

flusser@utia.cas.cz
sroubekf@utia.cas.cz

Prof. Ing. Jan Flusser, DrSc.
Prof. Ing. Filip Šroubek, DSc.

STROJOVÉ UČENÍ

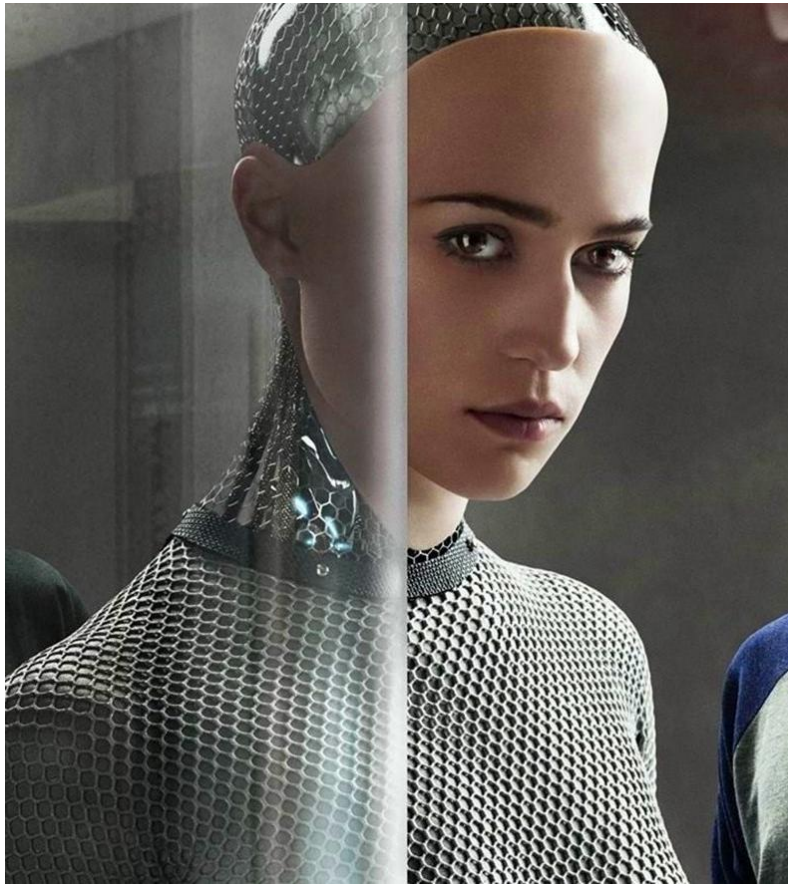
Předměty a návaznost

- SU1/USU ZS, 2+1
  Základní kurz. Existuje i jako ROZP2/ROZRF2
- SU2/HU ZS, 2+2
- DIZO LS
- SFTO LS

Umělá inteligence (AI)

- Systém (zařízení, stroj, bytost, …), který umí vnímat vnější prostředí, vyhodnocovat ho, rozhodovat se a vykonat akci ke splnění cíle, stanoveného uživatelem.





Umělá inteligence dnes

- Jazykové modely (ChatGPT)
- Call centra, vyřizování žádostí
- Generování fotografií a videa
- Autonomní vozidla, drony, roboti
- Specializovaná jednoúčelová zařízení





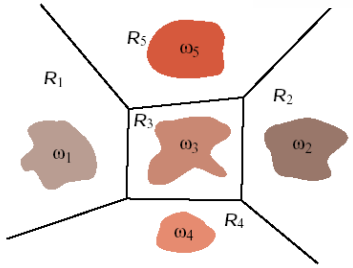


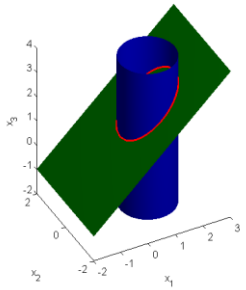


Umělá inteligence – obecné scéma

- Vnímání: přijímání a zpracování dat
  - senzory, úložiště, software
- Vyhodnocení situace
  - klasifikační problém
- Rozhodnutí
  - optimalizační problém
- Provedení
  - hardware









[Boston Dynamics]

Terminologie

- AI jako vědní obor
- Robotika
- Strojové učení (ML)
- Klasifikátory, rozpoznávání, analýza dat
- Kybernetika

Umělá inteligence

- **OBECNÁ**
- **SPECIÁLNÍ**

Základní odlišnost je v obecnosti zadání
a ve velikosti oblasti přípustných rozhodnutí

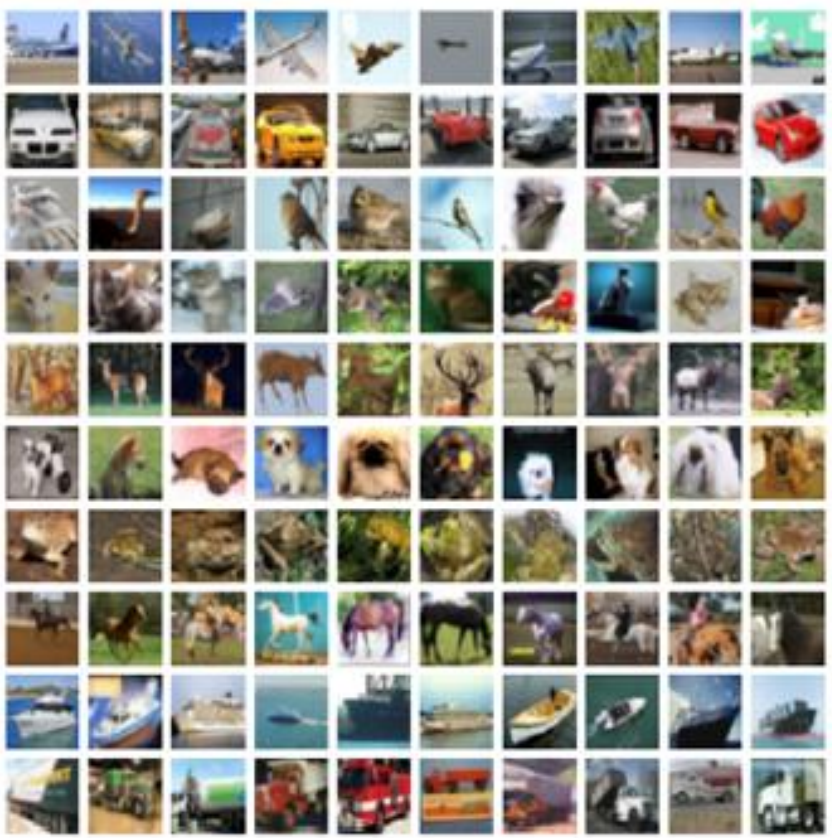
Strojové učení

Volba proměnných, typu a parametrů účelové funkce

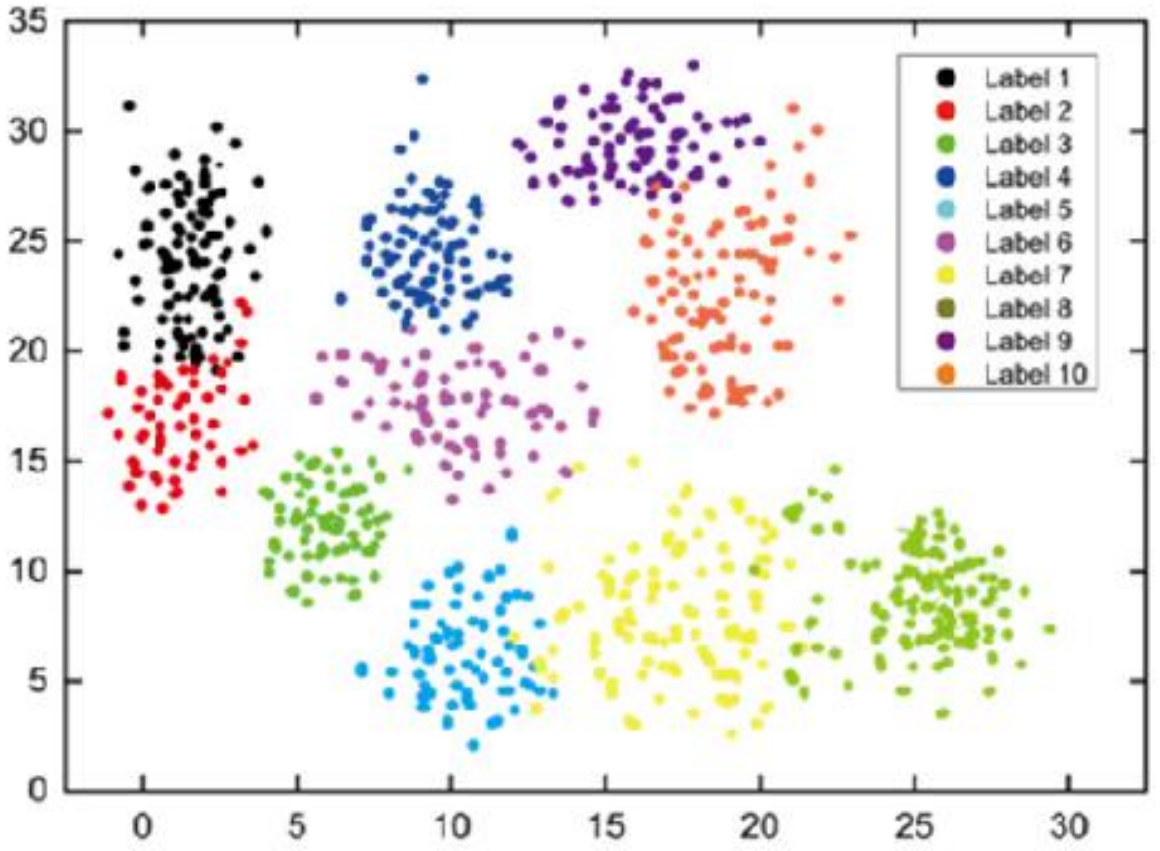
- **Klasické metody**: proměnné definuje uživatel (handcrafted features), parametry (někdy i typ) se trénují na datech

- **Hluboké sítě a podobné metody**: proměnné se určují trénováním (learned features)

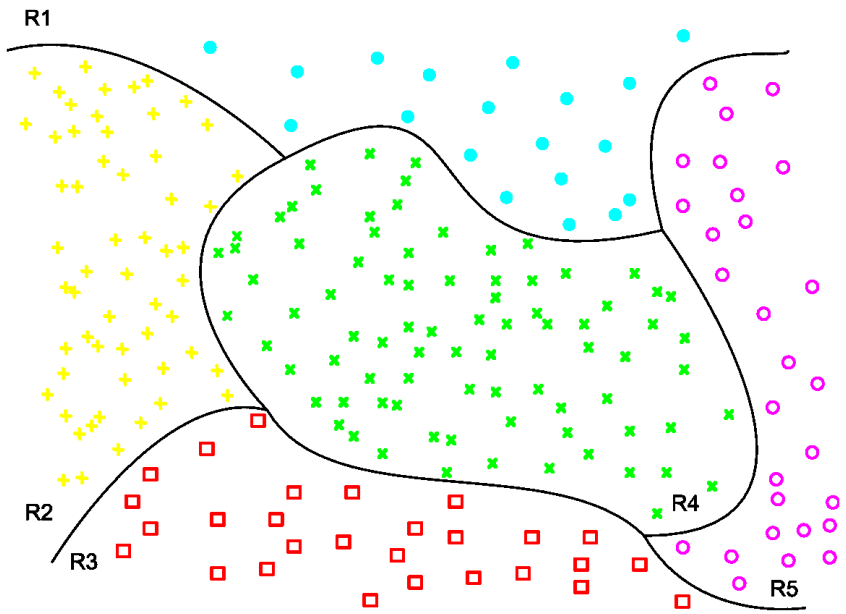
Object space and feature space

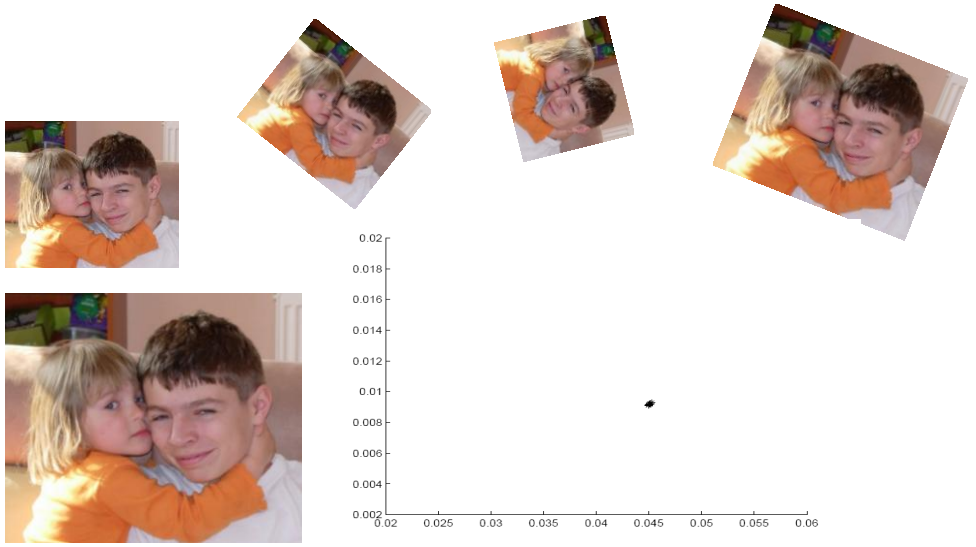


mapping



Handcrafted vs. Learned features





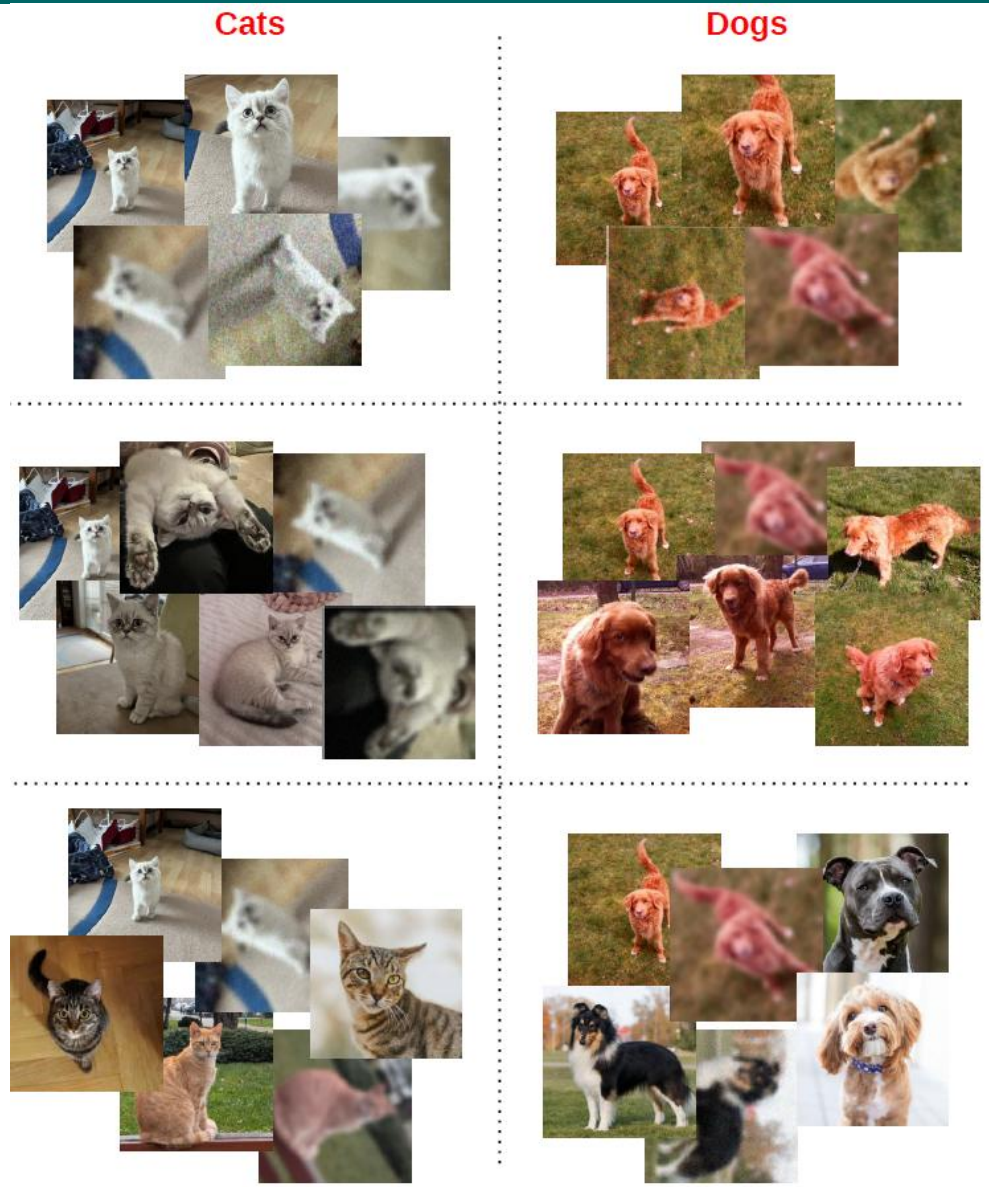




Handcrafted vs. Learned features

- Teoretický vhled
- Interpretovatelnost
- Model variací uvnitř tříd
- Závislost na datech
- Hledání příčin chyb
- Trénovací množiny
- Konkurenční nebo komplementární přístupy?

Crucial role of intra-class variations



- Single image in various versions (1)
- Single cat/dog in various positions (2)
- Various cats and dogs (3)

The intra-class variations determine the technique



- Exact model of the variations exist
- „Handcrafted“ global features invariant to them
- Traditional approach since 1970‘s

- We still have a model but it is too complex
- Problem with visibility
- „Handcrafted“ local invariants
- Since 1990‘s
- Deep learning also possible

- No general model
- Learning by examples
- Deep learning by CNN (since 2010)
- Equivariant and hybrid networks (since 2020)

Why is automatic object recognition difficult?

- Intra-class variations
- Atypical outliers
- The images of low quality, distorted, corrupted
- The objects are not in a “standard” position





Why is object recognition easy for humans?

- Learning by examples
- Life-time experience
- Fusion vision with other senses
- High robustness to (some) imperfections
- Ability to maintain intra-class variations



Development of **deep learning** methods

(2010's – now)

Netechnické aspekty AI

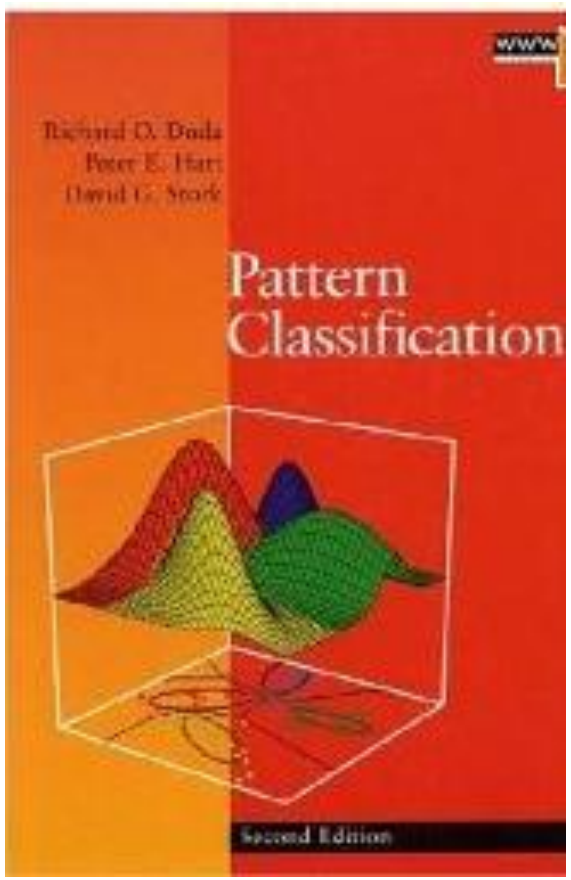
- Právní
- Sociální
- Etické
- Psychologické
- Politické

Literatura

Duda, Hart, Stork:

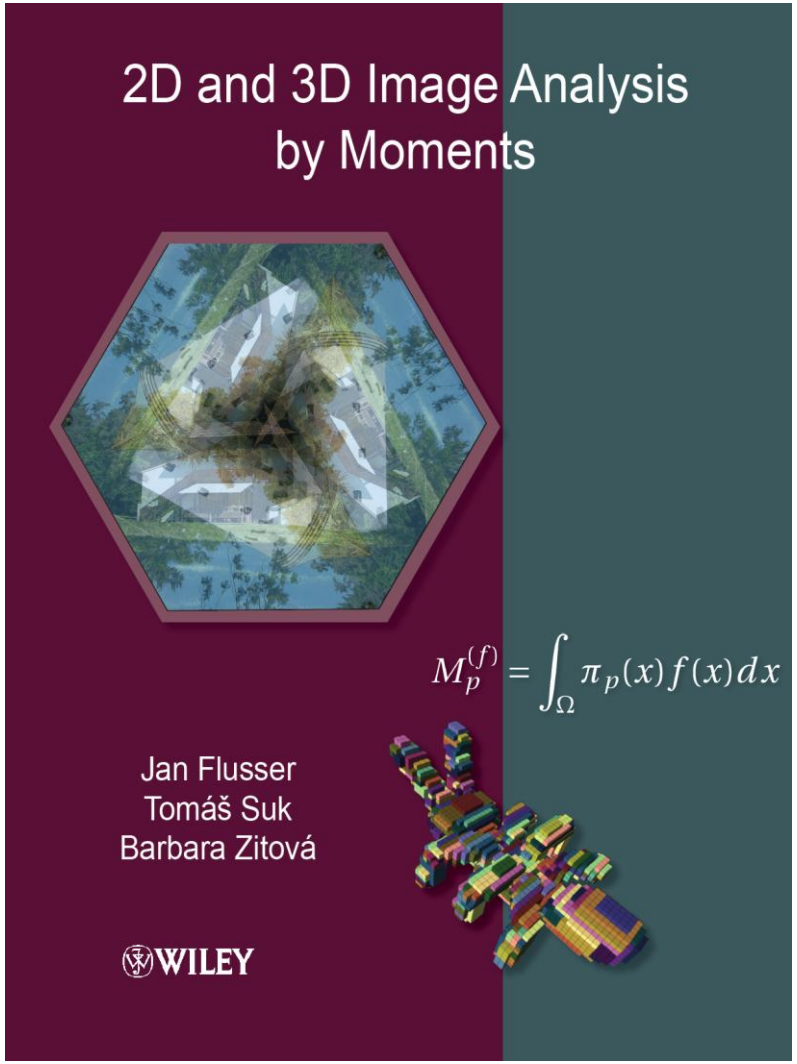
Pattern Classification,

2nd ed., Wiley, 2001



Literatura

Kapitoly 1 a 2 – stručný přehled



Pattern Recognition

- Recognition (classification) = assigning a pattern/object to one of pre-defined classes
- Statistical (feature-based) PR - the pattern is described by features (n-D vector in a metric space)

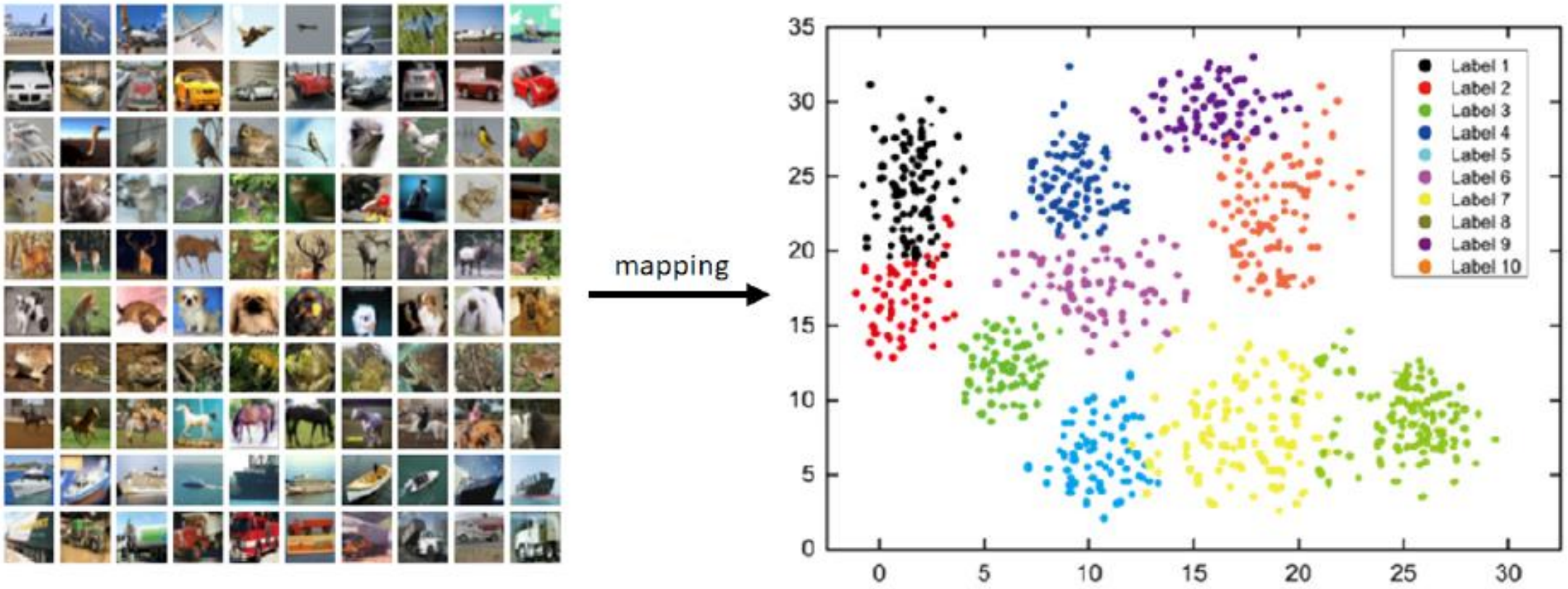
Pattern Recognition

- **Supervised PR**
  - **– training set available for each class**
- **Unsupervised PR (clustering)**
  - **– training set not available, No. of classes may not be known**

Desirable properties of the training set

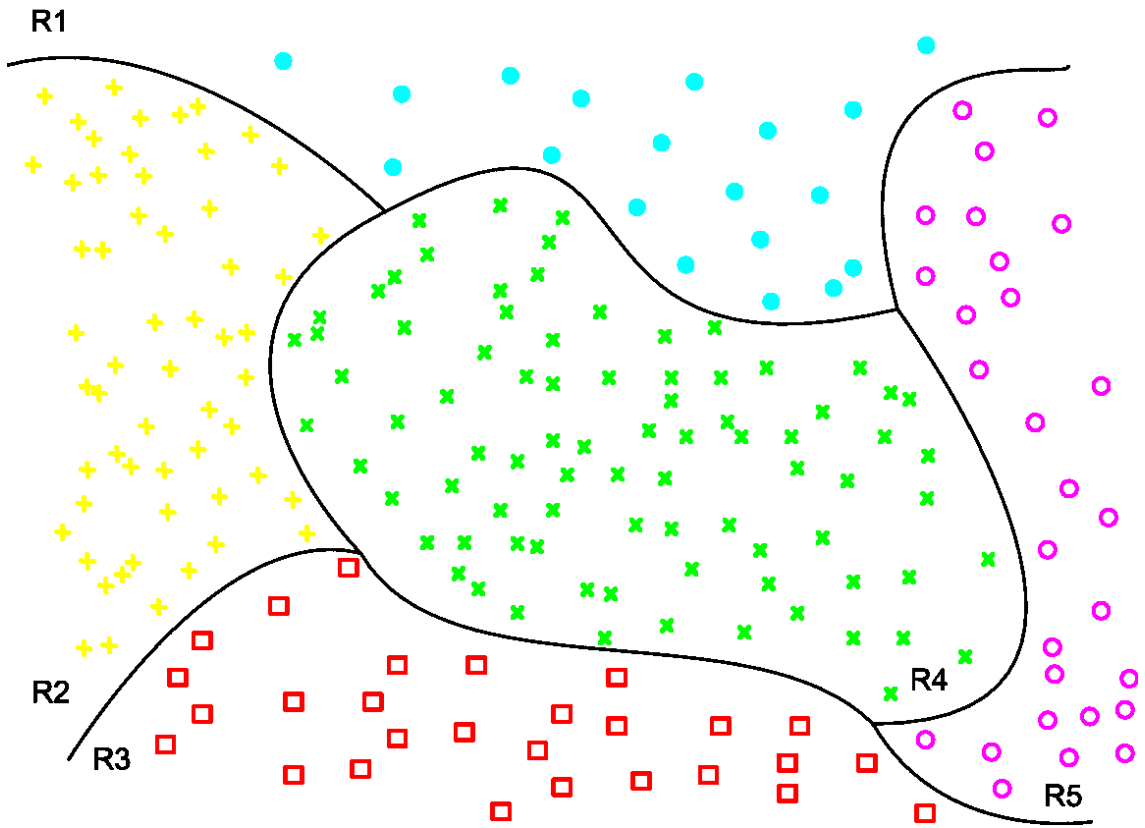
- It should contain typical representatives of each class including intra-class variations
- Reliable and large enough
- Should be selected by the domain experts

Image and feature space



Classification rule setup

Equivalent to a partitioning of the feature space



Independent of the particular application

Object Recognition System

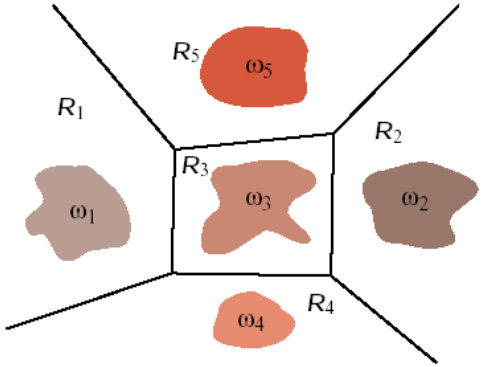
Image acquisition

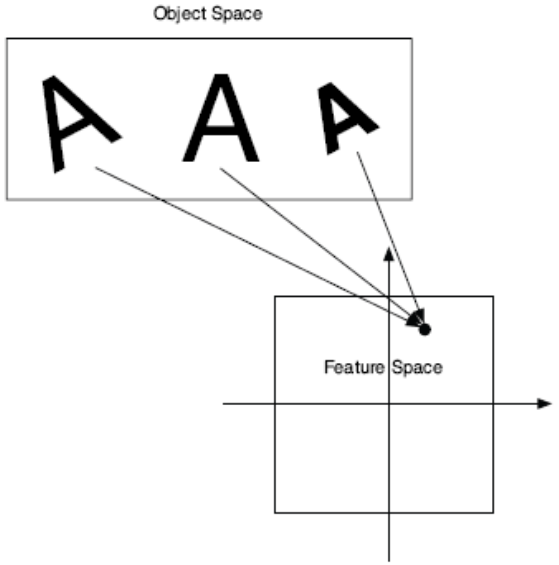
Preprocessing

Object detection



Computing of the features





Classification

Class label

Desirable properties of the features

- **Invariance**
- **Discriminability**
- **Robustness**
- **Efficiency, independence, completeness**

Example: TRS

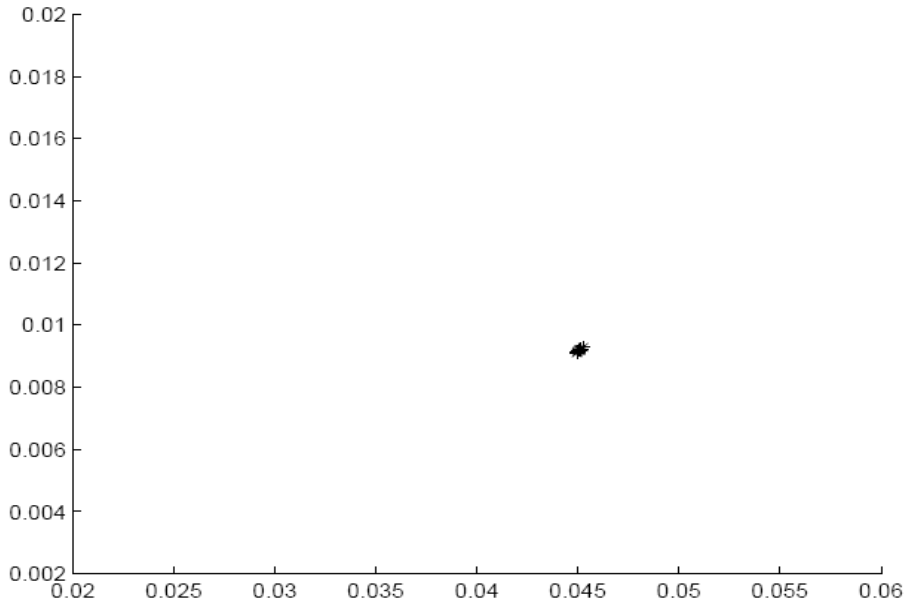




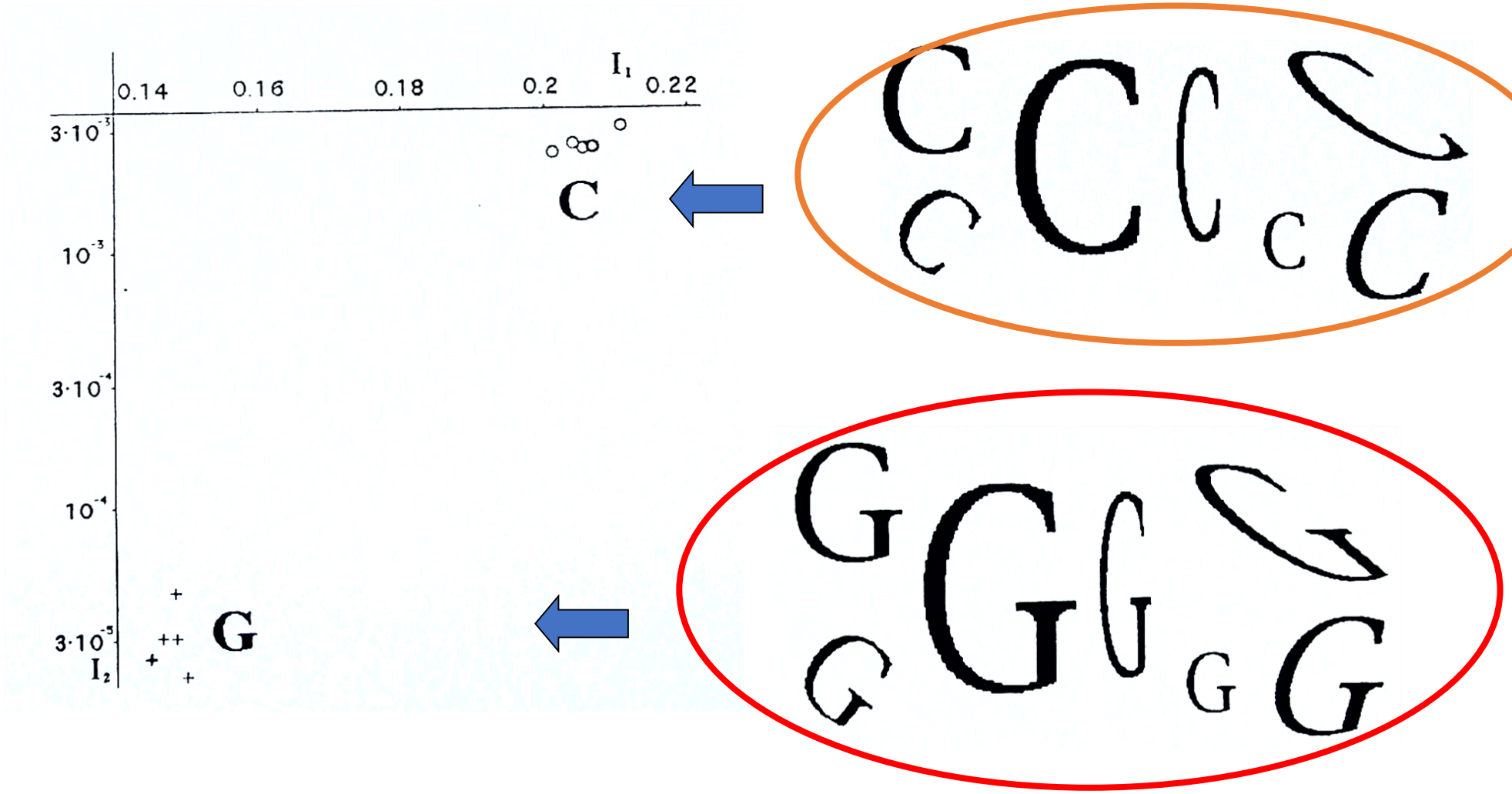







0.02
0.018
0.016
0.014
0.012
0.01
0.008
0.006
0.004
0.002
0.02
0.025
0.03
0.035
0.04
0.045
0.05
0.055
0.06

Discrimination power



What are invariants?

Invariants are **functionals** defined on the image space such that

$$I(D(f)) = I(f) \quad \text{for all admissible } D$$

$$I(D(f)) = D'(I(f)) \text{ - } \textit{equivariance}$$

Groups actions, orbits, equivalence

- Let *D* form a group D
- Group action D x I → I
- Orbit *D(f)*
- Orbits are *equivalence classes of I/D*

Orbits are “**classes**” of our classification problem

Complete invariants

Both factor spaces are equal iff

$I(f_1) \neq I(f_2)$ for any $f_1 \neq D(f_2)$

Classification and decision making

- Sensing, measuring the features



- Classification - Minimization of error probability





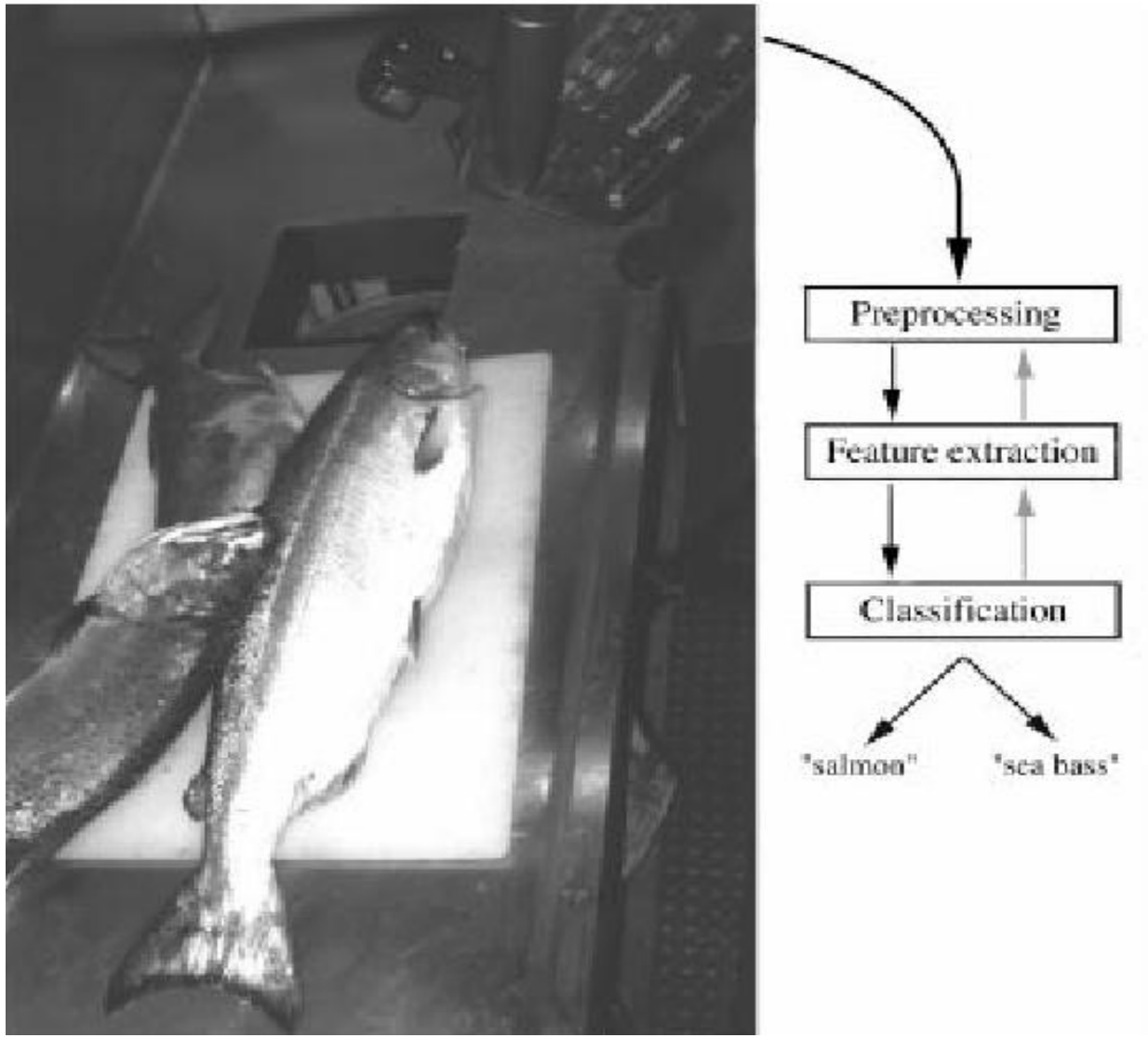


- Decision – Minimization of a loss function

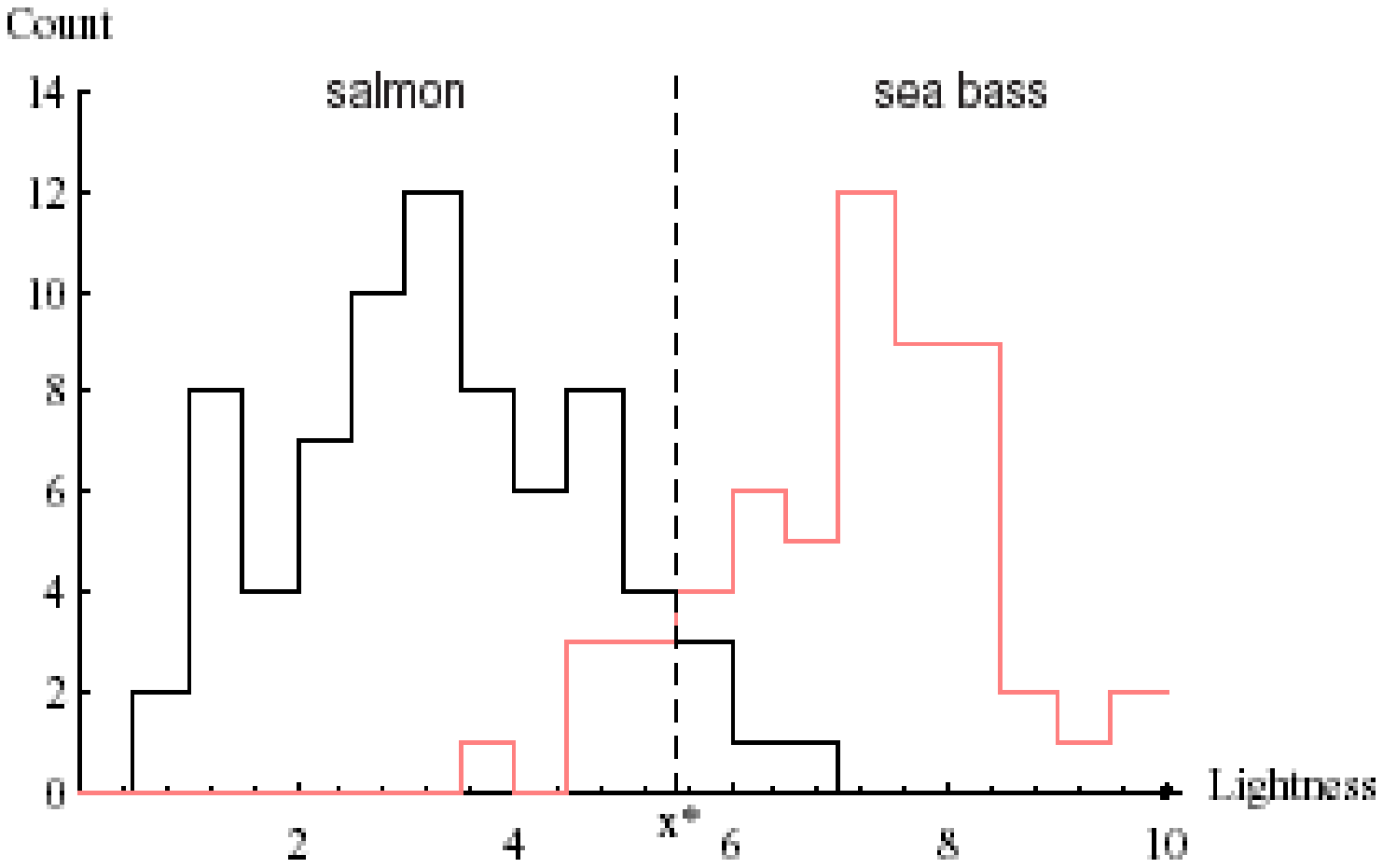


- Action - Hardware

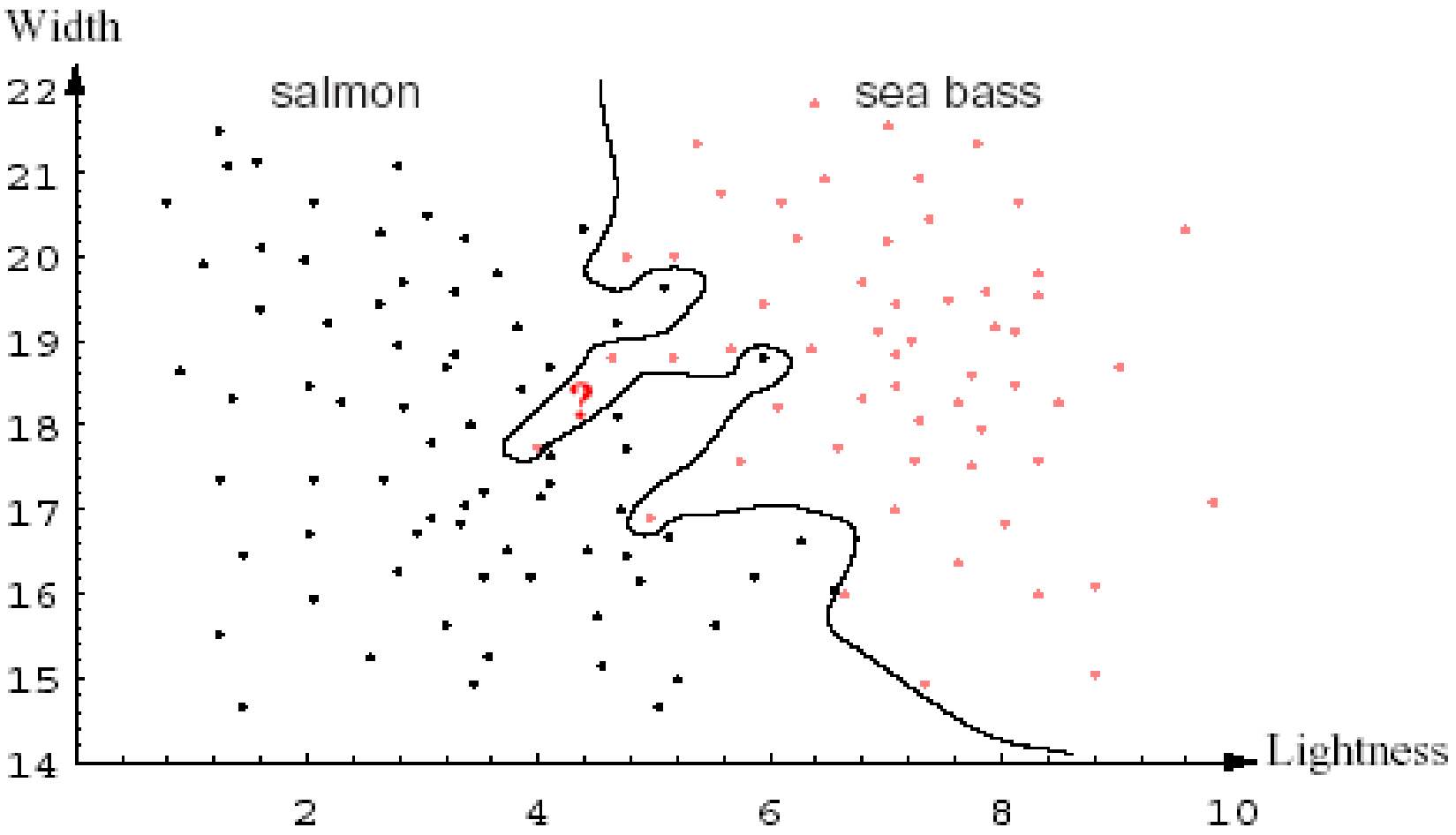
Example – Fish classification



The features: Length, width, brightness



2-D feature space



Empirical observation

- For a given training set, we can have several classifiers (several partitioning of the feature space)

