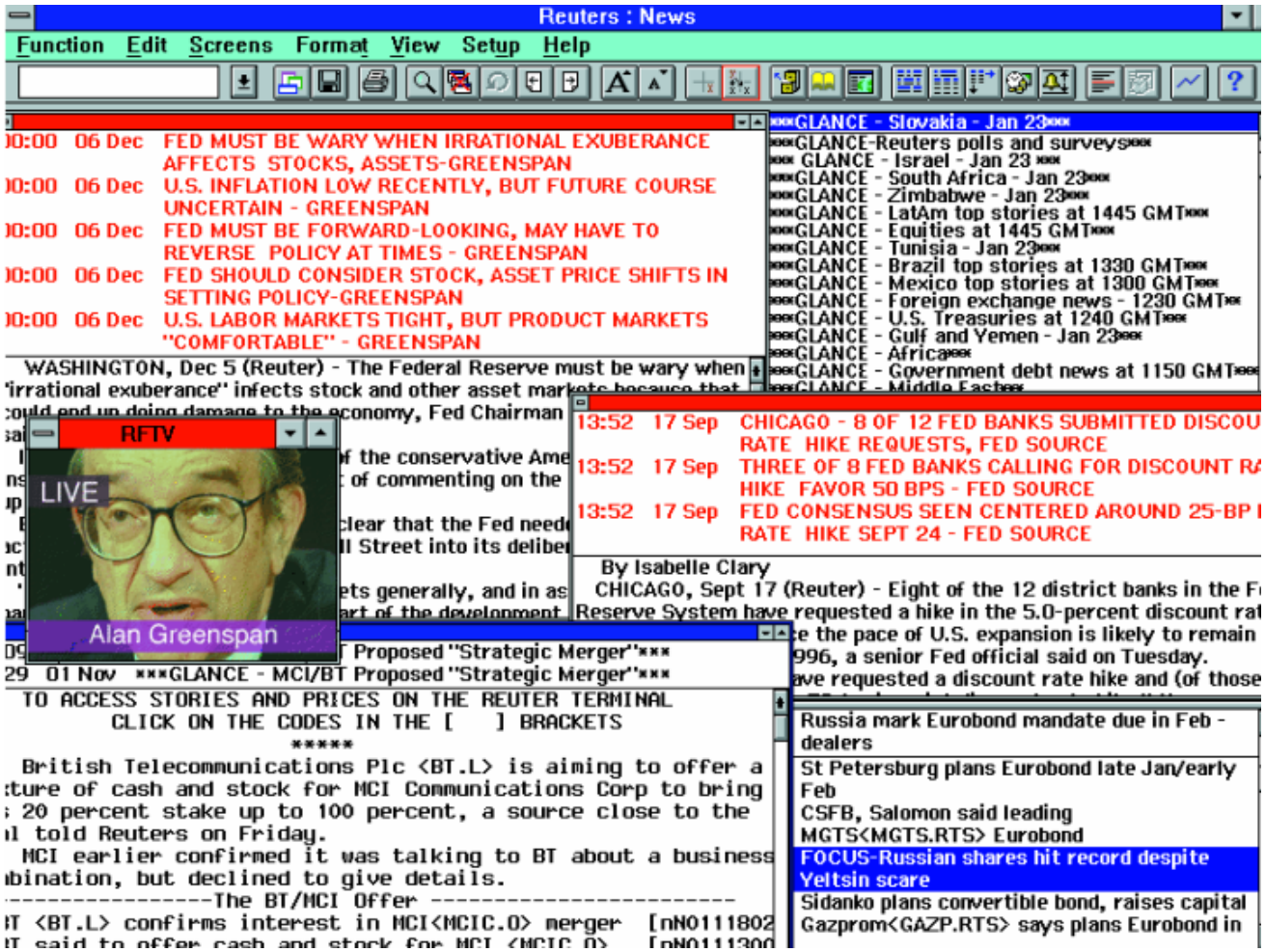
- $N = 100$, $M = 1000$. Data are organized in 10 clusters. In the last cluster in each data vector only one entry is equal to one


10
20
30
40
50
60
70
80
90
100
100
200
300
400
500
600
700
800
900
1000

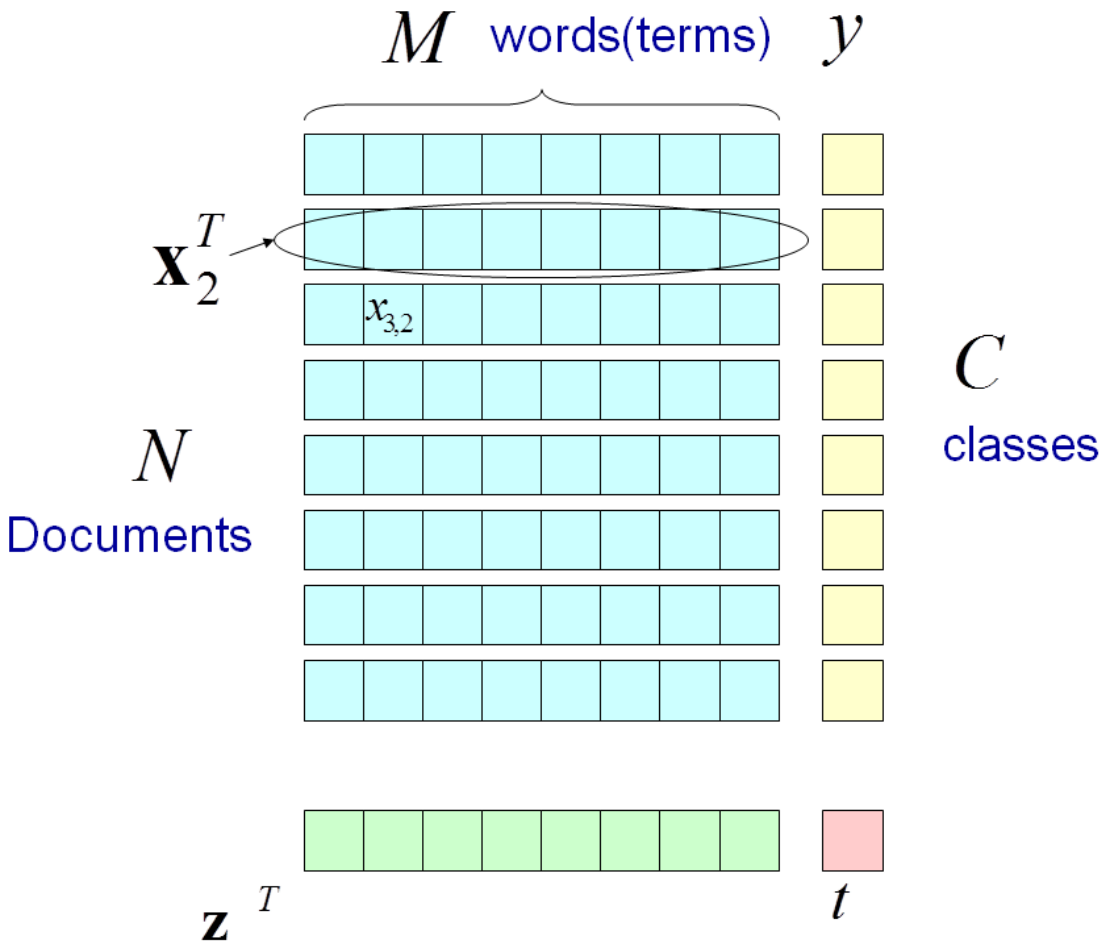
Distance between Documents. Dark: small distance


EUKLID
SIMPLE -00
COSINUS
PEARSON
20
40
60
80
100

Example: Classification of Documents



Vector Space Representation of a Document


M words(terms)
y
$\mathbf{x}_2^T$
$x_{3,2}$
C
classes
N
Documents
$\mathbf{z}^T$
t

Vector Space Representation of a Document

- Depending on preprocessing $x_{i,j}$ can represent slightly different quantities
- (A) $x_{i,j} \in \{0, 1\}$ is euqual 1, if word j is present in document i, otherwise 0
- (B) $x_{i,j} \in \{0, 1, 2, \ldots\}$ is the count of word j in document i (*term frequency* tf)
- (C) (B) $x_{i,j} \in \Re$ puts weights on tf. Most popular *inverse document frequency*:

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

  N is the number of documents. n_j is the number of documents in which word j occurs at least once. Thus a word obtains more weight if it is rare! A word that occurs in all documents gets a weight of zero.