Empirical observation

- For a given training set, we can have several classifiers (several partitioning of the feature space)
- The training samples are not always classified correctly

Empirical observation

- For a given training set, we can have several classifiers (several partitioning of the feature space)
- The training samples are not always classified correctly
- We should avoid overtraining of the classifier

Formal definition of the classifier

- Each class is characterized by its discriminant function *g(x)*
- Classification = maximization of *g(x)*

  Assign *x* to class *i* iff

$$g_i(\mathbf{x}) > g_j(\mathbf{x}) \qquad \forall j \neq i$$

- Discriminant functions define decision boundaries in the feature space

Minimum distance (NN) classifier

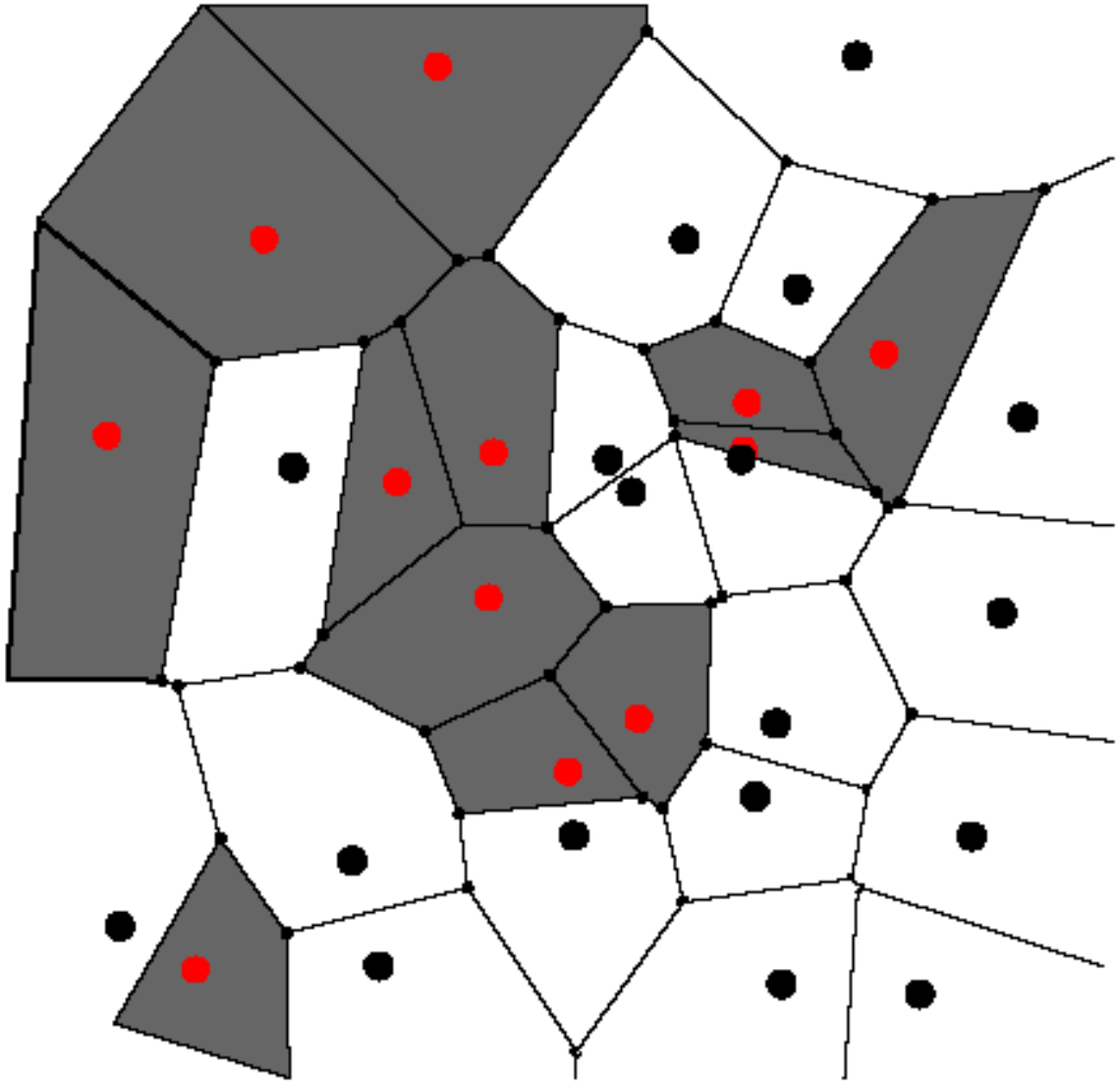
- Discriminant function

$$g_i(\mathbf{x}) = -d(\mathbf{x}, \omega_i)$$

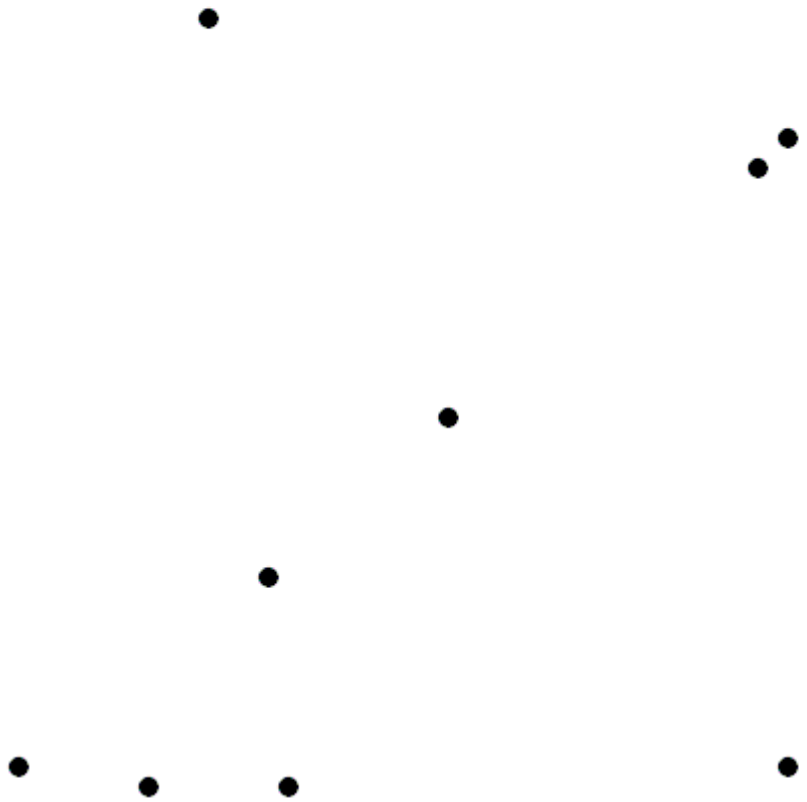
- No training/optimization
- Various definitions of $d(\mathbf{x}, \omega_i)$

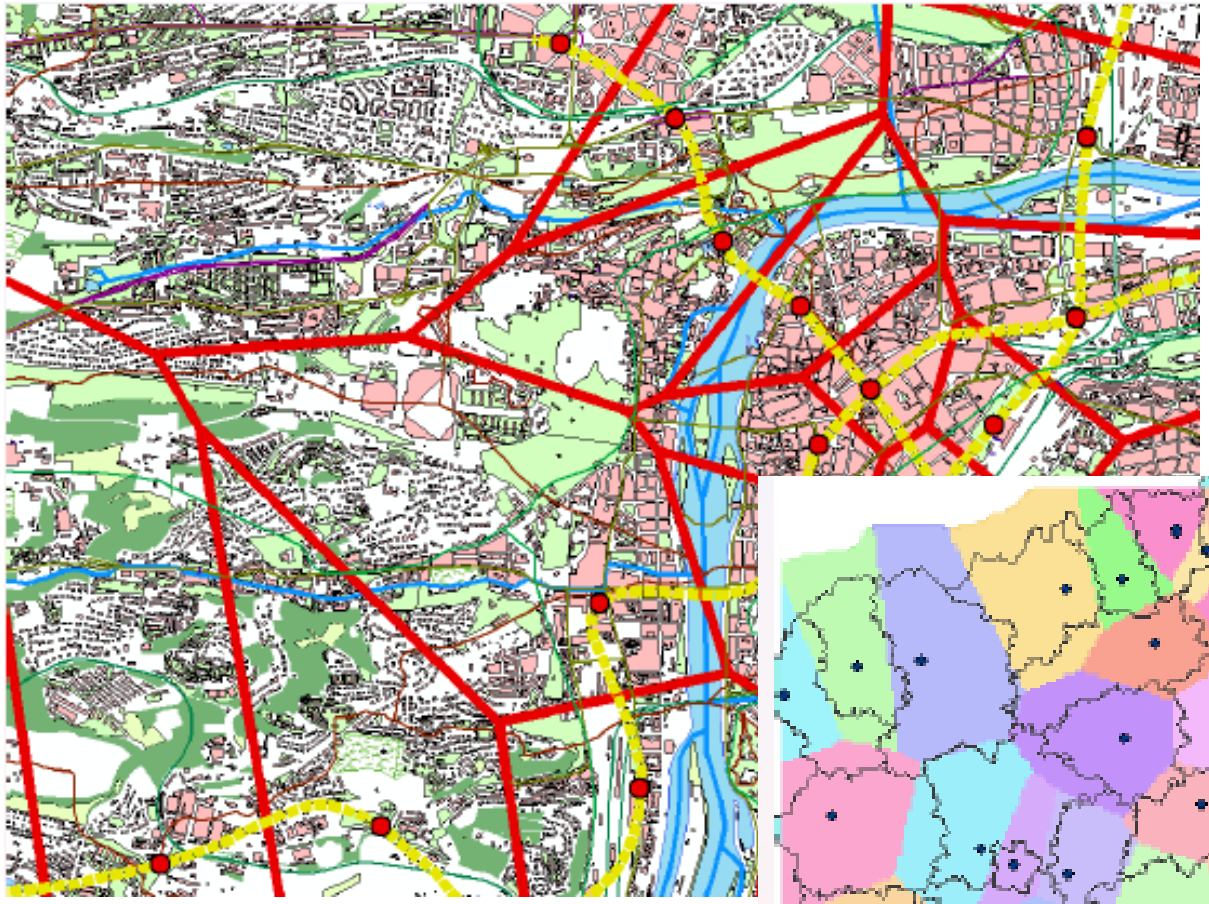
- One-element training set →

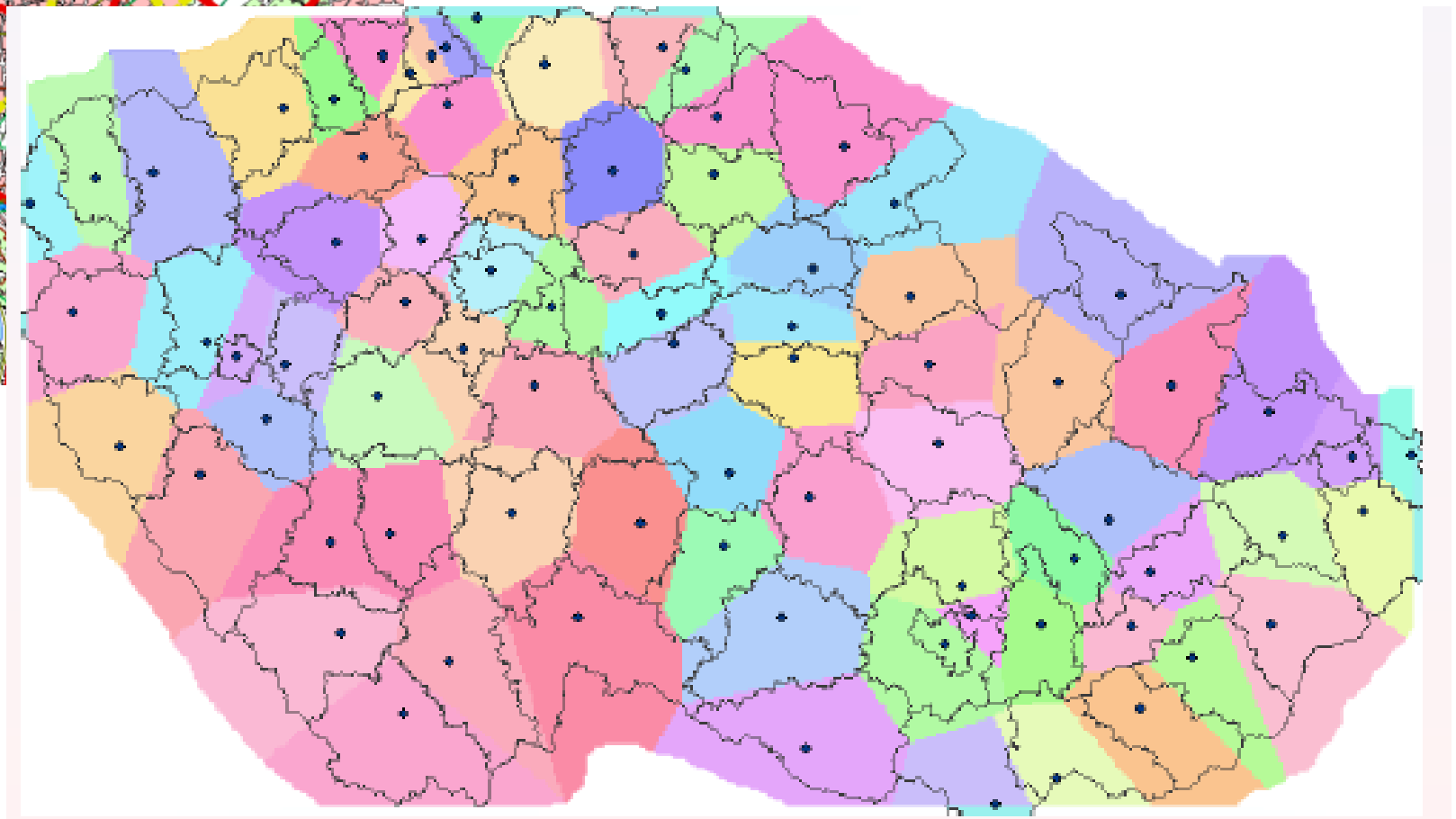
Voronoi polygons



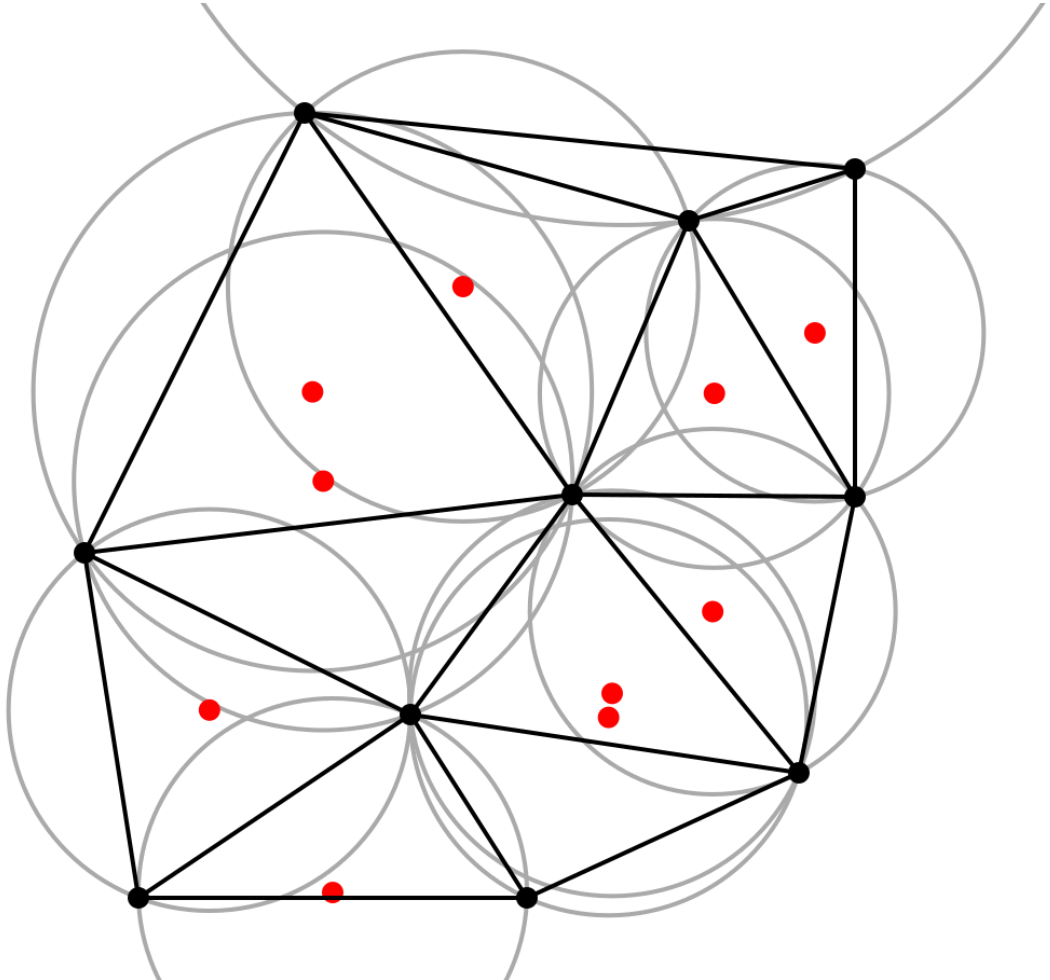
Voronoi polygons

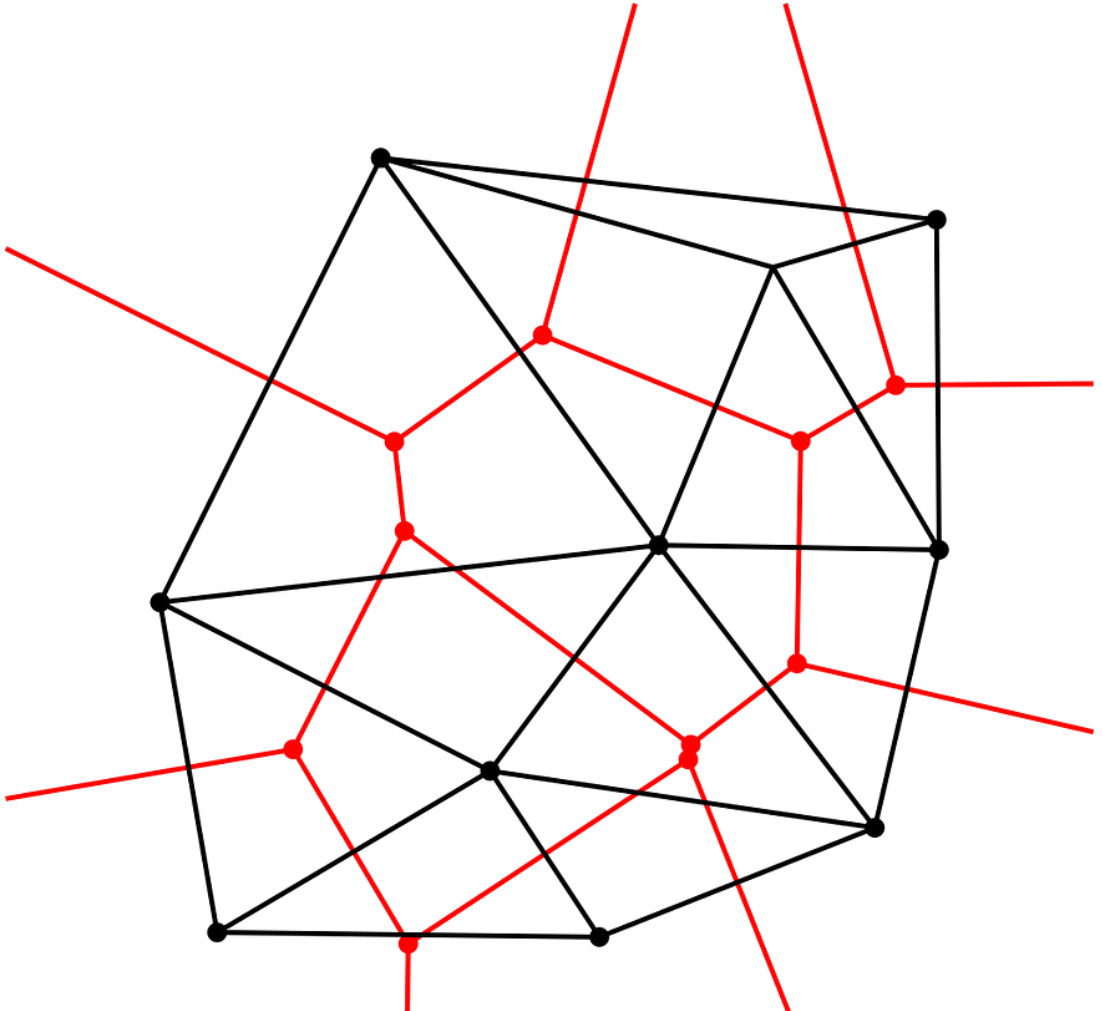






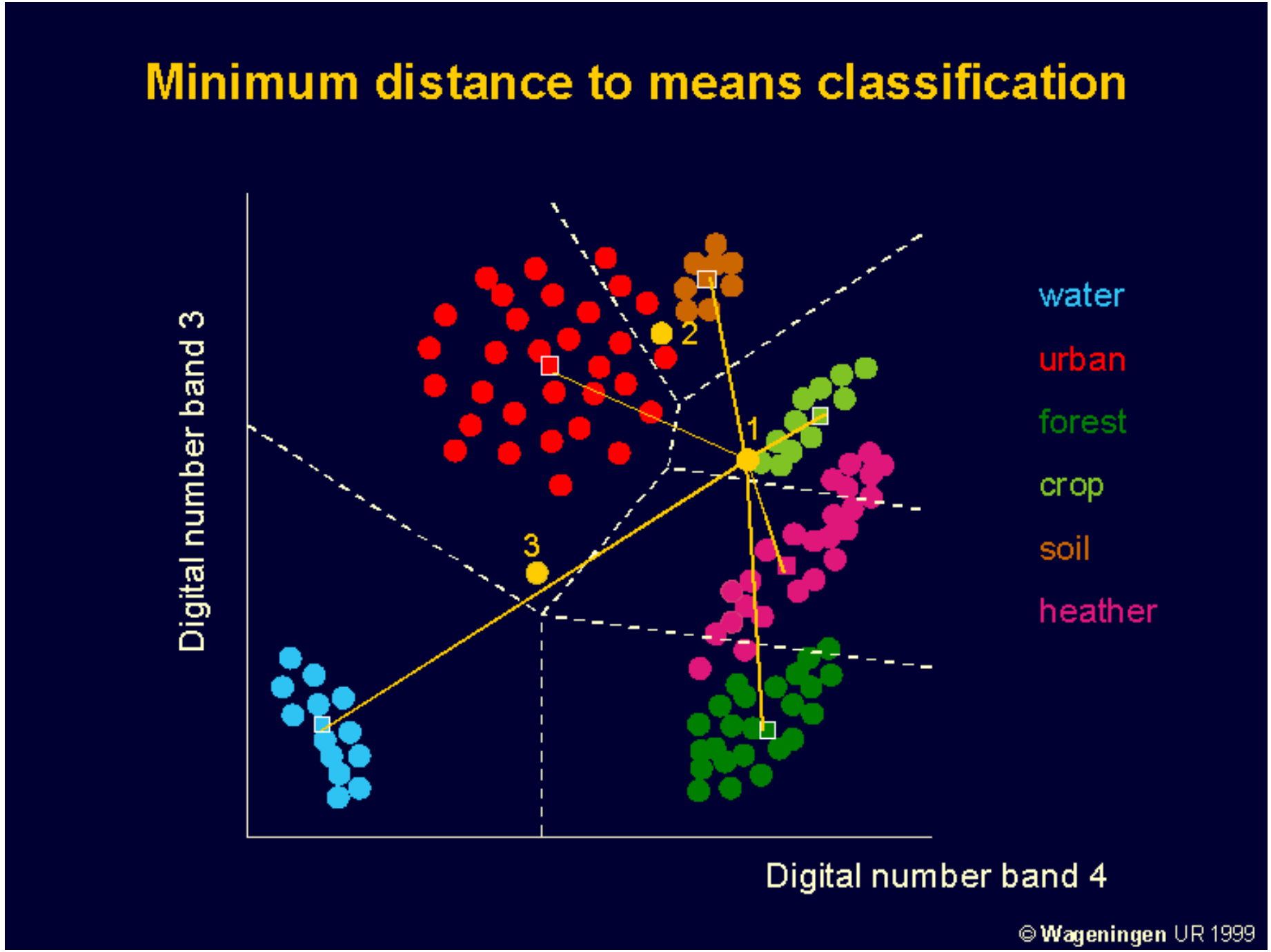
Off-topic: Delaunay triangulation

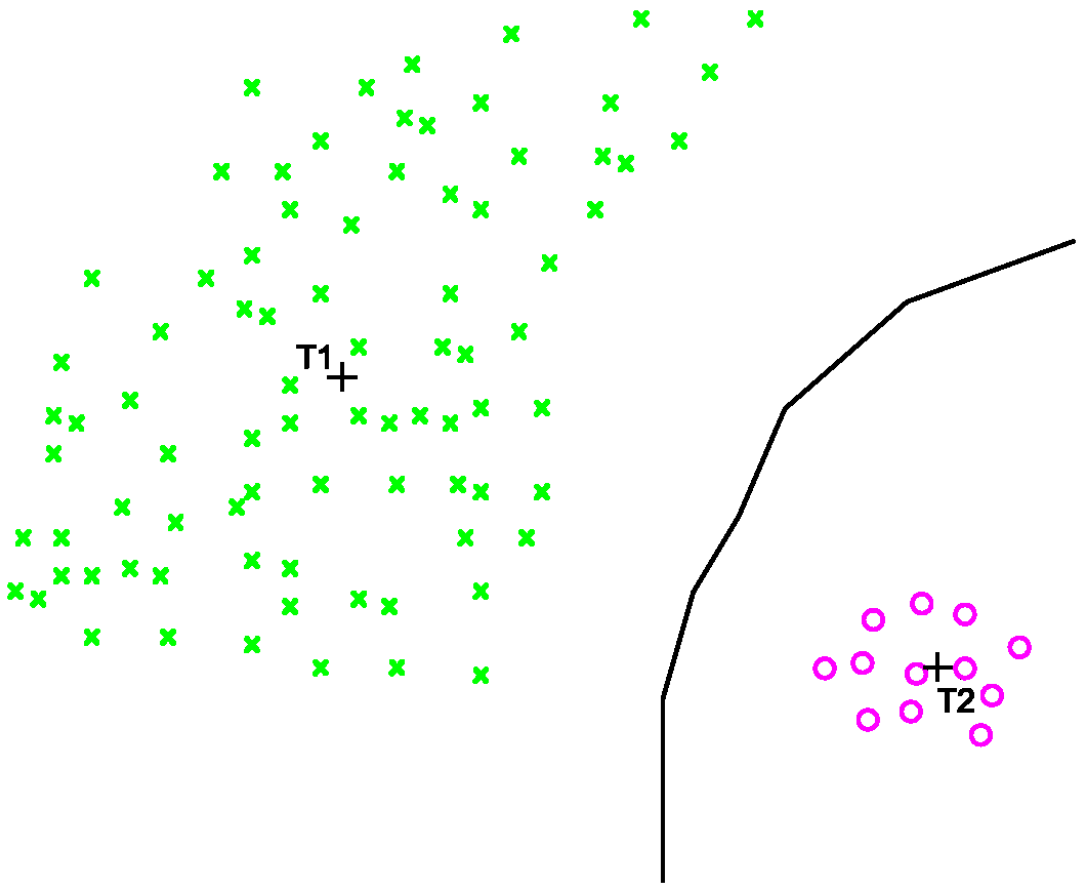


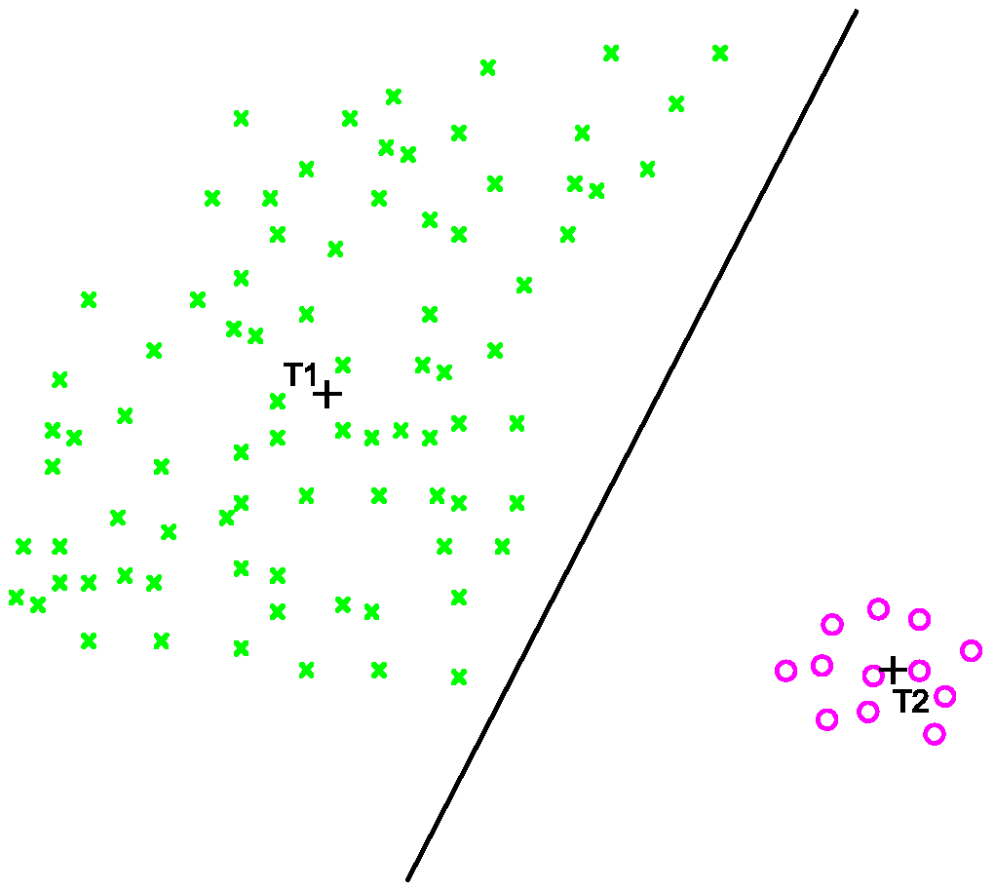


Minimum distance (NN) classifier

- Depending on $d(\mathbf{x}, \omega_i)$, NN classifier may not be linear
- NN classifier is sensitive to outliers → k-NN classifier


Minimum distance to means classification
Digital number band 3
Digital number band 4
1
2
3
water
urban
forest
crop
soil
heather
© Wageningen UR 1999

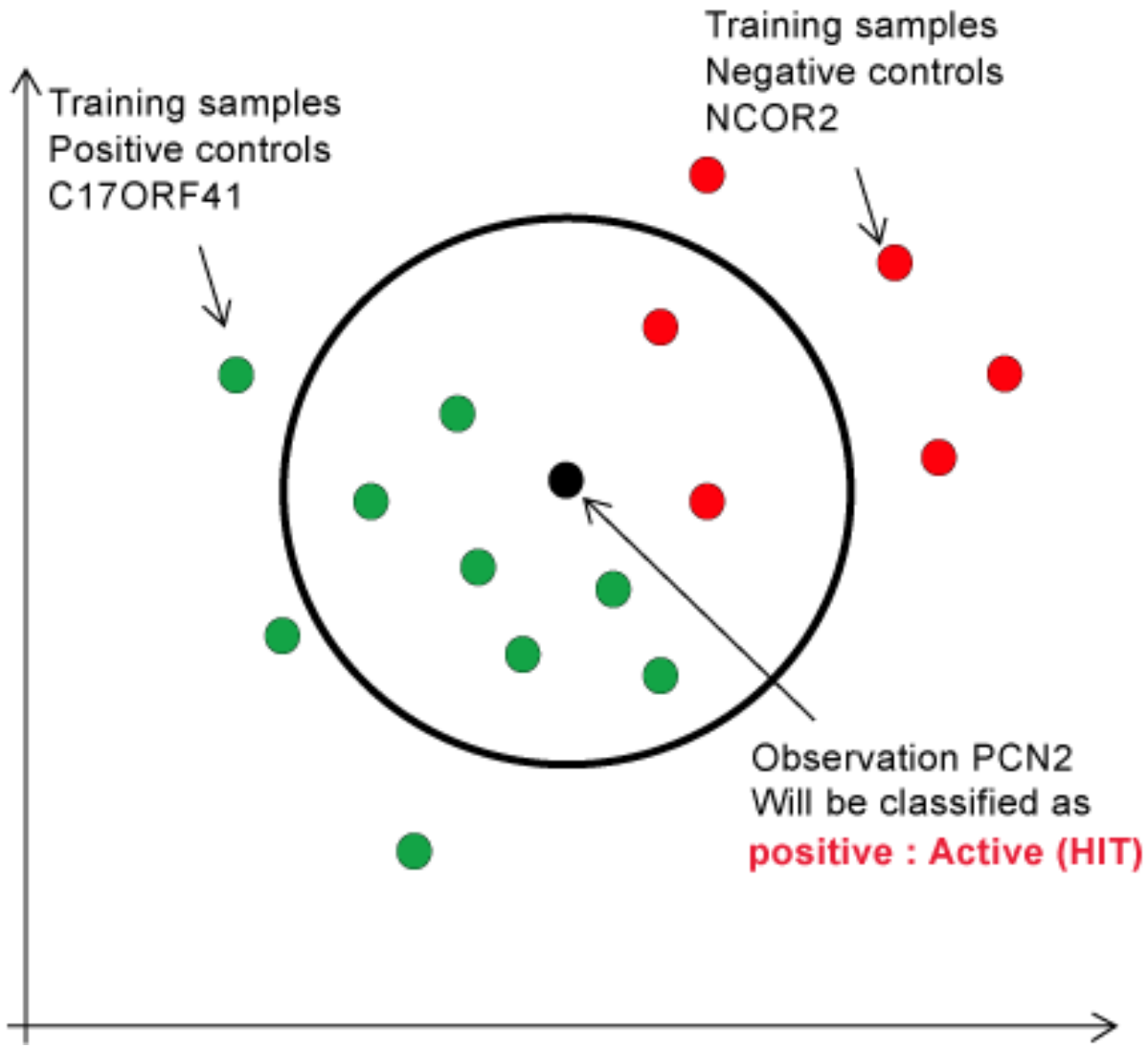

T1
T2


T1
T2

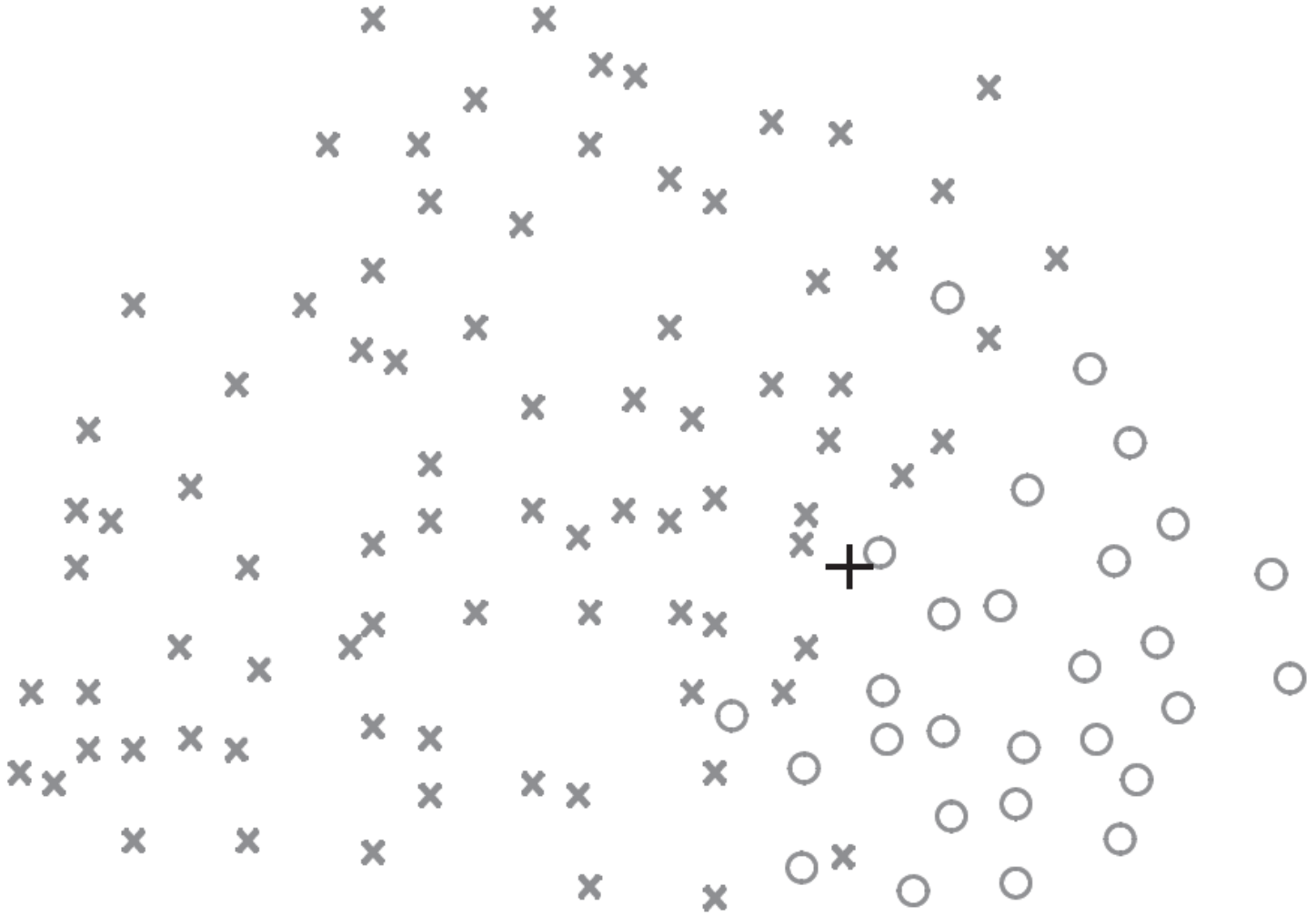
k-NN classifier

- NN classifier is sensitive to outliers → k-NN classifier
- It finds the nearest training points unless *k* samples belonging to one class has been reached

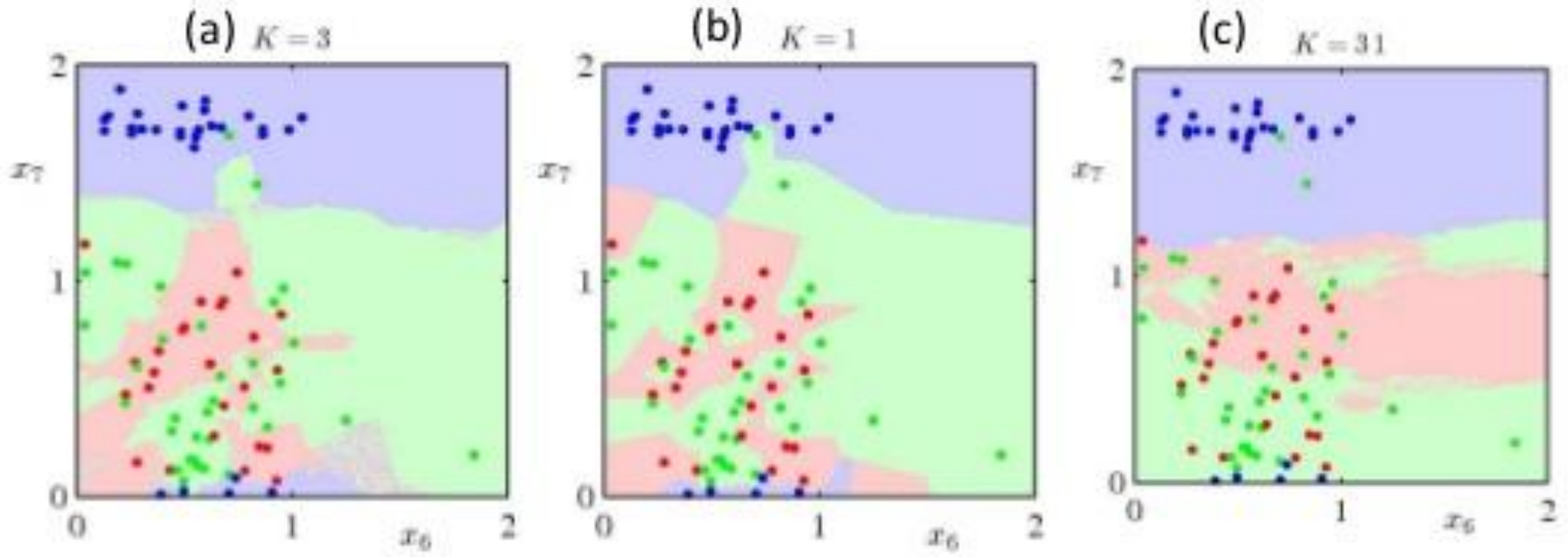
k-NN classifier



k-NN classifier



k-NN classifier



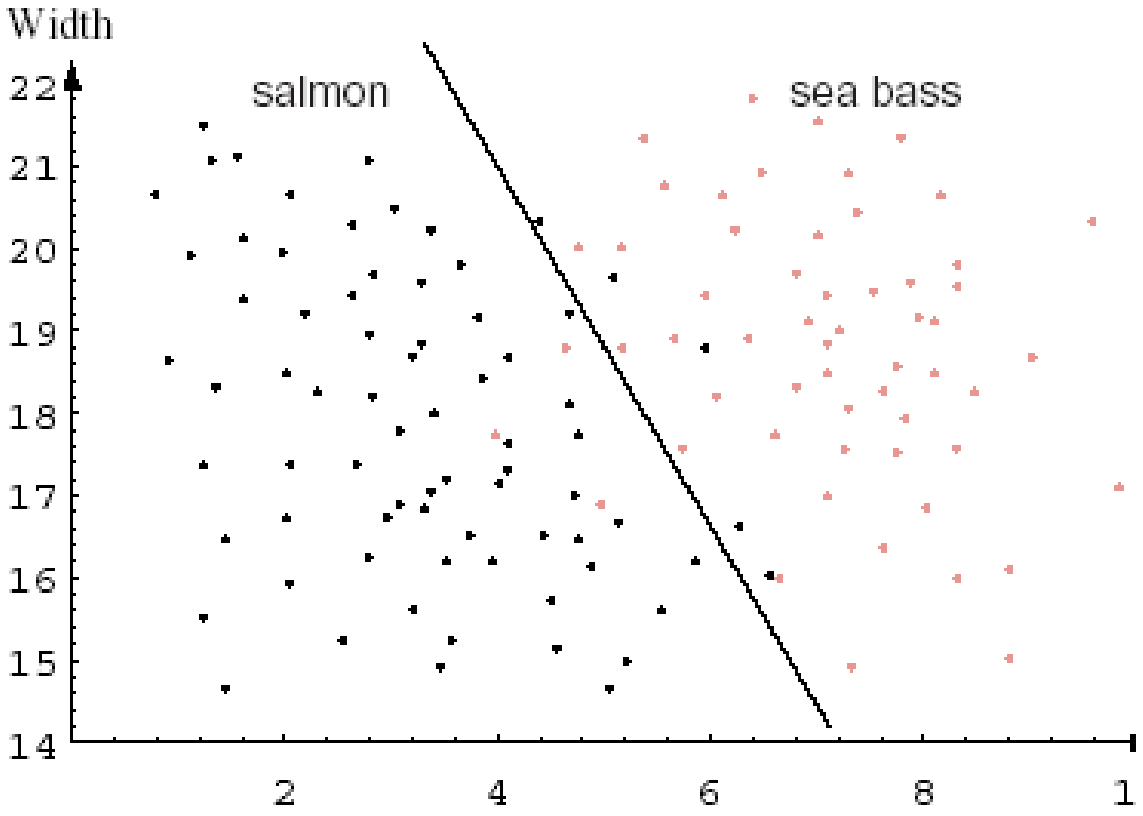
k-NN classifier

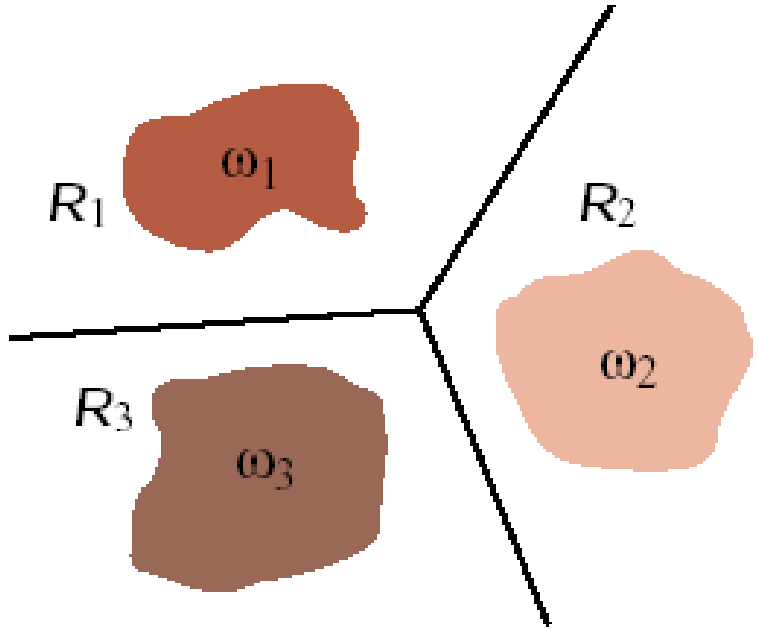
- Could be very slow, especially for high *k* and *n*

- Efficient algorithms: *n*-d tree, ball-tree, ANN (approximative)

Linear classifier

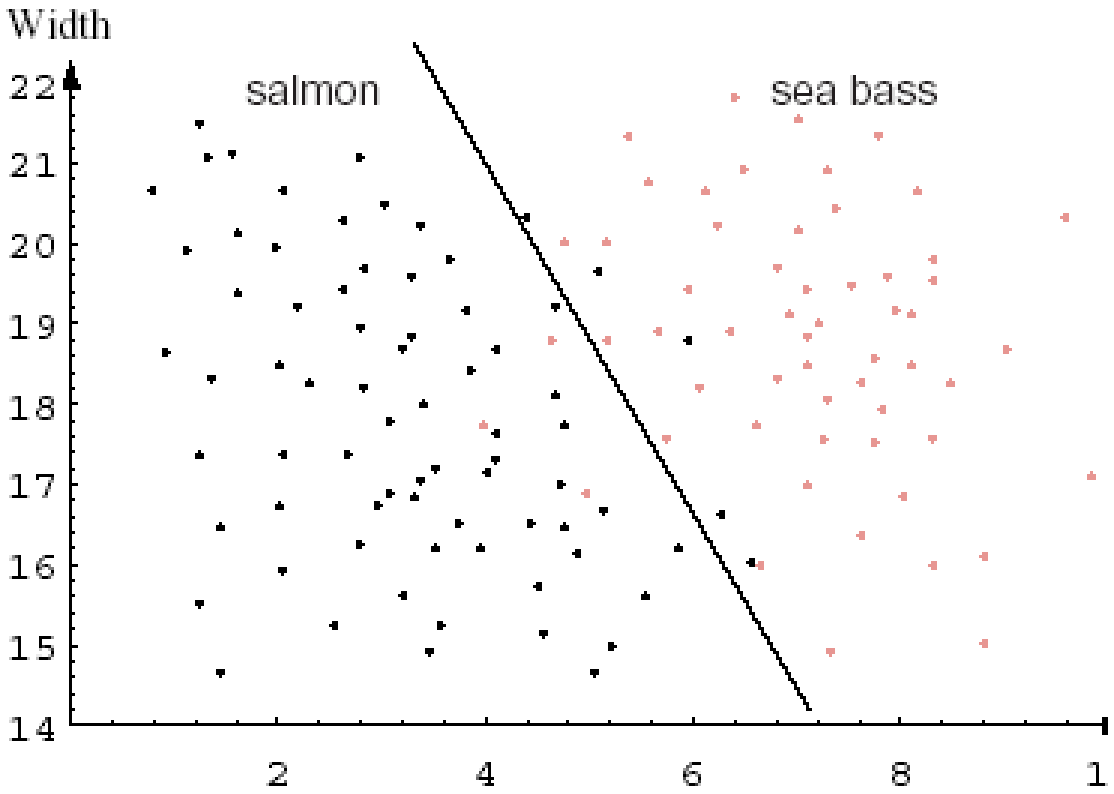
Discriminant functions g(x) are hyperplanes