- Preprocessing steps: stemming ("fishing", "fished", and "fisher" are all reduced to the root word, "fish"), elimination of *stop words* ("and", "or"); cosine similarity is commonly used

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 (cont'd)

- tf was used
- cosine as similarity measure
- C4.5, Ripper: Decision trees. PEBLS, VSM, WAKNN: different k-nearest-neighbor classifiers with word weights. Rainbow: naive-Bayes
- Reuters: News pices. fbis: Foreign Broadcast Information Service. trec6L: LA-Times news articles. west: law documents

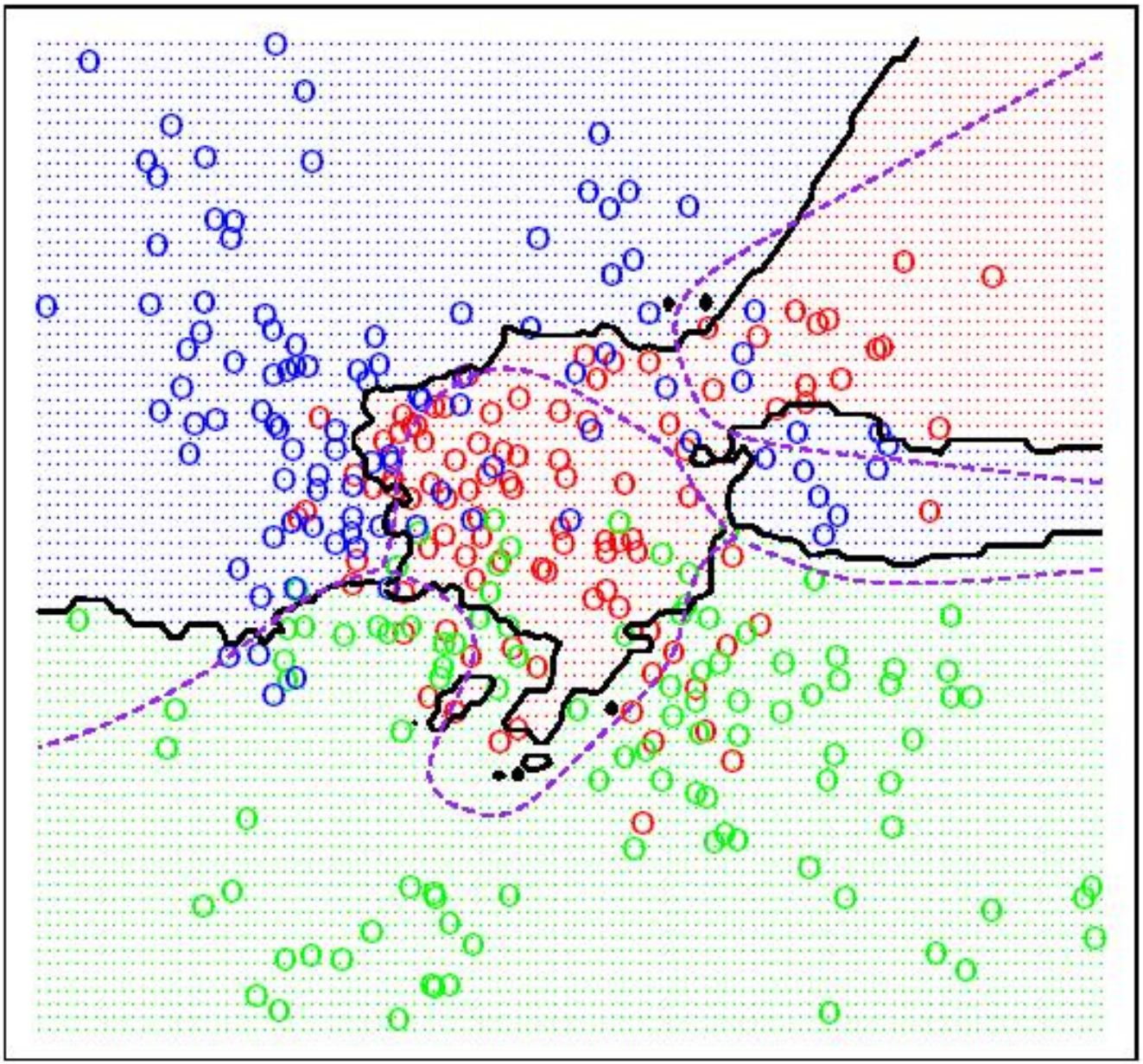
Text categorization (cont'd)

Table 1: Summary of data sets used.

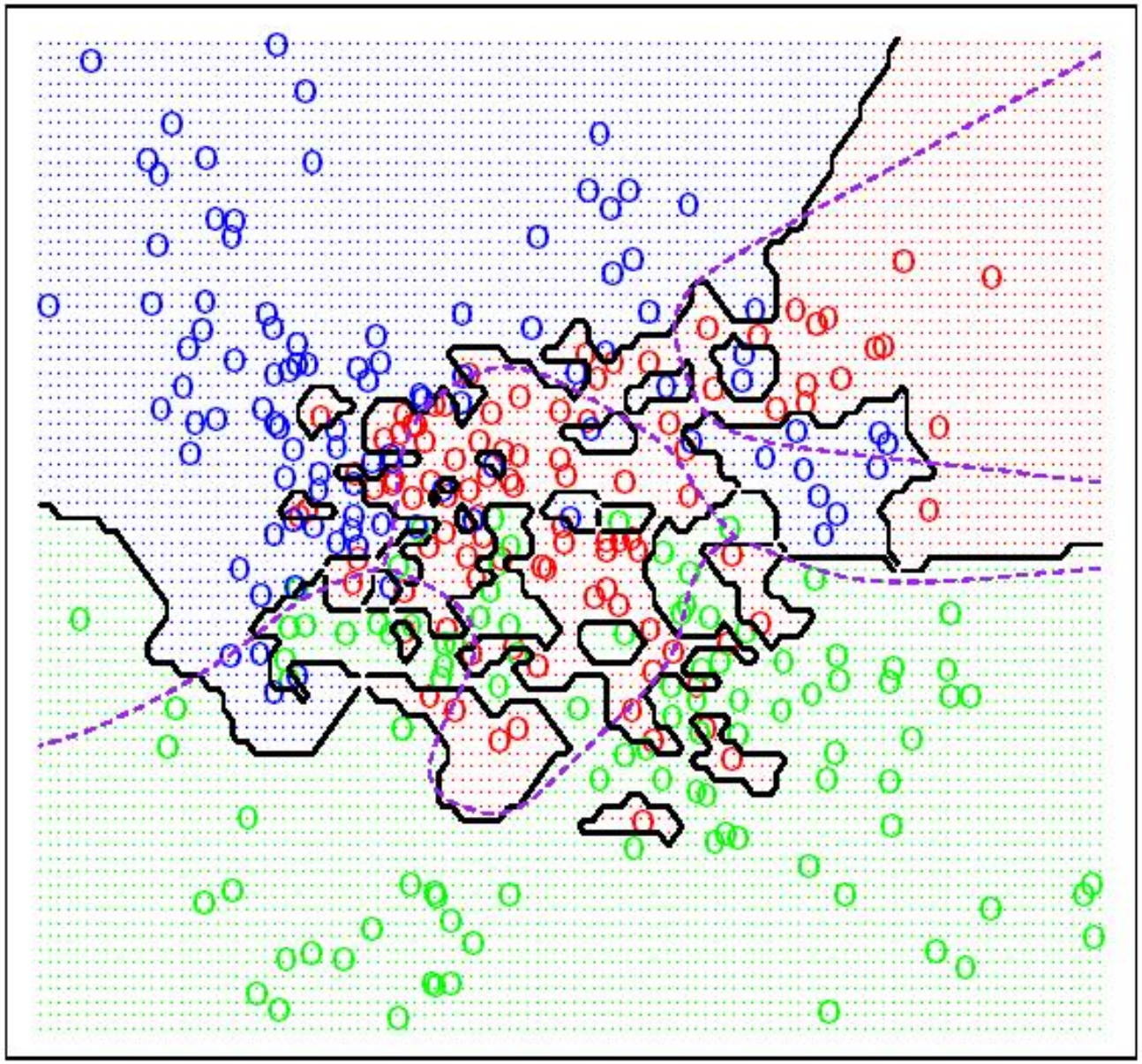
	C4.5	RIPPER	PEBLS	VSM	Rainbow	k-NN	WAKNN
west-1	85.50	84.47	78.50	85.20	84.40	76.73	**89.60**
west-2	71.30	68.33	67.80	77.44	72.11	68.33	**80.44**
west-3	79.60	75.92	72.70	86.53	80.00	70.61	**88.16**
west-4	81.80	77.14	78.60	87.86	**88.57**	73.93	85.00
west-5	84.60	89.71	86.80	89.71	85.21	84.57	**95.18**
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

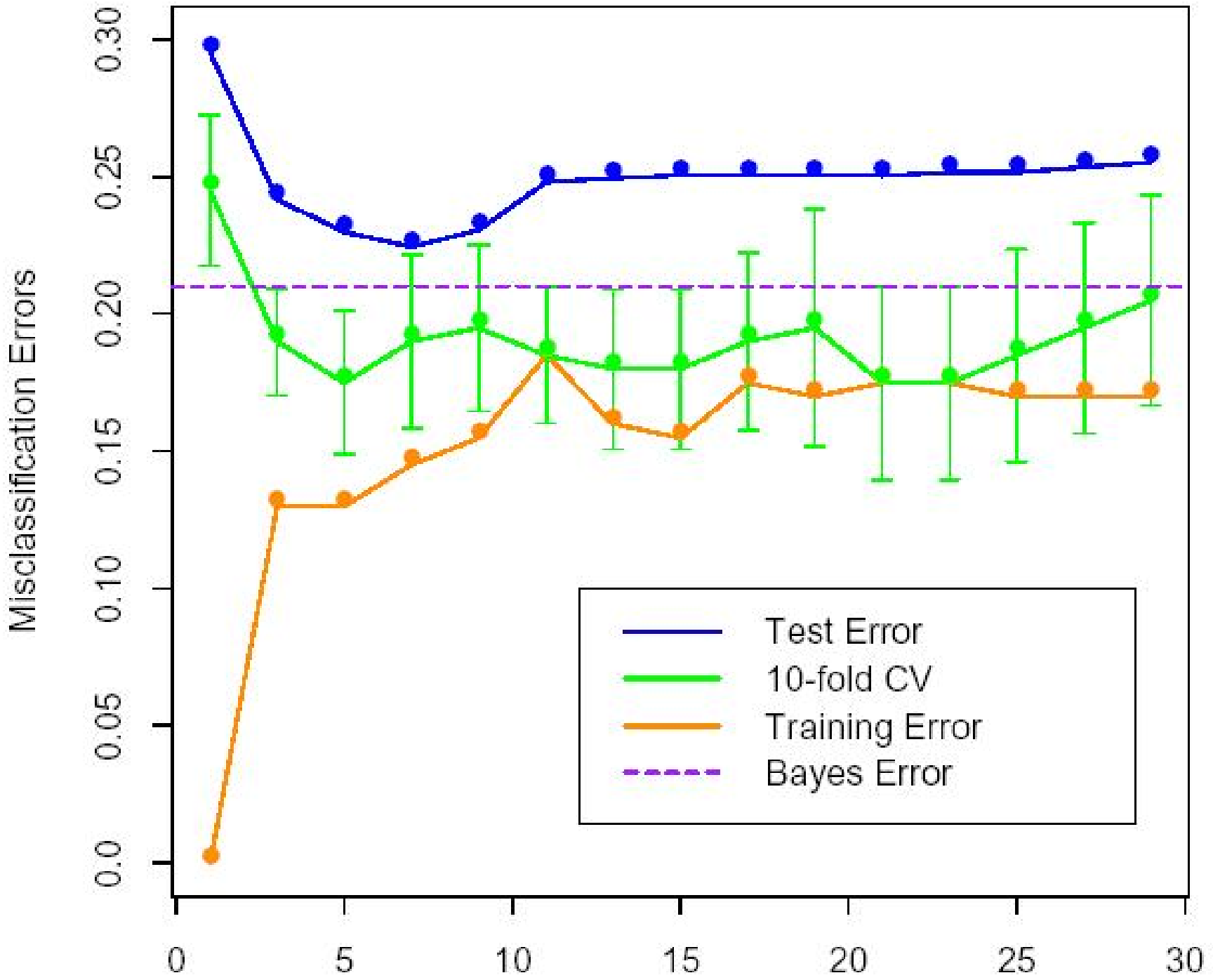
Optimizing k in k-Nearest-Neighbor Classification