Simple training algorithms

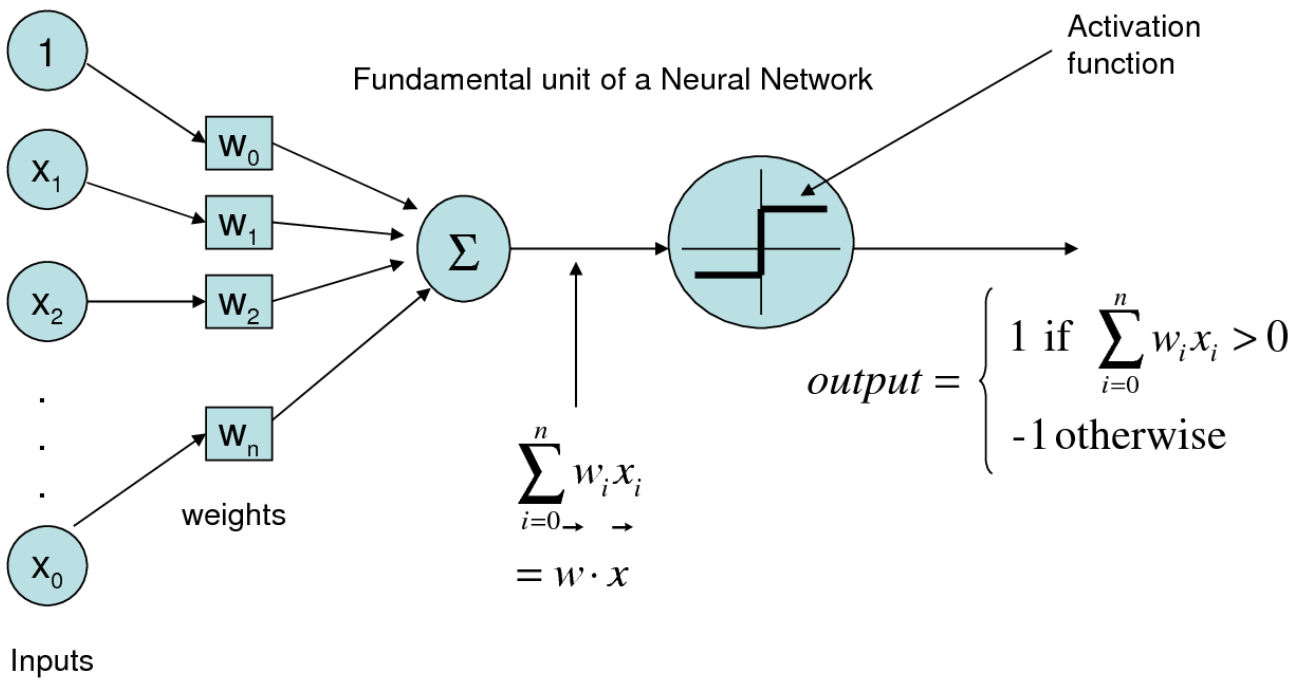
- Many possible hyperplanes
- Perceptron



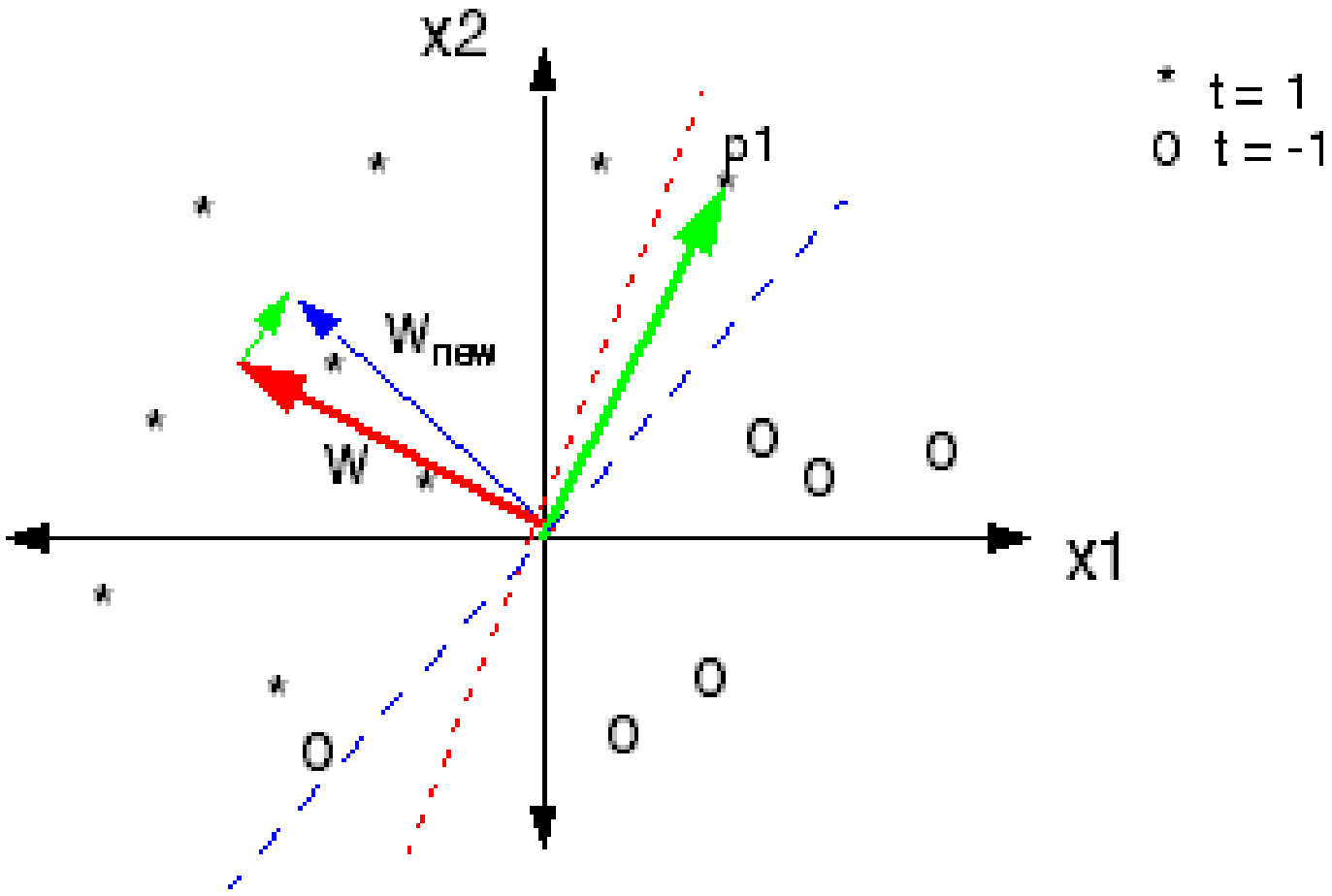
Perceptron

Artificial Neural Networks

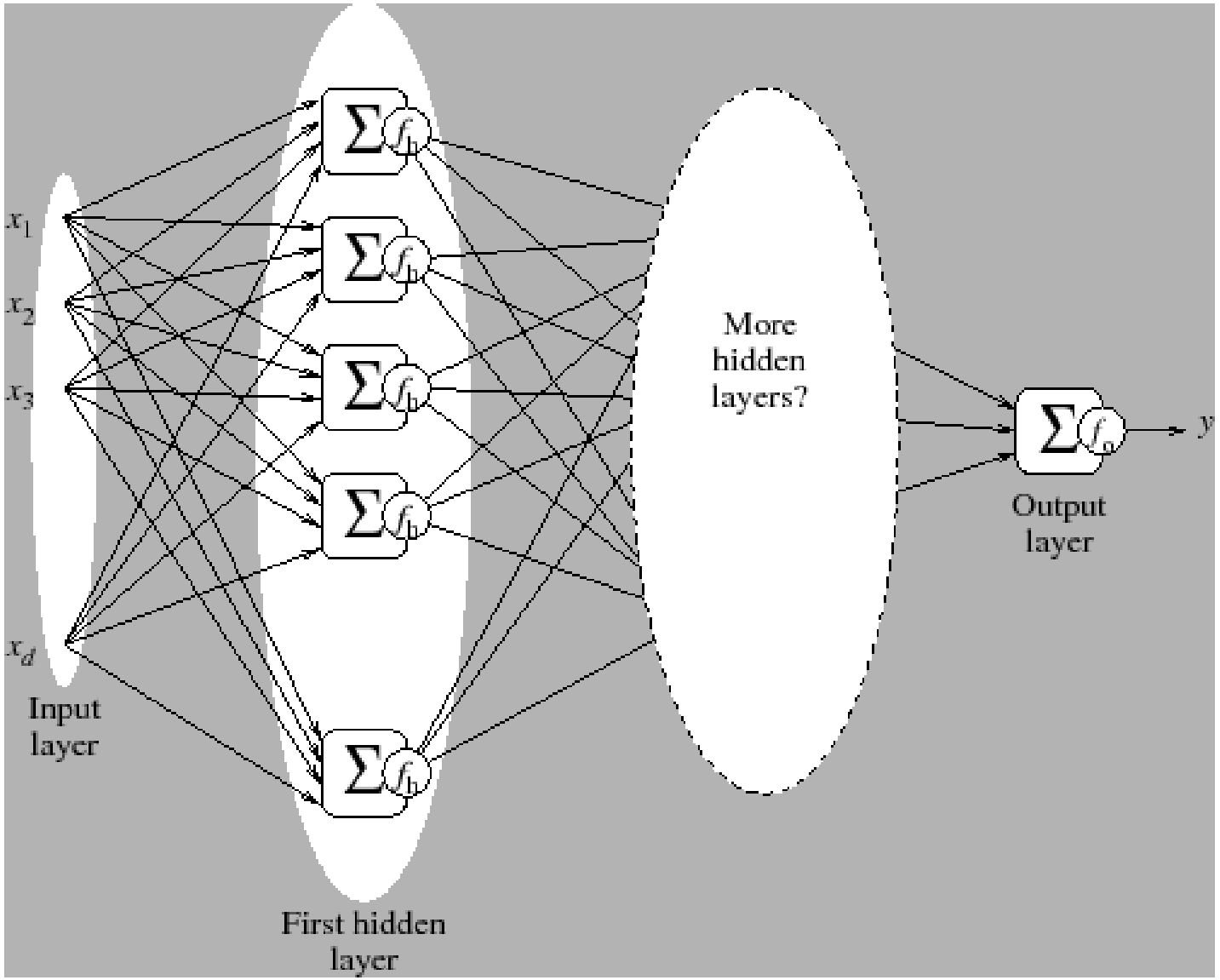
The Perceptron



Perceptron – an iterative trainig

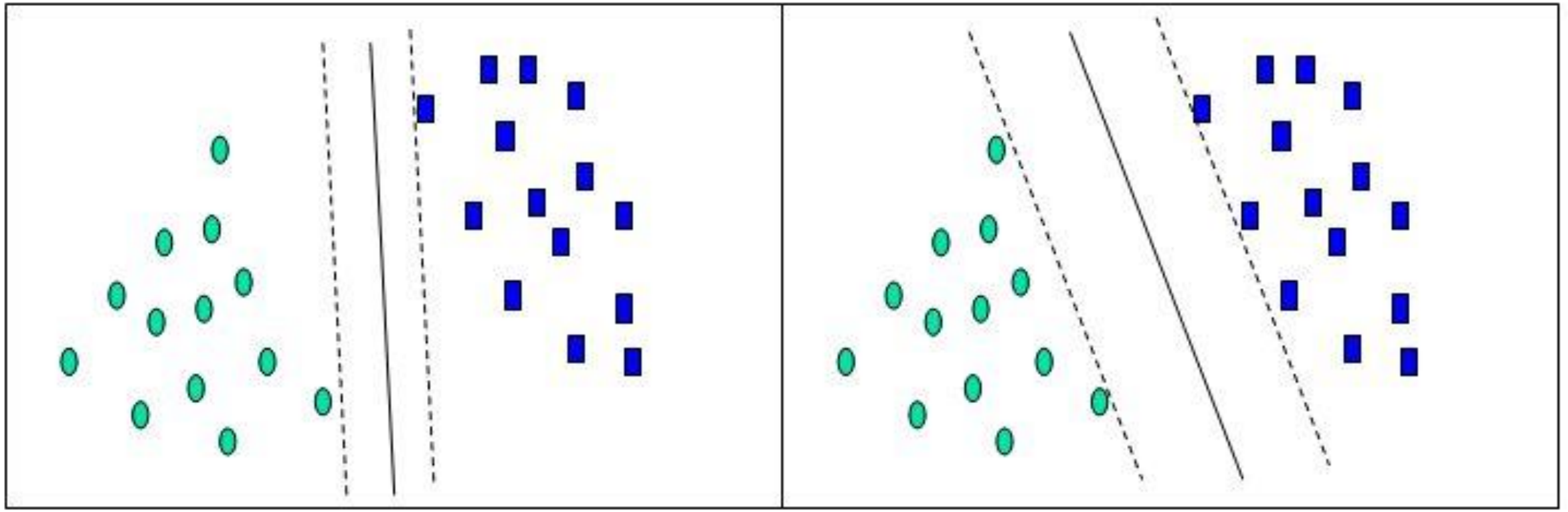


Multilayer perceptron



„Optimal“ linear classifier

Maximizing the margin



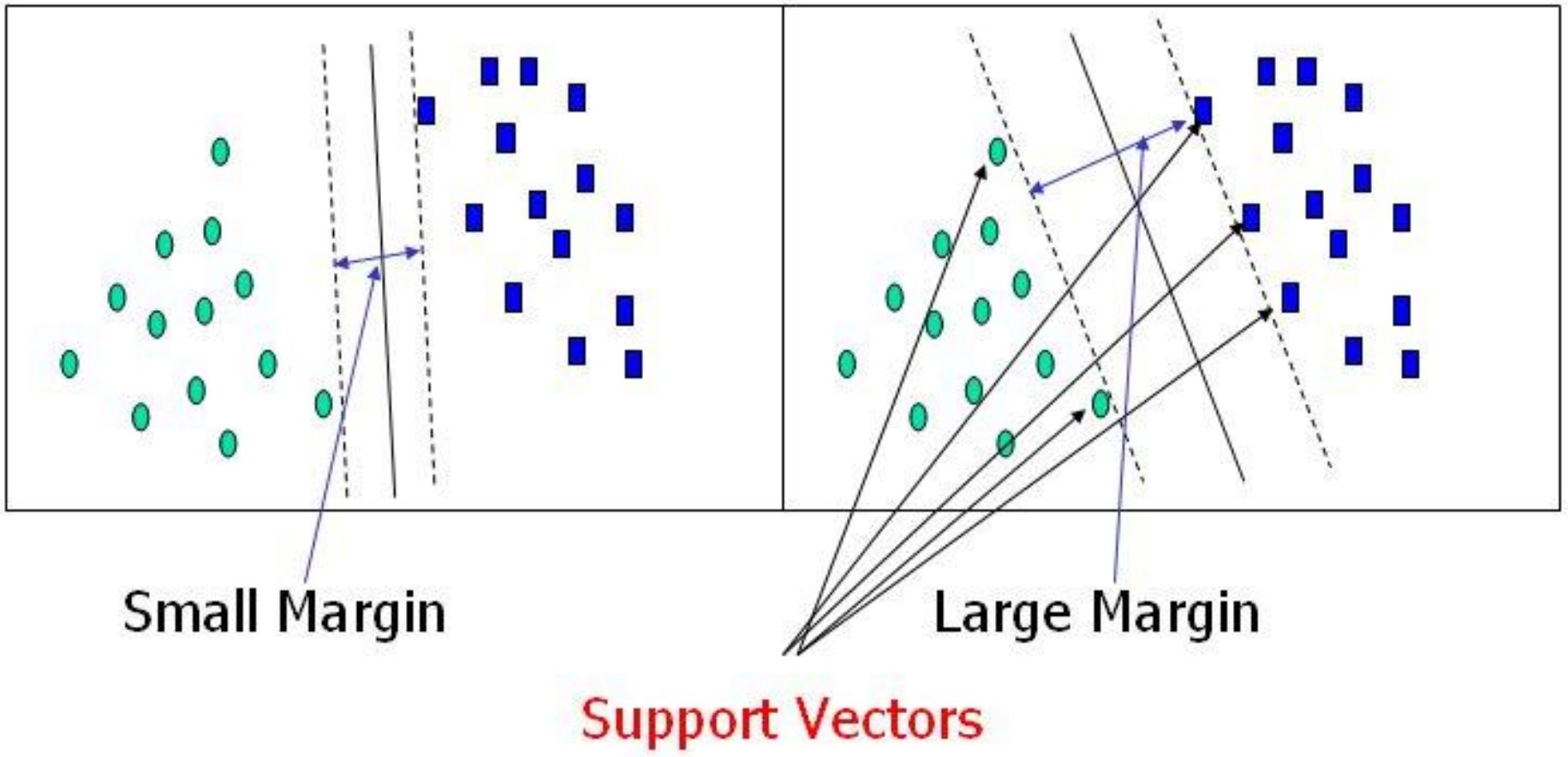
k-NN classifier

- Could be very slow, especially for high *k* and *n*

- Efficient algorithms: *n*-d tree, ball-tree, ANN (approximative)

Support vector machine (SVM)

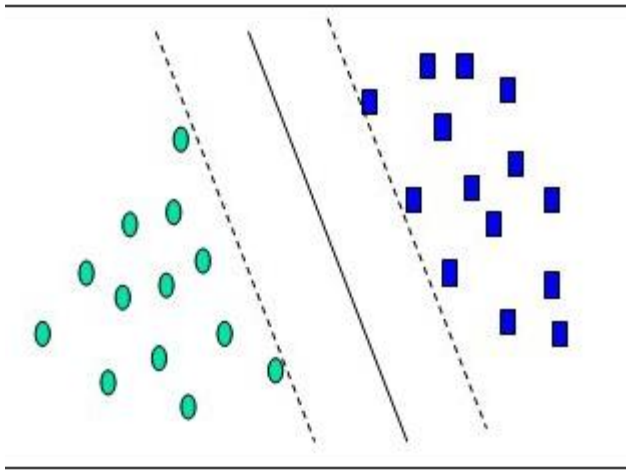
"Optimal" linear classifier



Support vector machine (SVM)

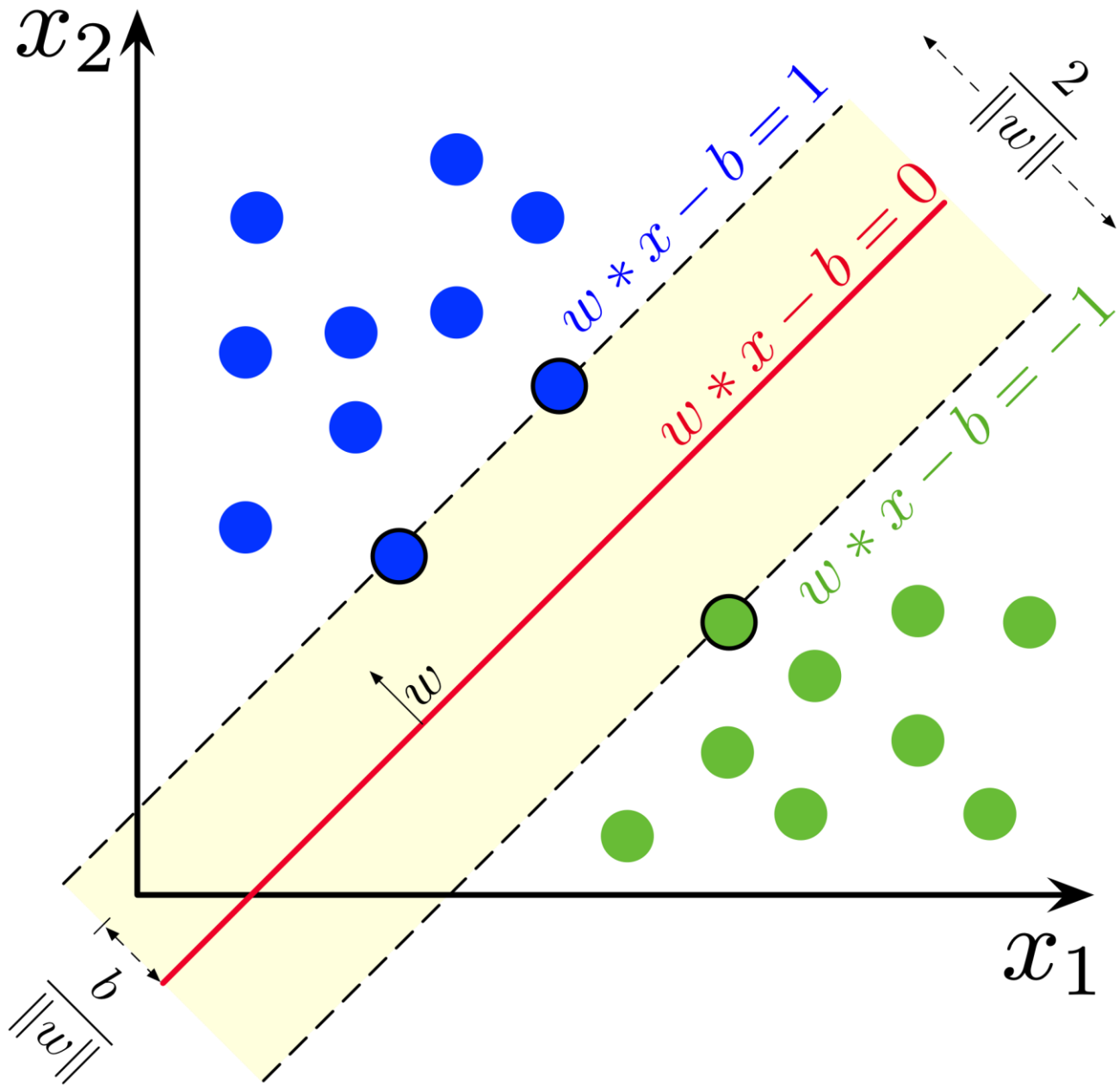
Training algorithm: Discrete optimization

- many versions

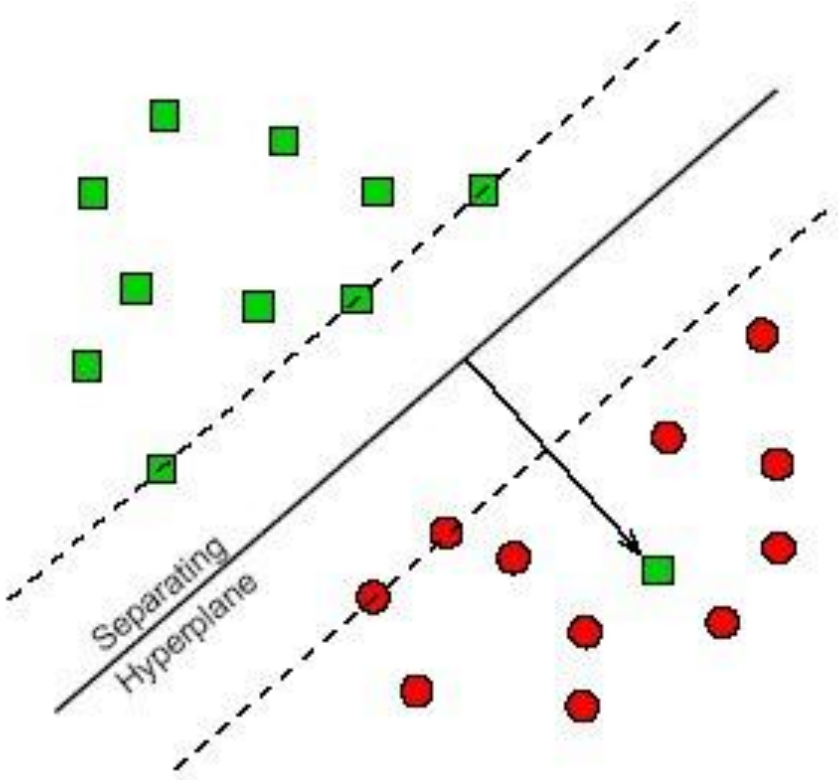


Drawbacks:

- very sensitive to outliers and noise
- does not consider the shape and the size of the training set


x_2
x_1
$w * x - b = 1$
$w * x - b = 0$
$w * x - b = -1$
w
$\frac{2}{\|w\|}$
$\frac{b}{\|w\|}$

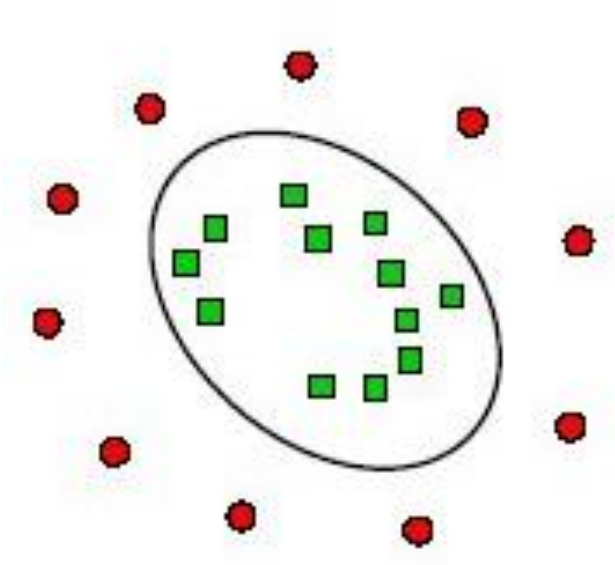
Soft margin SVM



Cost function penalizes the errors

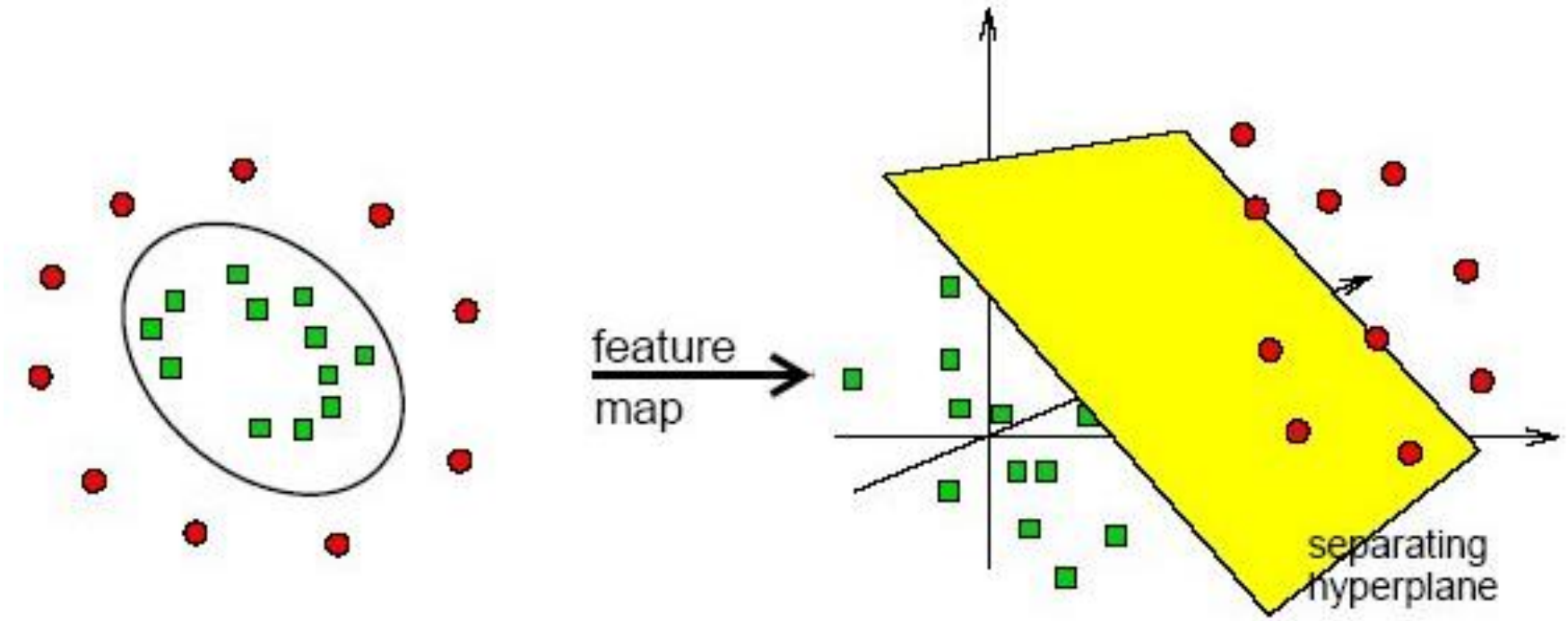
Regularization

SVM for linearly non-separable classes



How to make them linearly separable?

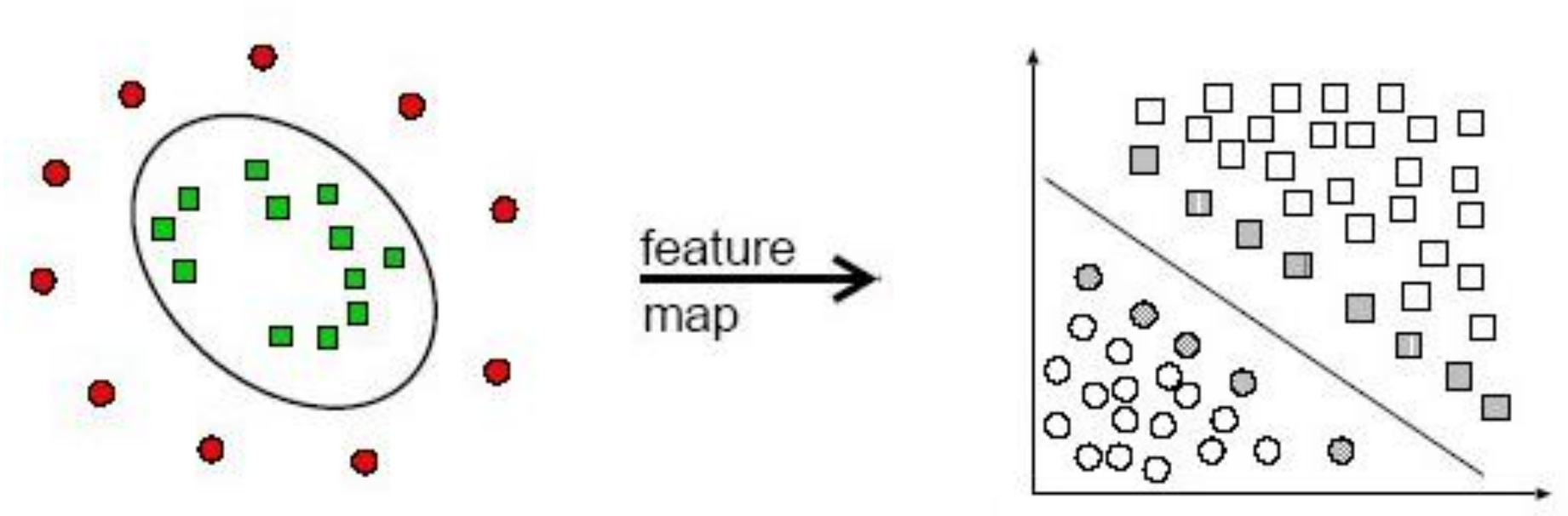
Increasing the number of features



“Curse of dimensionality”

Mapping the features into another space

“The kernel trick”

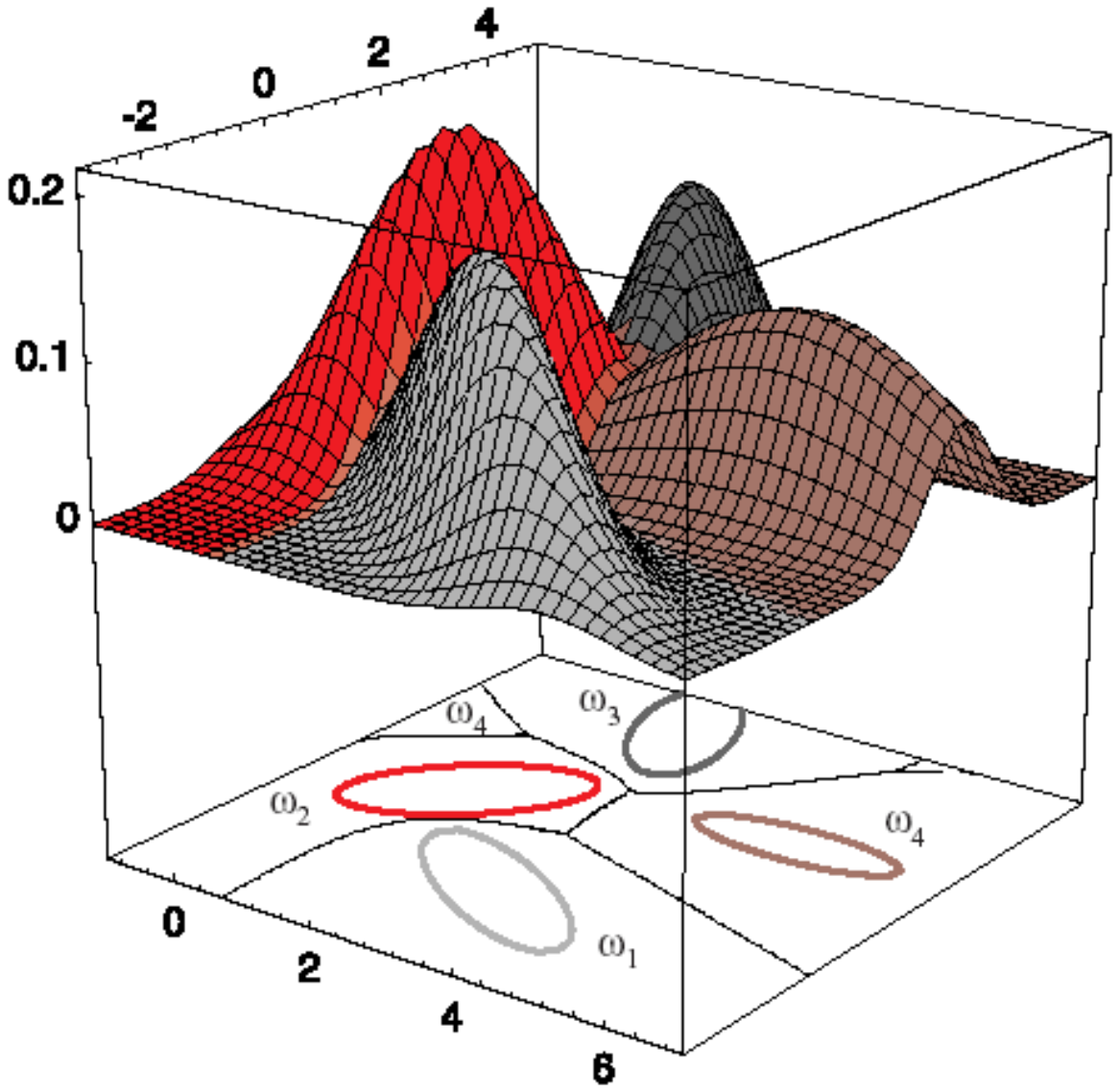


“Overtraining/undertraining”

Multiclass SVM

- One vs. one
- One vs rest
- Intrinsic multiclass

Statistical classifiers



Statistical classifiers

Assumptions:

Feature values are random variables

Training set is large, so it can be represented by pdf

Decision boundary is based on these pdf, not on individual data

The decision is probabilistic

Various optimization strategies to find the boundary

Bayesian classifier

Maximizes the aposteriory probability

Minimizes the mean classification error

It is based on the **Bayes rule**

The Bayes rule

Class-conditional probability

A posteriori probability

$$P(\omega_j|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_j)P(\omega_j)}{p(\mathbf{x})},$$

A priori probability

Total probability

$$p(\mathbf{x}) = \sum_{j=1}^{c} p(\mathbf{x}|\omega_j)P(\omega_j).$$

Bayesian classifier

Main idea: maximize posterior probability

$$P(\omega_j|\mathbf{x})$$

Since it is hard to do directly, we rather maximize

$$p(\mathbf{x}|\omega_j)P(\omega_j)$$

In case of equal priors, we maximize only

$$p(\mathbf{x}|\omega_j)$$

Equivalent formulation

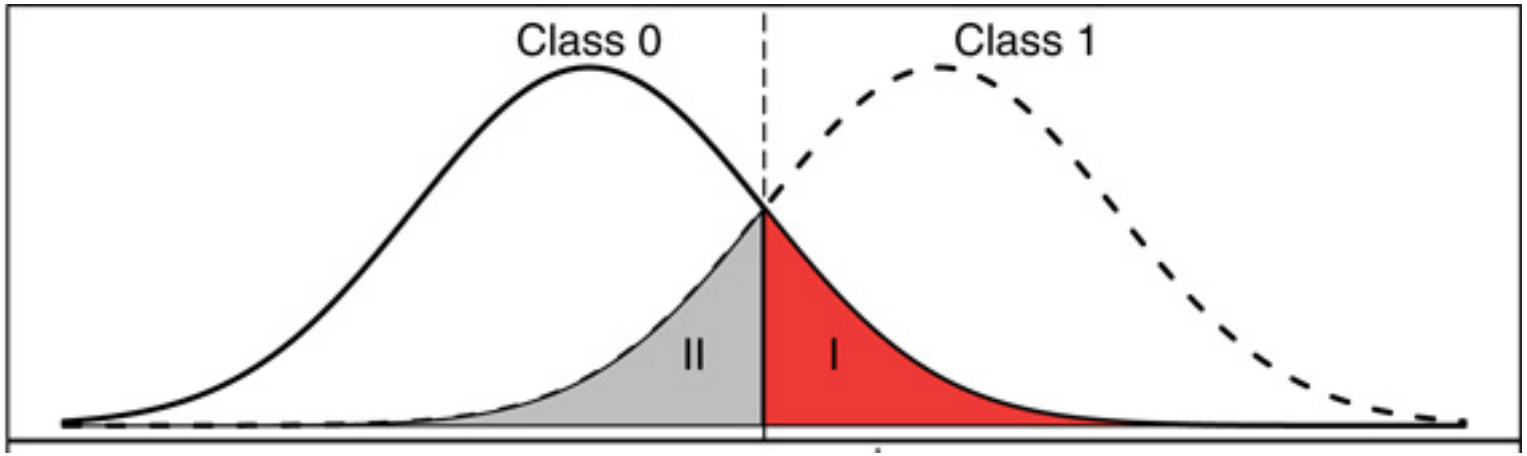
…in terms of discriminat functions

$$g_i(\mathbf{x}) = P(\omega_i|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_i)P(\omega_i)}{\sum_{j=1}^{c} p(\mathbf{x}|\omega_j)P(\omega_j)}$$

$$g_i(\mathbf{x}) = p(\mathbf{x}|\omega_i)P(\omega_i)$$

$$g_i(\mathbf{x}) = \ln\, p(\mathbf{x}|\omega_i) + \ln\, P(\omega_i),$$

Bayesian classifier



How to estimate $p(\mathbf{x}|\omega_j)P(\omega_j)$?

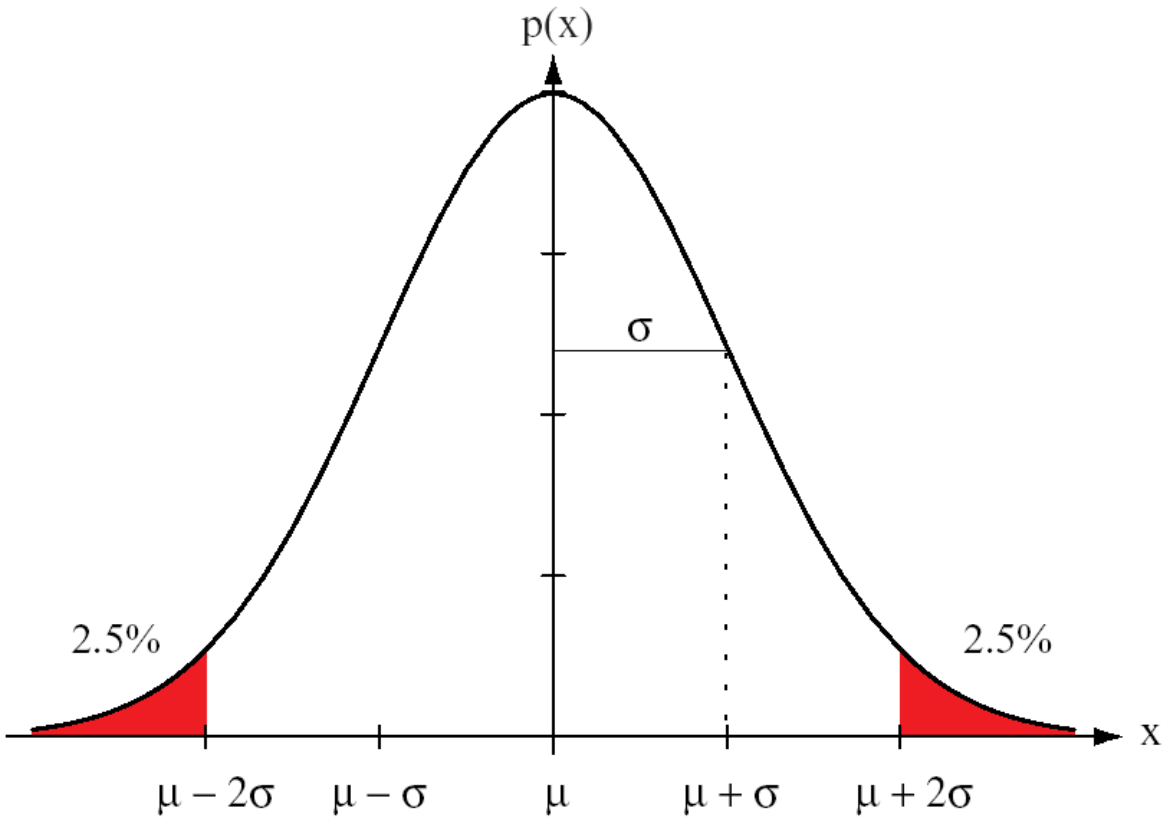
$P(\omega_j)$

- From the case studies performed before (OCR, speech recognition)
- From the occurence in the training set
- Assumption of equal priors

$p(\mathbf{x}|\omega_j)$

- Parametric estimate (assuming pdf is of a known form, e.g. Gaussian)
- Non-parametric estimate (pdf is unknown or too complex)

Parametric estimate of Gaussian $p(\mathbf{x}|\omega_j)$

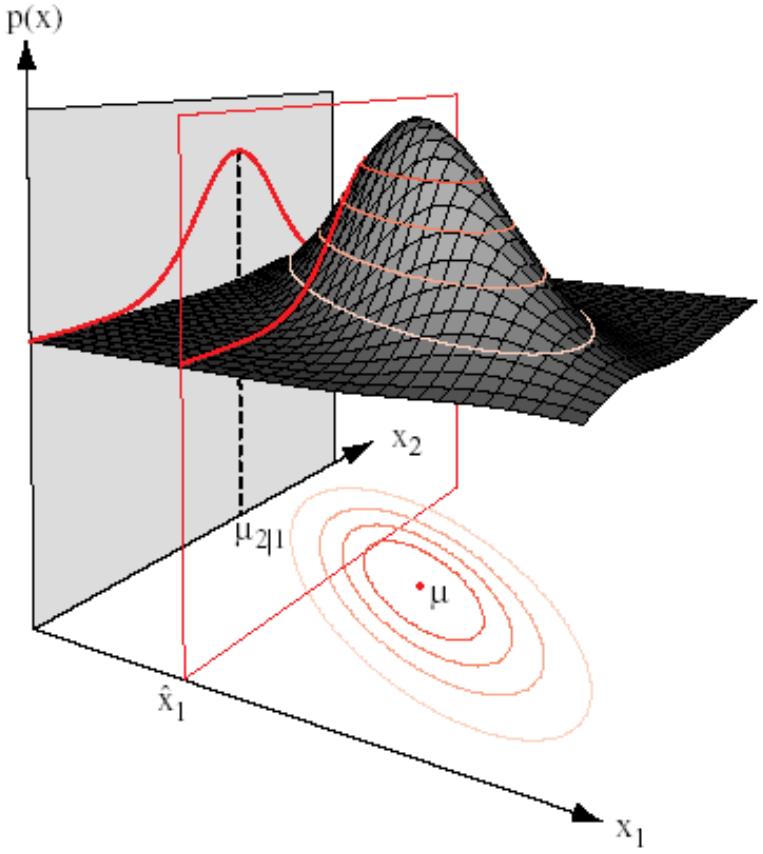


$$\hat{\mu} = \frac{1}{n}\sum_{k=1}^{n} x_k$$

$$\hat{\sigma}^2 = \frac{1}{n}\sum_{k=1}^{n}(x_k - \hat{\mu})^2$$

$$p(x) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left[-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right]$$

d-dimensional Gaussian pdf

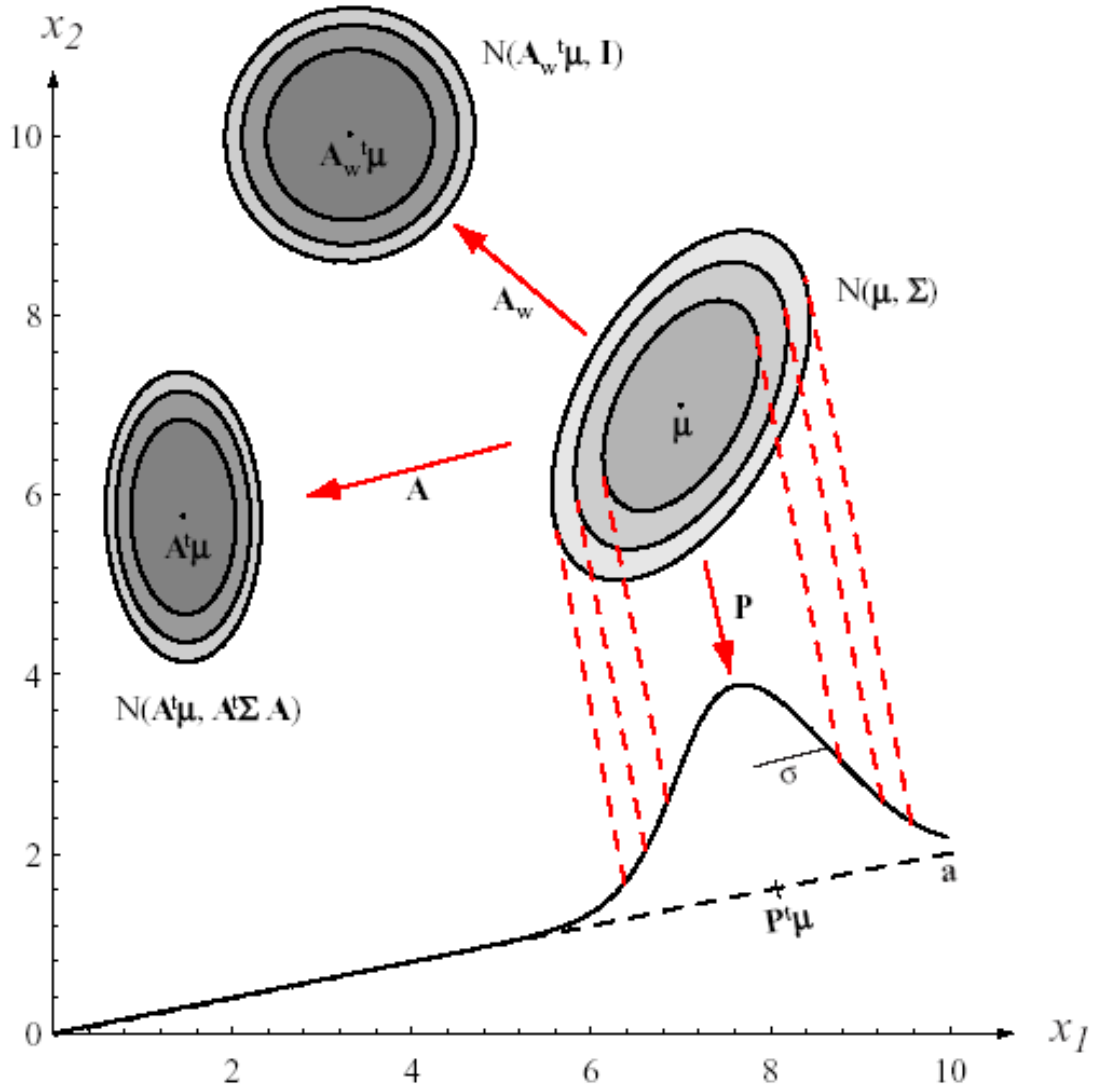


$$\hat{\boldsymbol{\mu}} = \frac{1}{n}\sum_{k=1}^{n}\mathbf{x}_k$$

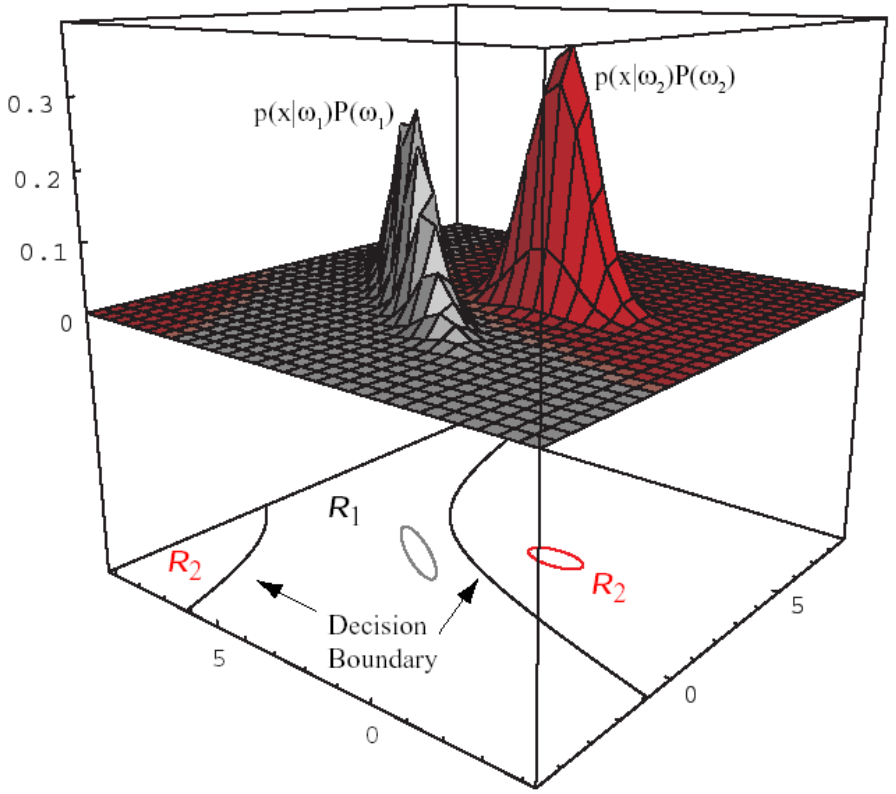
$$\hat{\boldsymbol{\Sigma}} = \frac{1}{n}\sum_{k=1}^{n}(\mathbf{x}_k - \hat{\boldsymbol{\mu}})(\mathbf{x}_k - \hat{\boldsymbol{\mu}})^t.$$

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}}\exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^t\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right]$$

The role of covariance matrix



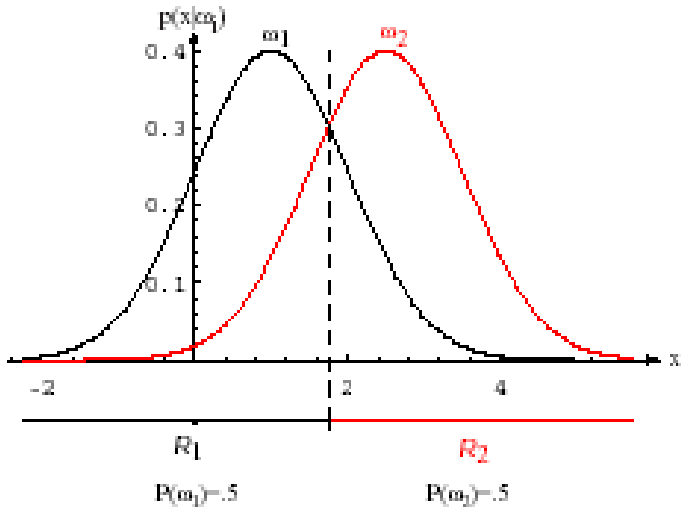
Two-class Gaussian case in 2D

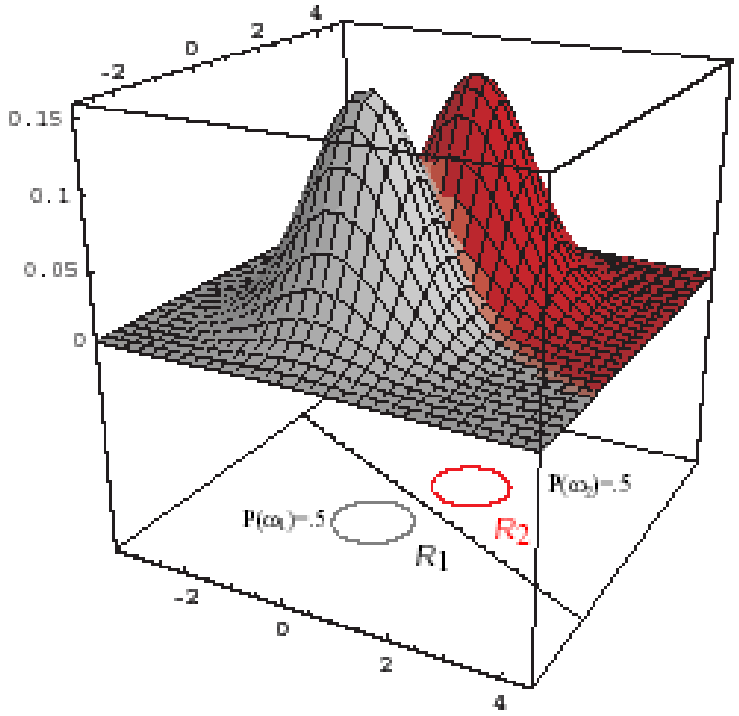


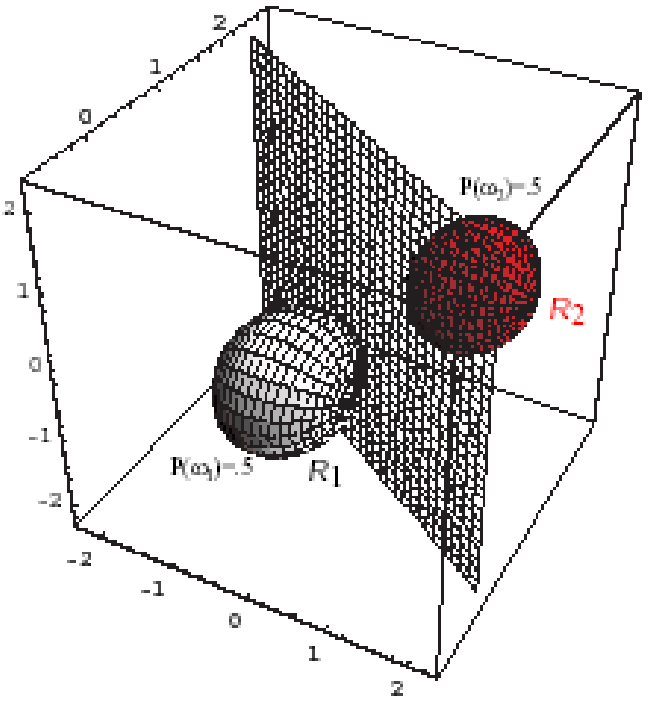
$$g_i(\mathbf{x}) = -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^t\boldsymbol{\Sigma}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i) - \frac{d}{2}\ln 2\pi - \frac{1}{2}\ln|\boldsymbol{\Sigma}_i| + \ln P(\omega_i).$$

Classification = comparison of two Gaussians

Two-class Gaussian case – Equal cov. mat.







$$g_i(\mathbf{x}) = -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_i)^t \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_i) + \ln P(\omega_i)$$

Linear decision boundary

Equal priors $P(\omega_j)$

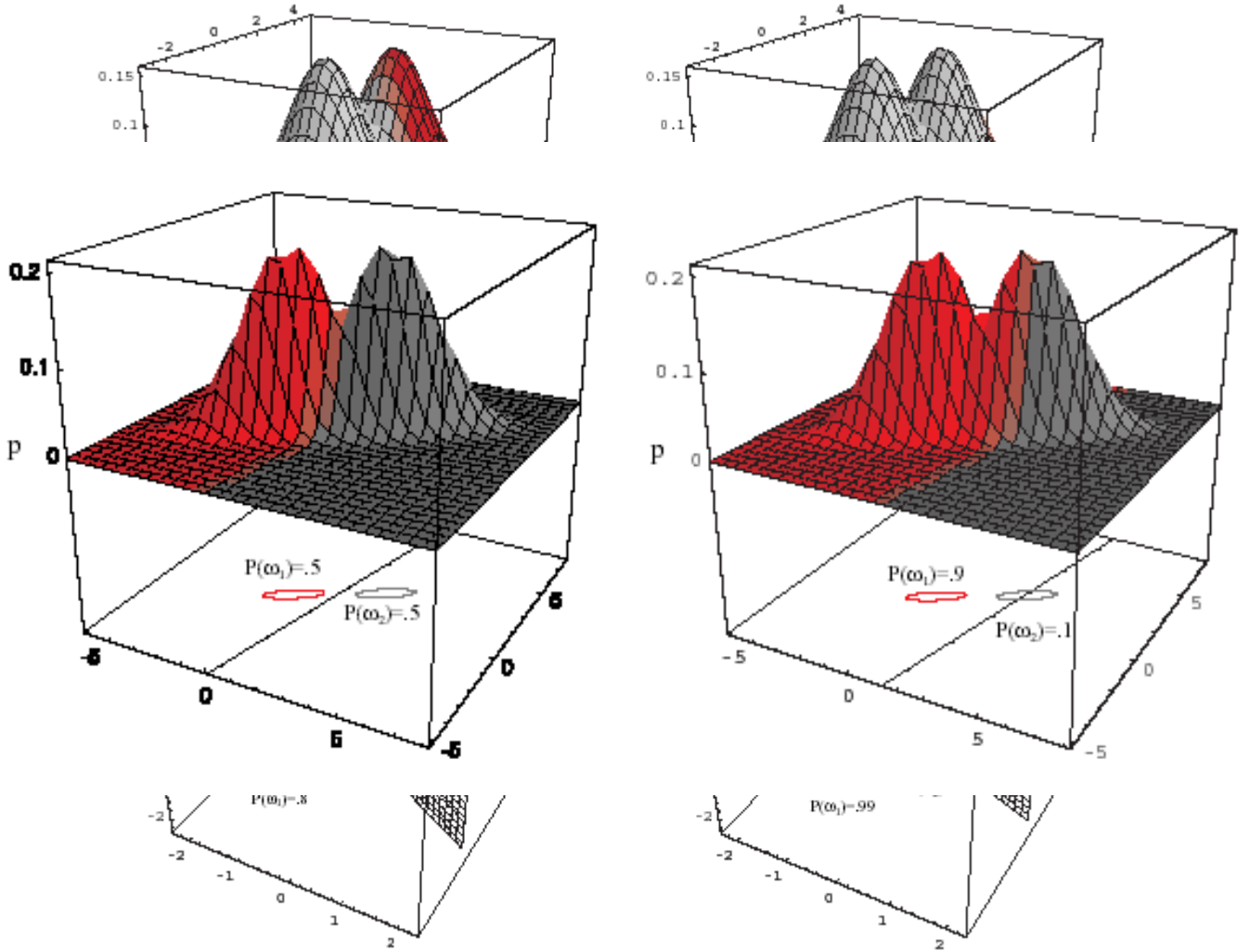
max $g_i(\mathrm{x}) = -\frac{1}{2}(\mathrm{x}-\boldsymbol{\mu}_i)^t\boldsymbol{\Sigma}^{-1}(\mathrm{x}-\boldsymbol{\mu}_i)$.

min $(\mathrm{x}-\boldsymbol{\mu}_i)^t\boldsymbol{\Sigma}^{-1}(\mathrm{x}-\boldsymbol{\mu}_i)$.

Classification by minimum **Mahalanobis distance**

If the cov. mat. is diagonal with equal variances then we get **“standard”** minimum distance rule

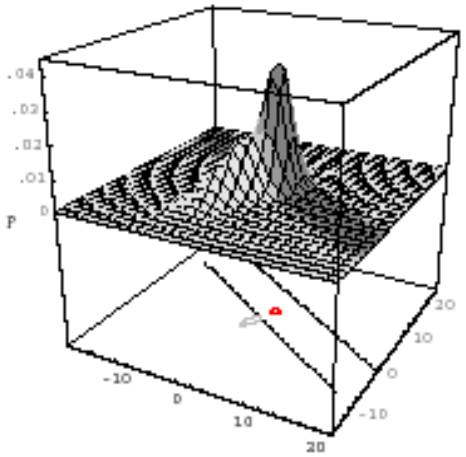
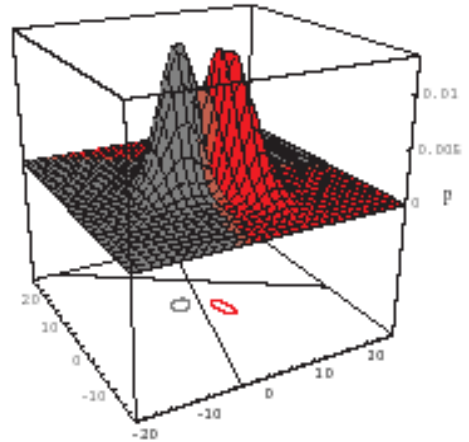
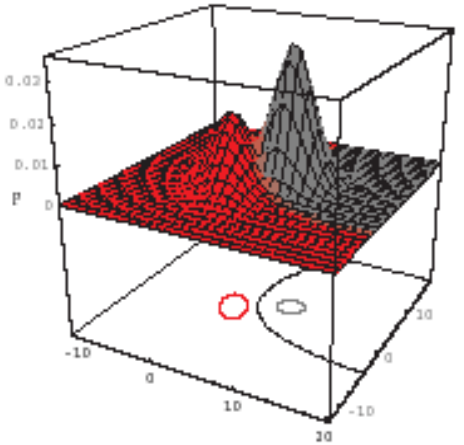
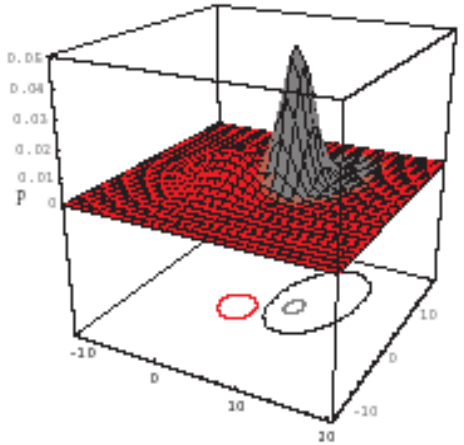
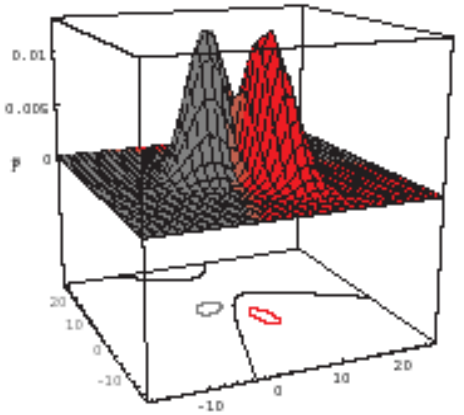
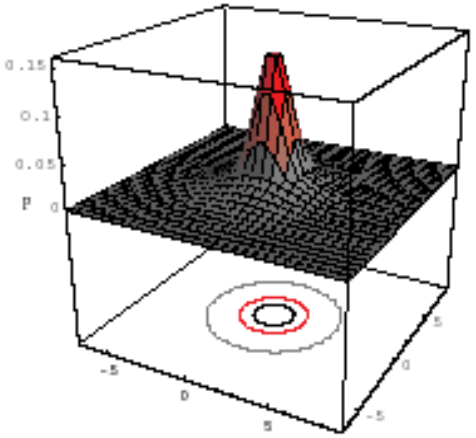
Non-equal priors



Linear decision boundary still preserved

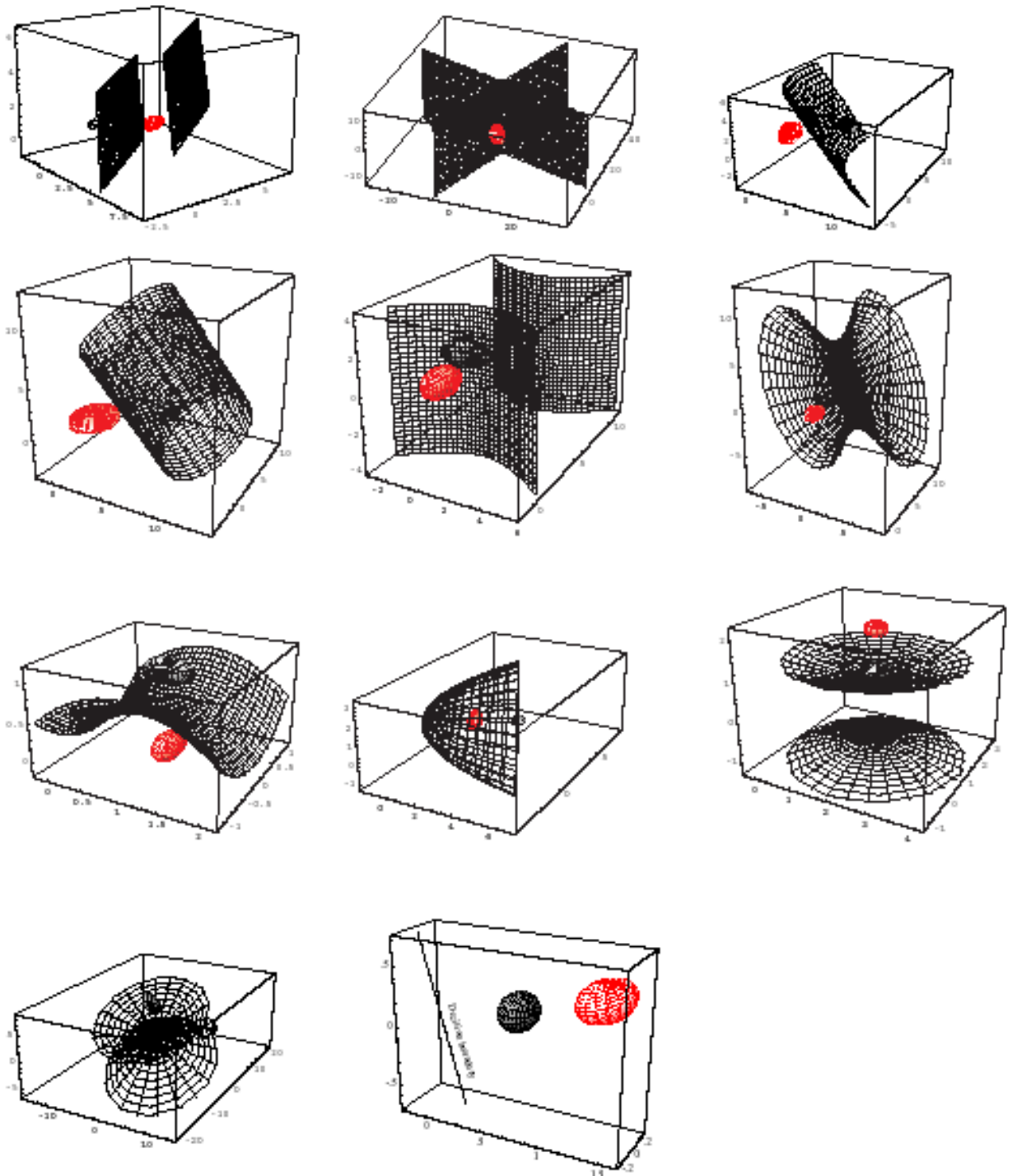
General G case in 2D

Decision boundary is a hyperquadric

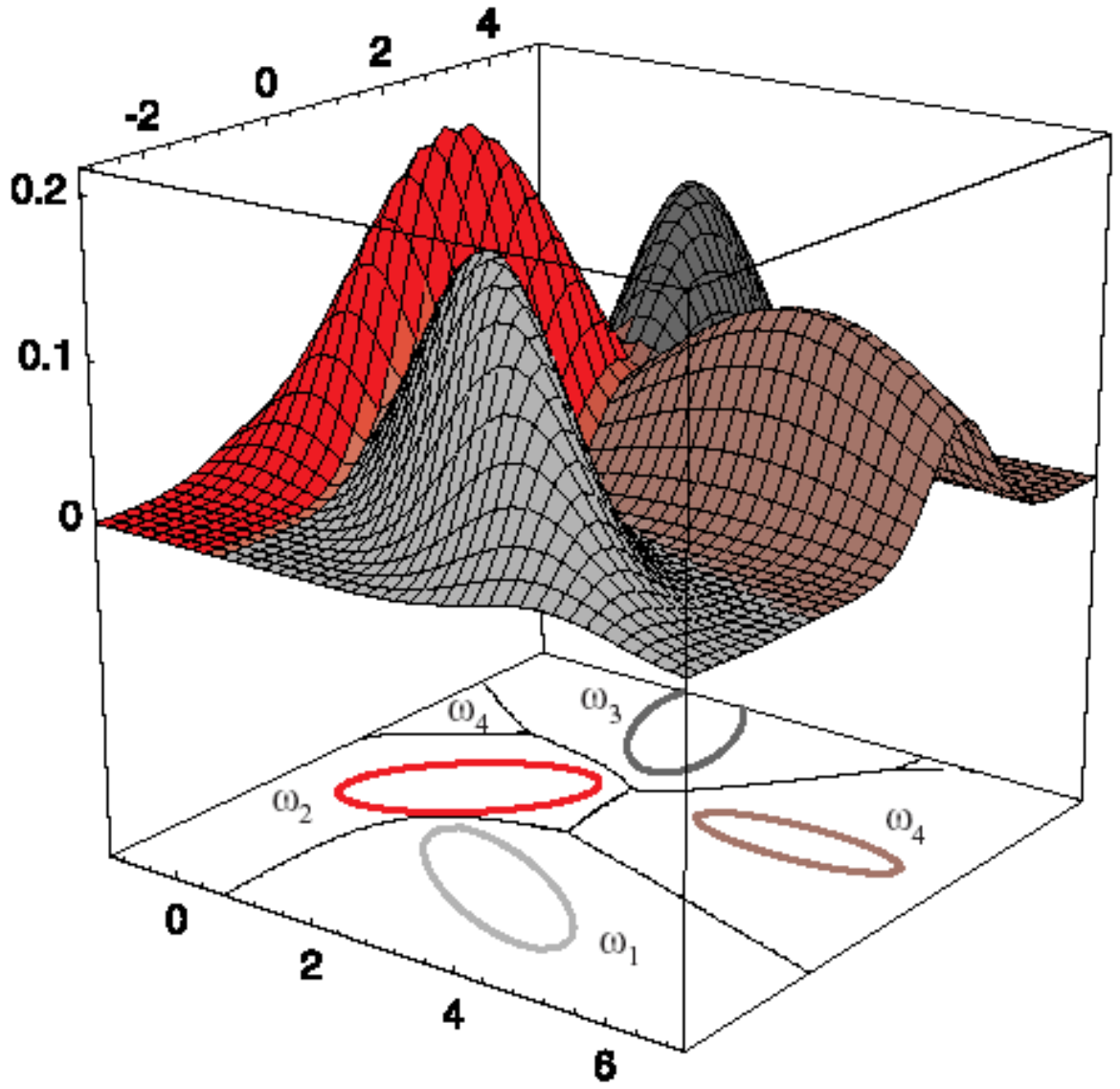







General G case in 3D

Decision boundary is a hyperquadric

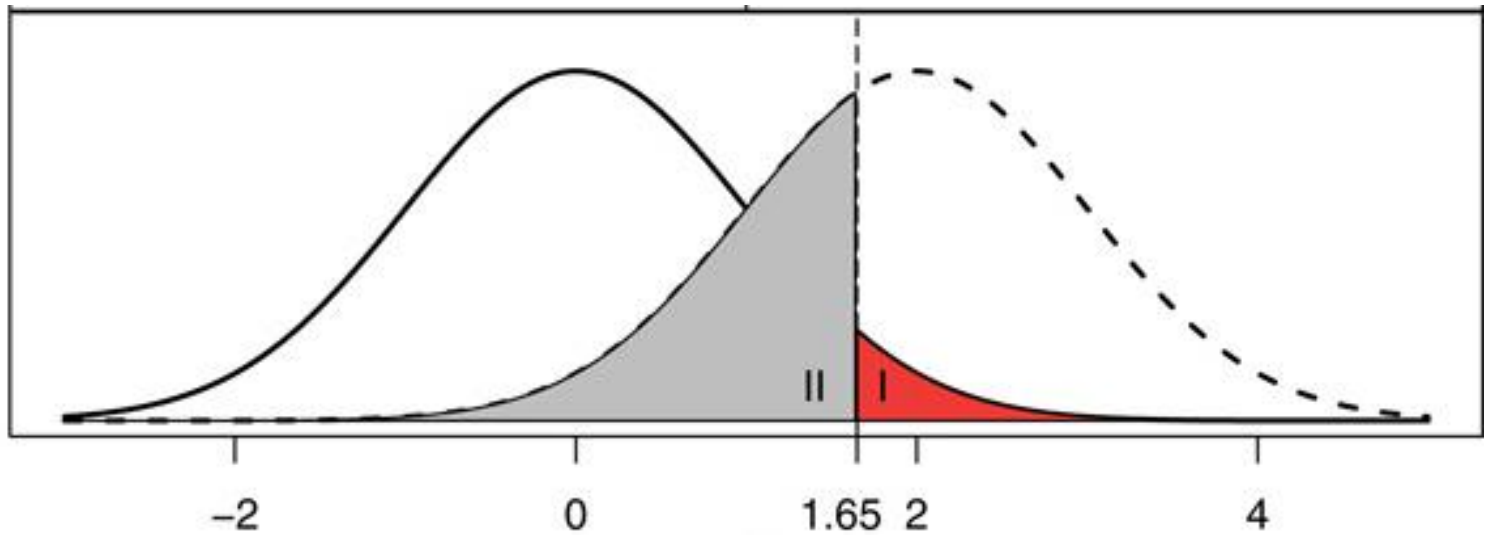


More classes, Gaussian case in 2D



Neyman-Pearson classifier

- Priors are unknown or do not exist
- Asymetric approach
- One error is constrained, the second one is optimized



Minimax classifier

- Minimizes the maximum classification error
- Often similar to Bayes

Recalling some previous topics

- k-NN : How to choose k?
- Compare Bayes, N.-P. and minimax classifiers
- Why not minmin classifier?
- Parameter estimation – maximum likelihood, MAP

Maximum likelihood estimation

$$f(x;\theta_1,\theta_2,...,\theta_k)$$

$$L(x_1,x_2,...,x_N|\theta_1,\theta_2,...,\theta_k)=L=\prod_{i=1}^{N} f(x_i;\theta_1,\theta_2,...,\theta_k)$$

$$i=1,2,...,N$$

$$\Lambda=\ln L=\sum_{i=1}^{N}\ln f(x_i;\theta_1,\theta_2,...,\theta_k)$$

$$\frac{\partial(\Lambda)}{\partial\theta_j}=0,\quad j=1,2,...,k$$

How to test the normality

…of the class-conditional distributions?

- Pearson's chi-square test (goodness-of-fit test)

$$X^2 = \sum \frac{(\text{observed} - \text{expected})^2}{\text{expected}}$$

- Moment-based tests
- Visual assessment

How to test the equality

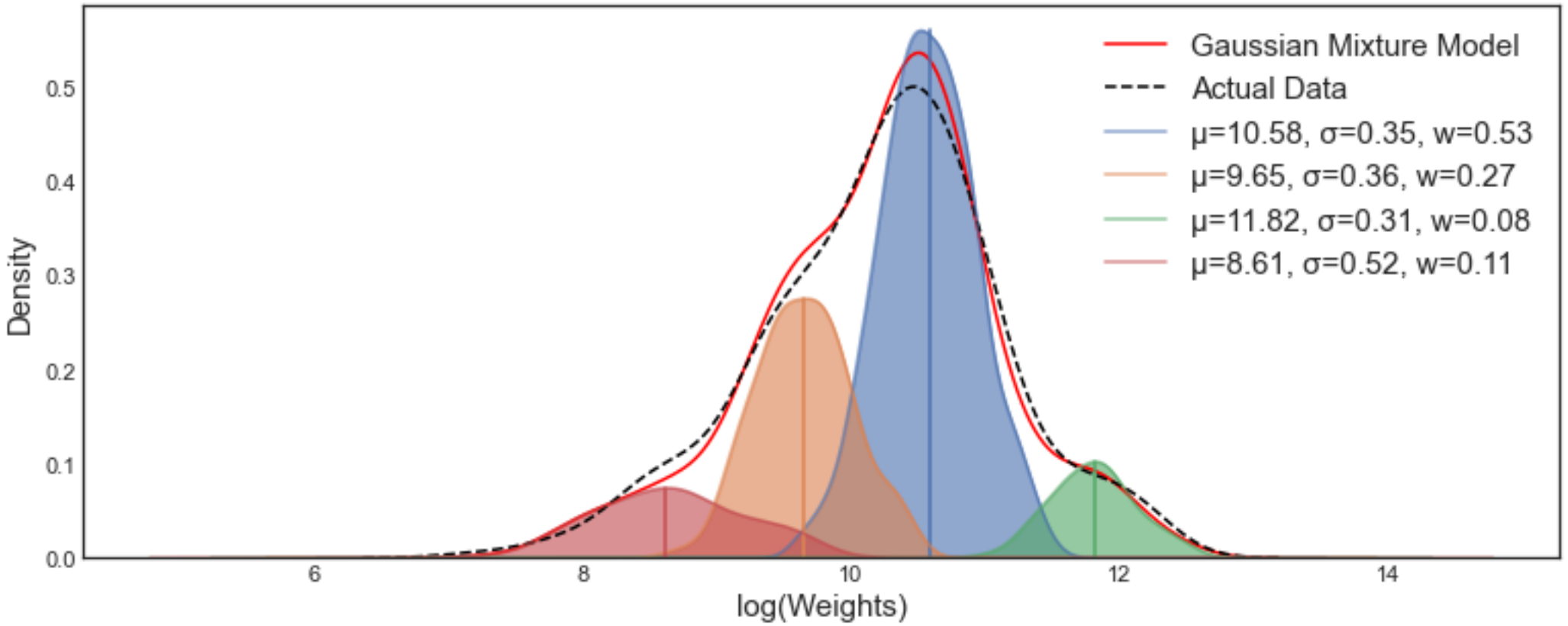
…of the covariance matrixes?

What to do

…if the classes are not normally distributed?

- Gaussian mixtures
- Parametric estimation of some other pdf
- Non-parametric estimation

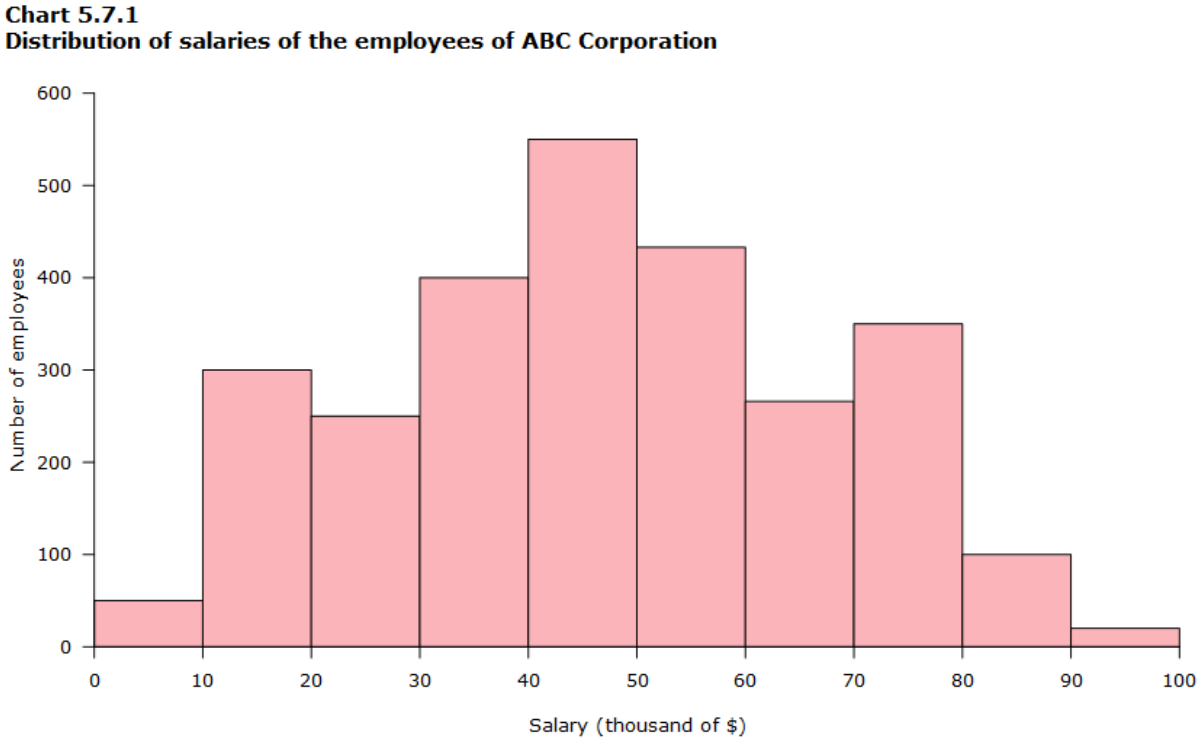
Gaussian mixture



Non-parametric estimation

Estimation of *p(x|w)* by relative frequency in volume *V*

- histogram method



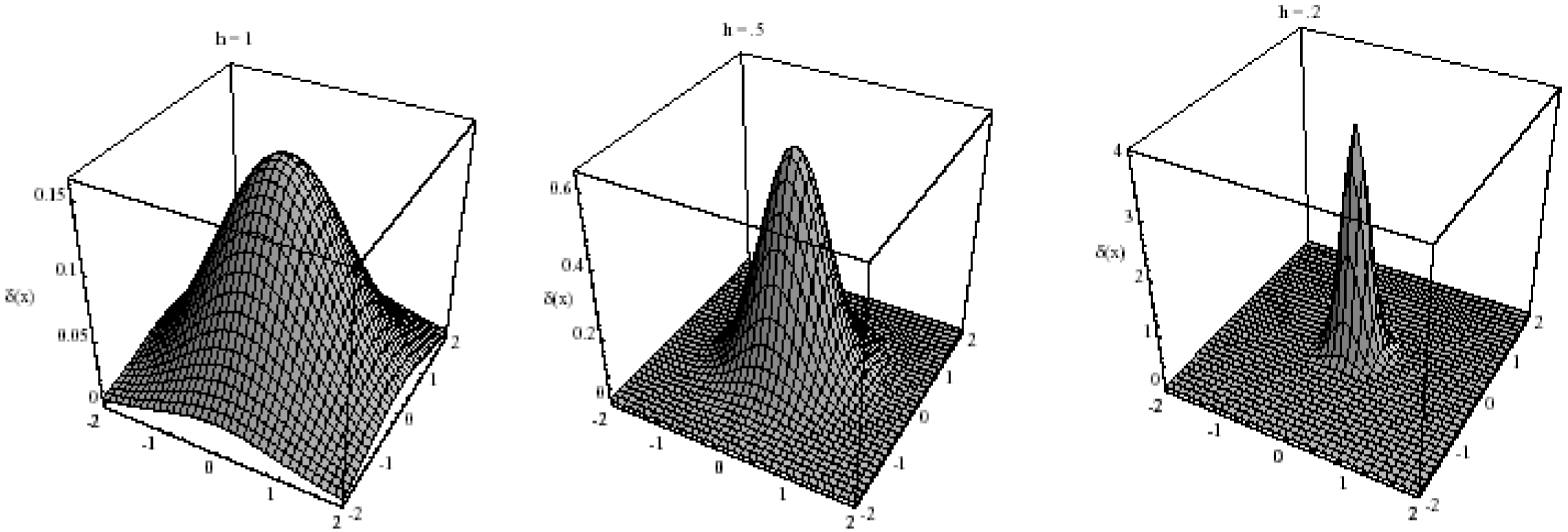
Non-parametric estimation

Estimation of *p(x|w)* by relative frequency in volume *V*

$$p(\mathbf{x}|\omega_j) \simeq \frac{k/n}{V}$$

Non-parametric estimation – Parzen window

Volume *V* is weighted by parametric Parzen window function



The role of the window size

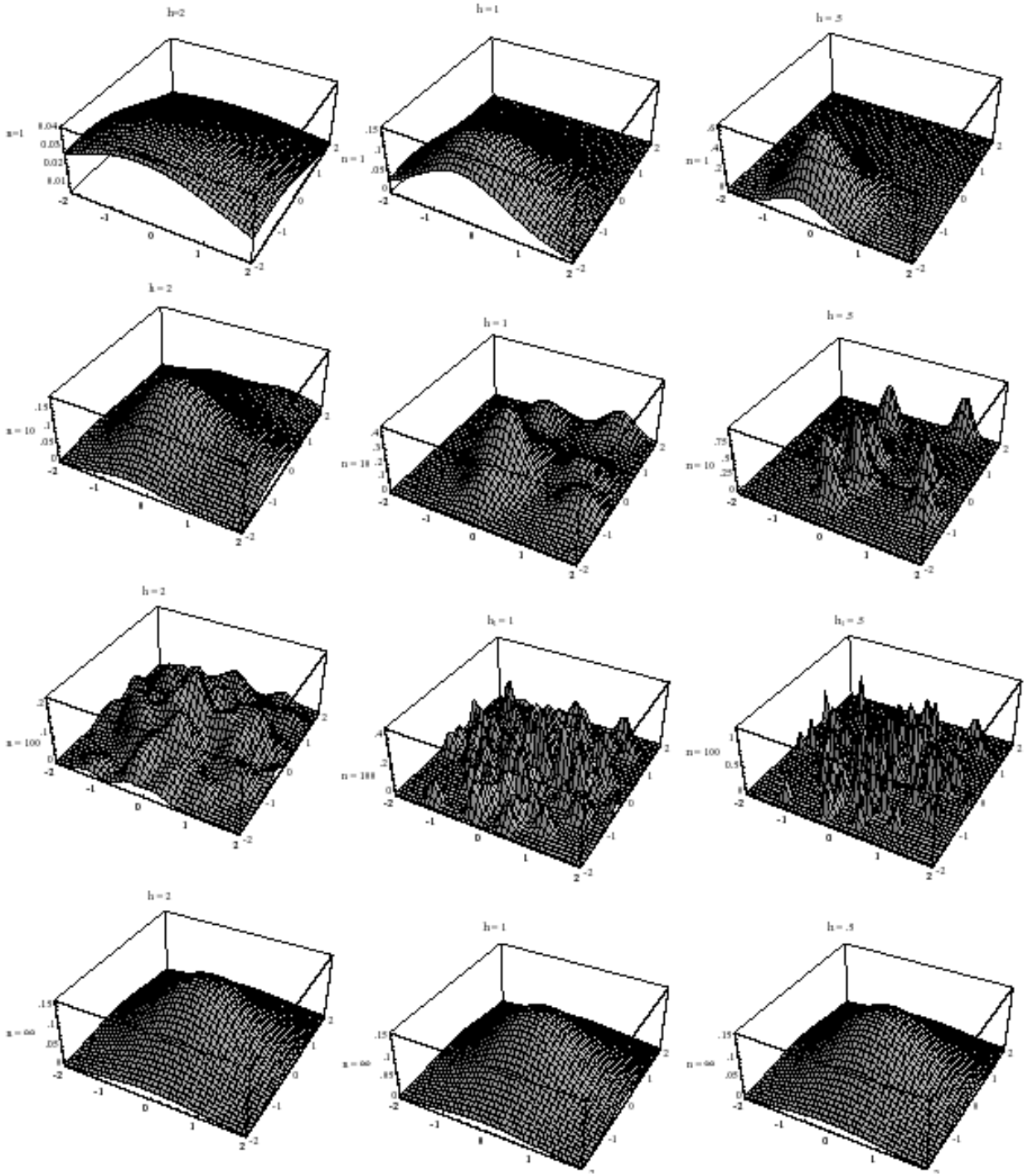
- Small window → overtaining
- Large window → data smoothing
- Continuous data → the size does not matter

n = 1

n = 10

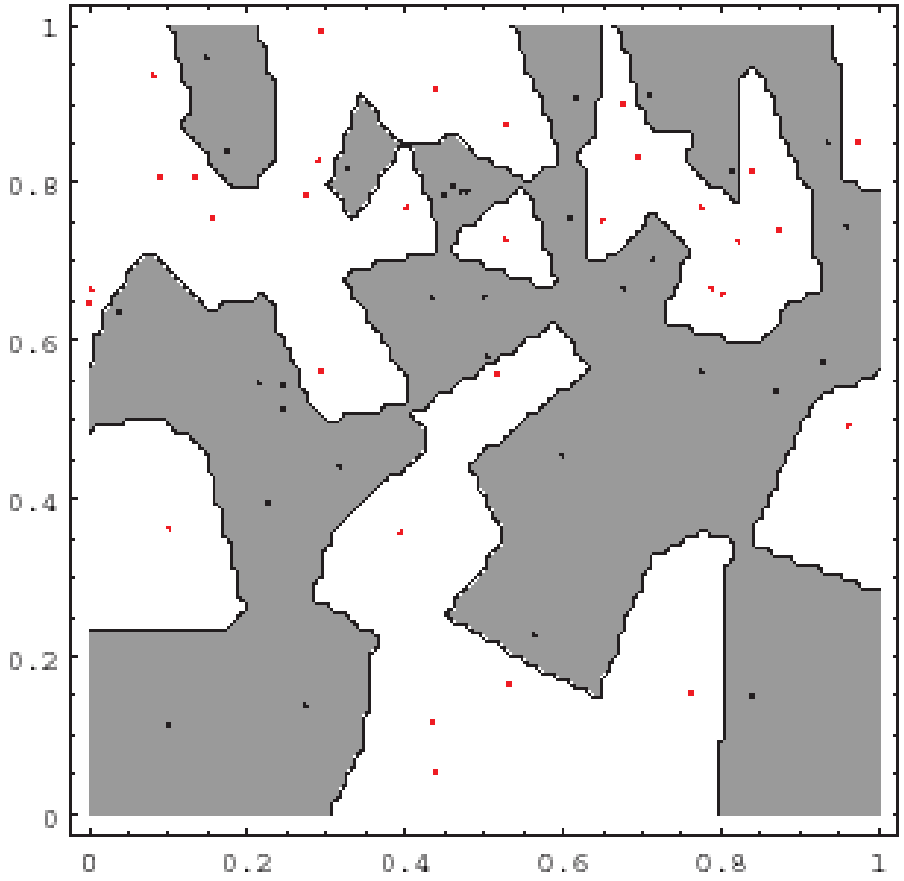
n = 100

n = ∞

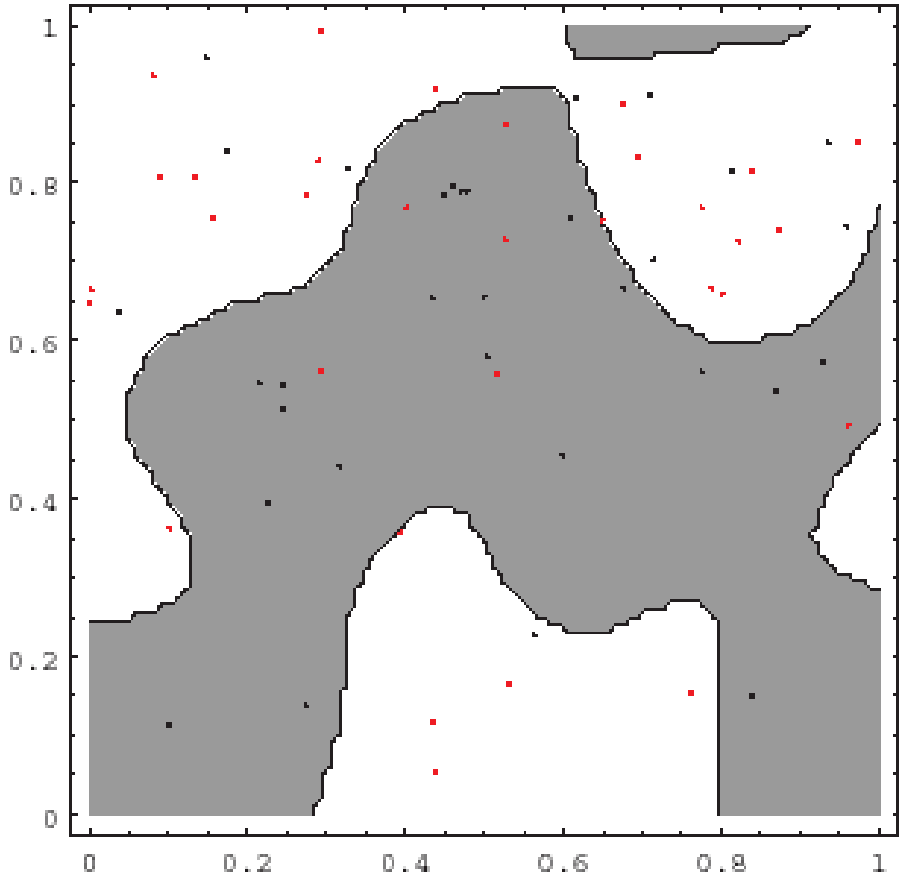


The role of the window size

Small window



Large window



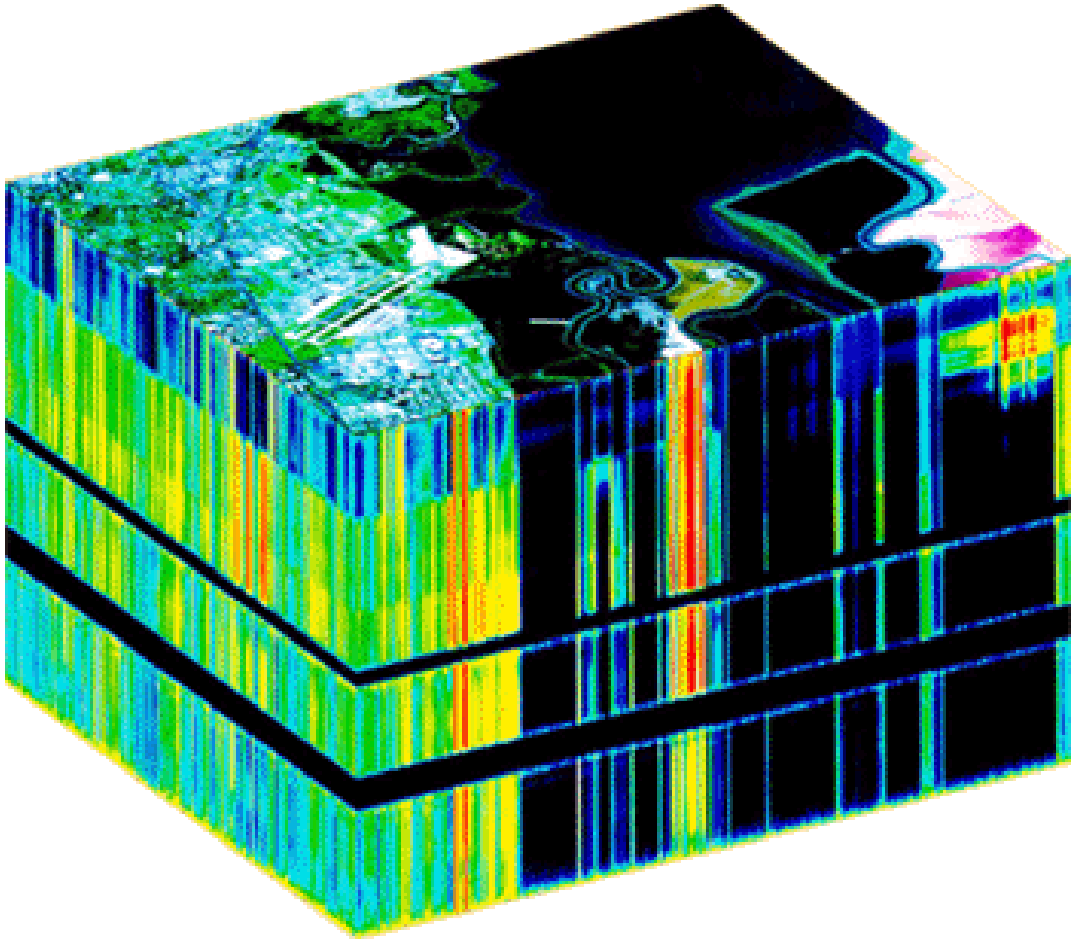
Applications of Bayesian classifier

…in multispectral remote sensing

- Objects = pixels
- Features = pixel values in the spectral bands (from 4 to several hundreds)
- Training set – selected manually by means of thematic maps (GIS), and on-site observation
- Number of classes – typically from 2 to 16