15-Nearest Neighbors

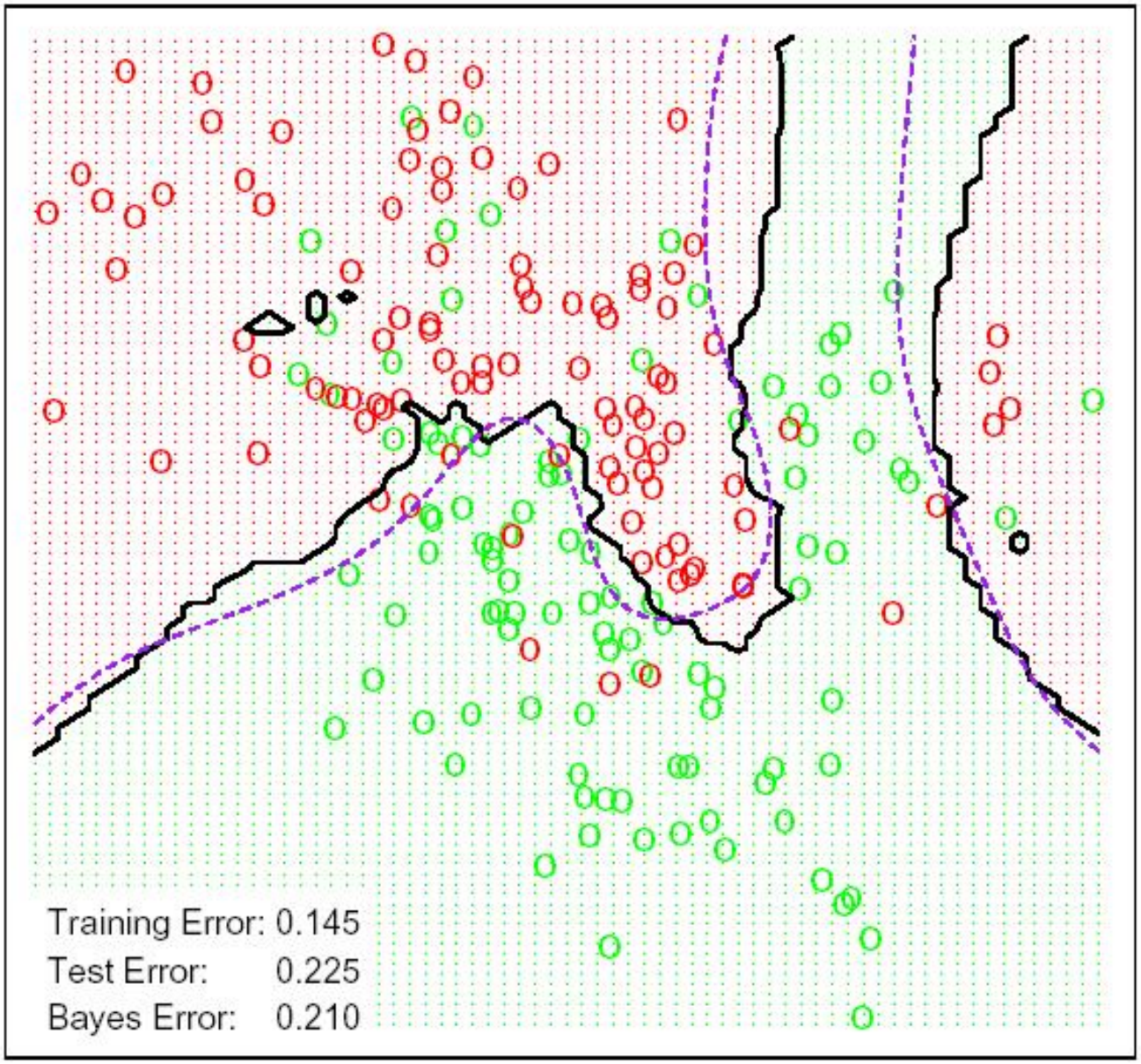


1-Nearest Neighbor




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

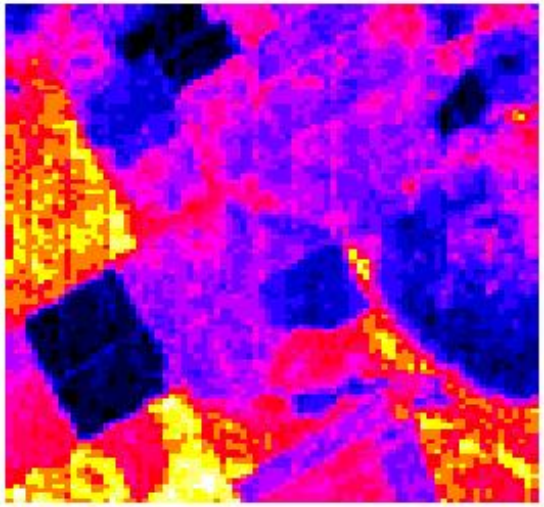
7-Nearest Neighbors



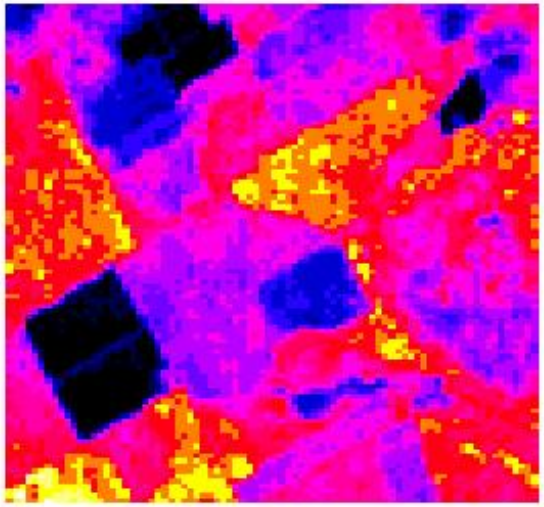
K-Nearest Neighbor Classification for LANDSAT Images

- Data: 4 spectral bands (infrared)
- A pixel is classified as on out off 7 classes: cotton, red soil, grey soil, ...
- Input: spectral bands of the pixel itself and of its 8 neighboring pixels Thus there are $4 \times (8 + 1) = 36$ inputs
- 5-nearest neighbor classifiers gave best results