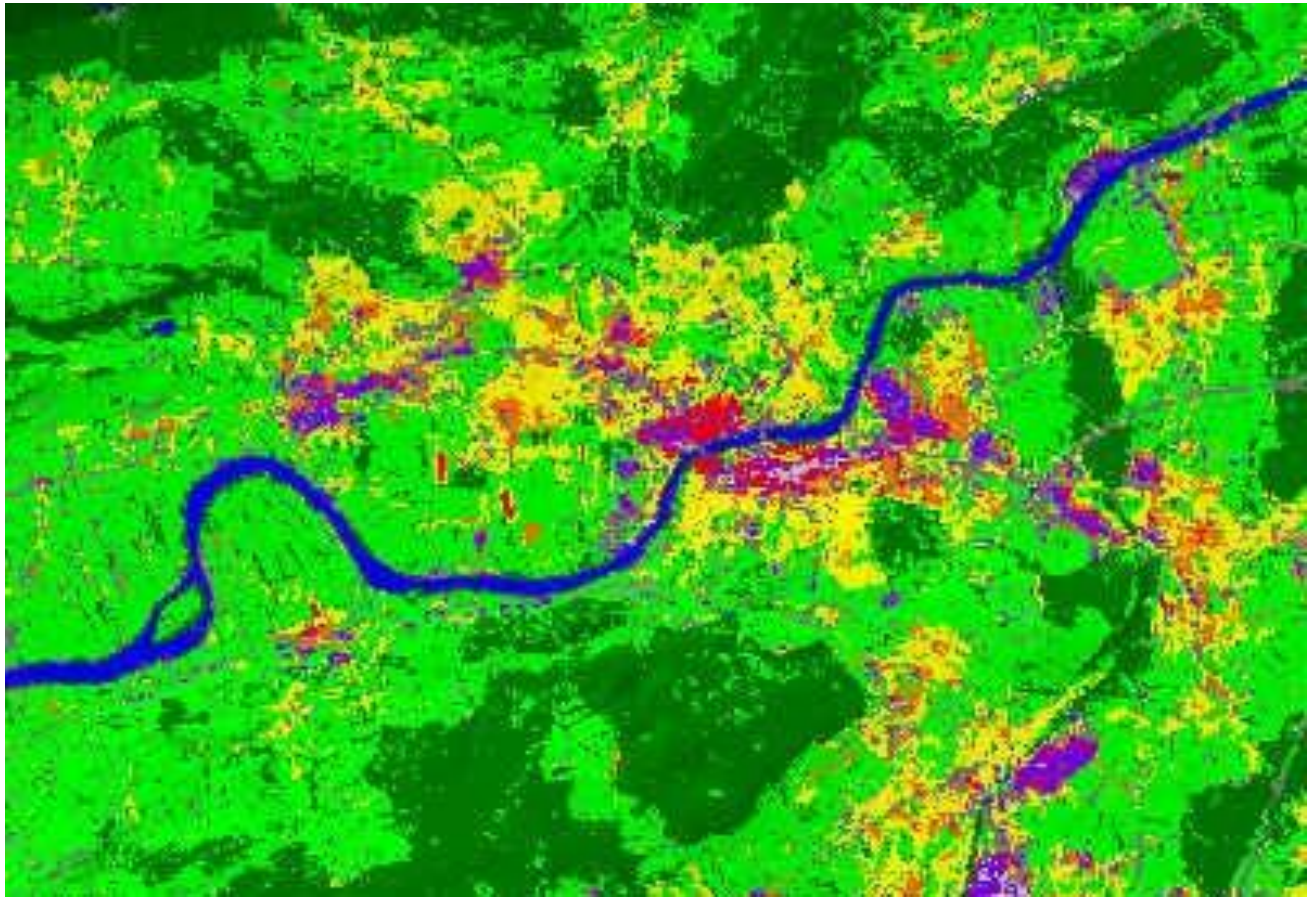
Applications of Bayesian classifier

…in multispectral remote sensing



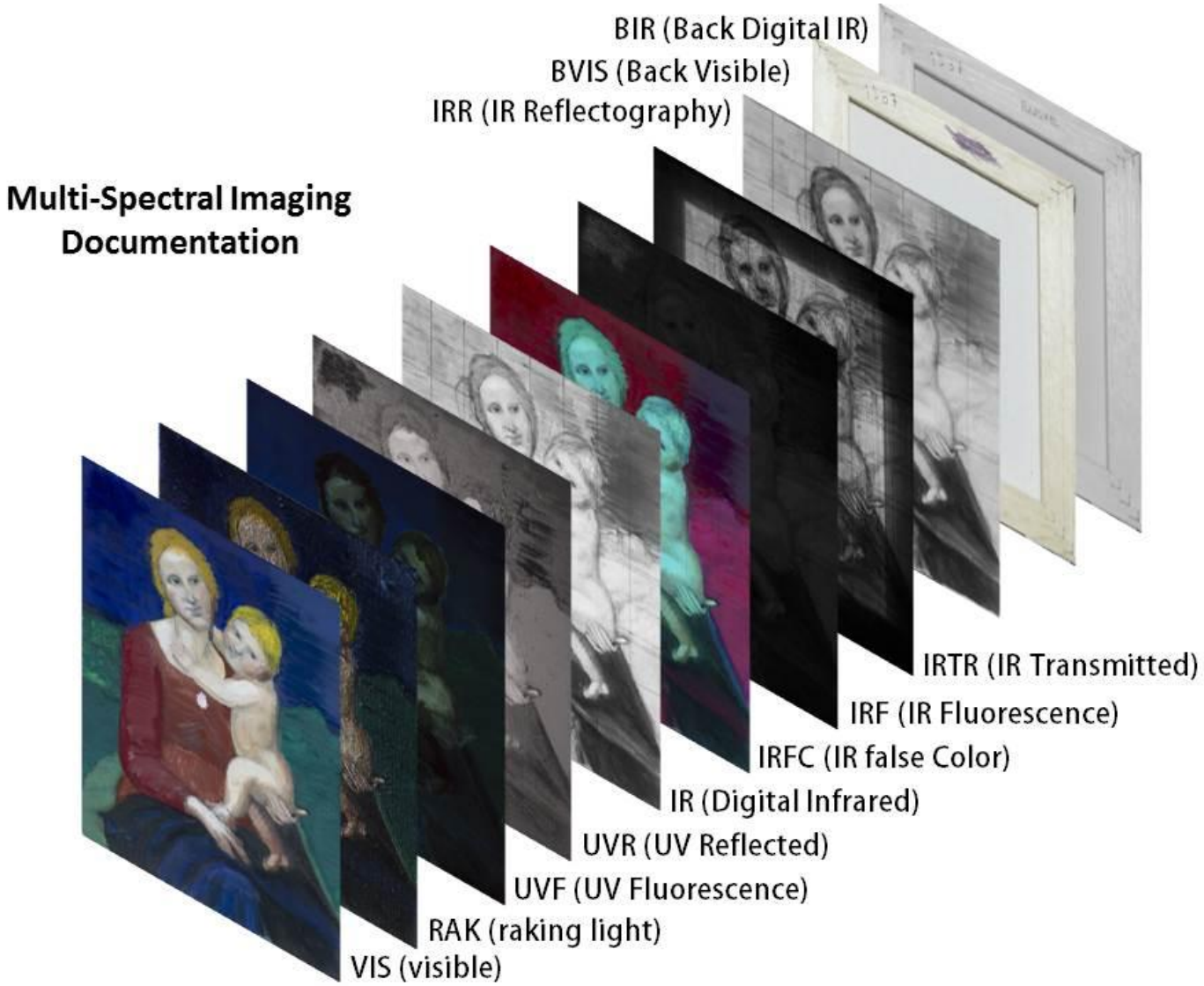
Applications of Bayesian classifier

…in multispectral remote sensing



Applications of Bayesian classifier

…in art



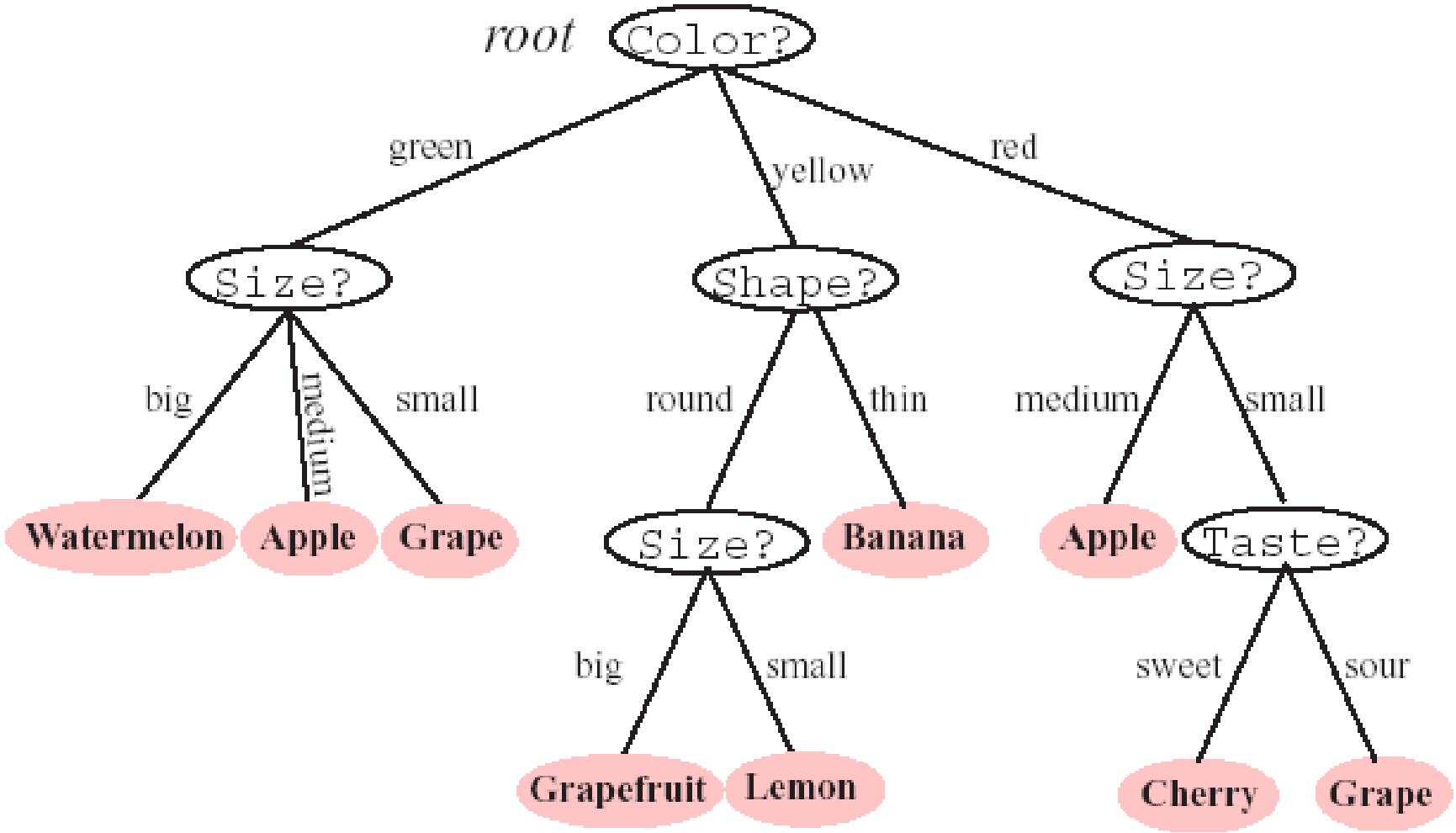
Non-metric classifiers

Typically for “YES – NO” features

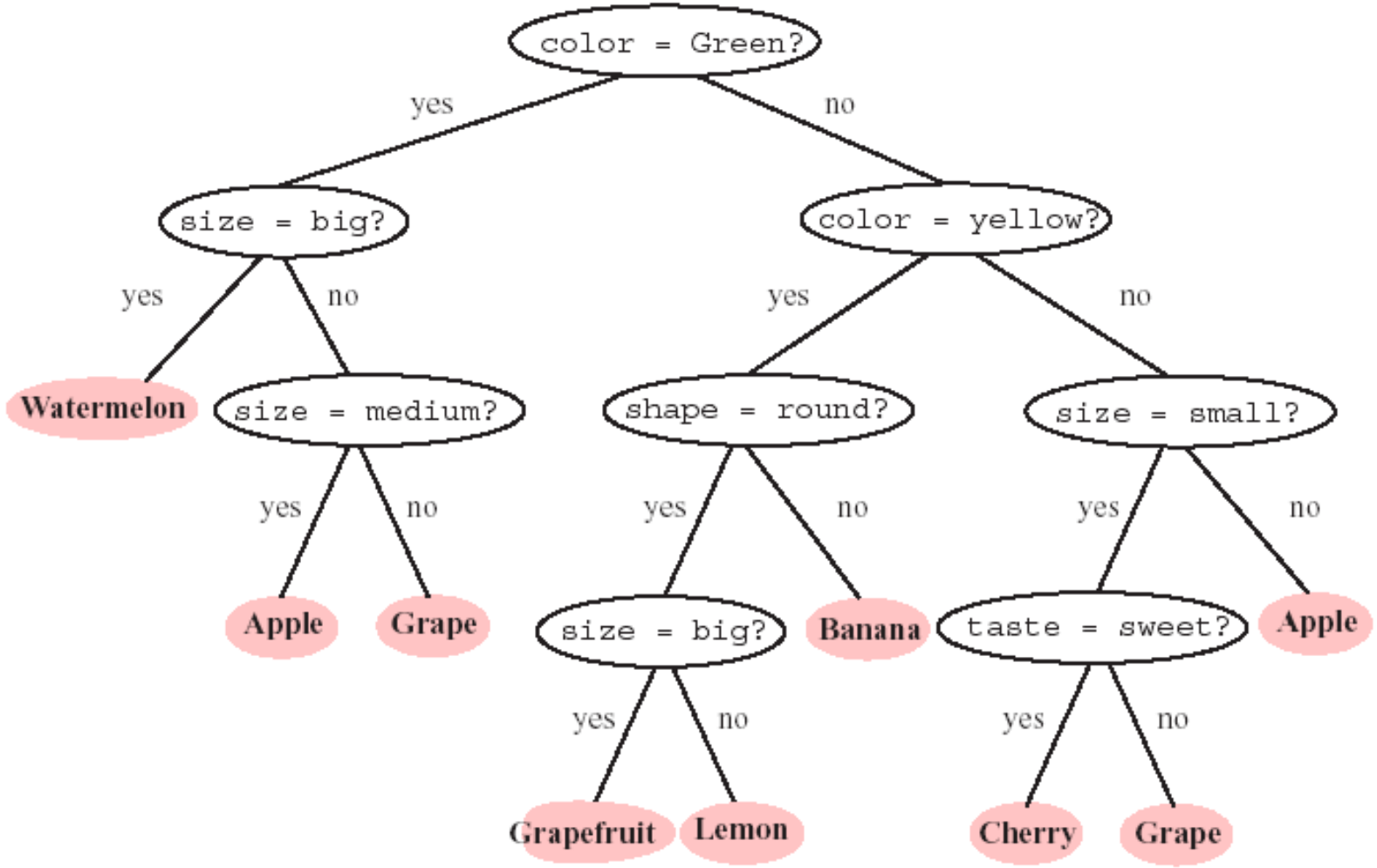
Feature metric is not explicitly defined

Decision trees

General decision tree



Binary decision tree

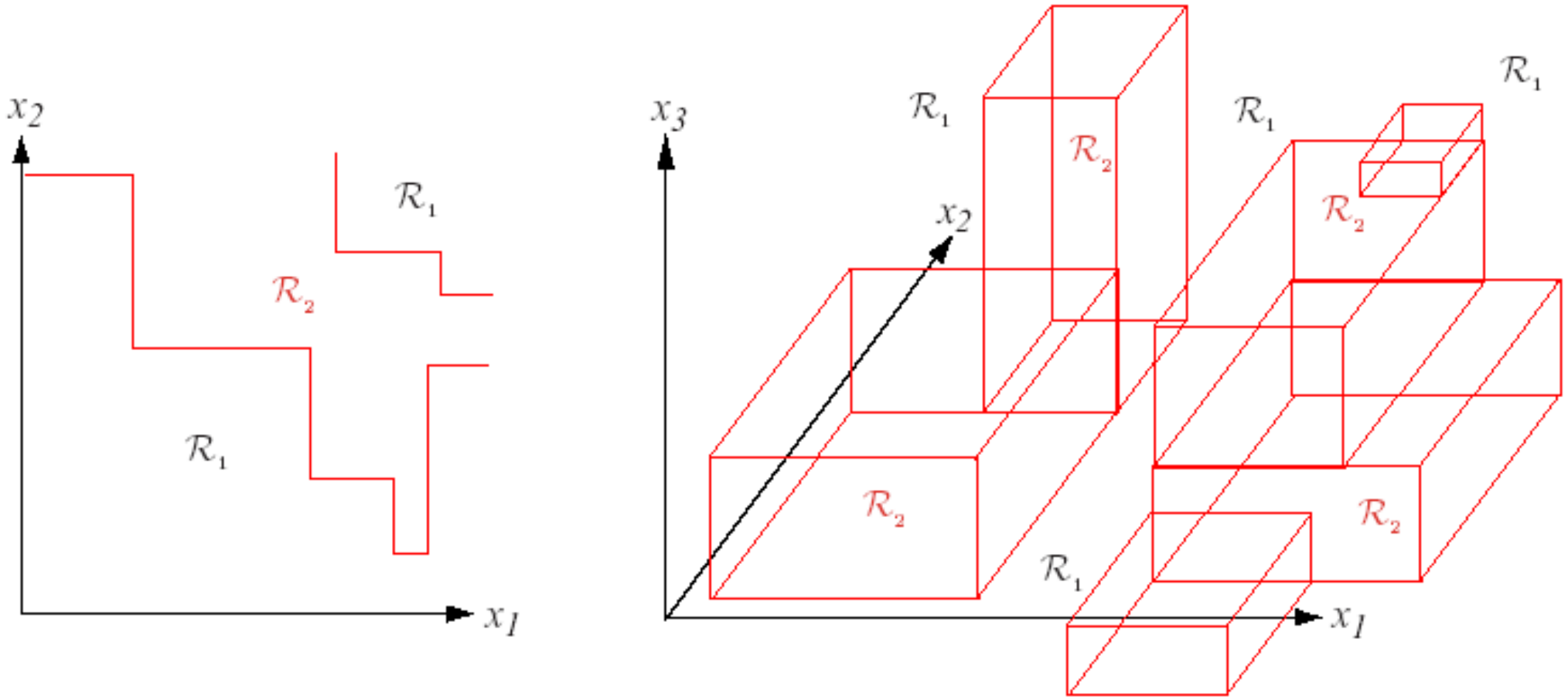


Any decision tree can be replaced by a binary tree

Real-valued features

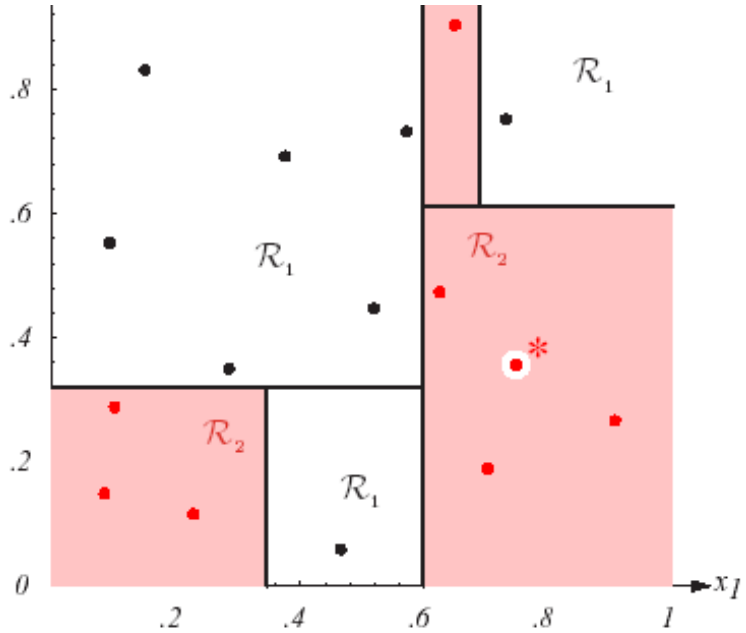
- Node decisions are in form of inequalities
- Training = setting their parameters
- Simple inequalities → stepwise decision boundary

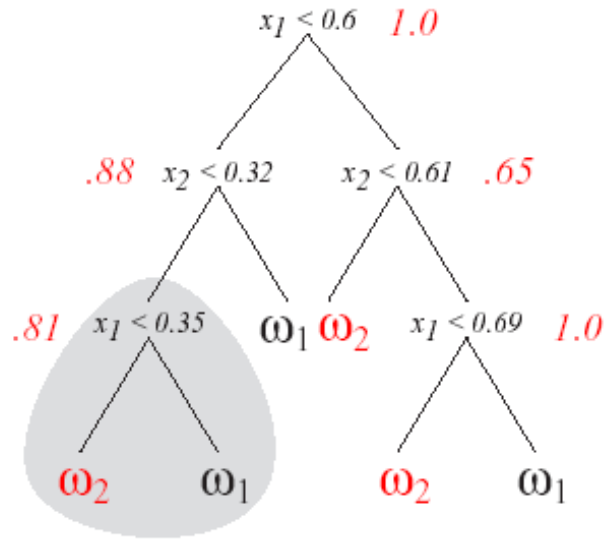
Stepwise decision boundary

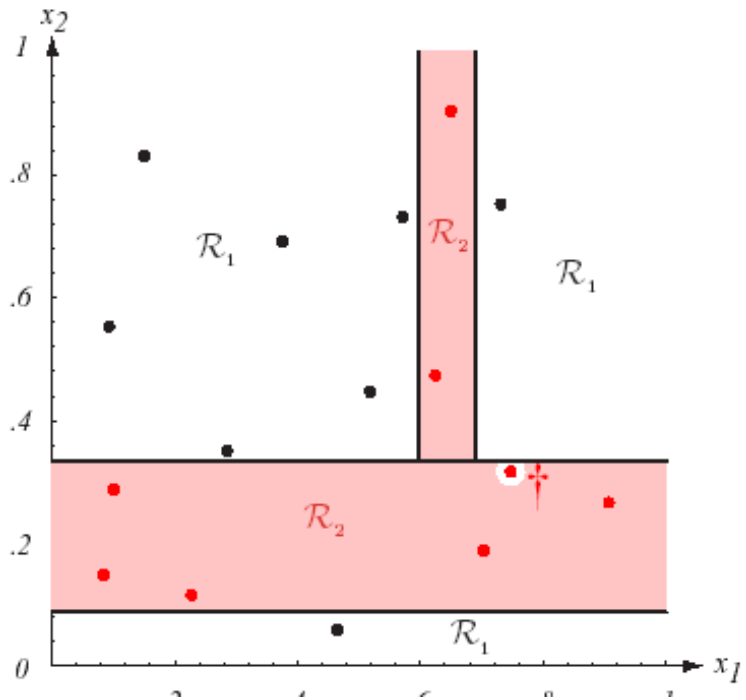


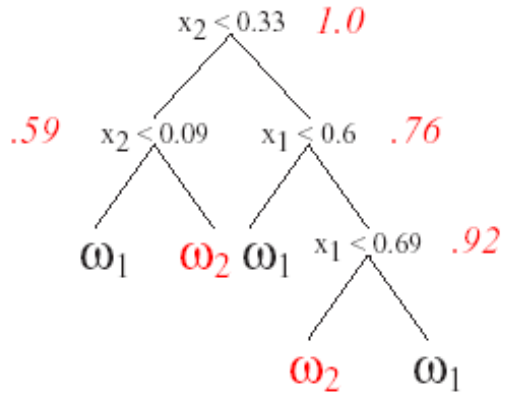
Real-valued features

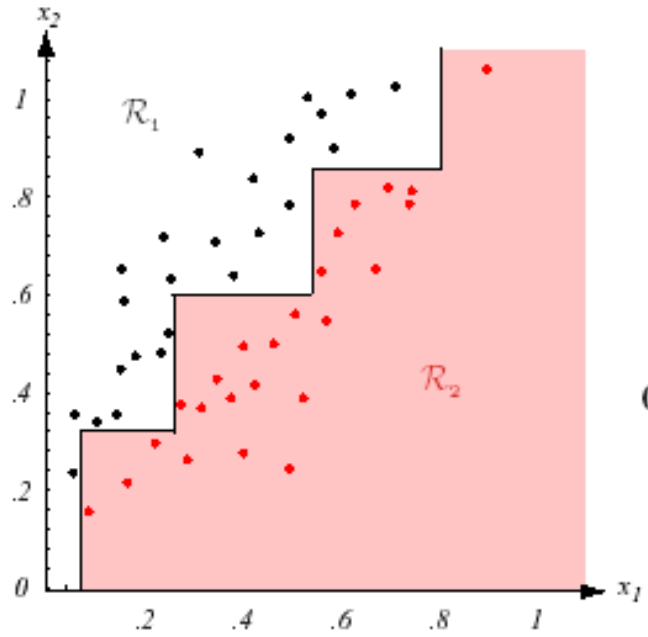
The tree structure and the form of inequalities influence both performance and speed.

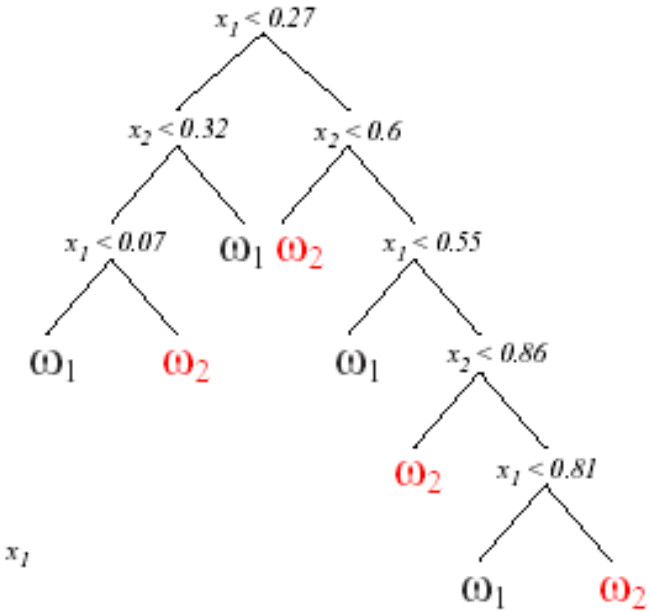

$\mathcal{R}_1$
$\mathcal{R}_2$
$\mathcal{R}_1$
$\mathcal{R}_2$
$\mathcal{R}_1$
0
.2
.4
.6
.8
1
x_1

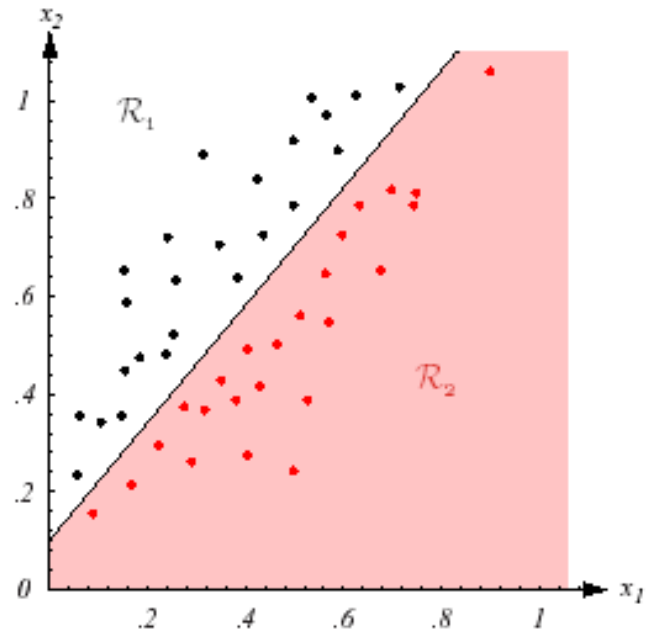

$x_1 < 0.6$ 1.0
.88
$x_2 < 0.32$
$x_2 < 0.61$
.65
.81
$x_1 < 0.35$
ω_1
ω_2
$x_1 < 0.69$
1.0
ω_2
ω_1
ω_2
ω_1

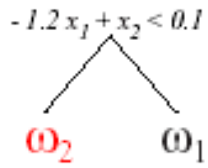

x_2
$\mathcal{R}_1$
$\mathcal{R}_2$
$\mathcal{R}_1$
$\mathcal{R}_2$
$\mathcal{R}_1$
0
.2
.4
.6
.8
1
x_1


$x_2 < 0.33$ 1.0
.59
$x_2 < 0.09$
$x_1 < 0.6$
.76
ω_1
ω_2
ω_1
$x_1 < 0.69$
.92
ω_2
ω_1


x_2
1
.8
.6
.4
.2
0
$\mathcal{R}_1$
$\mathcal{R}_2$
.2
.4
.6
.8
1
x_1


$x_1 < 0.27$
$x_2 < 0.32$
$x_2 < 0.6$
$x_1 < 0.07$
ω_1
ω_2
$x_1 < 0.55$
ω_1
ω_2
ω_1
$x_2 < 0.86$
ω_2
$x_1 < 0.81$
ω_1
ω_2


x_2
1
.8
.6
.4
.2
0
$\mathcal{R}_1$
$\mathcal{R}_2$
.2
.4
.6
.8
1
x_1


$-1.2\,x_1 + x_2 < 0.1$
ω_2
ω_1

Classification performance

How to evaluate the performance of the classifiers?

- Evaluation on the training set (optimistic error estimate).
- Evaluation on the test set (pessimistic error estimate). Intuitive evaluation is misleading, different errors may have different significance

Confusion matrix

		Predicted		
		Greyhound	Mastiff	Samoyed
Actual	Greyhound	250	25	18
	Mastiff	21	320	24
	Samoyed	22	12	180

Confusion matrix 2 x 2

Machine learing \ Manual counting	True	False
True	True Positive (TP)	False Positive (FP)
False	False Negative (FN)	True Negative (TN)

Equations:

$$\text{False positive rate (FPR)} = \frac{FP}{FP+TN}$$

$$\text{False negative rate (FNR)} = \frac{FN}{FN+TP}$$

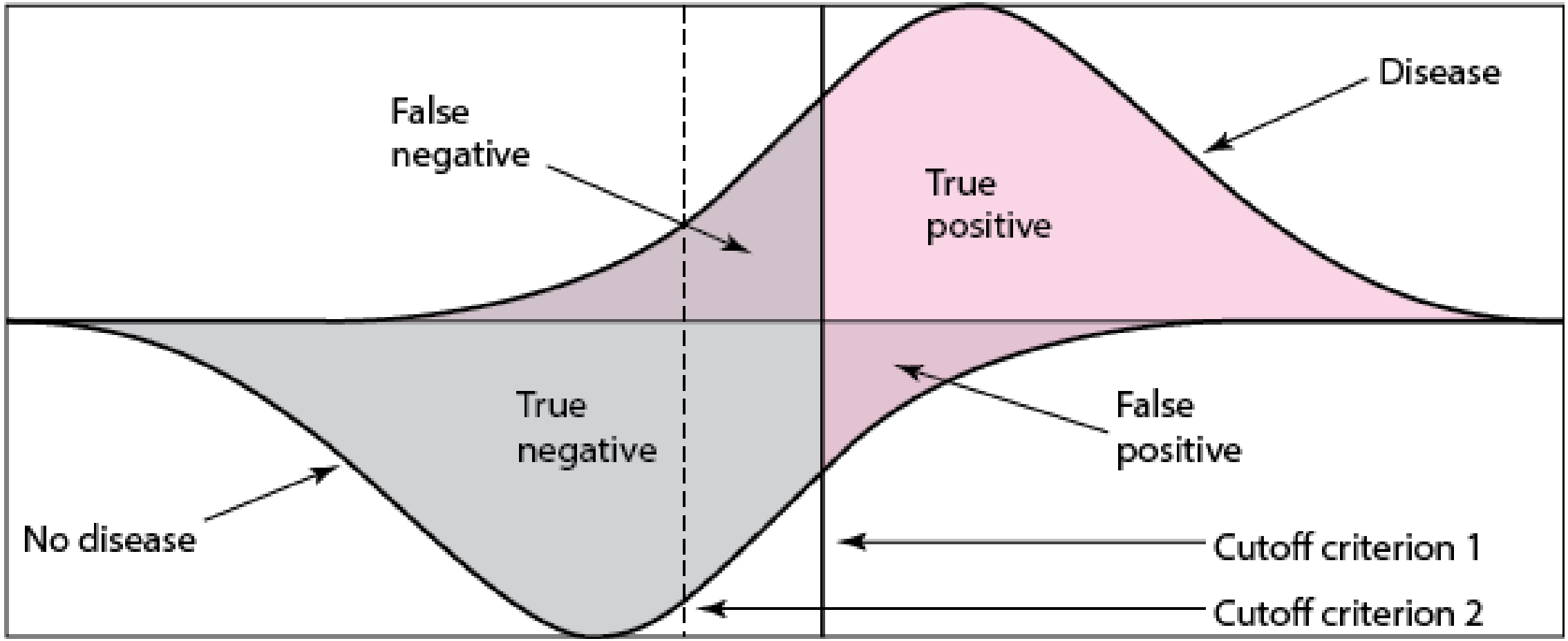
$$\text{Sensitivity} = \frac{TP}{TP+FN}$$

$$\text{Specificity} = \frac{TN}{TN+FP}$$

$$\text{Youden index} = \text{Sensitivity} + \text{Specificity} - 1$$

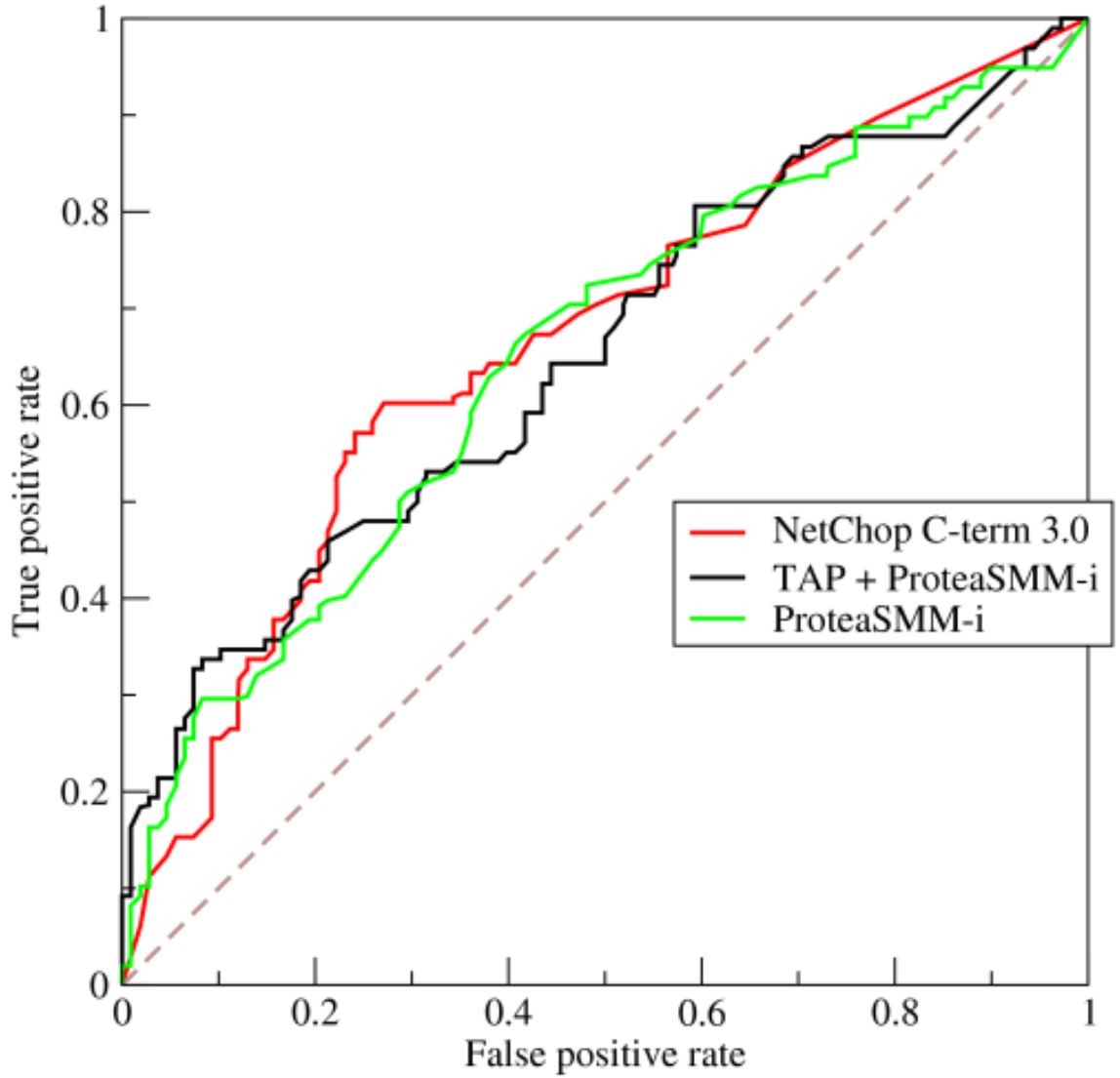
$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN}$$

Tuning the classifier

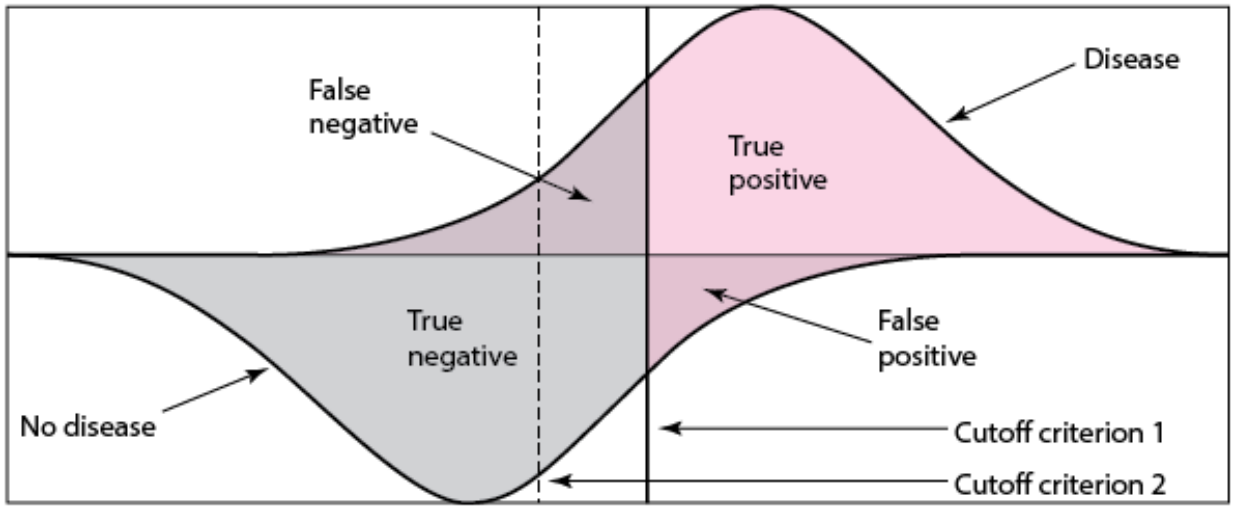


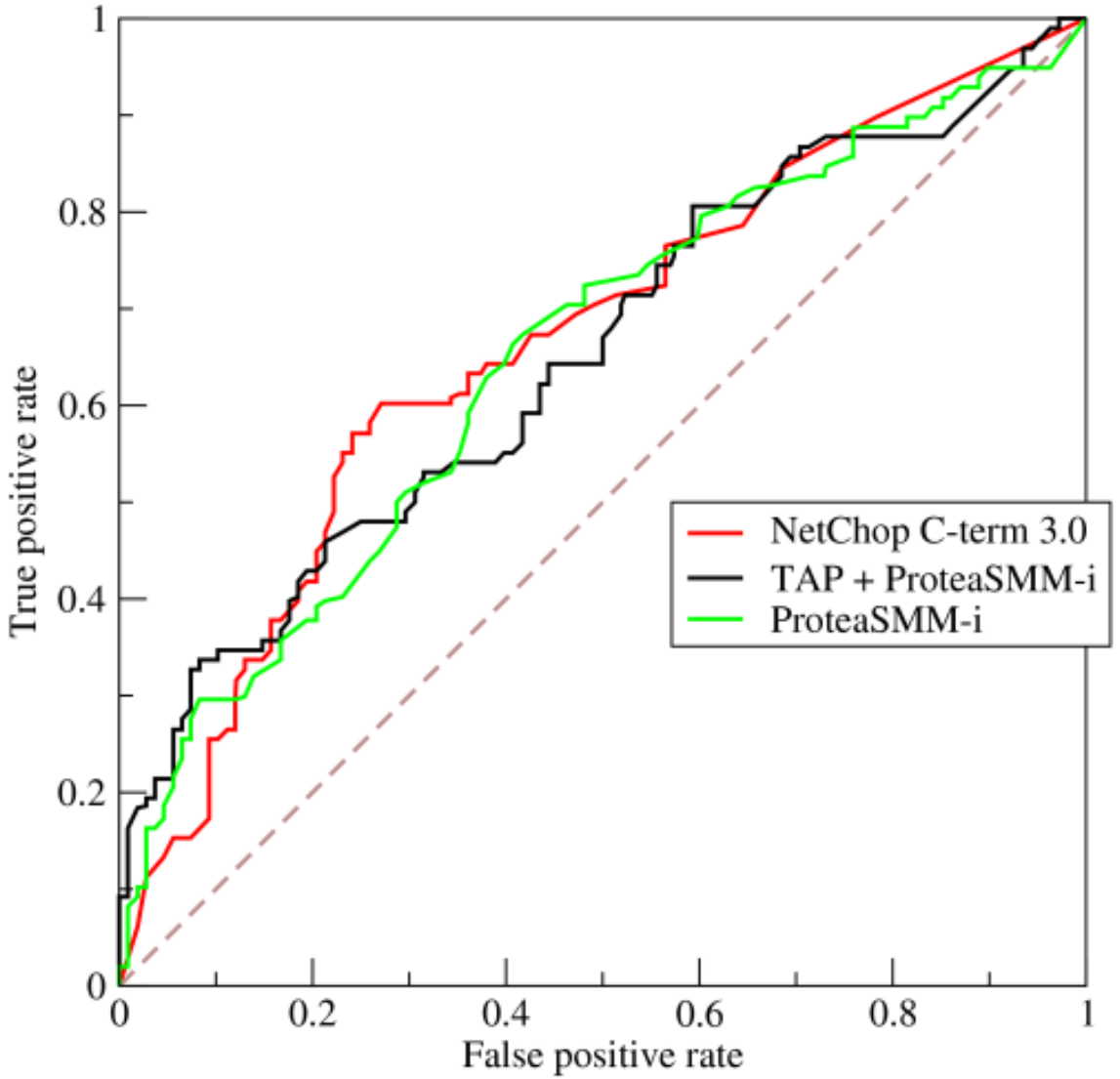
ROC curve

Receiver Operating Characteristic



ROC construction





Classification performance

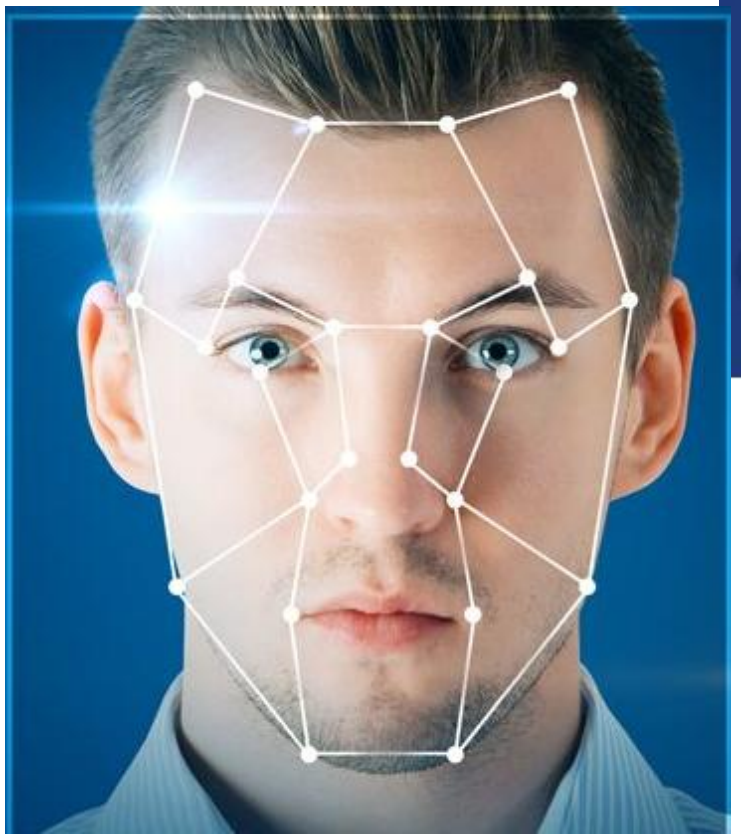
How to increase the performance?

- other features
- more features (dangerous – curse of dimensionality!)
- other (larger, better) training sets
- other parametric models
- other classifiers
- **combining different classifiers**

Combining classifiers

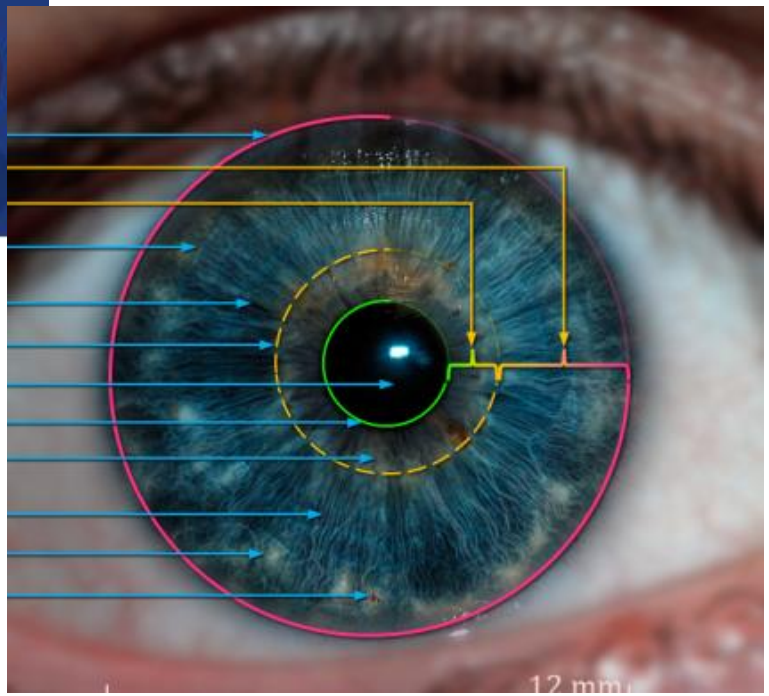
- Several *independent* classifiers
- Averaging of noisy results
- Suppression of extremes
- No guarantee of improvement over the best classifier

Combining classifiers in biometrics

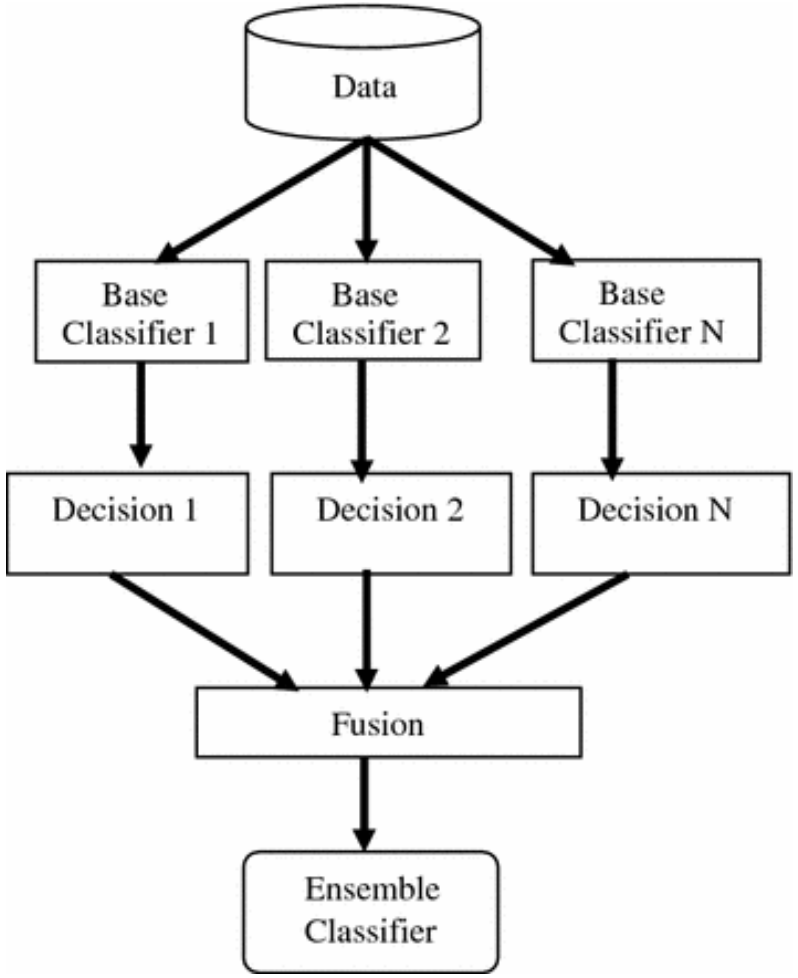








Combining deterministic decisions: voting

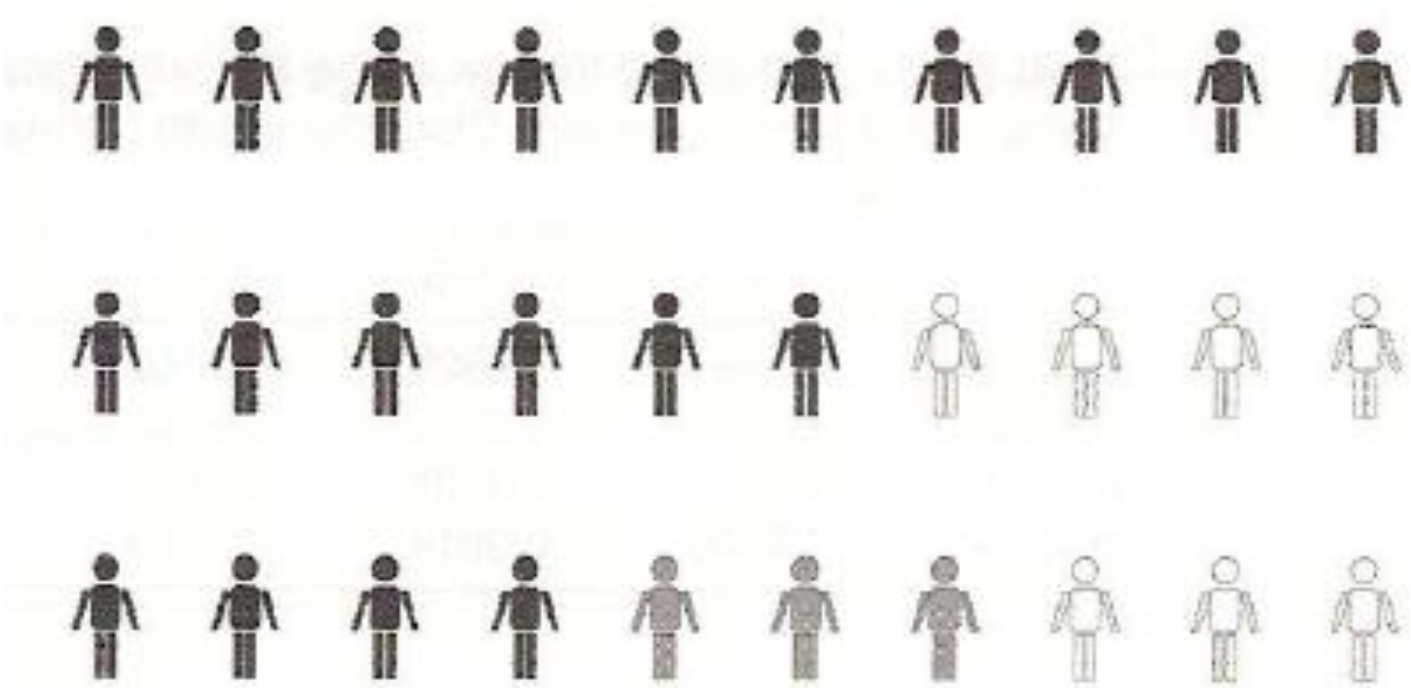




$$P_{maj} = \sum_{m=\lfloor L/2 \rfloor + 1}^{L} \binom{L}{m} p^m (1-p)^{L-m}$$

$P > p$ iff $p > 0.5$

Combining deterministic classifiers: voting



Weighted voting: incorporating the expert knowledge

Majority vote – probability of success

Většinové hlasování :

$$P_{maj} = \sum_{m=\lfloor L/2 \rfloor+1}^{L} \binom{L}{m} p^m (1-p)^{L-m}$$

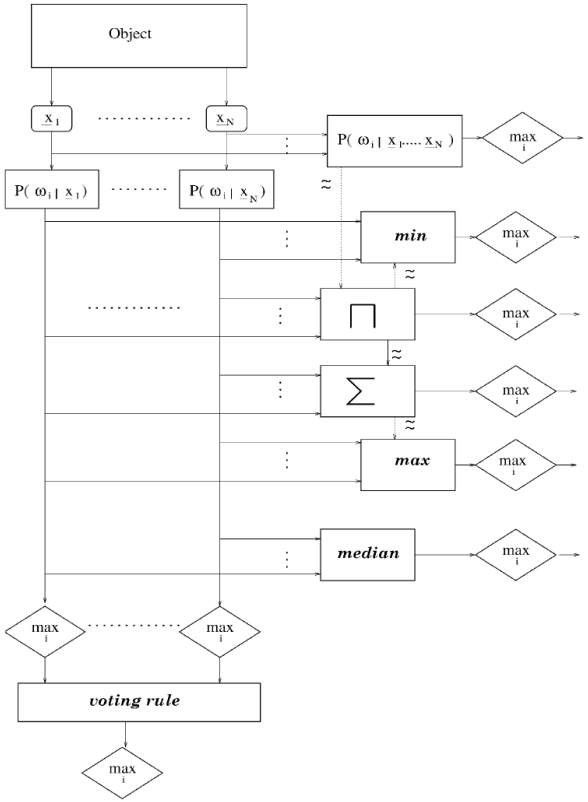
kde L je počet **nezávislých** klasifikátorů a p je jejich úspěšnost.

	$L=3$	$L=5$	$L=7$	$L=9$
$p=0.6$	0.6480	0.6826	0.7102	0.7334
$p=0.7$	0.7840	0.8369	0.8740	0.9012
$p=0.8$	0.8960	0.9421	0.9667	0.9804
$p=0.9$	0.9720	0.9914	0.9973	0.9991

Combining probabilistic classifiers

$$\max \quad p(\omega_i|x_1,\cdots,x_C)$$

$$p(\omega_i)\cdot p(x_1,\cdots,x_C|\omega_i) \quad p(x_1,\cdots,x_C|\omega_i)=\prod_{j=1}^{C} p(x_j|\omega_i)$$



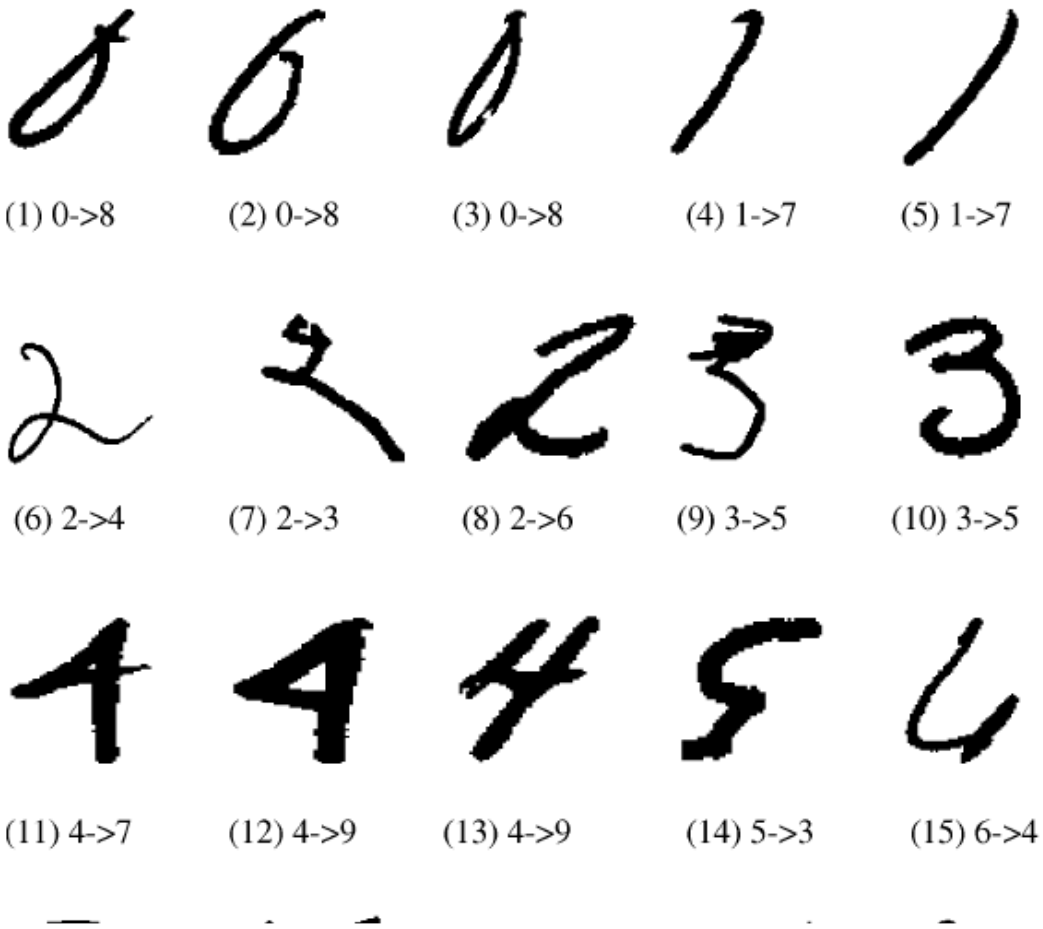
Handwritten digit recognition

TABLE 2
THE CLASSIFICATION RATE FOR EACH CLASSIFIER

Individual classifier	Classification rate %
Structural:	90.85
Gaussian:	93.93
Neural Net:	93.2
HMM:	94.77

TABLE 3
THE CLASSIFICATION RATE USING DIFFERENT COMBINING SCHEMES

Combining rule	Classification rate %
Majority Vote:	97.96
Sum rule:	98.05
Max rule:	93.93
Min rule:	86.00
Product rule:	84.69
Median rule:	98.19



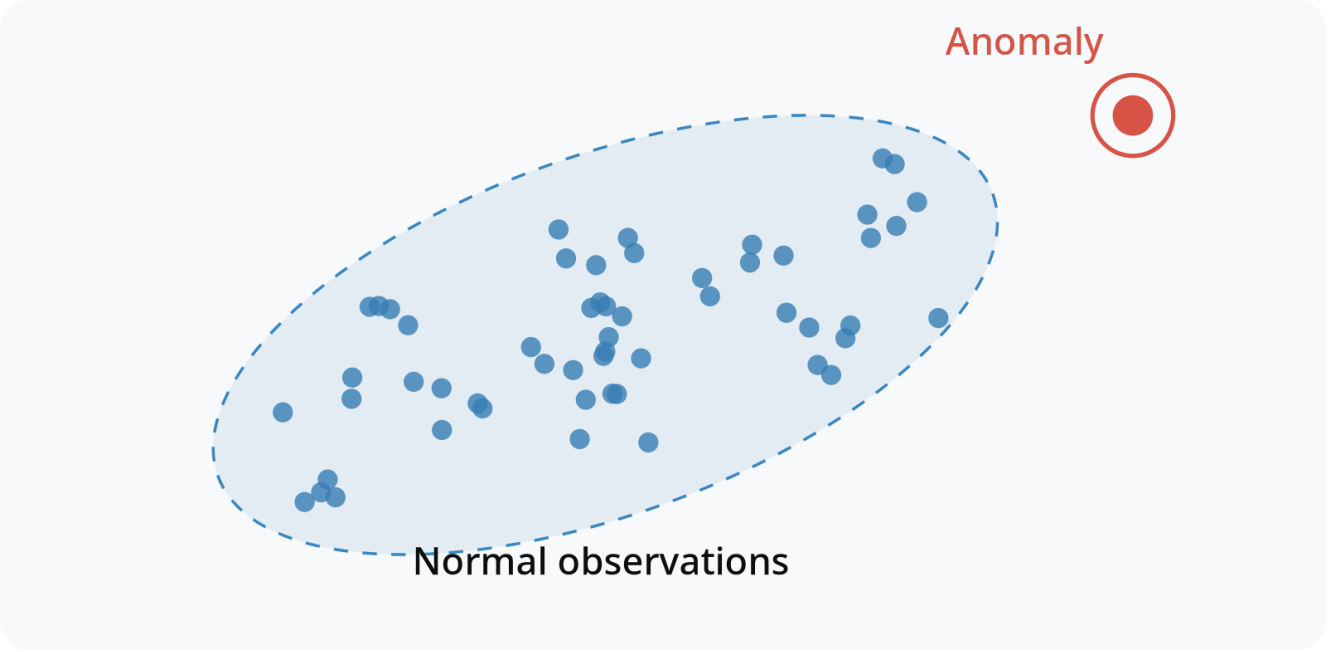
Reprinted from J. Kittler et al., PAMI, 1998

Anomaly detection

Single class of „normal“ objects

Anomaly – an object „far from the expectation“

Training on normal objects only



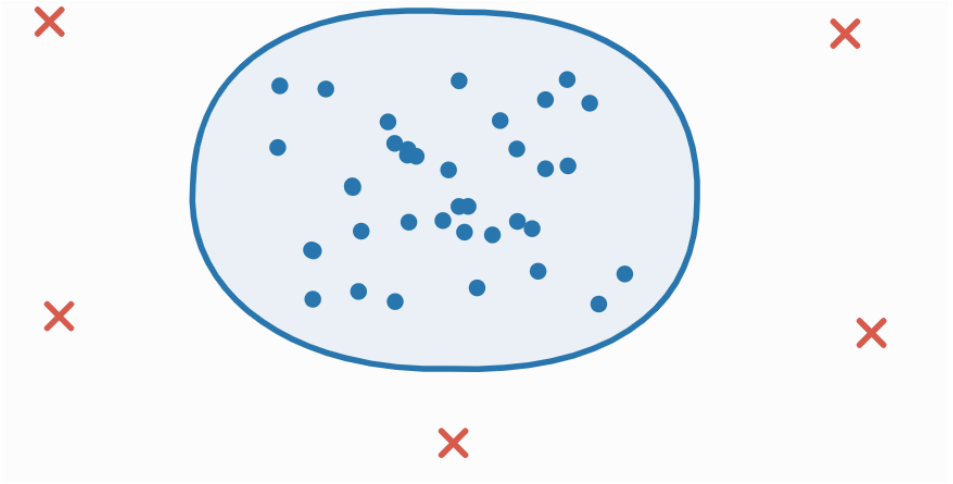
Anomaly detection

Learn normality $\longrightarrow$ Measure deviation $\longrightarrow$ Decide anomaly.

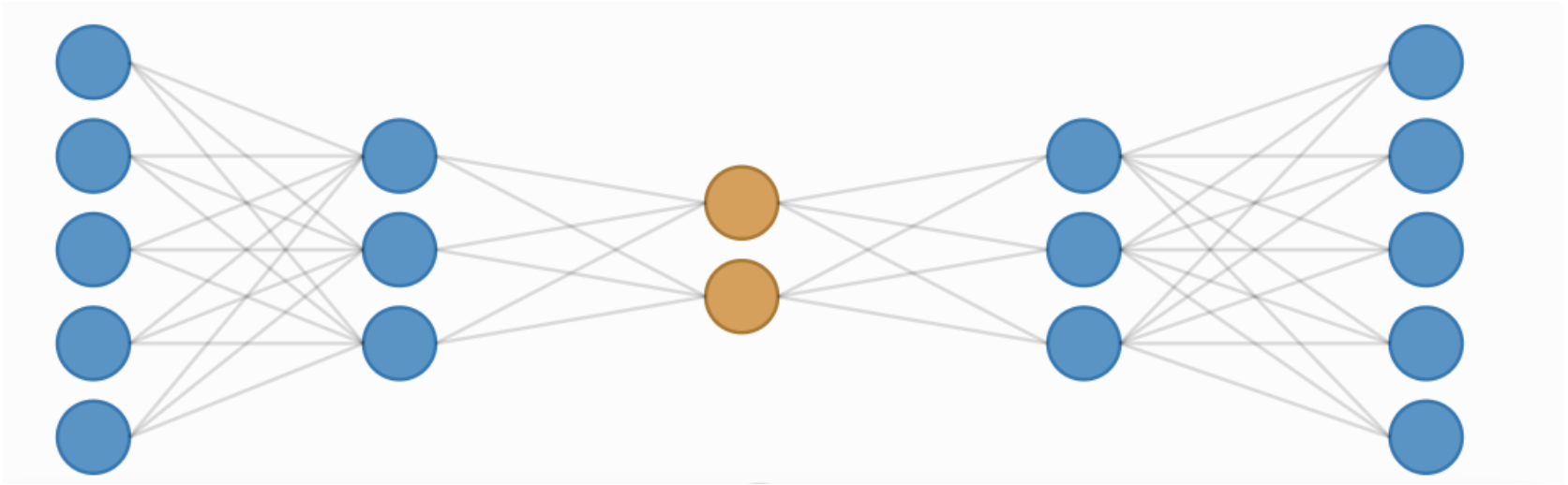
Distance measure and thresholding

$$d_M^2(x) = (x - \mu)^T C^{-1} (x - \mu)$$

One-class SVM

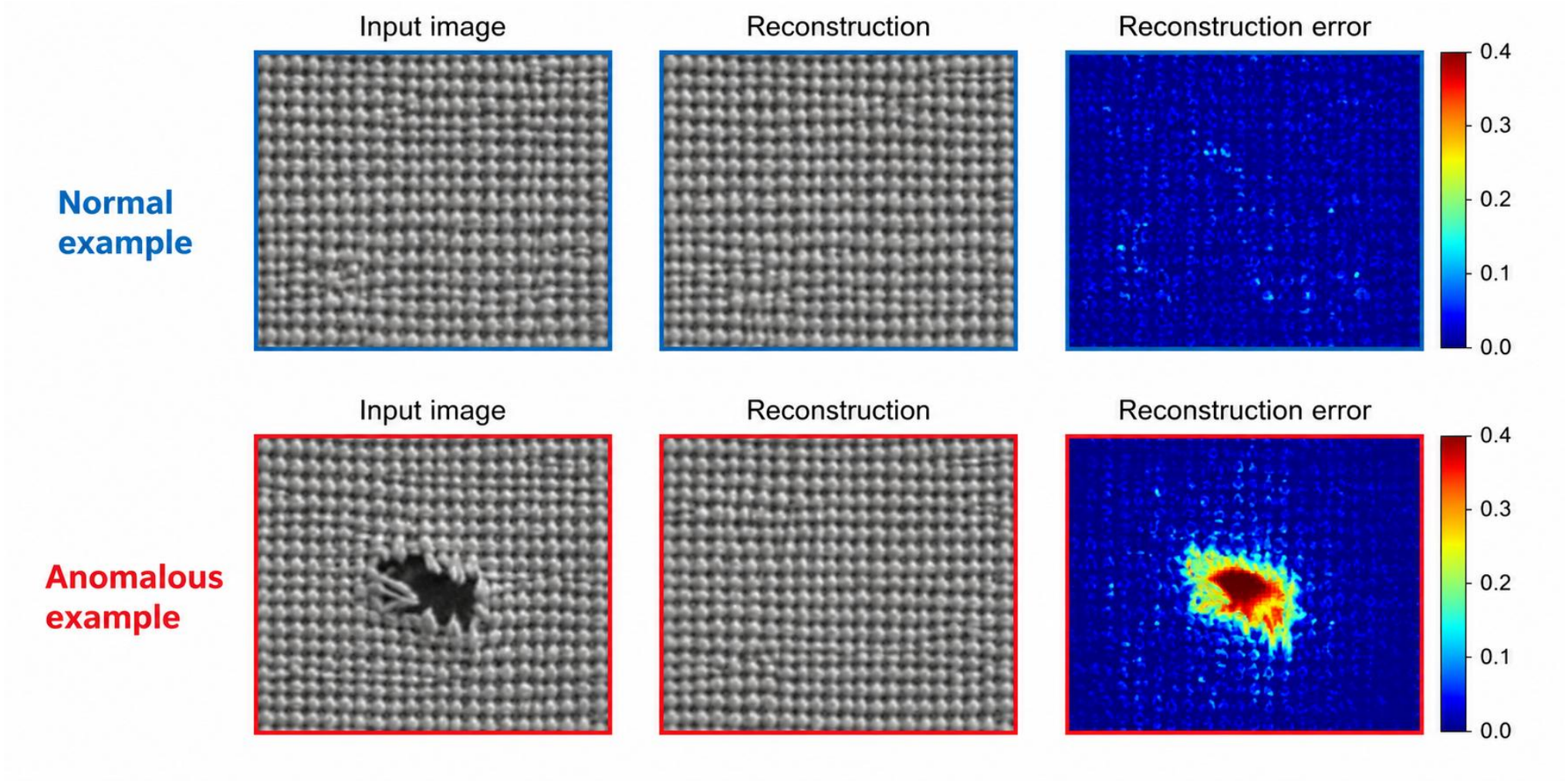


Autoencoder



- Normal data are reconstructed with small error
- Anomalies -> big error
- What is „big“?

Autoencoder



Unsupervised Classification

Cluster analysis

Training set is not available, No. of classes may not be *a priori* known

What are clusters?

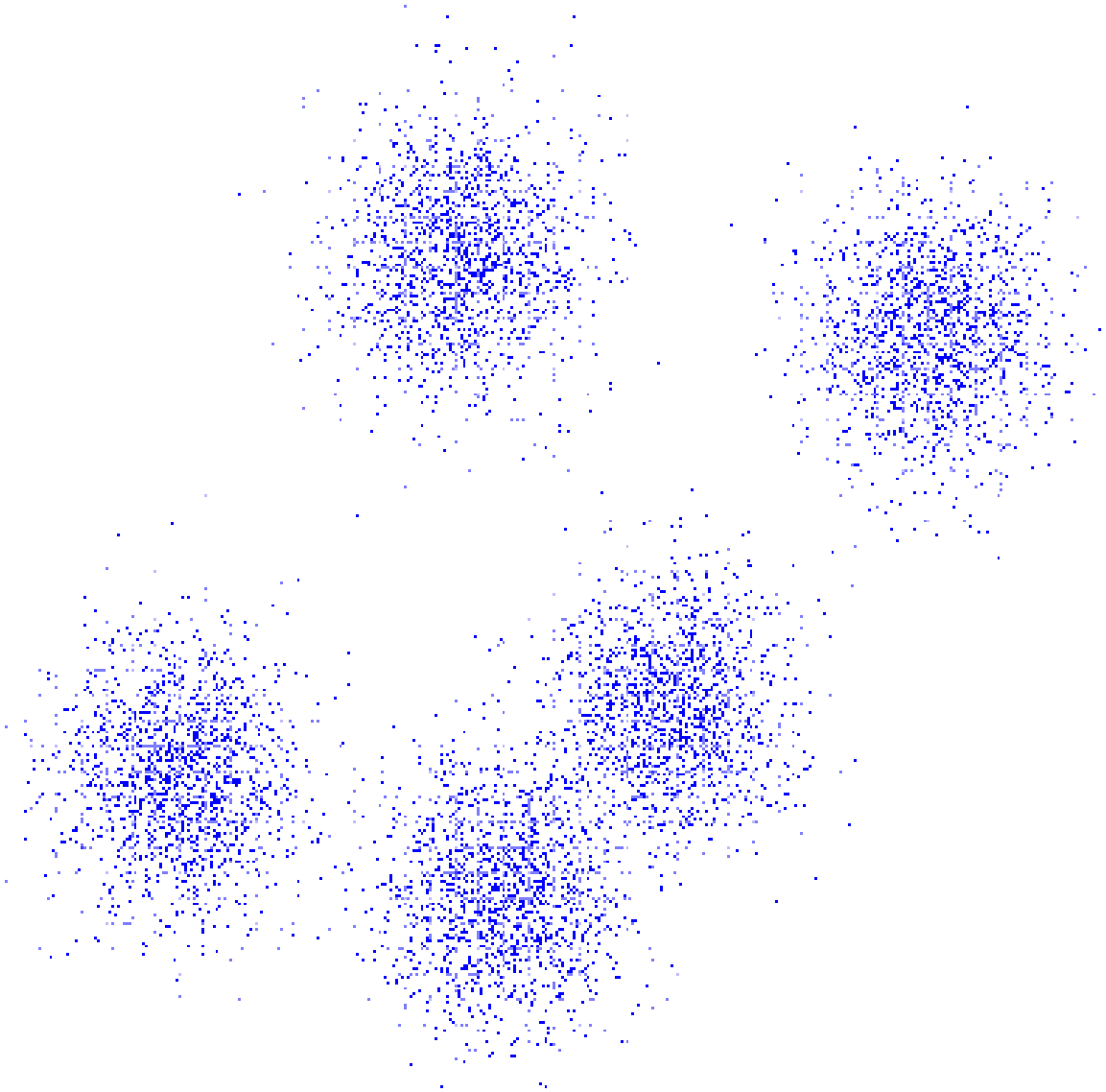
Intuitive meaning

- compact, well-separated subsets

Formal definition

- many attempts, mostly unsatisfactory
  (*t*-connectivity, diameter constraint, ...)
- any partition of the data into disjoint subsets

What are clusters?



How to compare different clusterings?

Ward criterion

Variance measure *J* should be minimized

$$J = \sum_{i=1}^{N} \sum_{\mathbf{x} \in C_i} \|\mathbf{x} - \mathbf{m}_i\|^2$$

Drawback – only clusterings with the same *N* can be compared.
Global minimum

J = 0 is reached in the degenerated case.

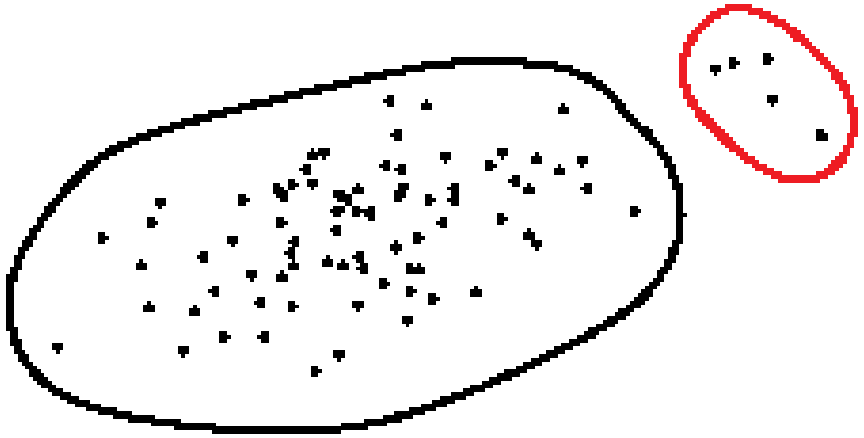
Minimization of *J*

Drawbacks

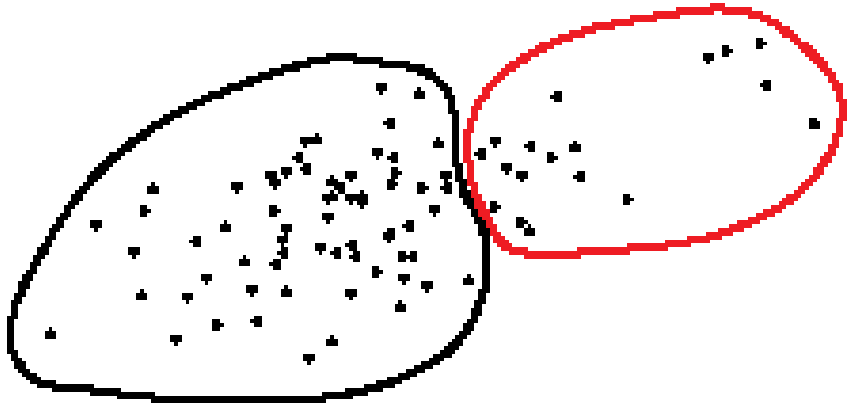
The results are sometimes “intuitively wrong”

because *J* prefers clusters with approx the same size

An example of a „wrong“ result

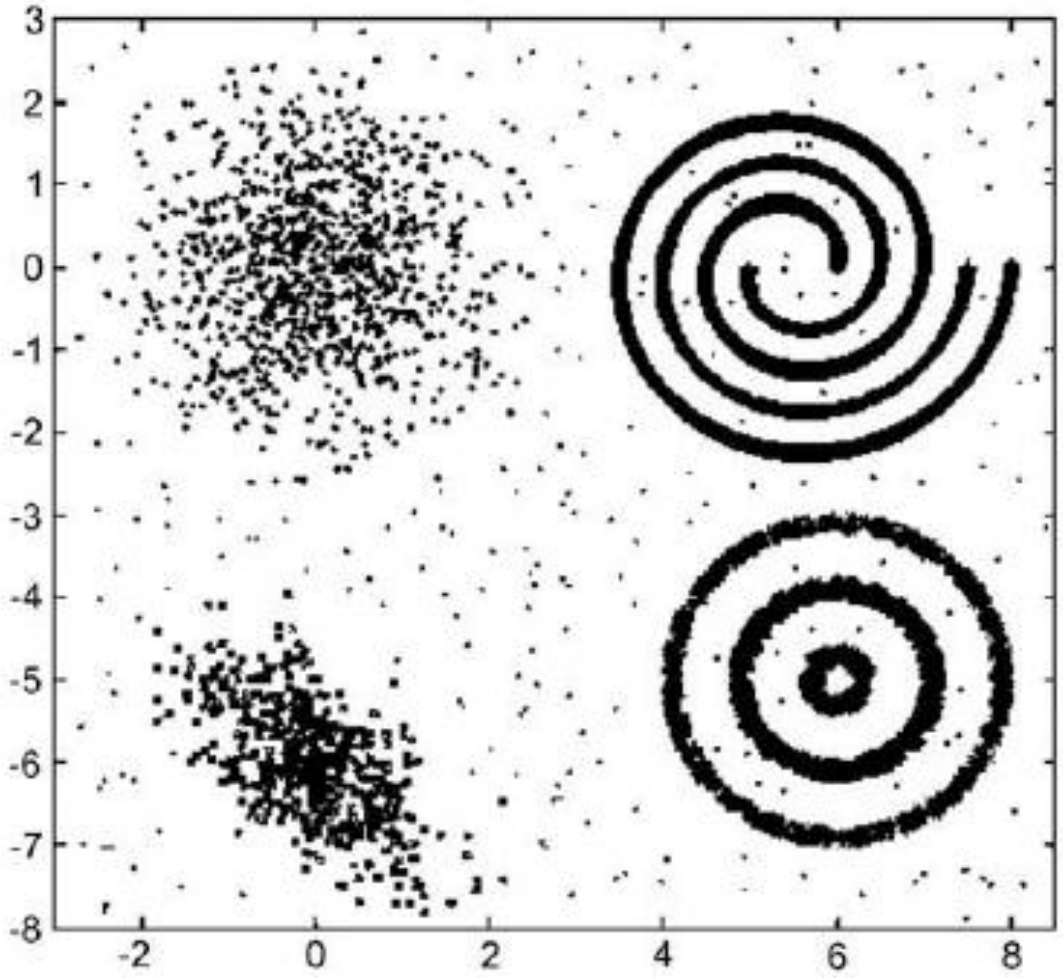


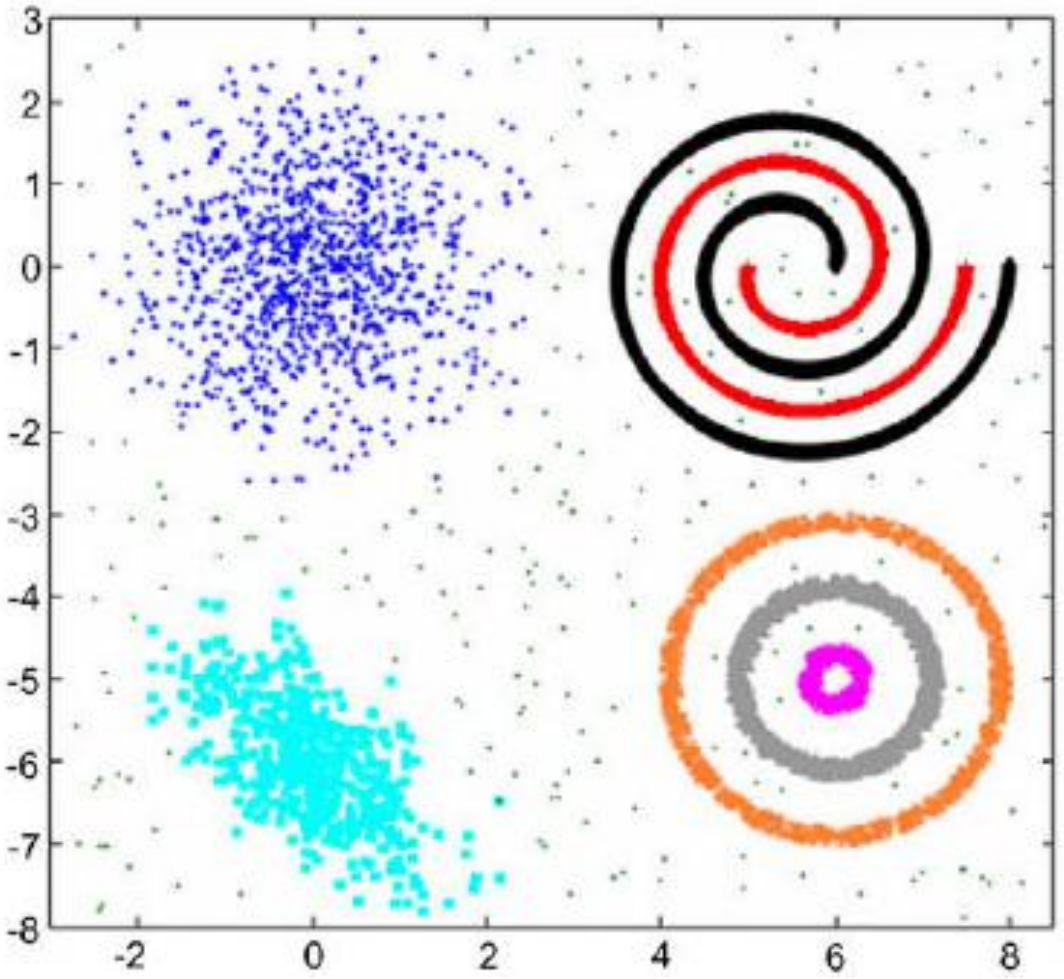
$J_e = large$



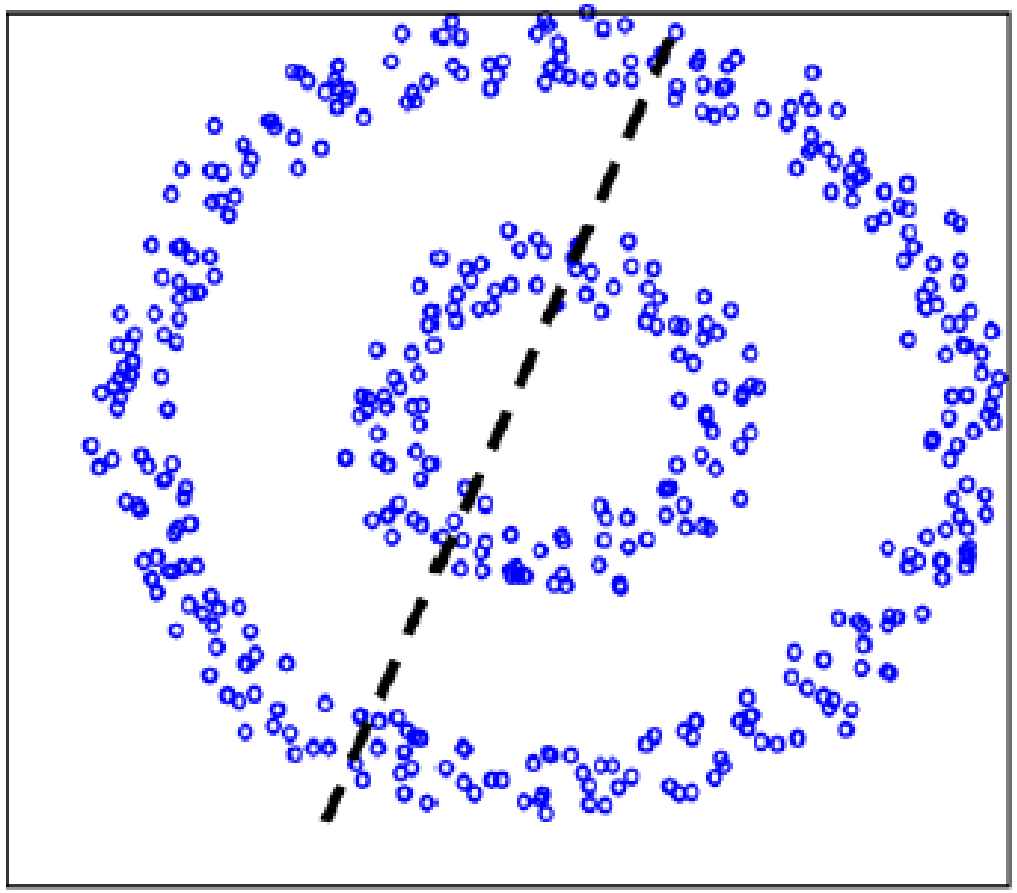
$J_e = small$

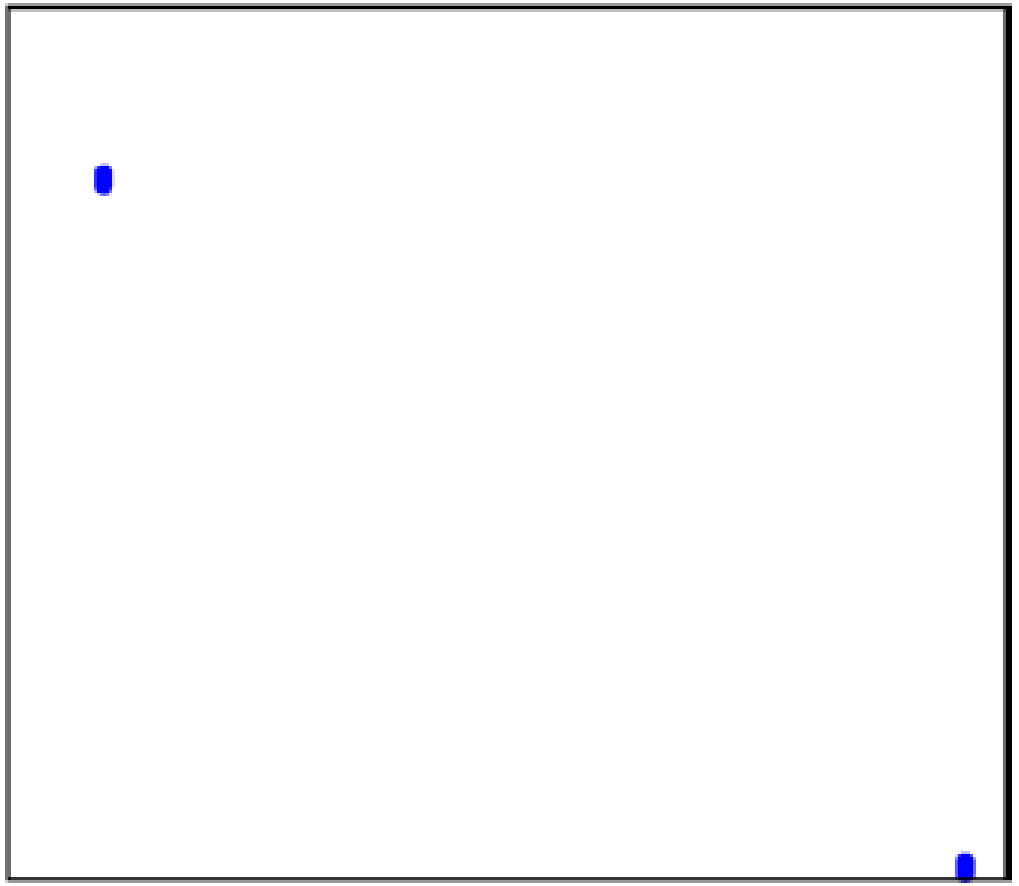
Drawback – assumes uncorrelated features and “convex” clusters





Solution – proper transform of the features





Clustering techniques

Iterative methods

- typically if *N* is given

Hierarchical methods

- typically if *N* is unknown

Other methods

- sequential, graph-based, branch & bound, fuzzy, genetic, model-based, etc.

Sequential clustering

- *N* may be unknown
- Very fast but not very good
- Each point is considered only once

Idea:

A new point is either added to an existing cluster or it forms a new cluster. The decision is based on the user-defined distance threshold.