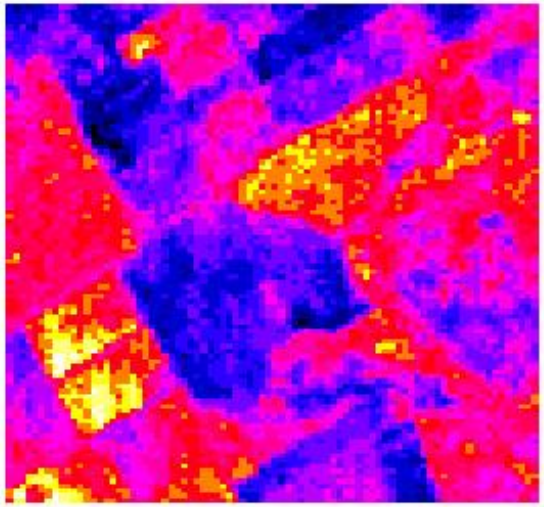
Spectral Band 1



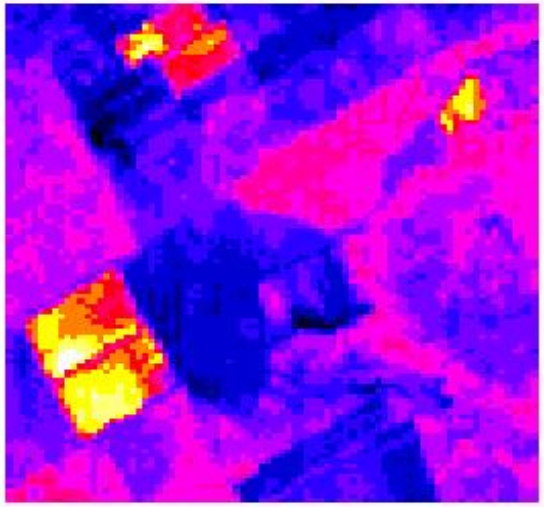
Spectral Band 2



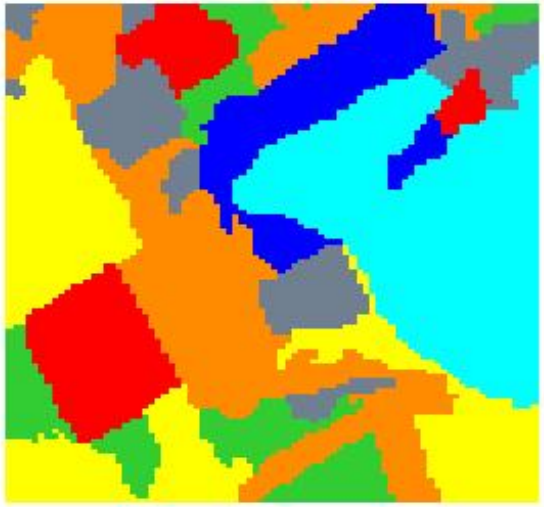
Spectral Band 3



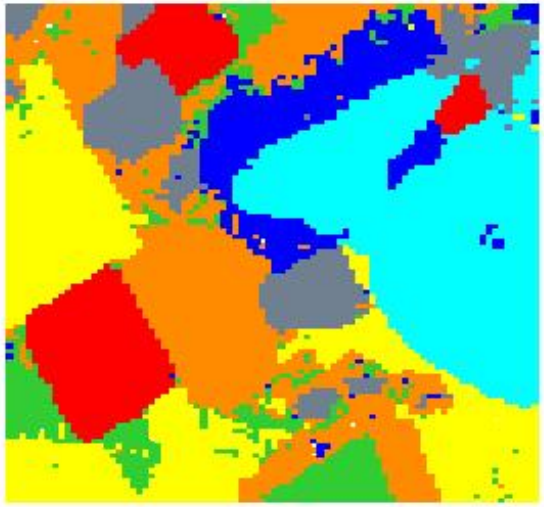
Spectral Band 4

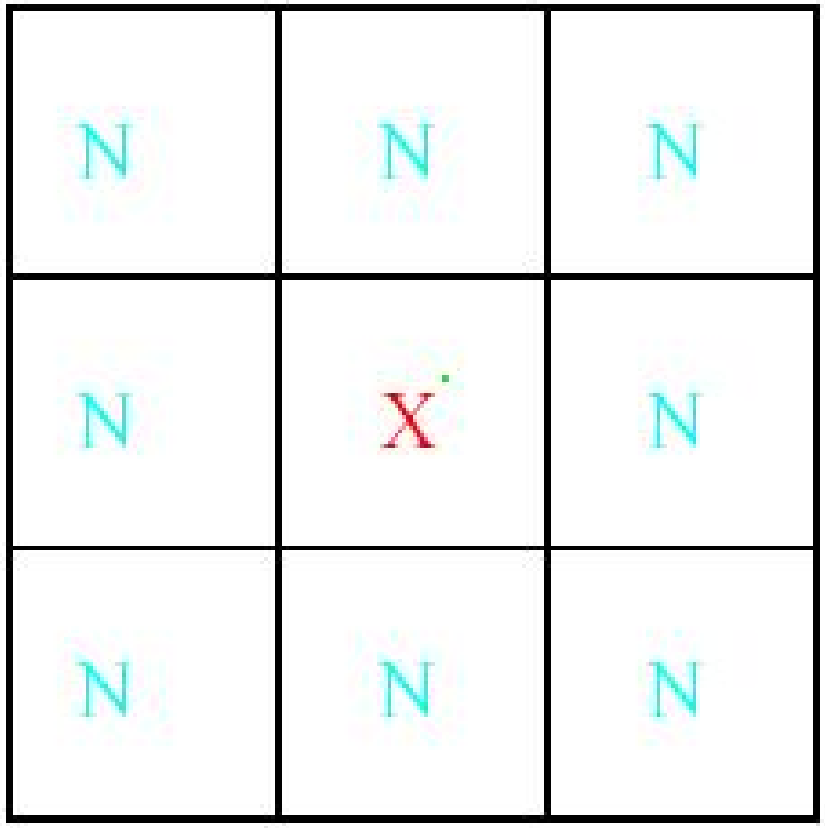


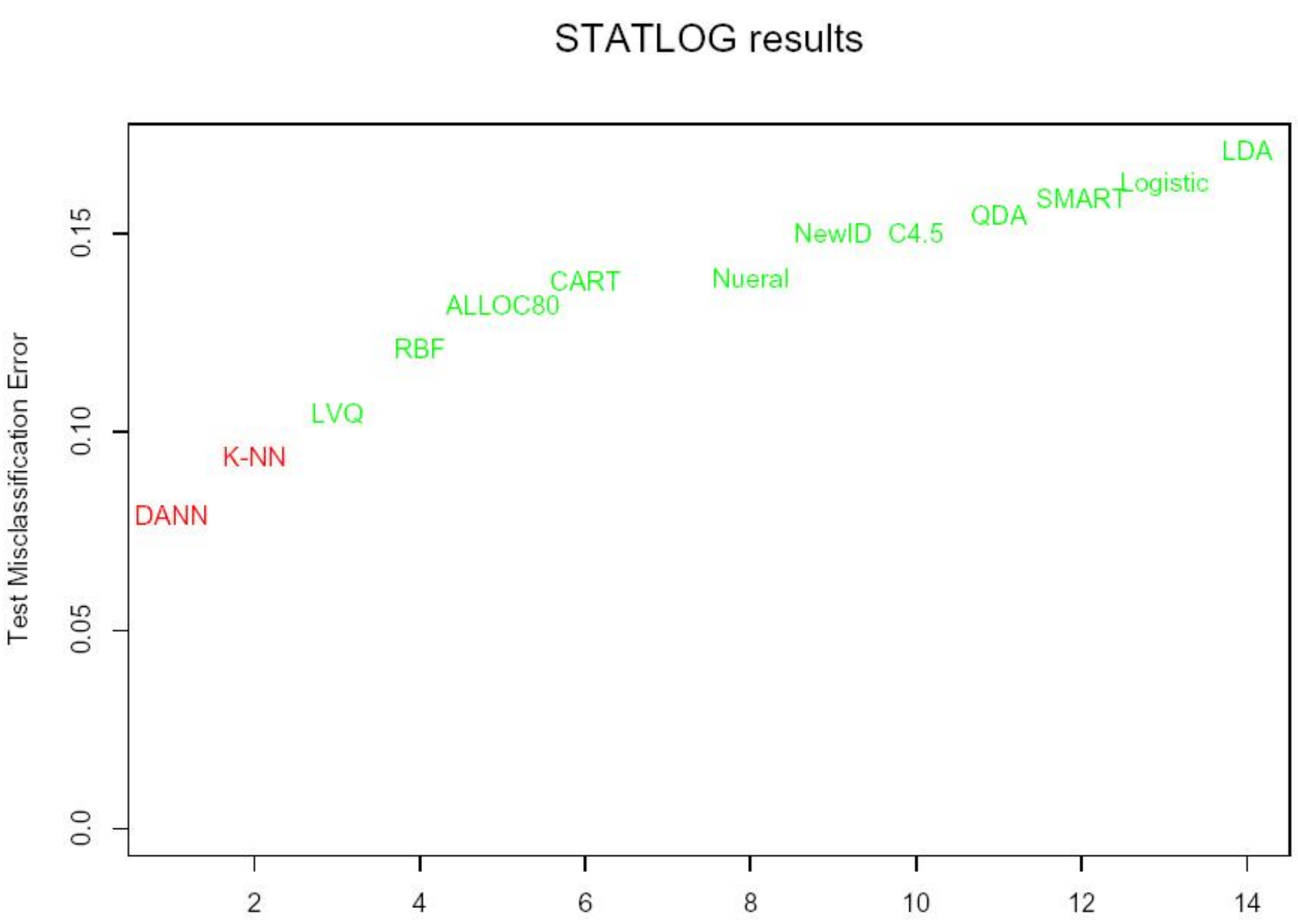
Land Usage



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

Grandmother Cells

- Hypothesis 1: the brain stores a 3-D model of an object. During recognition the brain compares an image of an object to the 3-D model. This approach is efficient in terms of storage but computationally involved during recognition
- Hypothesis 2: The brain stores several 2-D view of an object and compares an image of an object to the different 2-D views. This approach might require more storage but can be done in parallel at the time of recognition. For each view there is a specific neuron: grandmother cell
- Bag-of-words model in computer vision: image features (maybe calculated at a set of regions of interests) are treated as words, an image is represented as image feature counts, and similarity is calculated based on a distance metrics. Popular features: Scale-invariant feature transform (or SIFT) features, published by David Lowe in 1999

Kernels and Similarity Functions

- Recall that the similarity function (often: $sim(x, z) = 1 - dist(x, z)$) should reflect similarity in terms of the function to be approximates
- If $sim(x, z)$ is a positive definite function, we can think of it as a kernel function $k(x, z)$
- Since one interpretation of a kernel is

$$k(x, z) = cov(f(x), f(z))$$

  a similarity function might have the same interpretation and a distance function should imply a small distance for locations with correlated functional values
- A kernel classifier can converge towards a nearest neighbor classifier for local kernels if $k(x_i, x_j) \to 0$ for training points x_i and x_j. Example: Gaussian kernel
- If we start with a $d(x, z)$ a nearest-neighbor classifier implies a kernel that corresponds to a high-dimensional feature space

Class-specific Distributions and Distance Functions

- Consider that $P(z|y_i) = \mathcal{N}(z; \mu_i, \sigma^2 I)$ are Gaussians with diagonal covariance
- The the maximum probably posterior class is

$$\arg\max_i \left(\log P(i) - \frac{1}{2\sigma^2} \|z - \mu_i\|^2 \right)$$

- Thus, if we assume that the class probabilities $P(i)$ are equal, we decide for the class with the smallest distance to the data point
- Conversely, given a distance function we might imply a class-specific probability distribution

$$P(z|y_i) = \frac{1}{Z} \exp -\frac{1}{S} d(z, \mu_i)^n$$

would all lead to nearest neighbor rules where $S > 0$, $n \geq 1$, and Z normalizes the distribution