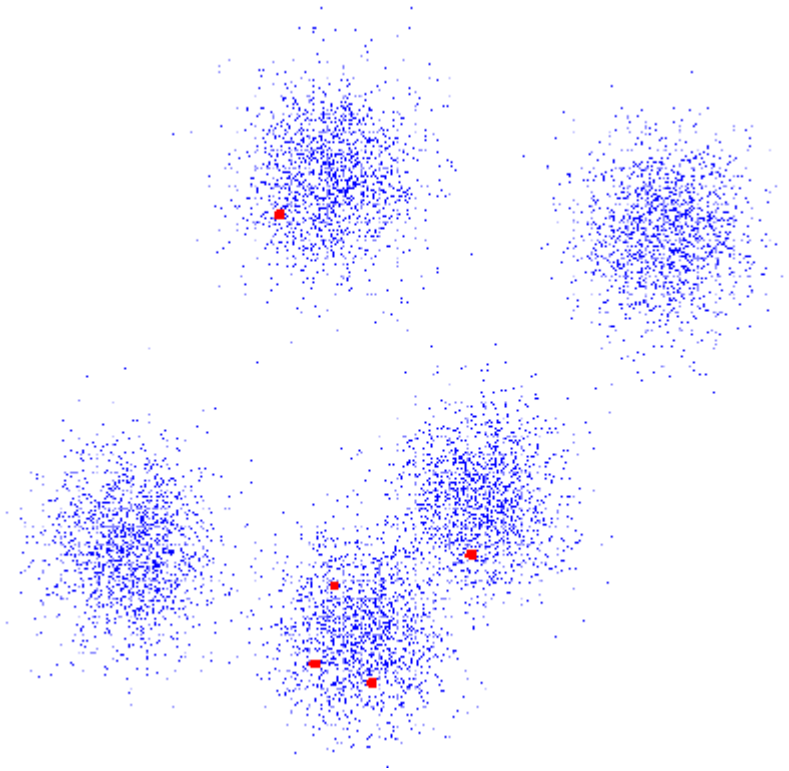
Sequential clustering

Drawbacks:

- Dependence on the distance threshold
- Dependence on the order of data points

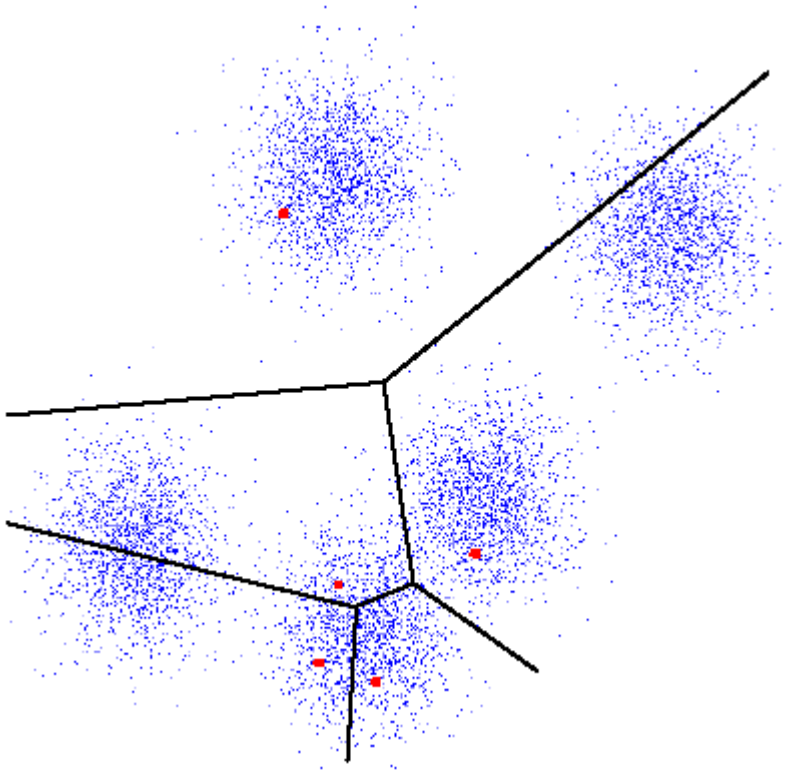
N-means clustering

1. Select *N* initial cluster centroids.



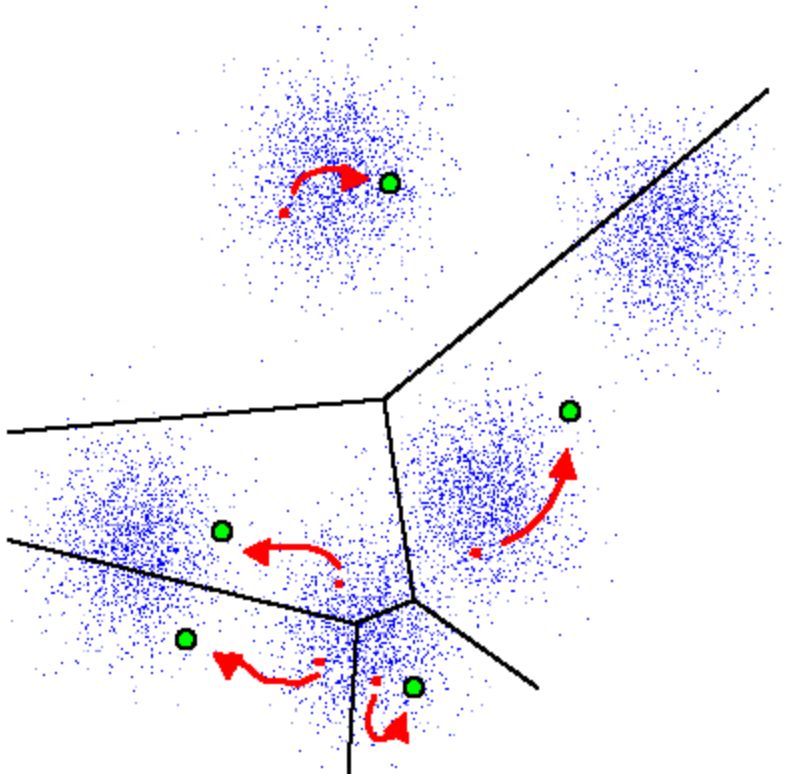
N-means clustering

2. Classify every point *x* according to minimum distance.



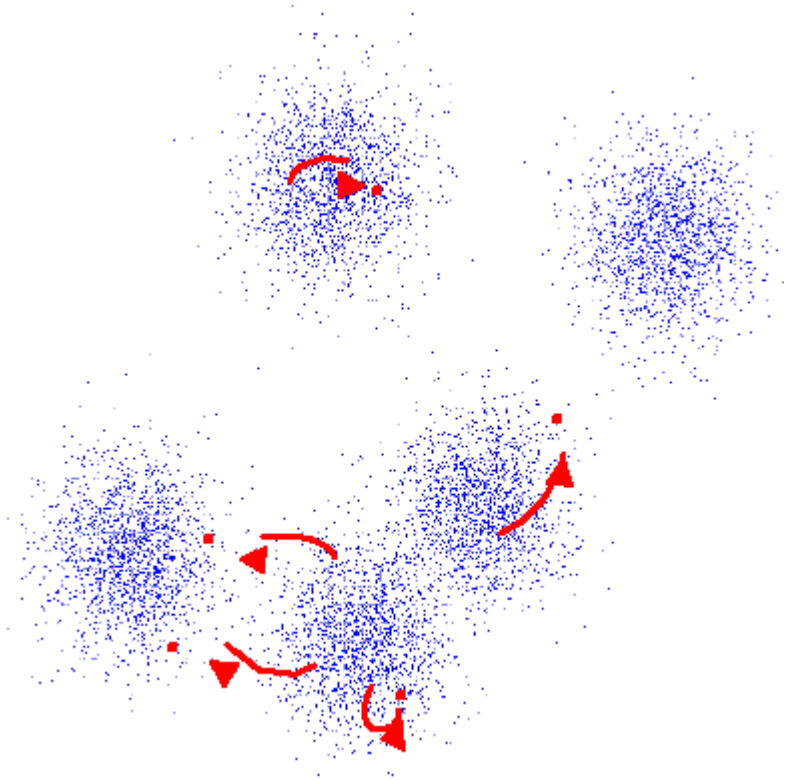
N-means clustering

3. Recalculate the cluster centroids.



N-means clustering

4. If the centroids did not change then STOP else GOTO 2.



N-means clustering

Drawbacks

- The result depends on the initialization
- The alg terminates in a finite time in a local minimum of J
- The results are sometimes “intuitively wrong”

N-means ++ initialization

Initialization that takes into account the distribution of data

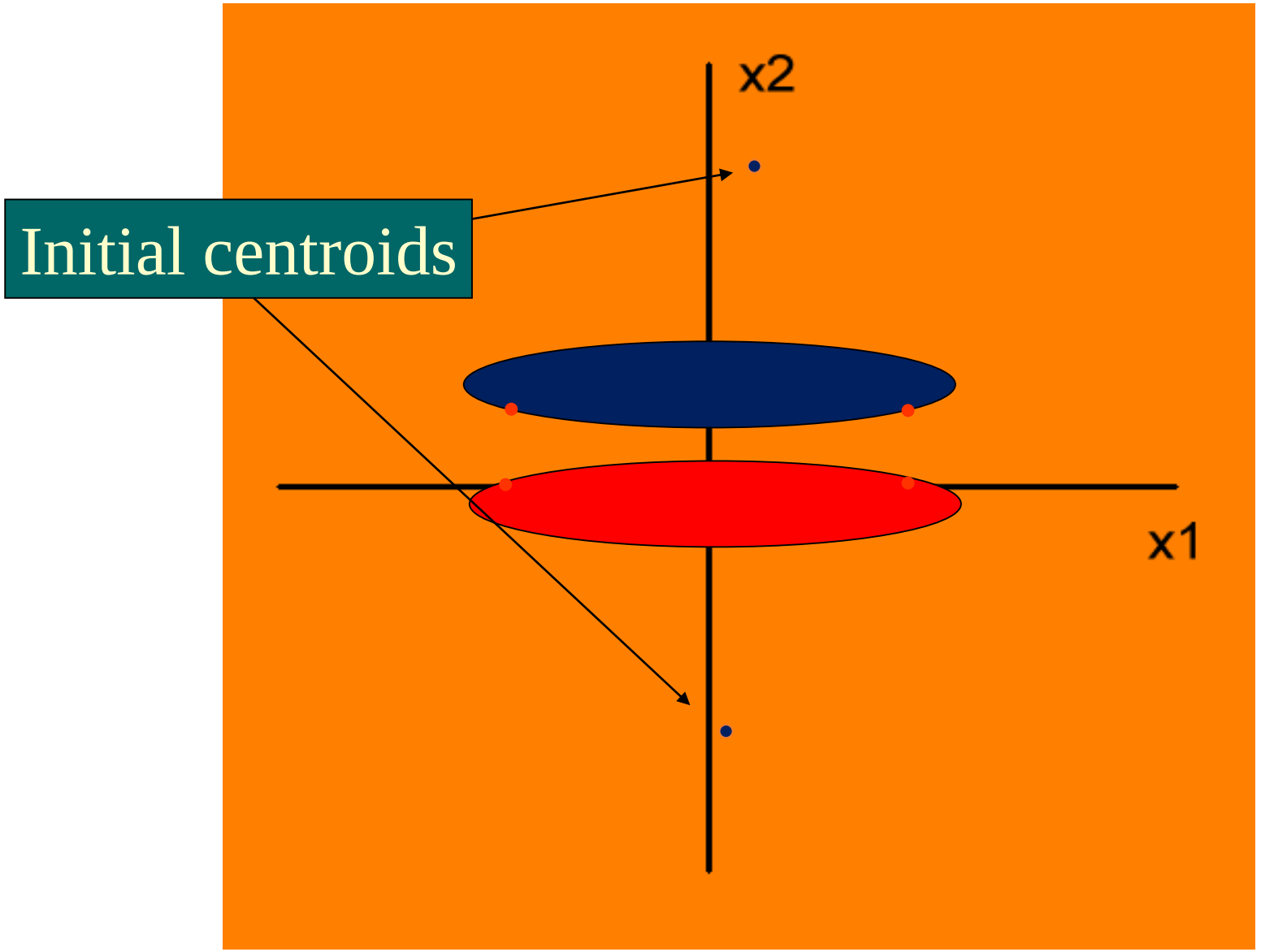
1. Choose the first centroid randomly
2. For each datapoint, calculate the minimum distance
3. Choose the next centroid randomly with probability proportional to its minimum distance
4. Repeat until *N* centroids have been chosen

N-means clustering – an example

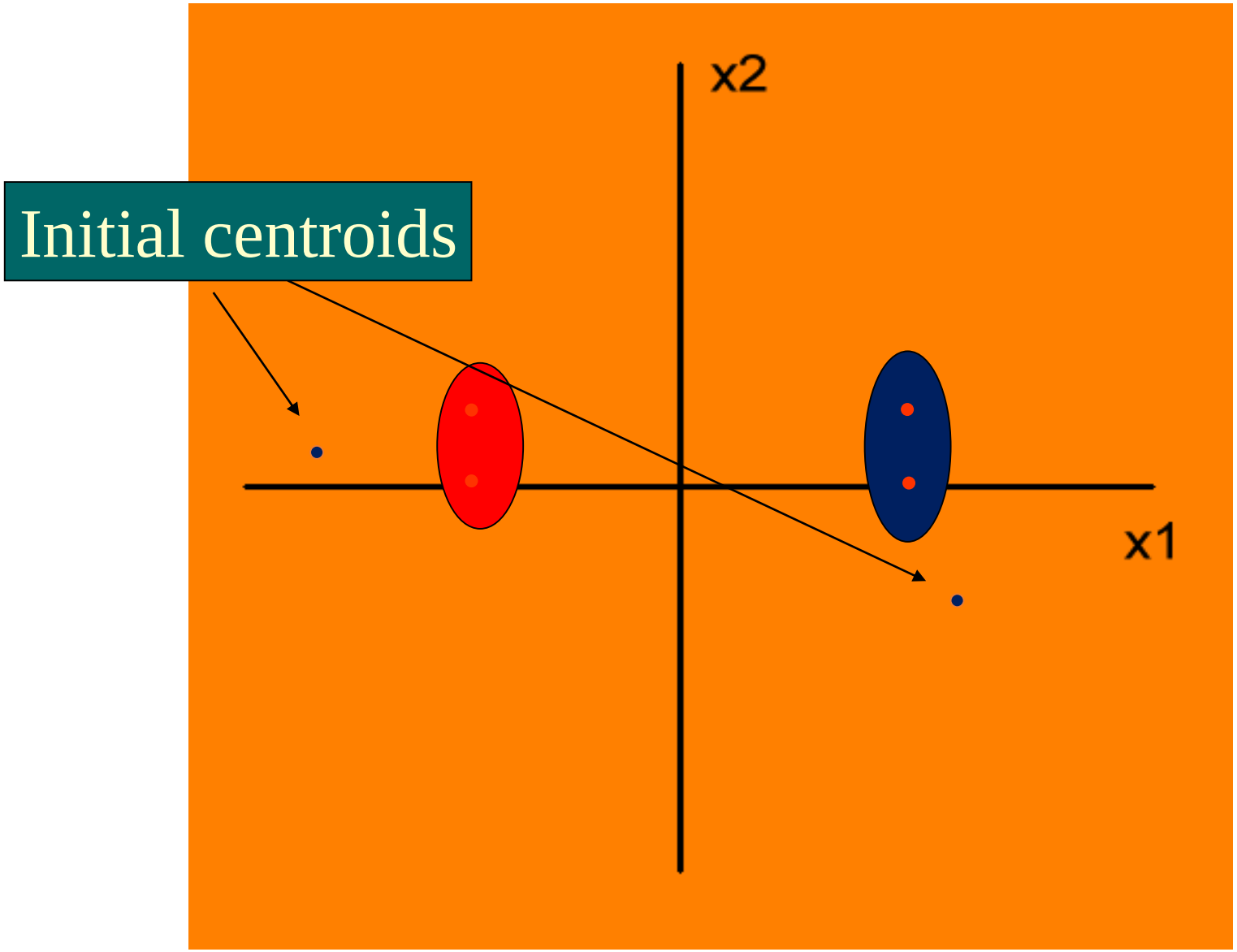
Two features, four points, two clusters ($N = 2$)

Different initializations → different clusterings

N-means clustering – an example



N-means clustering – an example



Iterative minimization of *J*

1. Let's have an initial clustering (by *N*-means)
2. For every point *x* do the following:

   Move *x* from its current cluster to another cluster, such that the decrease of *J* is maximized.
3. If all data points do not move, then STOP

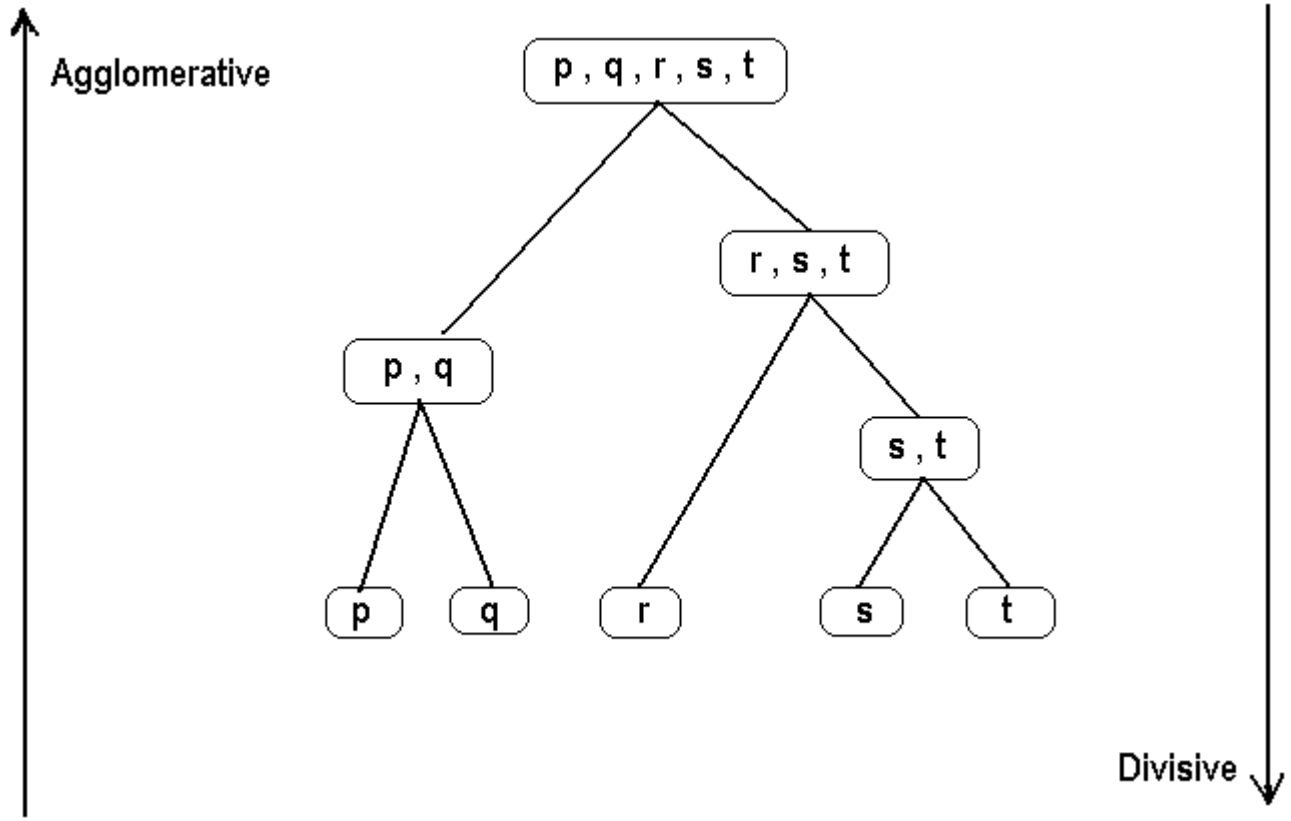
Iterative minimization of J

Drawbacks

The algorithm is optimal in each step but
in general global minimum of J is not reached.

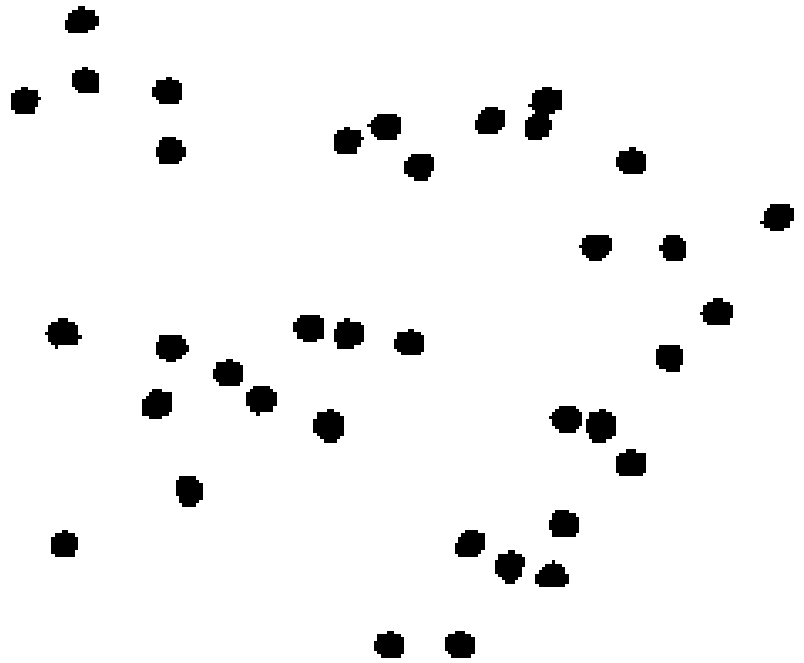
Hierarchical clustering

- Agglomerative clustering
- Divisive clustering



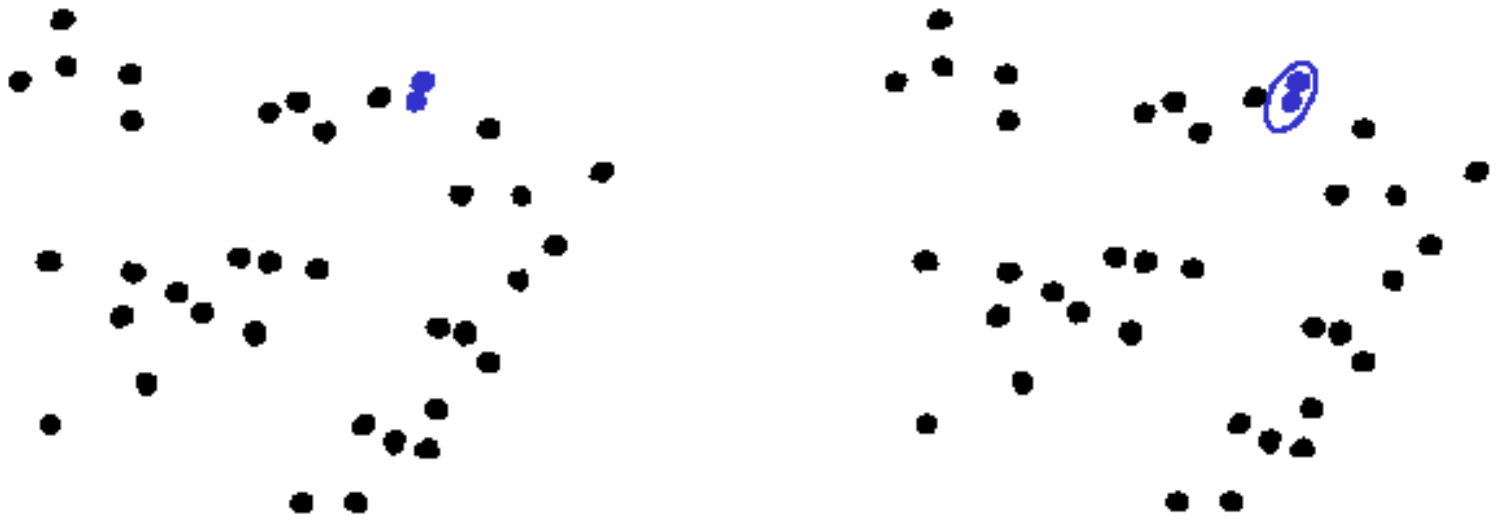
Basic agglomerative clustering

1. Each point = one cluster



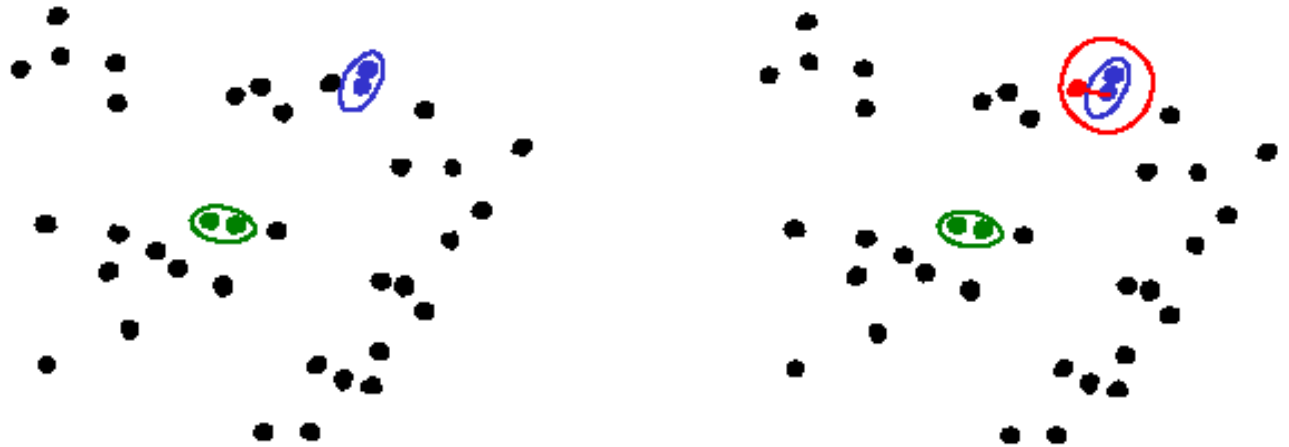
Basic agglomerative clustering

1. Each point = one cluster
2. Find two “nearest” or “most similar” clusters and merge them together



Basic agglomerative clustering

1. Each point = one cluster
2. Find two “nearest” or “most similar” clusters and merge them together
3. Repeat 2 until the stop constraint is reached



Basic agglomerative clustering

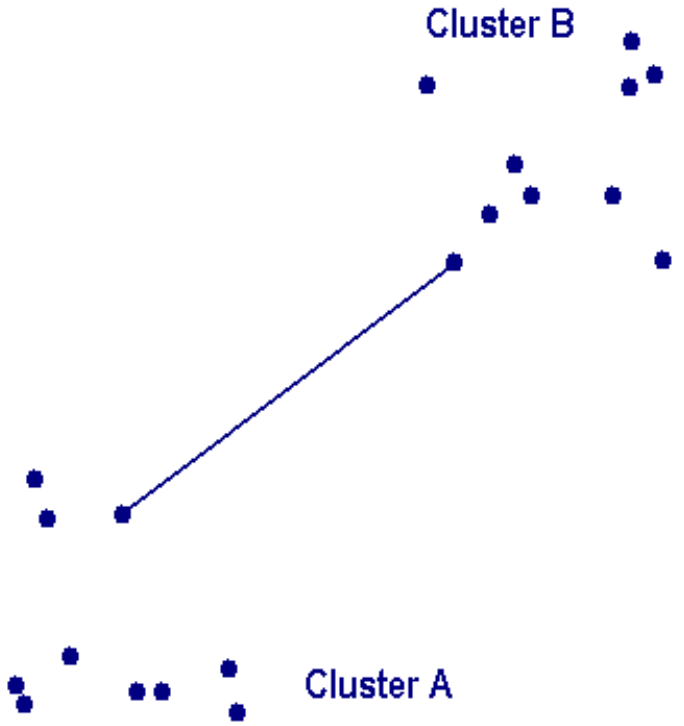
Particular implementations of this method differ from each other by:

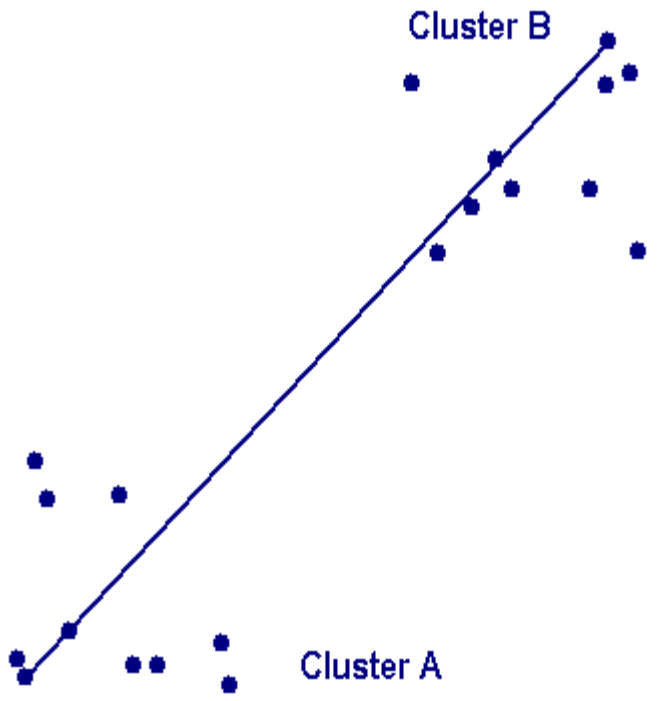
- The STOP constraints
- The distance/similarity measures used

Simple between-cluster distance measures

$$d(A,B) = d(m_1,m_2)$$

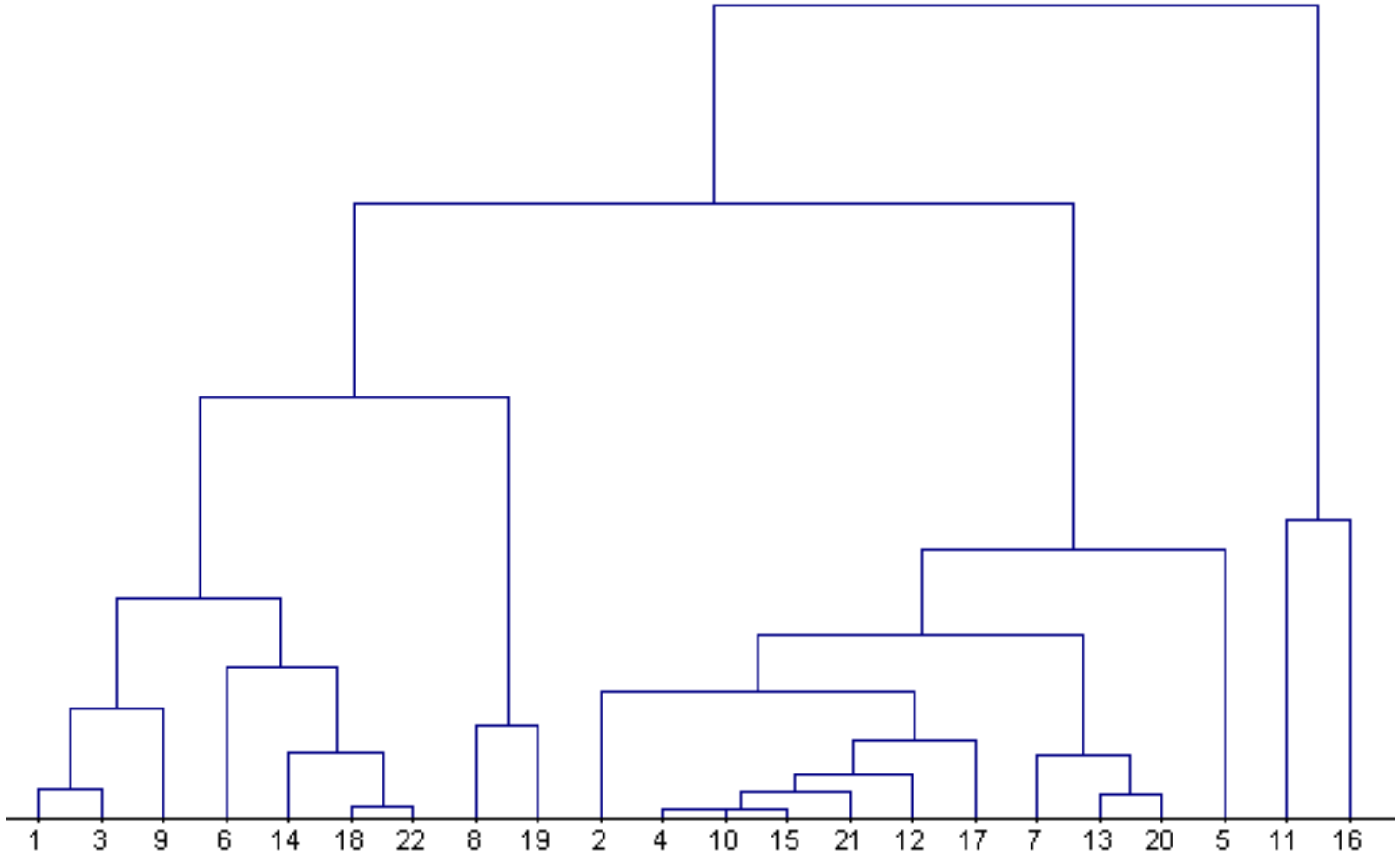
$d(A,B) = \min d(a,b)$ $\quad$ $d(A,B) = \max d(a,b)$





Agglomerative clustering

Representation by a clustering tree (dendrogram)



Definite agglomerative clustering

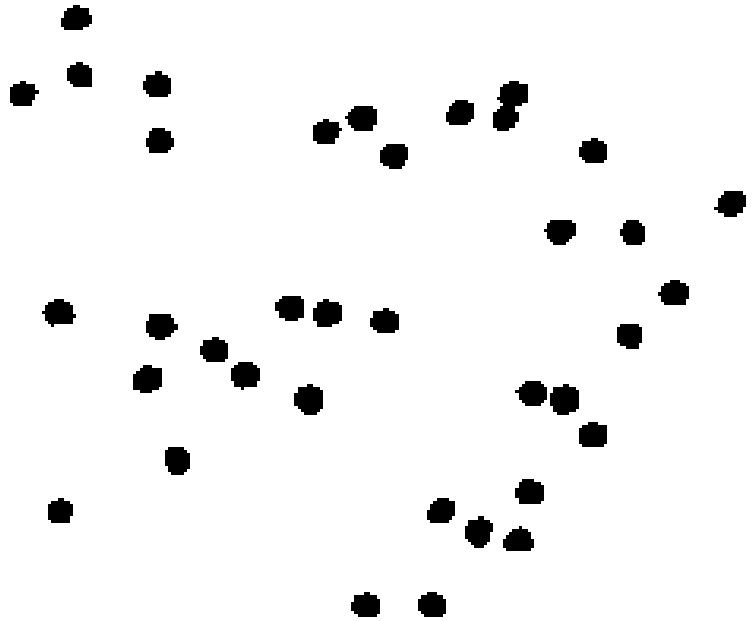
Basic algorithm – at a certain level, there may be multiple candidates of merging.

Random selection leads to ambiguities.

Definite algorithm – **all** such candidates are merged at that level → the number of new clusters emerging at one level may be greater than 1.

Basic divisive clustering

1. All points = one cluster



Basic divisive clustering

1. All points = one cluster
2. Divide the cluster into two parts A,B such that *d(A,B)* is maximized
3. Select a new cluster to split
4. Apply 2 to the selected cluster
5. Repeat 3-4 until the stop constraint is reached

Full search in STEP 2 is very expensive – O(2^n)

Suboptimal STEP 2 of divisive clustering

1. Find point *p* such that the mean of *d(p,x)* is maximized
2. Divide C into A U B, where B={*p*} (*p* is a seed point of a new cluster B)
3. For *x* from A calculate the mean dist *d(x,A)* and *d(x,B).* If *d(x,A)* > *d(x,B)* move *x* into B.
4. Repeat 3 for each *x* from A.

Suboptimal STEP 2 of divisive clustering

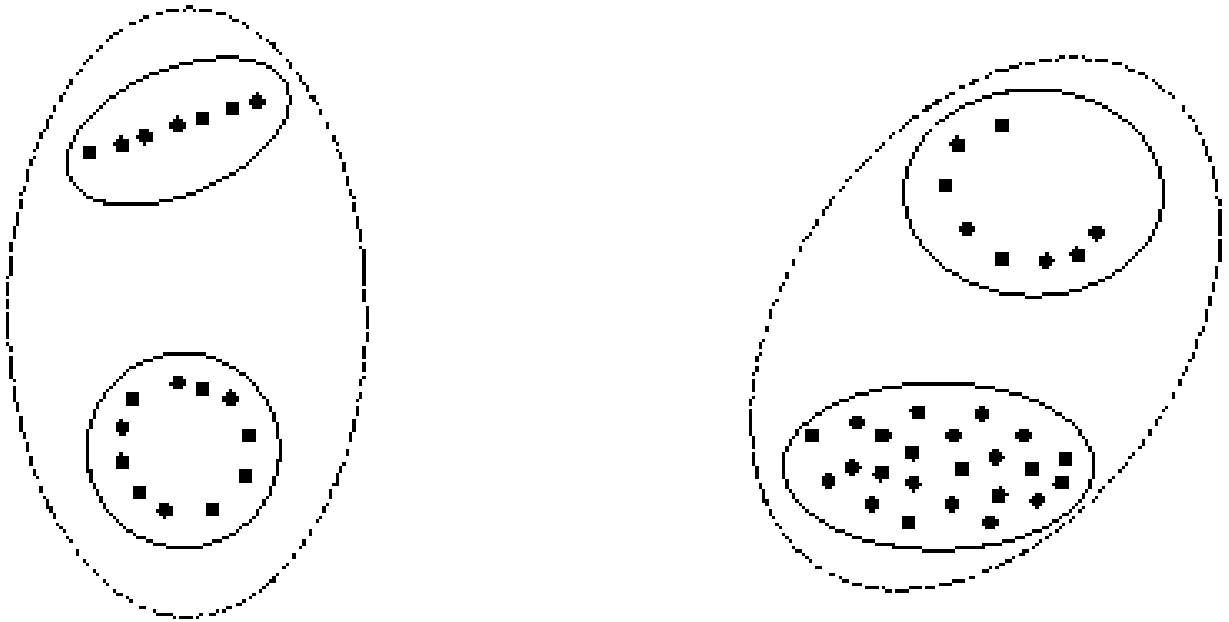
Alternatively, any iterative algorithm for N=2 can be used (N-means, J - minimization, ...)

STEP 3 of divisive clustering

Selecting the next cluster to split

- randomly (when performing complete decomposition)
- by maximum diameter
- by maximum variance
- by maximum *J*

How many clusters are there?



2 or 4?

Clustering is a very subjective task

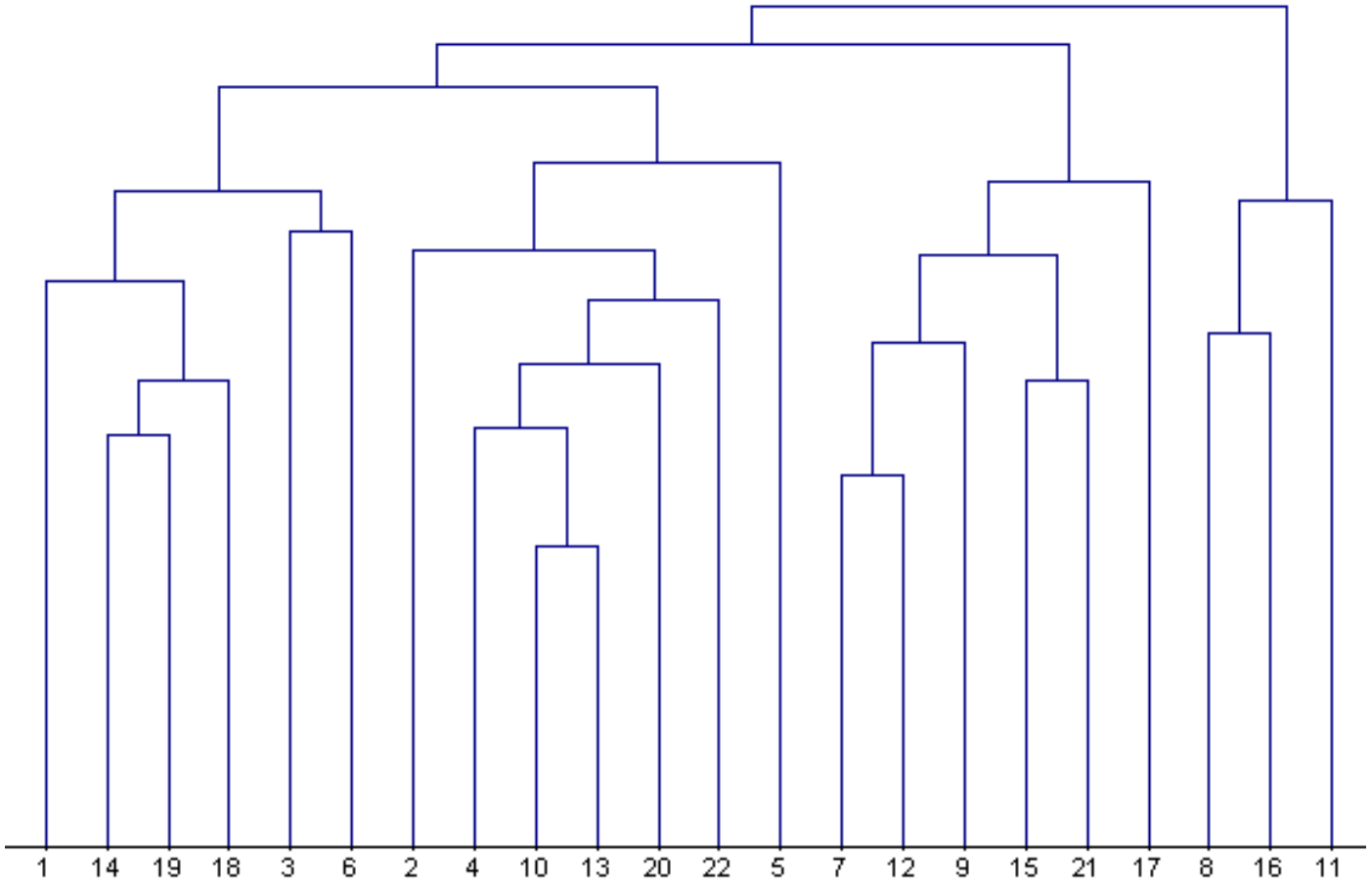
How many clusters are there?

- Difficult to answer even for humans
- “Clustering tendency”, “cluster validity”
- Hierarchical methods:

  N can be estimated from the complete dendrogram
- The methods minimizing a cost function

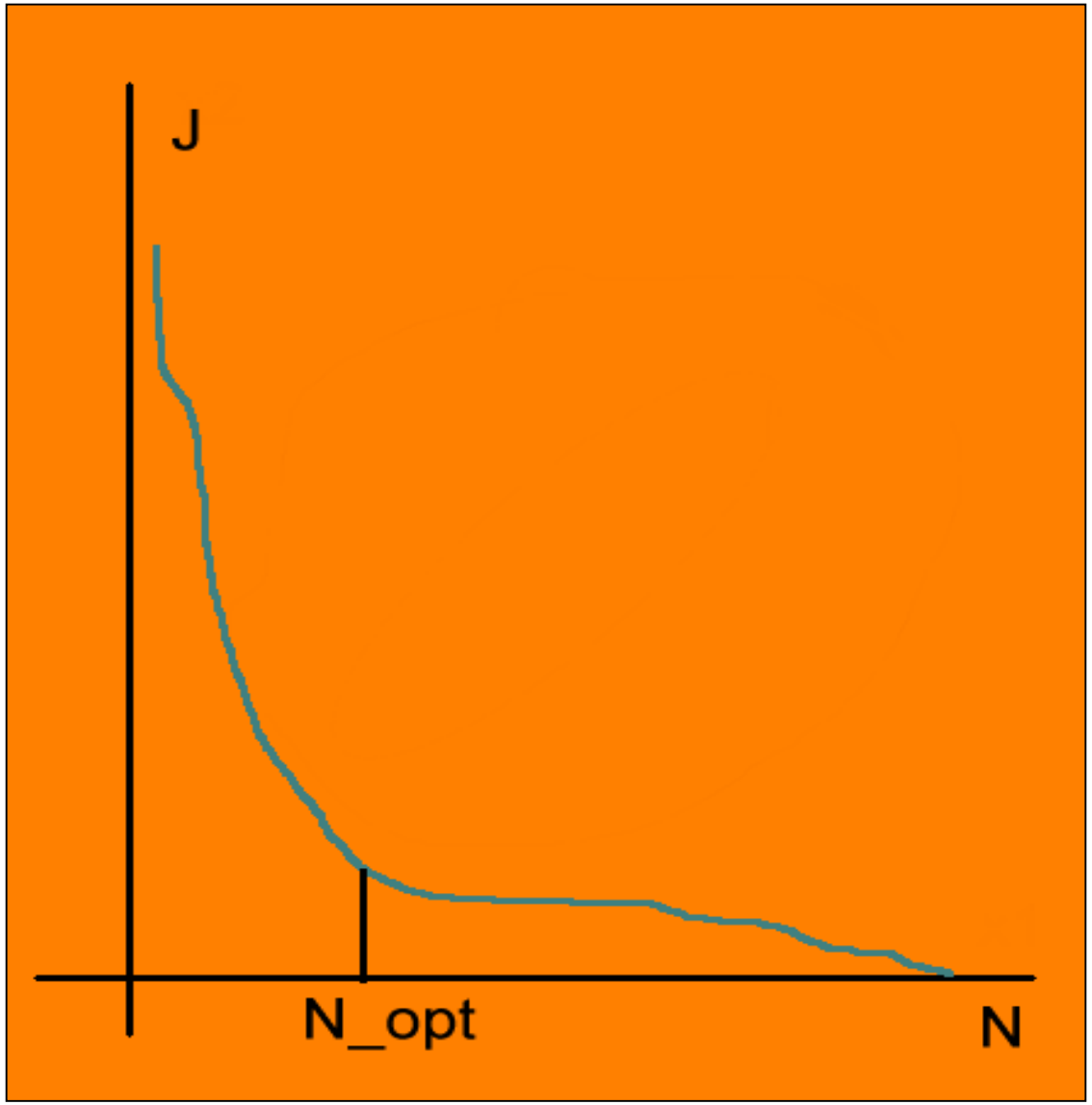
  N can be estimated from the “knees” in *J*-*N* graph

Life time of the clusters



Optimal number of clusters = 4

Optimal number of clusters



Other clustering criteria

Scatter matrices

- between cluster matrix

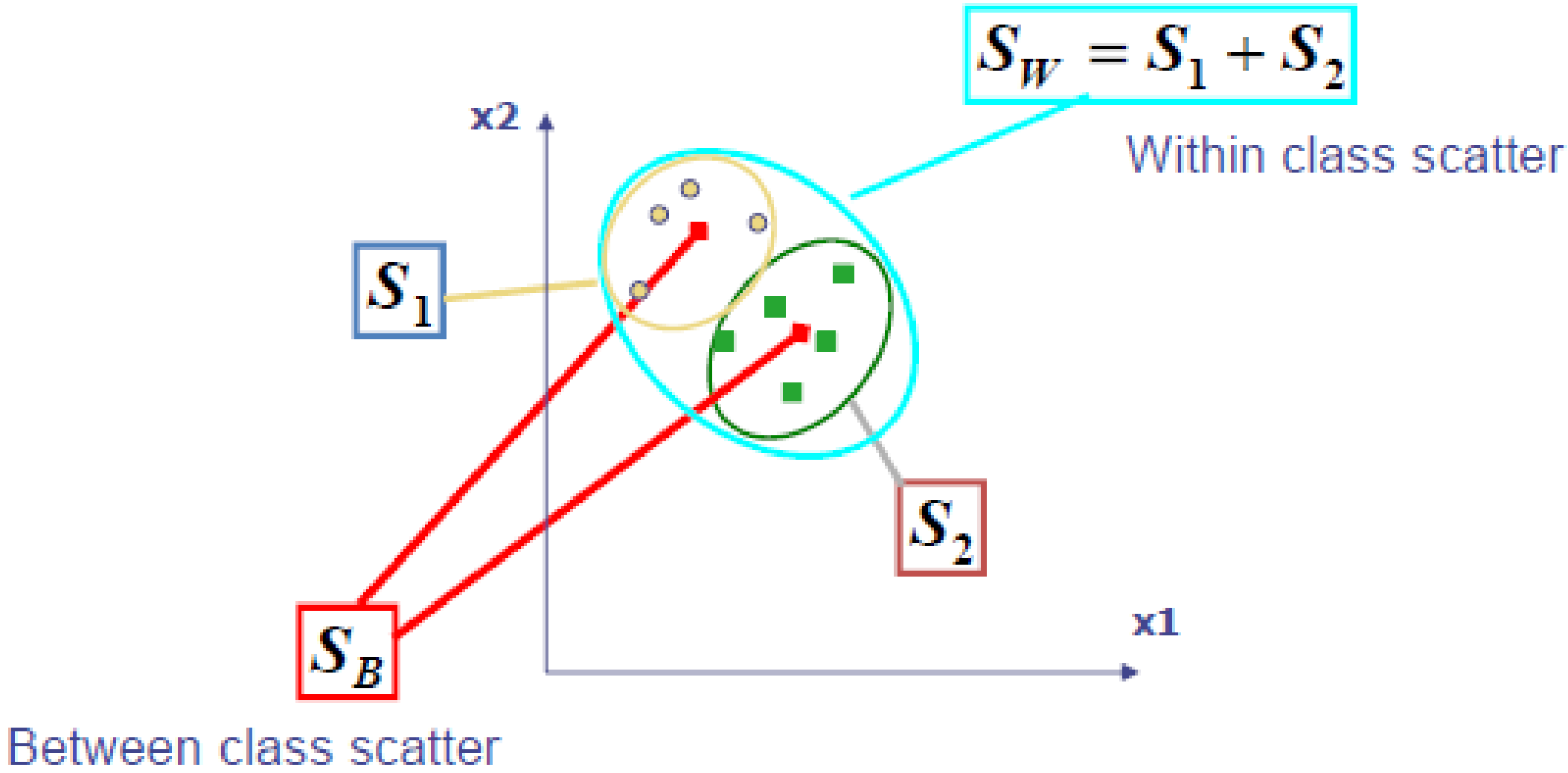
$$B = \sum_{i=1}^{N} n_i(\mathbf{m}_i - \mathbf{m})(\mathbf{m}_i - \mathbf{m})^t$$

- within cluster matrix

$$W = \sum_{i=1}^{N} W_i \qquad W_i = \sum_{\mathbf{x} \in C_i} (\mathbf{x} - \mathbf{m}_i)(\mathbf{x} - \mathbf{m}_i)^t$$

- total scatter matrix

$$T = \sum_{\mathbf{x}} (\mathbf{x} - \mathbf{m})(\mathbf{x} - \mathbf{m})^t$$


$S_W = S_1 + S_2$
Within class scatter
x2
S_1
S_2
S_B
x1
Between class scatter

Other clustering criteria

min $\operatorname{tr}(W) = \sum_{i=1}^{N} \operatorname{tr}(W_i) = \sum_{i=1}^{N} \sum_{\mathbf{x} \in C_i} \|\mathbf{x} - \mathbf{m}_i\|^2 = J$

min $\det(W)$

$T = W + B$

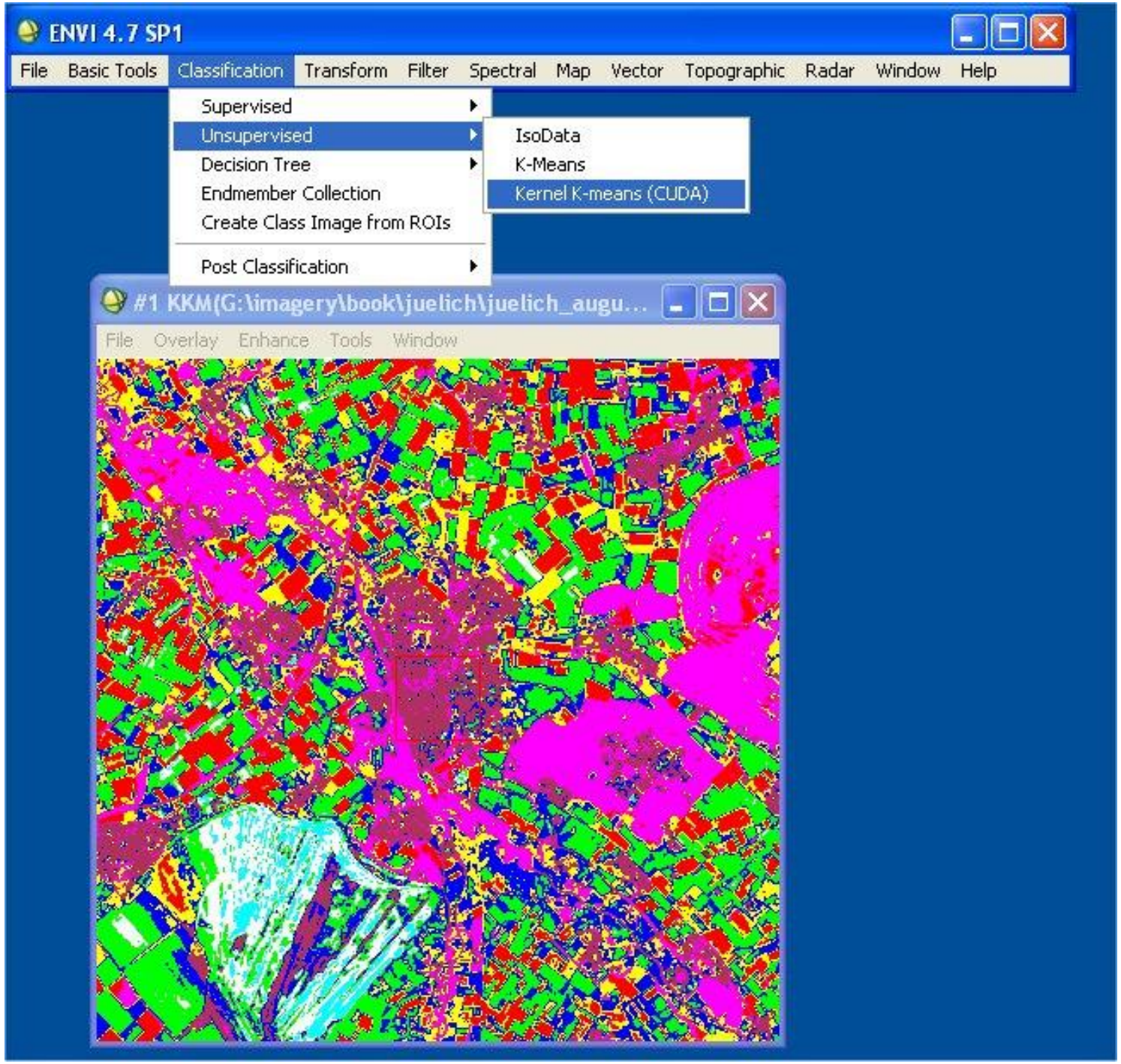
max $\operatorname{tr}(W^{-1}B)$

Applications of clustering in image proc.