Parzen Windows as Density Estimators

- We can also model

$$P(z|y_i) = \frac{1}{N_i}\frac{1}{Z}\sum_{y(x_k)=y_i} \exp -\frac{1}{S}d(z, x_k)^n$$

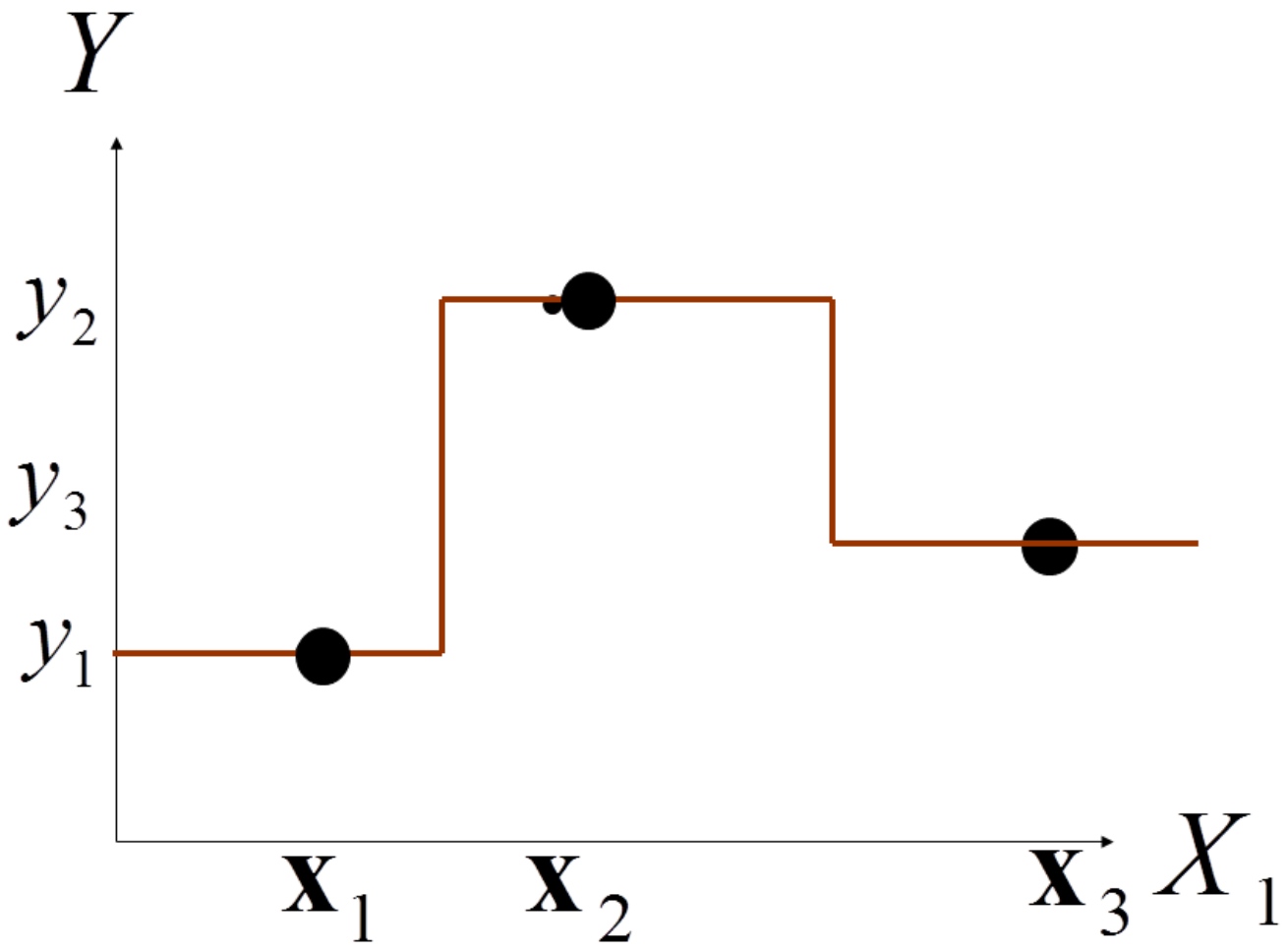
  by defining a probability distribution for each training data point, where N_i is the number of training data points in class i

- With $S \to 0$ we get a nearest neighbor classifier

Instance-based Approaches for Regression

Nearest-neighbour Approaches for Regression

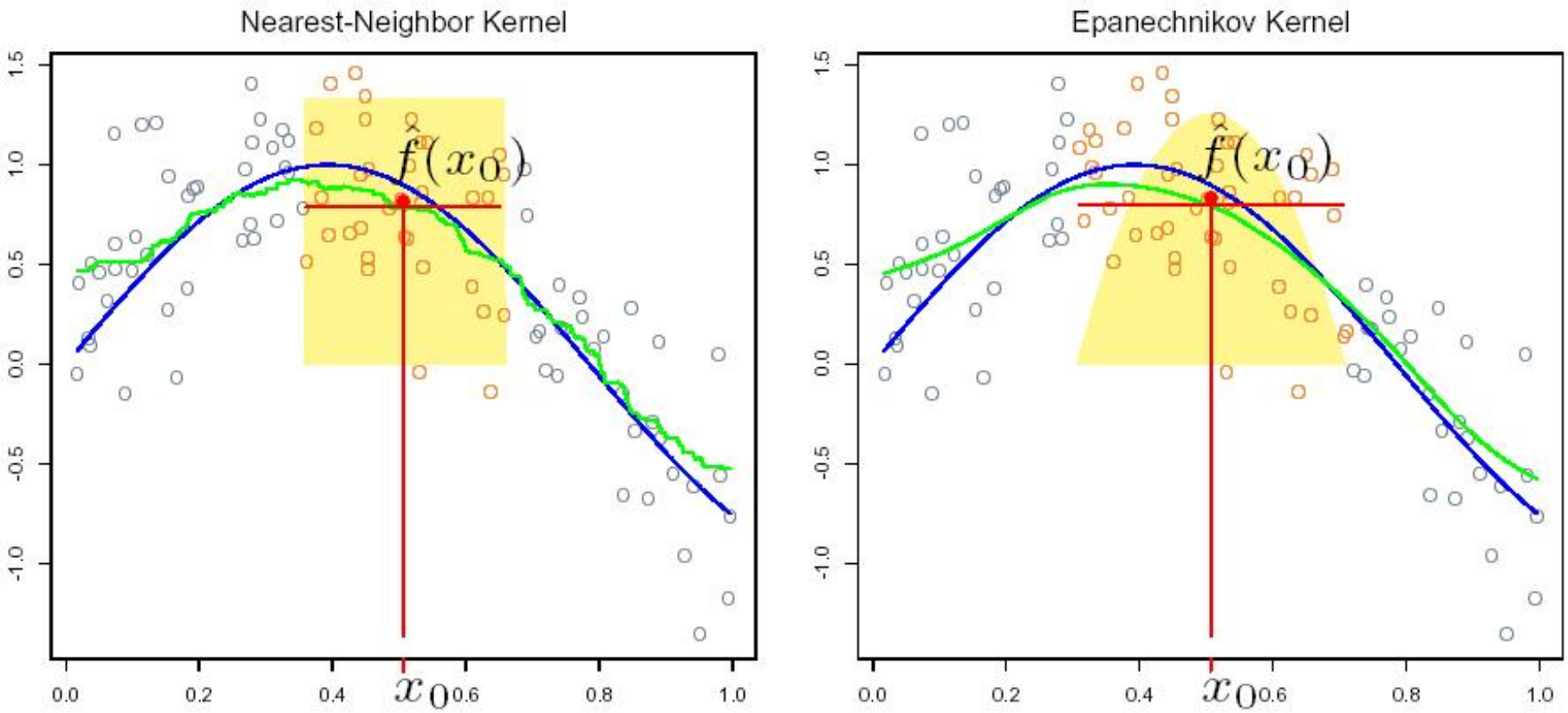
- The solution is piecewise constant


Y
y2
y3
y1
x1
x2
x3
X1

Comments

- Regression: $y \in \Re$
- The performance of a k-nearest-neighbor system is typically not very good
- Let $J(\mathbf{z})$ be the indices of the k-nearest neighbors to $\mathbf{z}$
- $k-$ nearest-neighbor smoother:

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


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

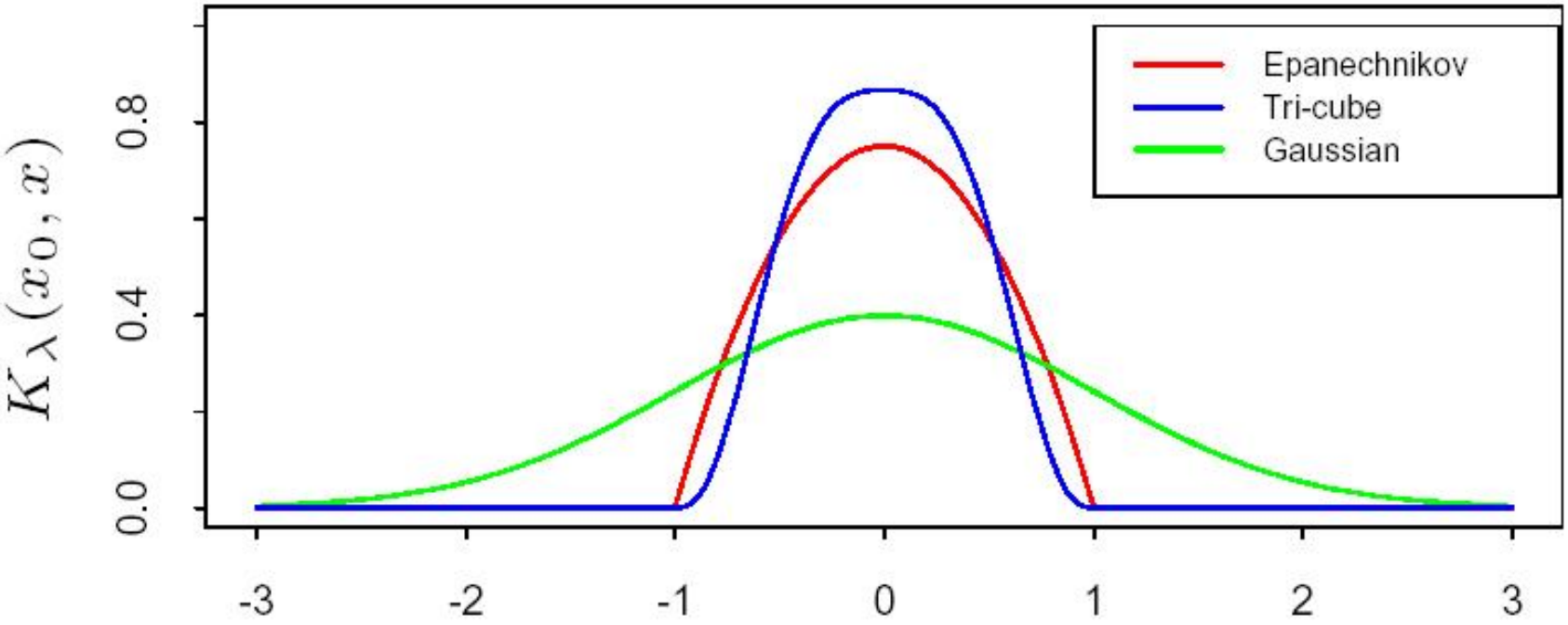
Kernel Smoother:

- Nadaraya-Watson smoother

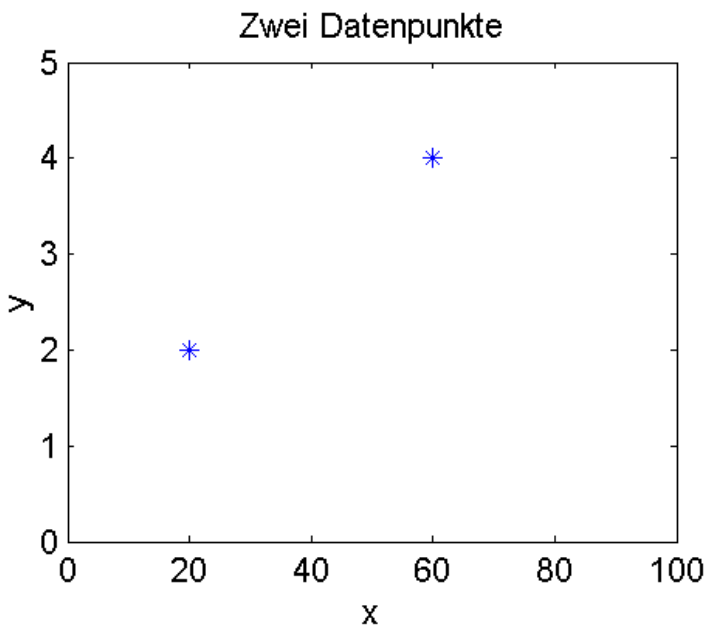
$$\hat{t} = \frac{1}{\sum_{i=1}^{N} K_\lambda(\mathbf{z}, \mathbf{x}_i)} \sum_{i=1}^{N} K_\lambda(\mathbf{z}, \mathbf{x}_i) y_i$$

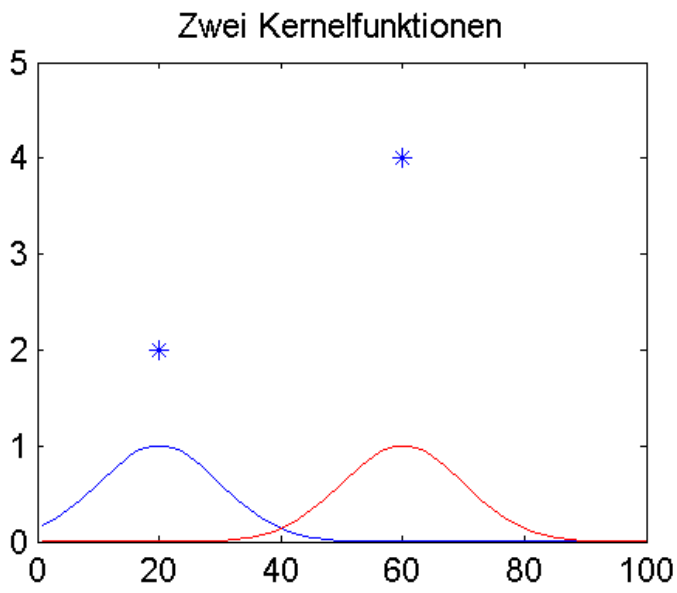
- Example: Gaussian kernel with

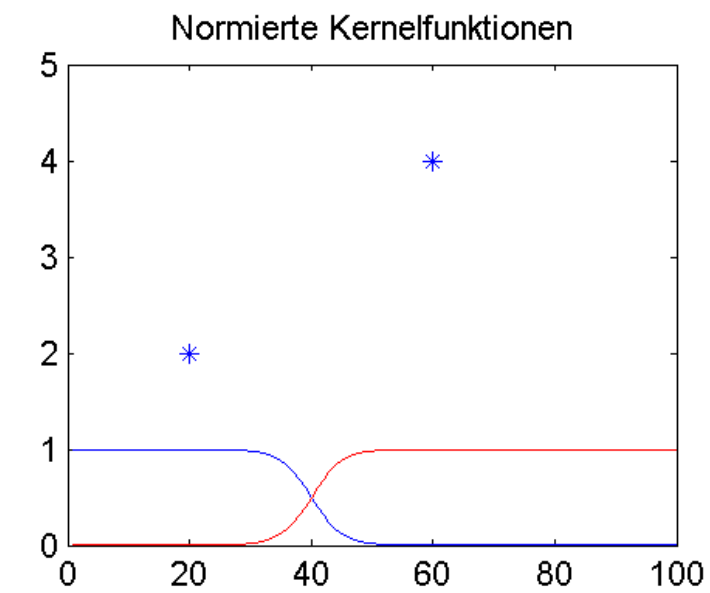
$$K_\lambda(\mathbf{z}, \mathbf{x}_i) = \exp\left(-\frac{1}{2\lambda^2}|\mathbf{z} - \mathbf{x}_i|^2\right)$$

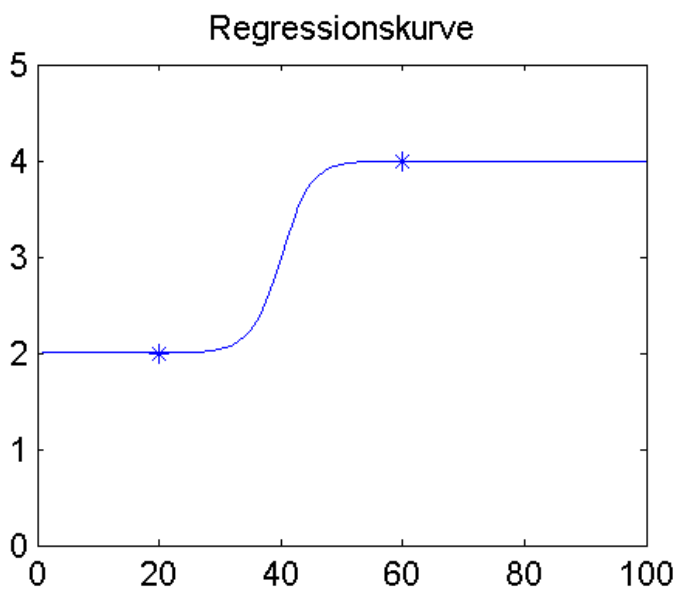

$K_\lambda(x_0, x)$
0.0
0.4
0.8
-3
-2
-1
0
1
2
3
Epanechnikov
Tri-cube
Gaussian

Kernel Smoother: 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

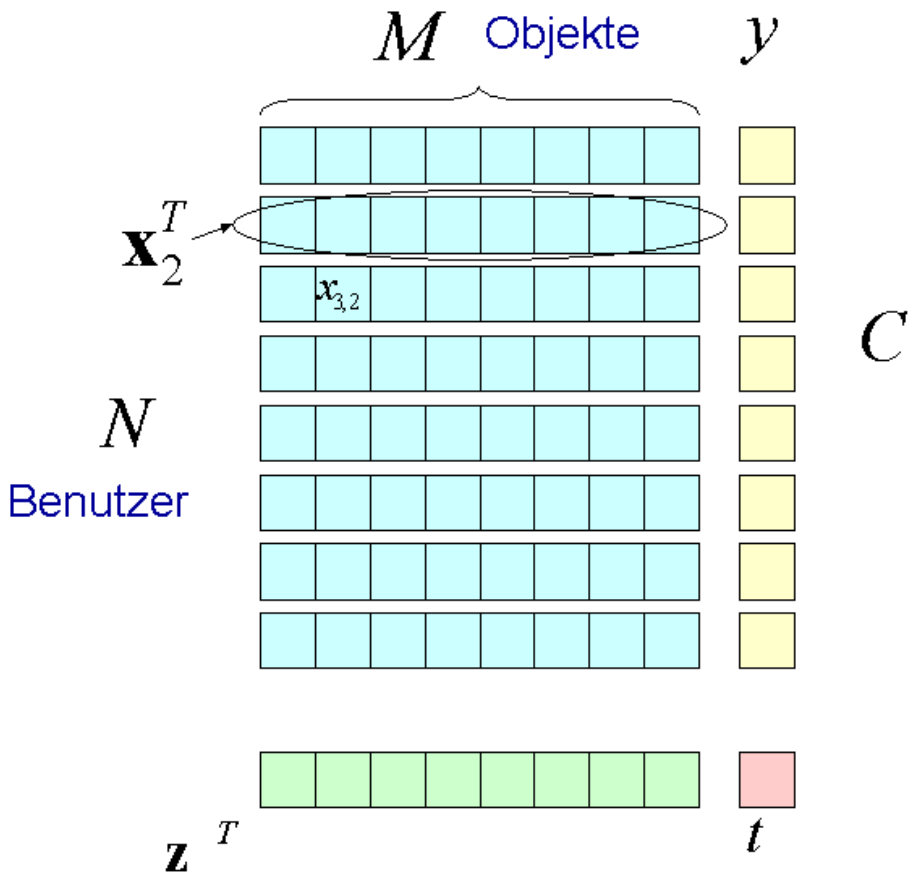
- (1) Two data points $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2)$
- (2) $K_\lambda(\mathbf{z}, \mathbf{x}_1)$, $K_\lambda(\mathbf{z}, \mathbf{x}_2)$
- (3)

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

- (4)

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

Example: Collaborative Filtering



- CF: Recommensation systems, who can work without user or item properties
- $x_{i,j}$ is the evaluation of user i for item j-te; typically most ratings are missing
- The goal is to predict the evaluation of the active user for a item the active user has not rated yet
- In the example $\mathbf{z}$ is the evaluation of the active user

Empirical Analysis of Predictive 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 visited or not visited
- TeleVision: Show watched or not
- EachMovie: movie evaluation: $1, \ldots, 5$ Points

CF+: Instance-based System for CF

- Let a be the active user and let's assume that we want to calculate the prediction for the active user for item q

$$\widehat{x}_{a,q} = mean_a + \frac{1}{\sum_{i:i \text{ rated } q} w(\mathbf{x}_i, \mathbf{x}_a)} \sum_{i:i \text{ rated } q} w(\mathbf{x}_i, \mathbf{x}_a)(x_{i,q} - mean_i)$$

  $mean_i$ is the mean evaluation of the user i

  $mean_a$ is the mean evaluation of the active user i

- The weight $w(\mathbf{x}_i, \mathbf{z})$ is the Pearson-Correlation calculated over movies that were both rated by user i and the active user

Recommender System

- One recommends the K top-ranked items to the active user
- BN:Bayes net, CR+: our algorithm , VSIM: similar bit with cosine, BC: clustering, POP: recommend the most popular item
- RD (required difference): minimum distance for a significant difference
- Given 5: the number of items that the active user has rated

	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.

Results: 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.

Results: 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.

Locally Weighted Regression

- This is even more expensive to calculate at prediction time but sometimes yields very good results
- Also known as *lowess* or *loess* (locally weighted scatter plot smoothing)
- Prediction is obtained via simple linear regression

$$y(z) = \sum_j w_j(z) z_i$$

but the weights $w(z)$ are dependent on the query

- Let $k(x_i, z)$ be some similarity function and $K(z)$ be a diagonal matrix with $K(z)_{ii} = k(x_i, z)$, then

$$w(z) = \left(X^T K(z) X + \lambda I\right)^{-1} X^T K(z) y$$

Kernel Smoothers from Joint Density Models

- Consider that we model the joint input-output distribution as a sum of Gaussians

$$P((z^T, t)^T) = \frac{1}{N} \sum_i N((z^T, t)^T; (x_i^T, y_i)^T, \sigma^2 I)$$

we get

$$E(t|z) = \frac{\sum_i y_i N(z; x_i, \sigma^2 I)}{\sum_i N(z; x_i, \sigma^2 I)}$$

which is exactly the Nadaraya-Watson smoother smoother

Kernel Smoothers and Kernel Systems

- One can consider

$$k_i(z, x_i) = \frac{N(z; x_i, \sigma^2 I)}{\sum_k N(z; x_k, \sigma^2 I)}$$

  to be kernels derived from data. These are in general non-Mercer kernels since the resulting kernel matrix is not necessarily positive definite

- The weights could then be calculated, e.g., by regularized least squares and give

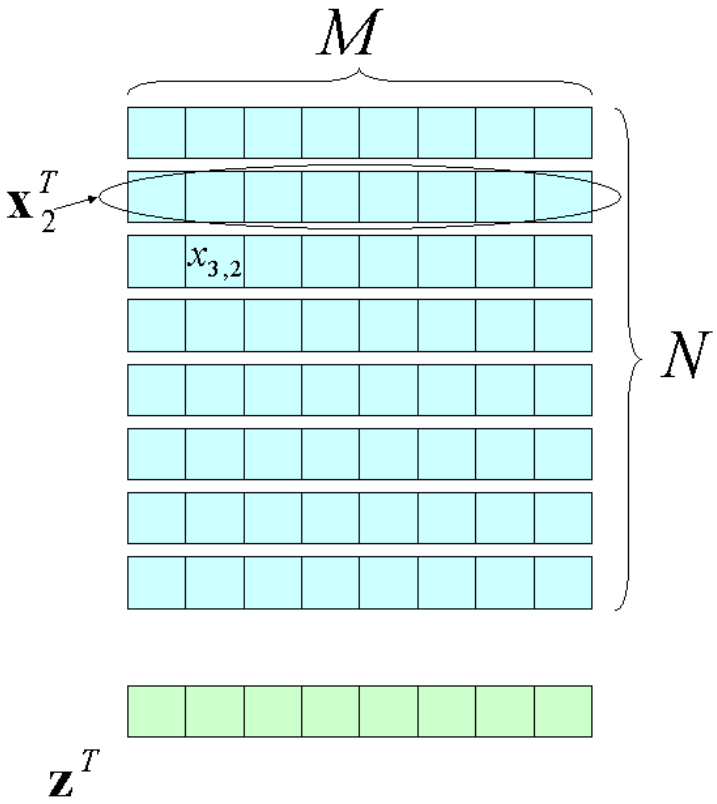
$$t(z) = \sum_i v_i k_i(z, x_i)$$

Instance-based Approaches for Density Estimation (One-Class Classification)

One-Sided Data

- In some applications one can only obtain examples for one class label
- Many normal user data but no fraud data
- Data from normal plant operation, but no data from plan failure
- Sometimes it makes sense to consider all data which are not in the training set as negative examples; this makes sense if there are only a finite number of potential data. Another option is to randomly sample data from some assumed distribution and treat them as negative data
- Another option is to model $P(x|class = 1)$ and then to evaluate if the new observed data seems to come from this distribution

Data Matrix for Unsupervised Learning



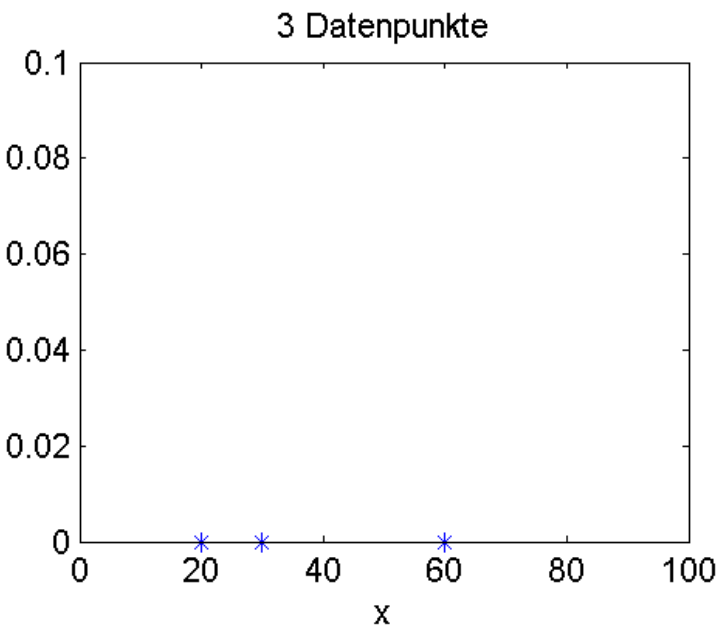
X_j	j-te Variable
X	$= (X_1, \ldots, X_M)^T$ Vector of variables
M	number of Variables
N	number of data points
$\mathbf{x}_i$	$= (x_{i,1}, \ldots, x_{i,M})^T$ i-th data point
$x_{i,j}$	j-th component of $\mathbf{x}_i$
D	$= \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ (trainings-) data set
$\mathbf{z}$	test vector

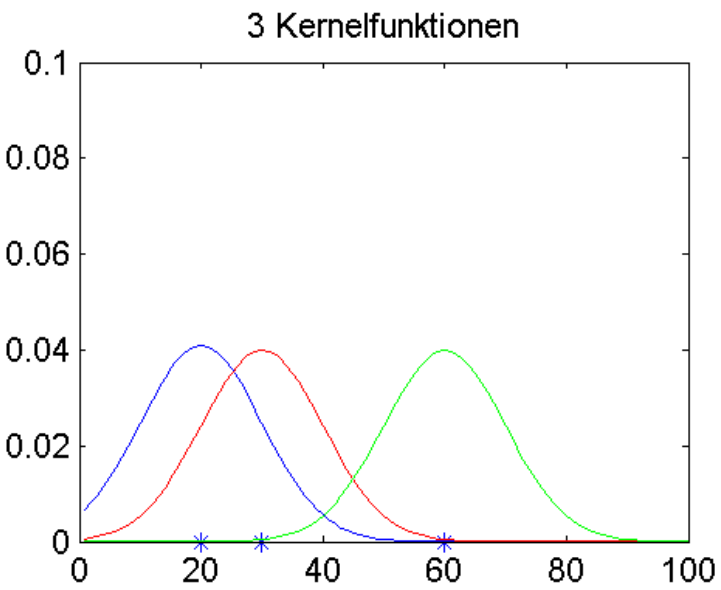
Kernel Density Estimation (cont'd)

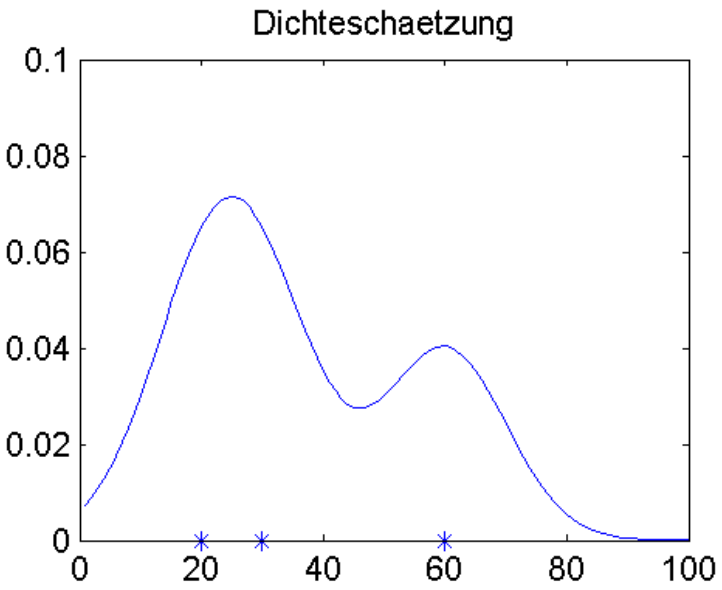
- Parzen estimator

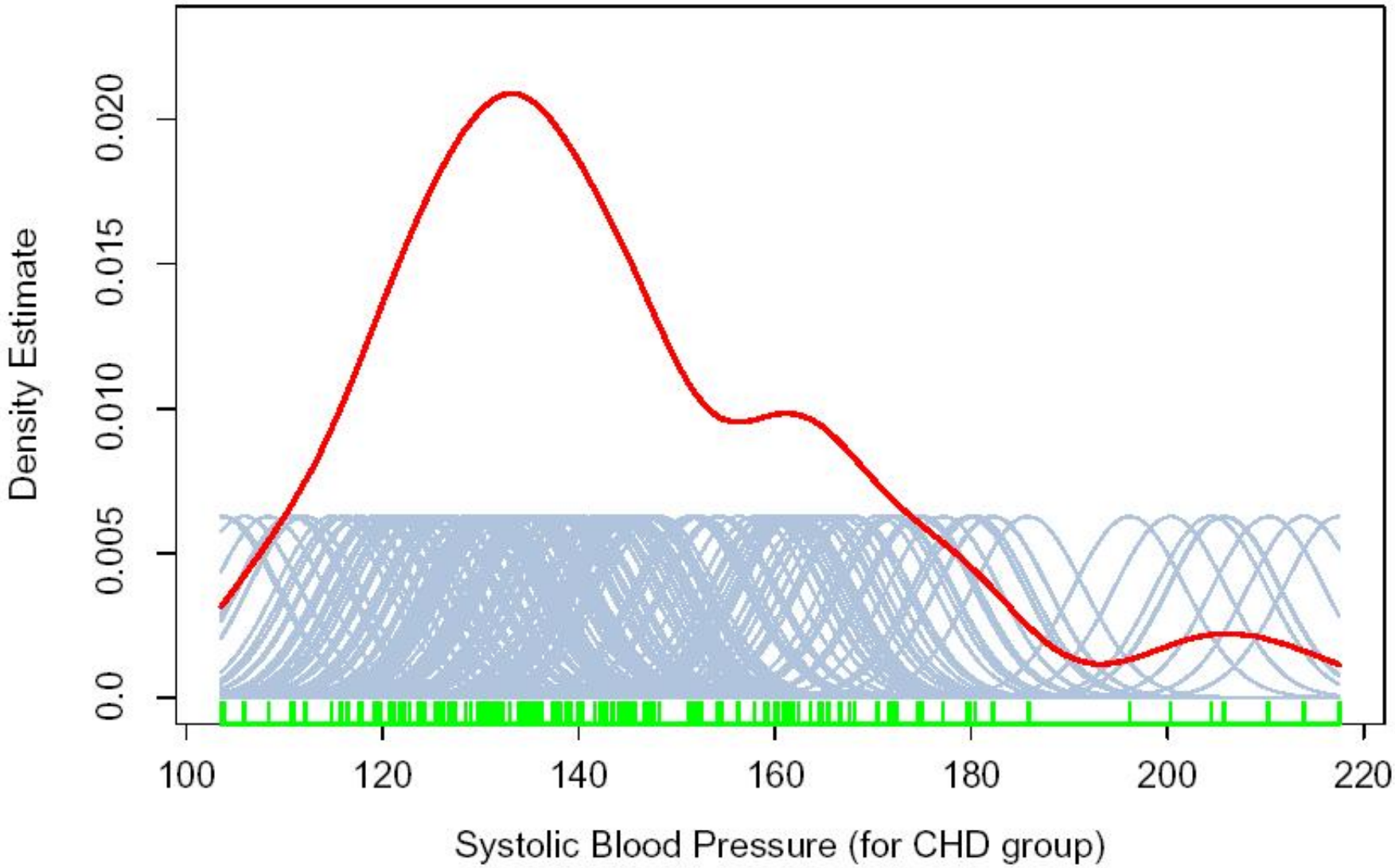
$$\hat{P}(\mathbf{z}) = \frac{1}{c \times N} \sum_{i=1}^{N} K_\lambda(\mathbf{z}, \mathbf{x}_i)$$

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

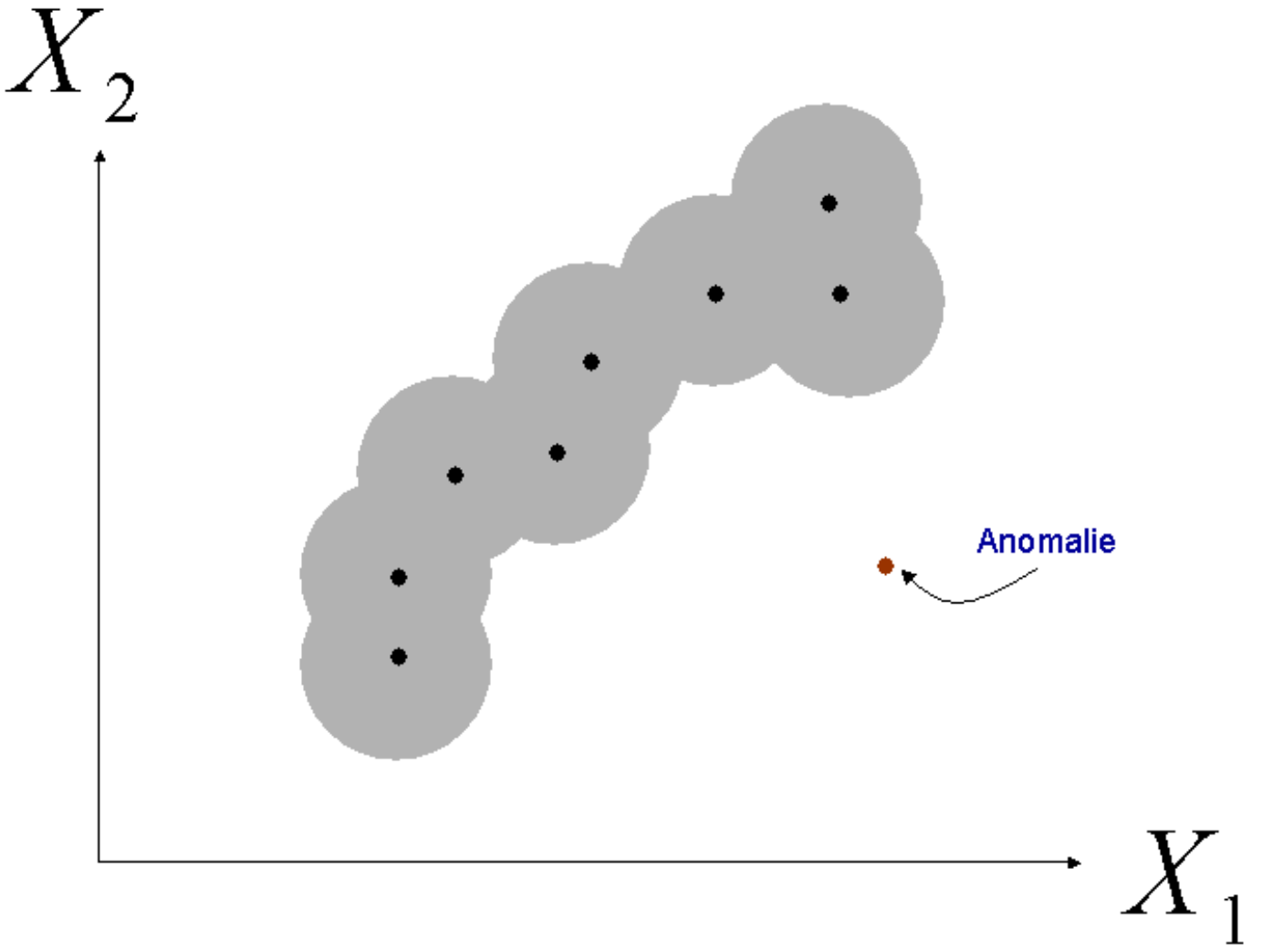

3 Datenpunkte
0.1
0.08
0.06
0.04
0.02
0
0
20
40
60
80
100
x


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 Smoother: Anomaly Detection



Concluding Remarks

- Instance-based approaches have their advantages when the input dimensions are not too large but the functions/class boundaries are complex. Examples: spatio-temporal data
- With large N one needs to build appropriate index structures (k-d trees) to quickly find neighboring data points