- **Segmentation – clustering in color space**
- **Preliminary classification of multispectral images**
- **Clustering in parametric space – RANSAC, image registration and matching**

Numerous applications are outside image processing area




ENVI 4.7 SP1
File
Basic Tools
Classification
Transform
Filter
Spectral
Map
Vector
Topographic
Radar
Window
Help
Supervised
Unsupervised
Decision Tree
Endmember Collection
Create Class Image from ROIs
Post Classification
IsoData
K-Means
Kernel K-means (CUDA)
#1 KKM(G:\imagery\book\juelich\juelich_augu...
File
Overlay
Enhance
Tools
Window

Reduction of the Feature Space Dimensionality

Problem formulation

Having *D* features, we want to reduce their number to *n*, where $n \ll D$

$$(x_1, x_2, \cdots, x_D) \longrightarrow (y_1, y_2, \cdots, y_n)$$

Why to reduce the number of the features

- **Pros:** Lower computing complexity
  Improvement of the classification performance
- **Cons:** Possible loss of information

Basic approaches to DR

- **Feature extraction**

  Transform $T: \quad R^D \rightarrow R^n$

  Creation of a new feature space. The features lose their original meaning.

Basic approaches to DR

- **Feature extraction**

  Transform $T: \quad R^D \rightarrow R^n$

  Creation of a new feature space. The features lose their original meaning.

Example: $n = 1$

$$y_1 = \sum_{i=1}^{D} x_i$$

Basic approaches to DR

- **Feature extraction**

  Transform $T: \quad R^D \rightarrow R^n$

  Creation of a new feature space. The features lose their original meaning.

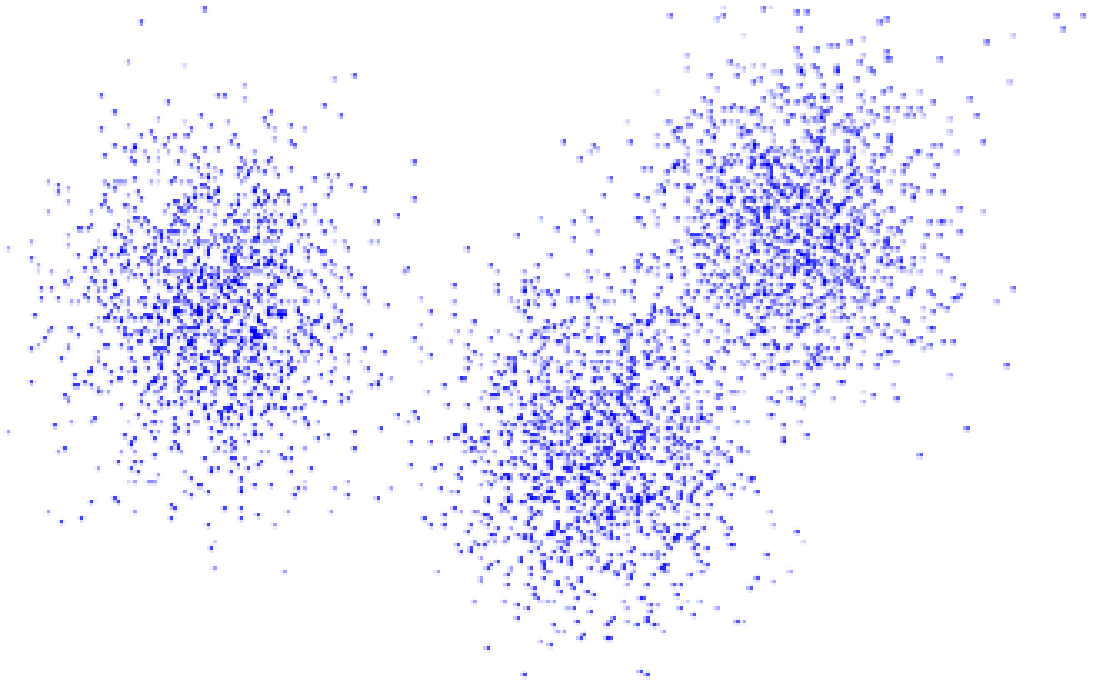
- **Feature selection**

  Selection of a subset of the original features.

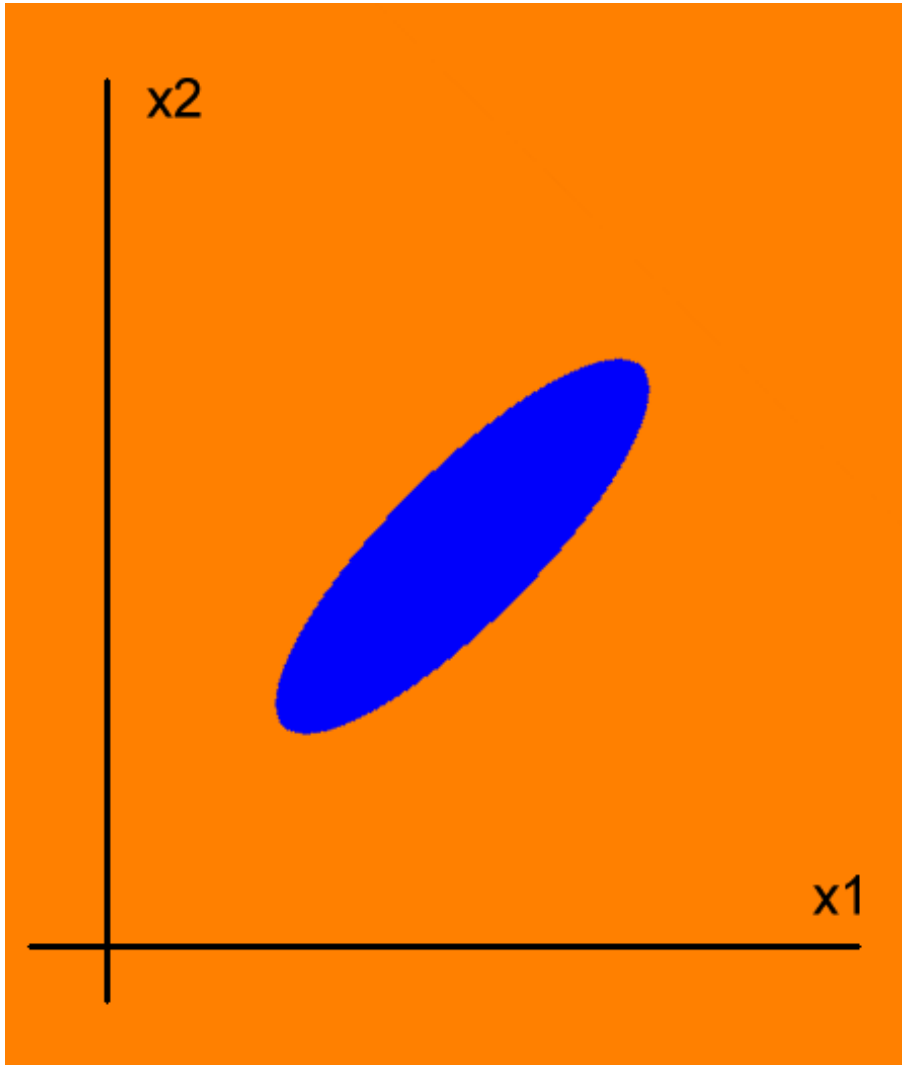
Principal Component Transform

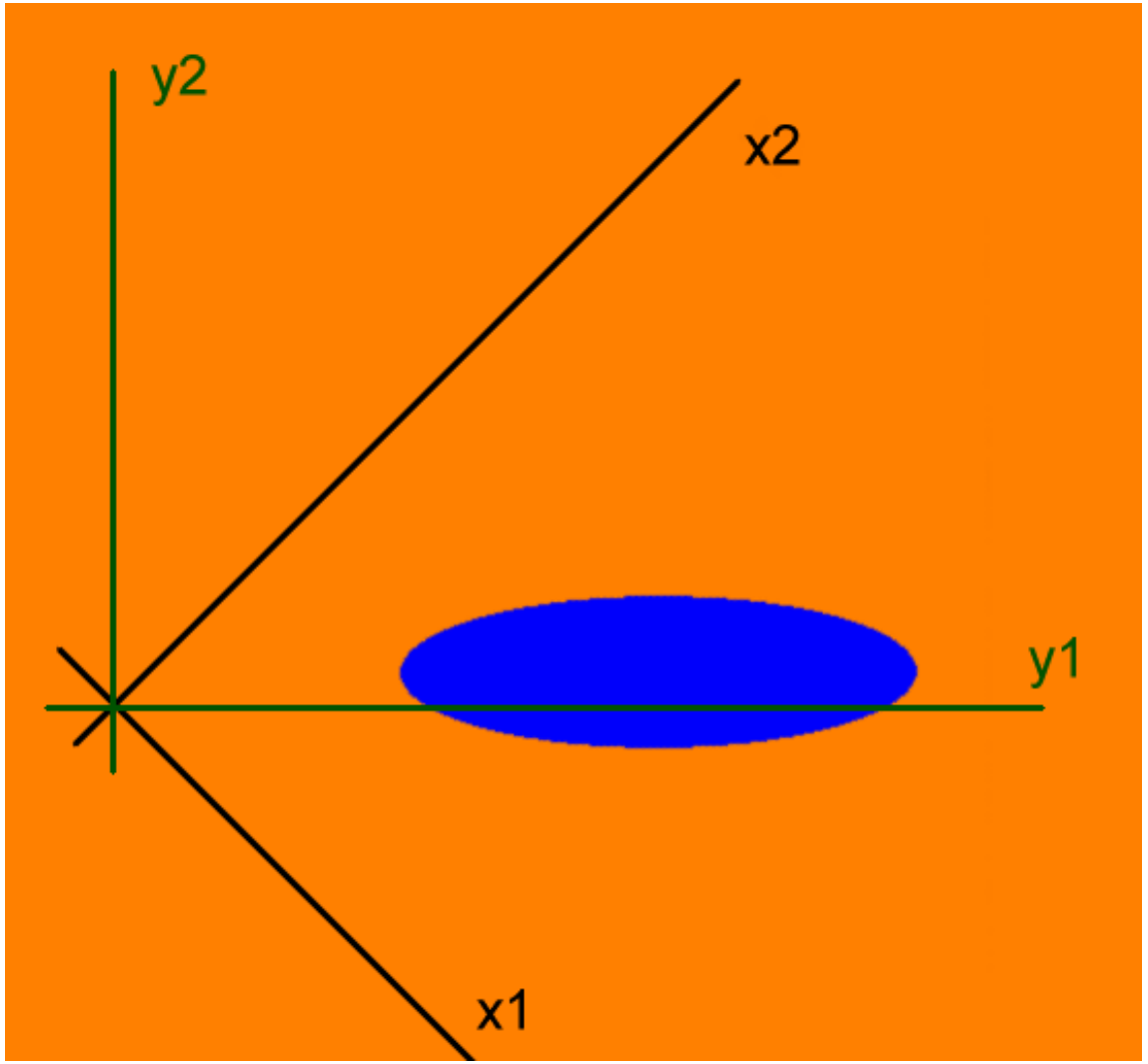
Karhunen-Loeve Transform

PCT is a method for “one-class” problem, i.e. for non-structured data (no class representatives, no training sets, just a cloud of data points in the feature space)



Principal Component Transform





Principal Component Transform

- PCT is a rotation of the feature space

$$y = T'x,$$

such that the new features y are **uncorrelated**, i.e. covariance matrix C_y is diagonal.

- This is always possible since the original covariance matrix C_x is symmetric and can be diagonalized in the orthonormal basis of its eigenvectors

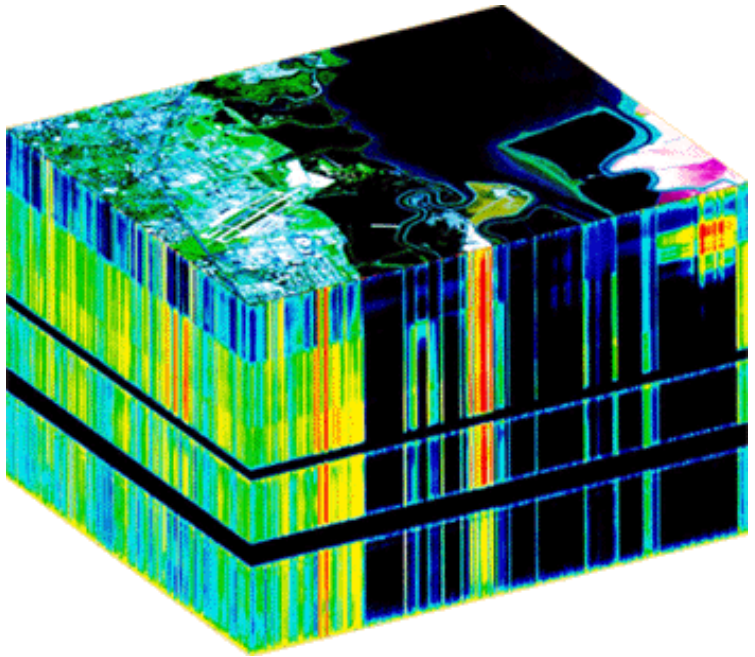
$$C_y = T'C_x T$$

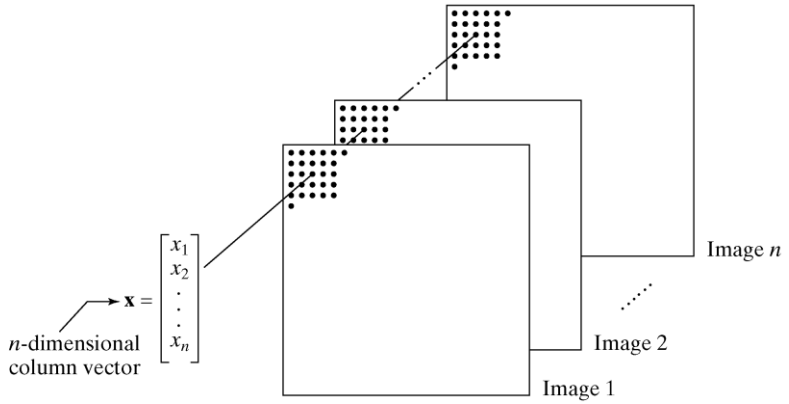
- Features with the highest variances are called **principal components**. First n PC's are kept.

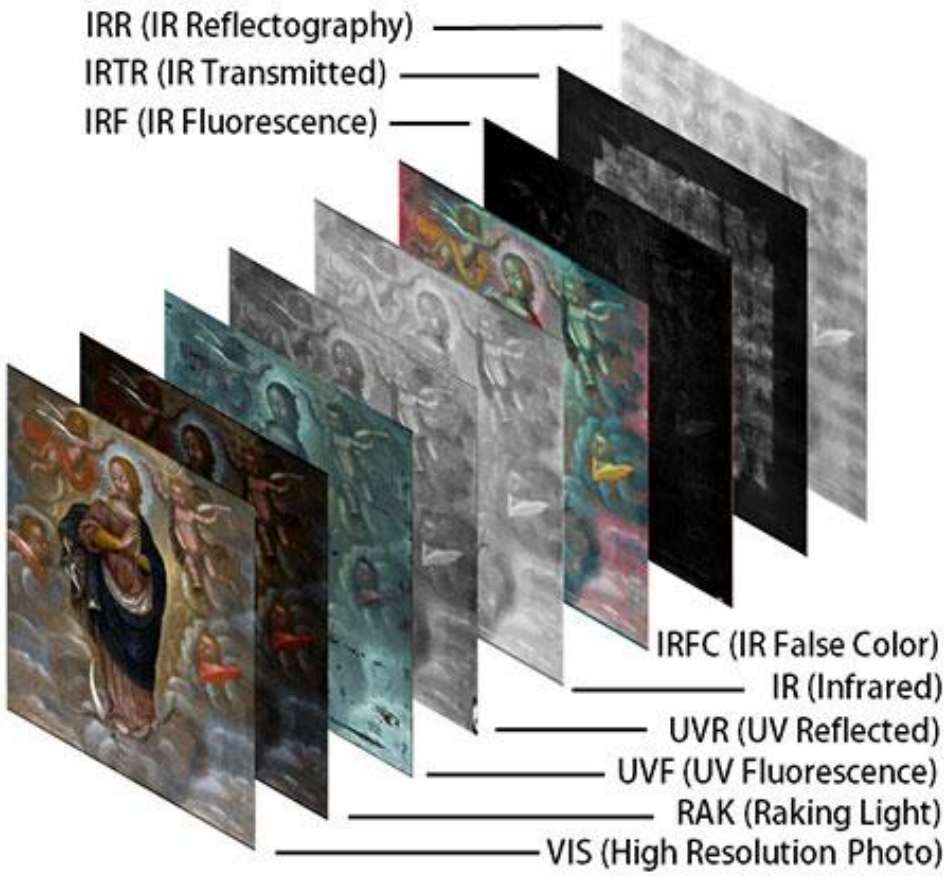
Applications of the PCT

- “Optimal” data representation
- Visualization and compression of multimodal images

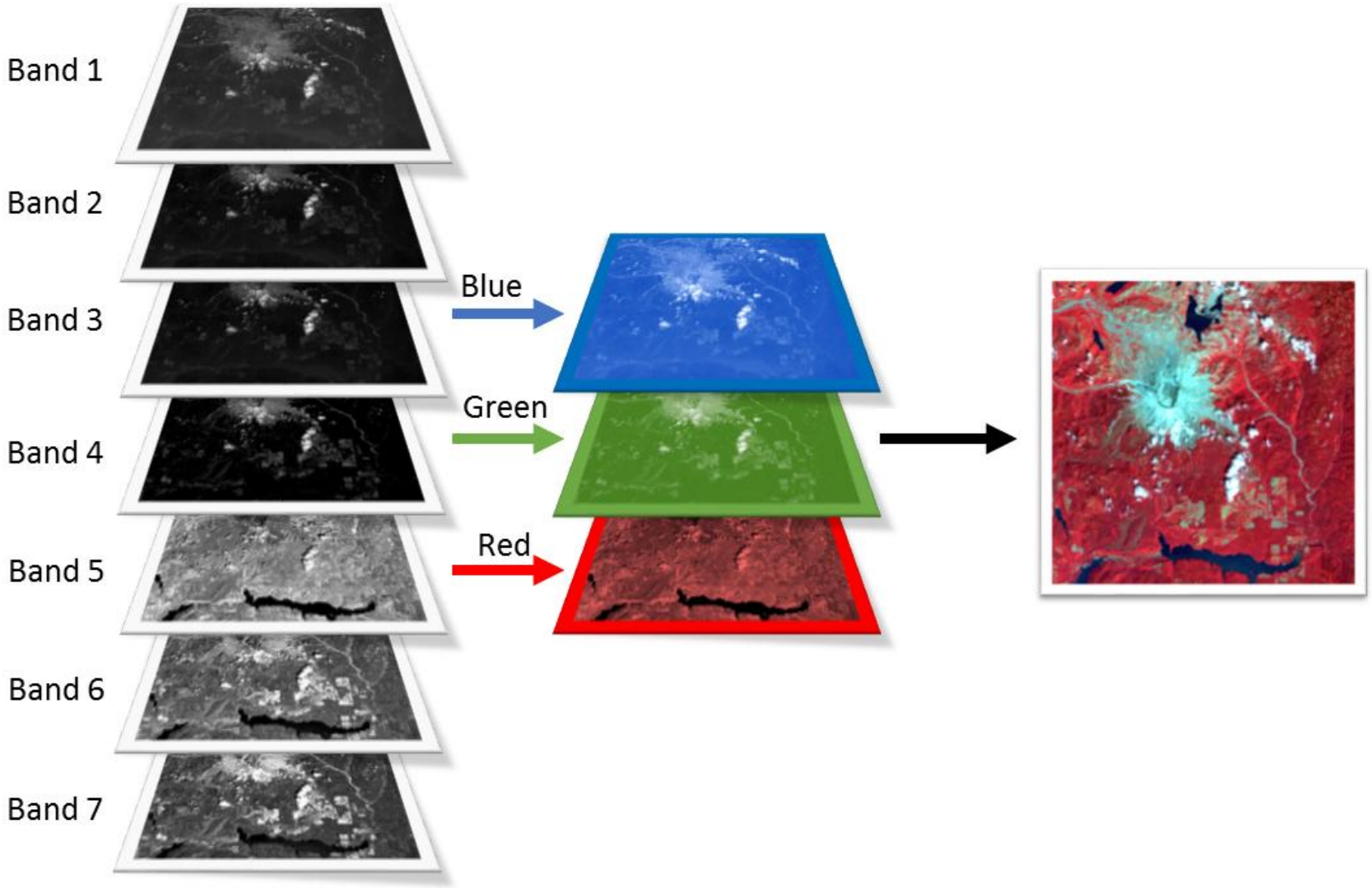
PCT of multispectral images



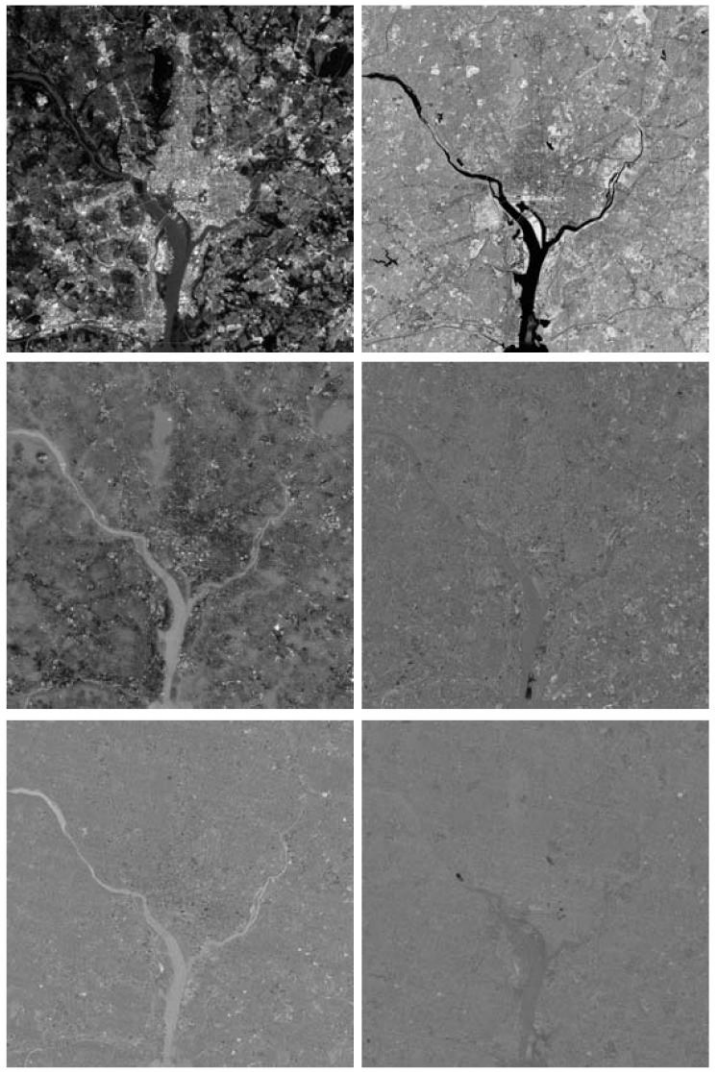




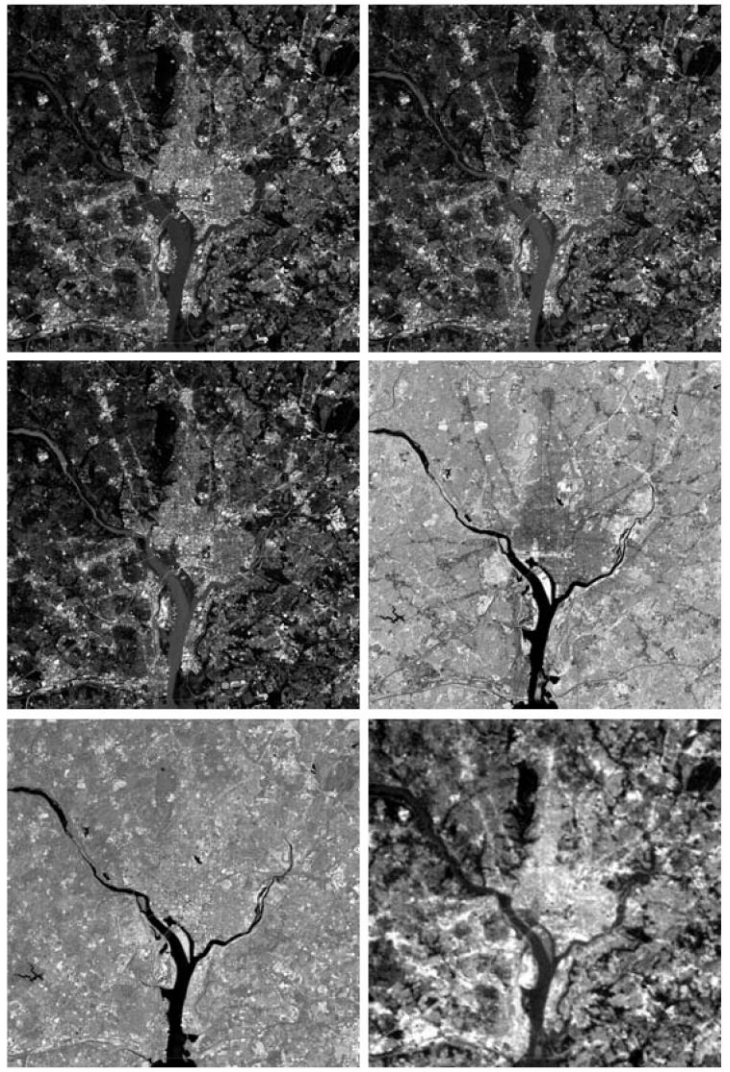
PCT for visualization



PCT



Reconstruct Original in two PC's

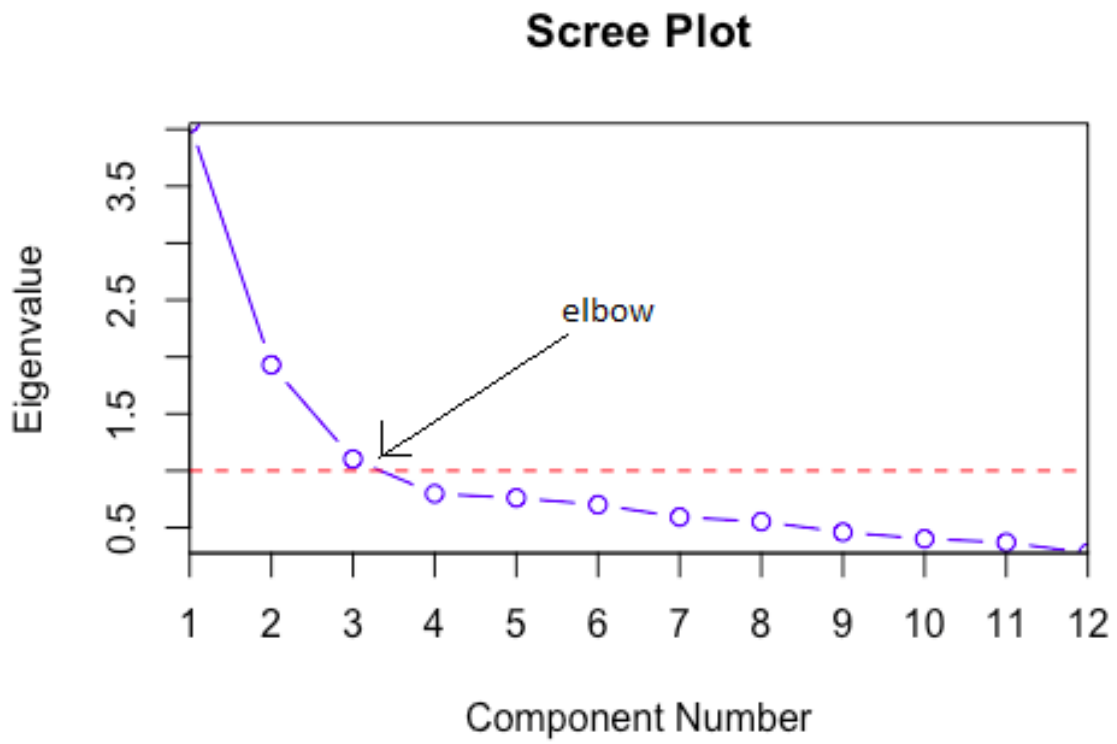


λ_1	λ_2	λ_3	λ_4	λ_5	λ_6
10352	2959	1403	203	94	31

PCT: How many components?

80% of total variation

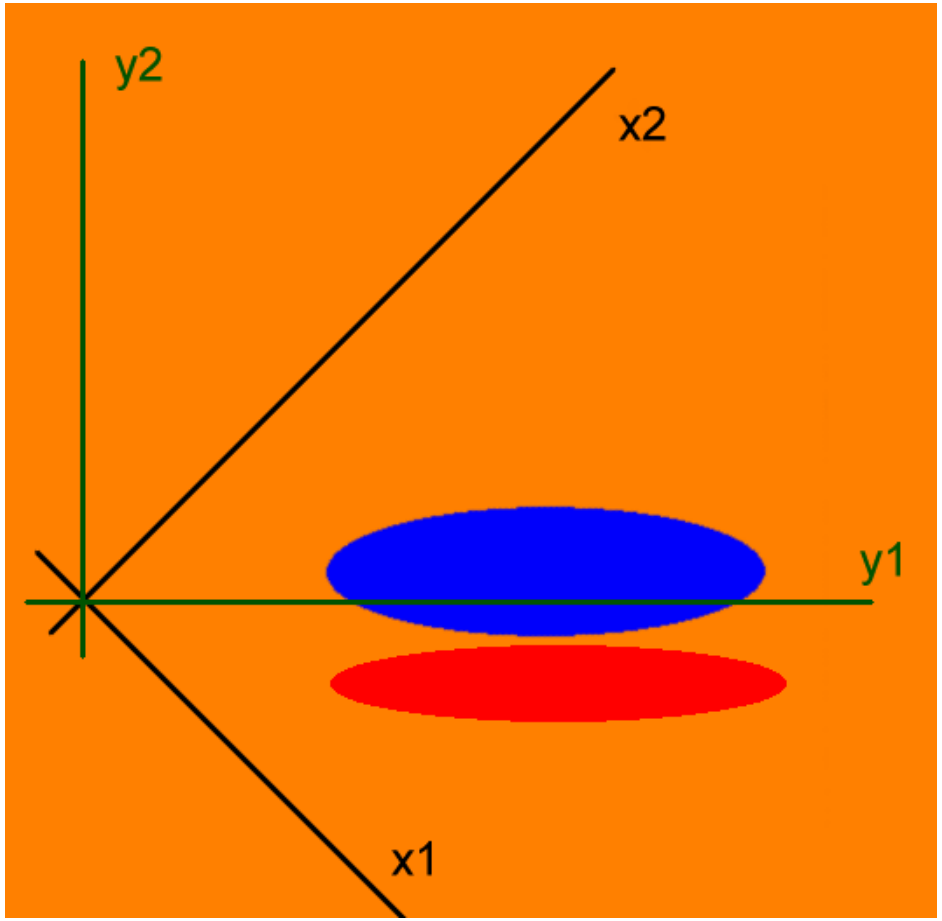
The “elbow” rule

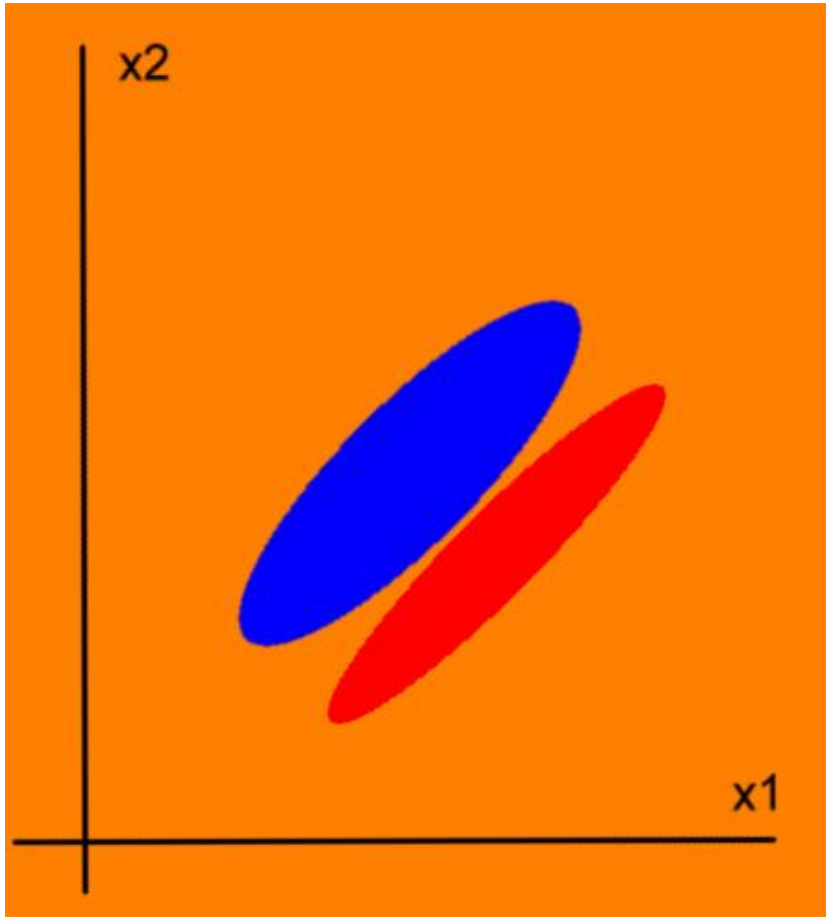


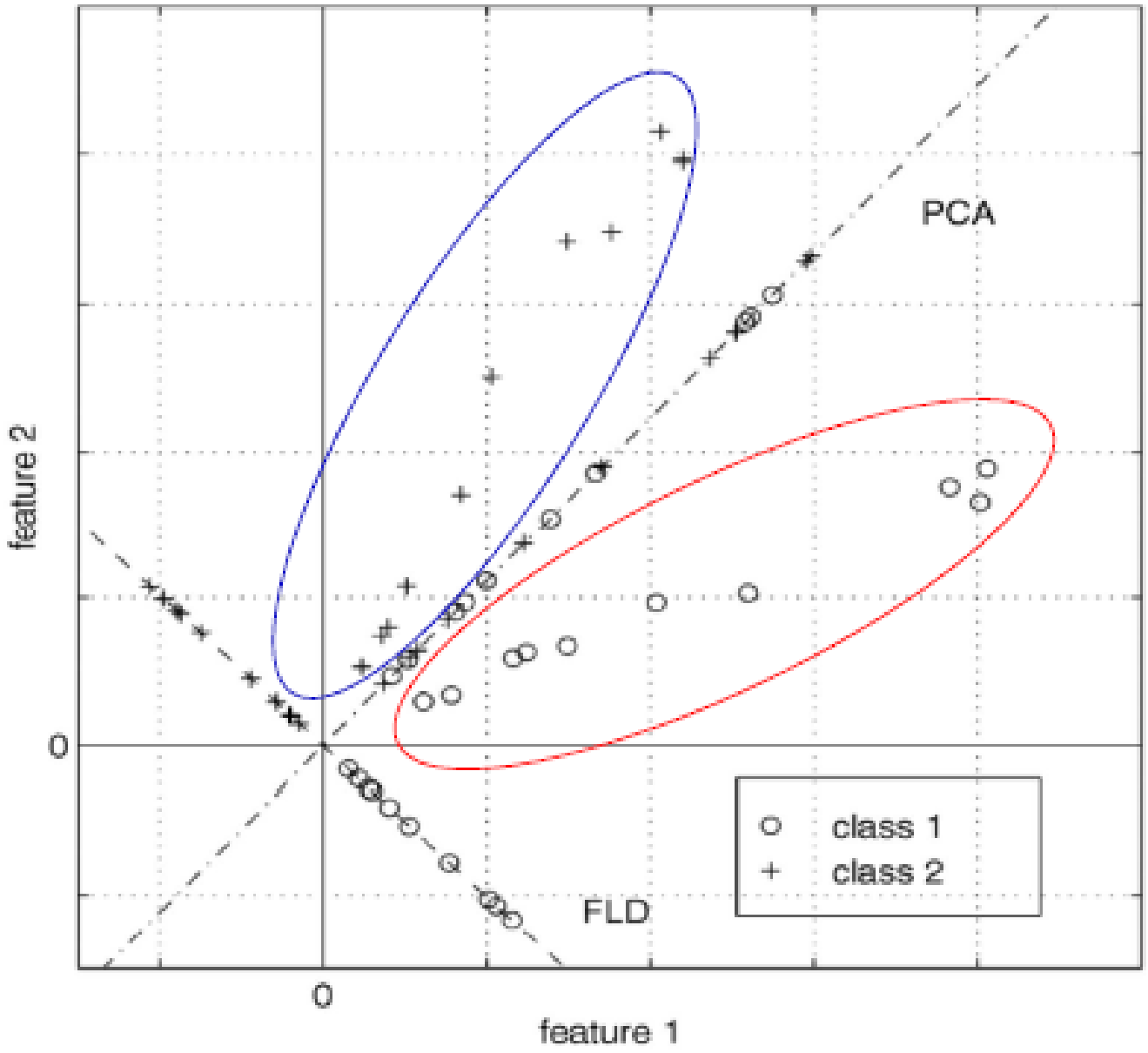
Why is PCT not suitable for classification?

PCT evaluates the contribution of individual features solely by their variance, which may be different from their **discrimination power**.

Why is PCT not suitable for classification?





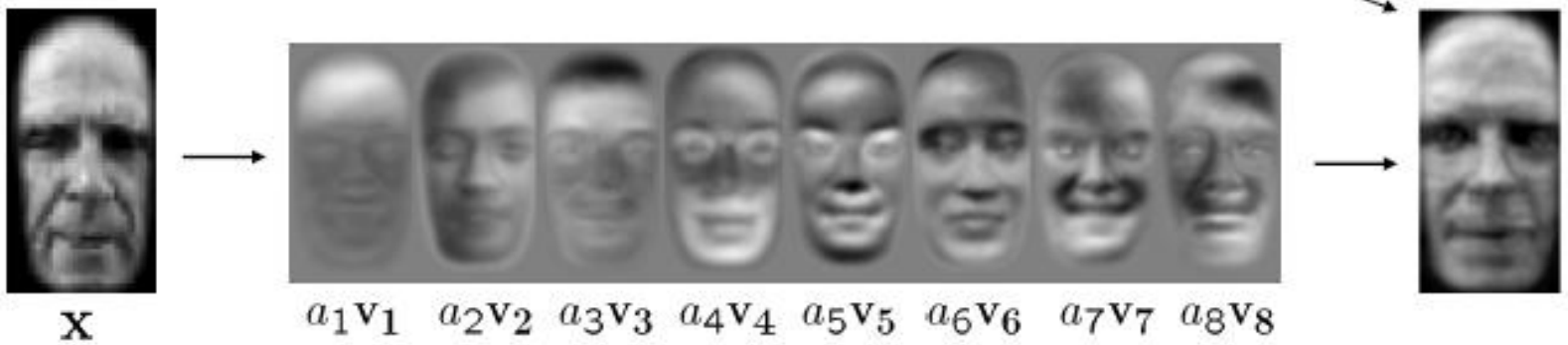

PCA
FLD
class 1
class 2
feature 1
feature 2
0
0

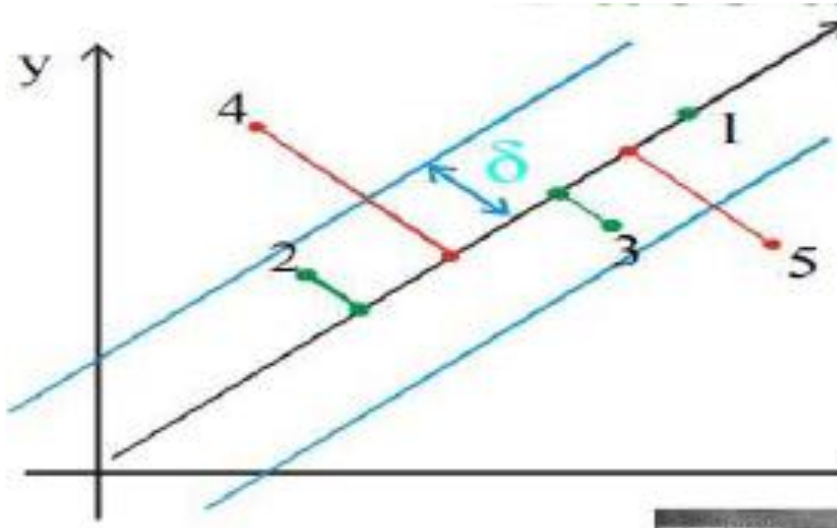
Face recognition by eigenfaces



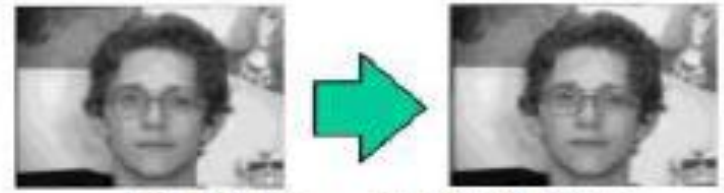


$$\mathbf{x} \approx \overline{\mathbf{x}} + a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + \ldots + a_K\mathbf{v}_\mathbf{K}$$





original projection



A face, used for training

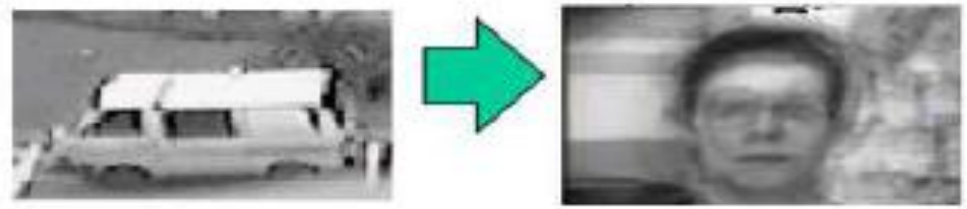


A face, not used for training

1 *The best case: on the subspace*

2,3 *Close enough*

4,5 *Too far – not a face*



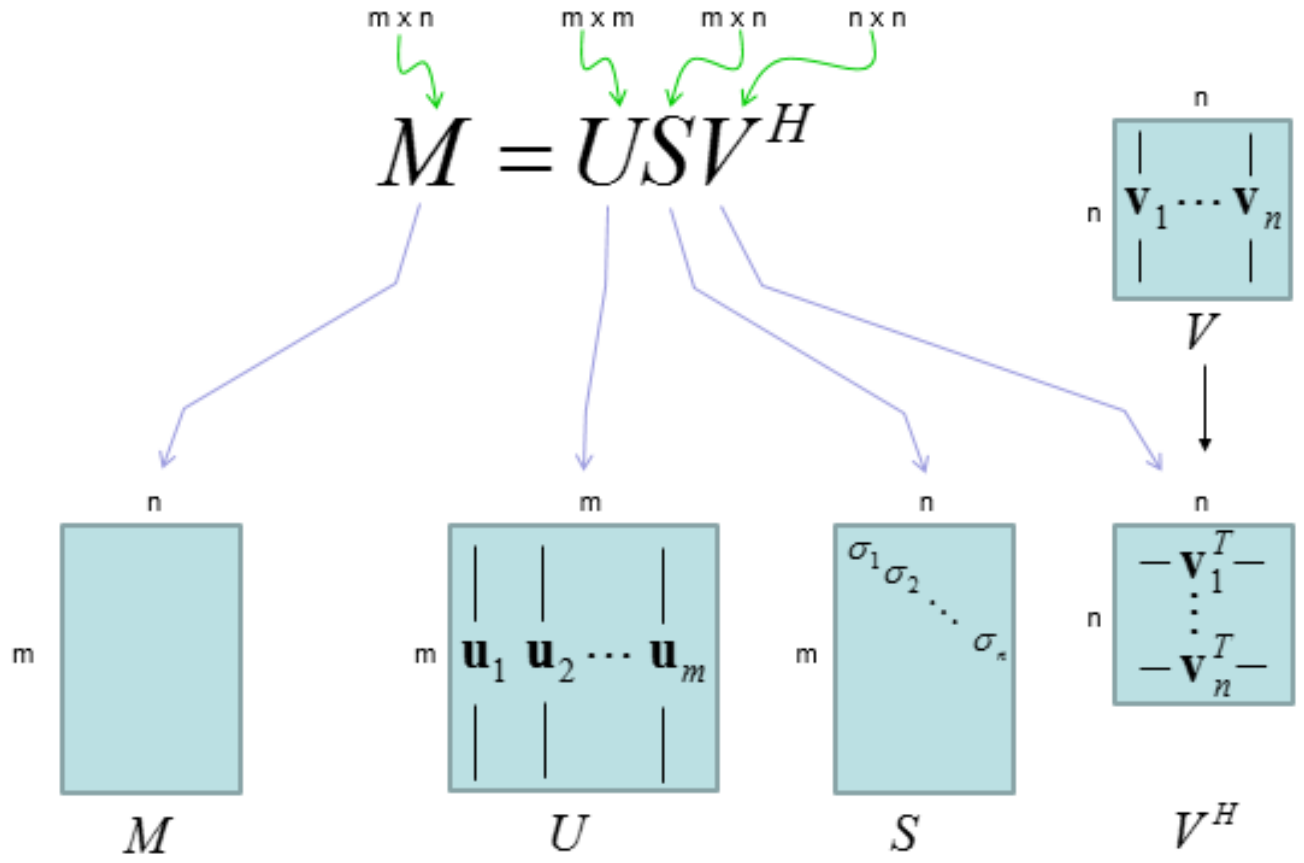
Not a face, not used for training

PCT vs. SVD

M – arbitrary matrix

U, V – orthogonal

S - diagonal

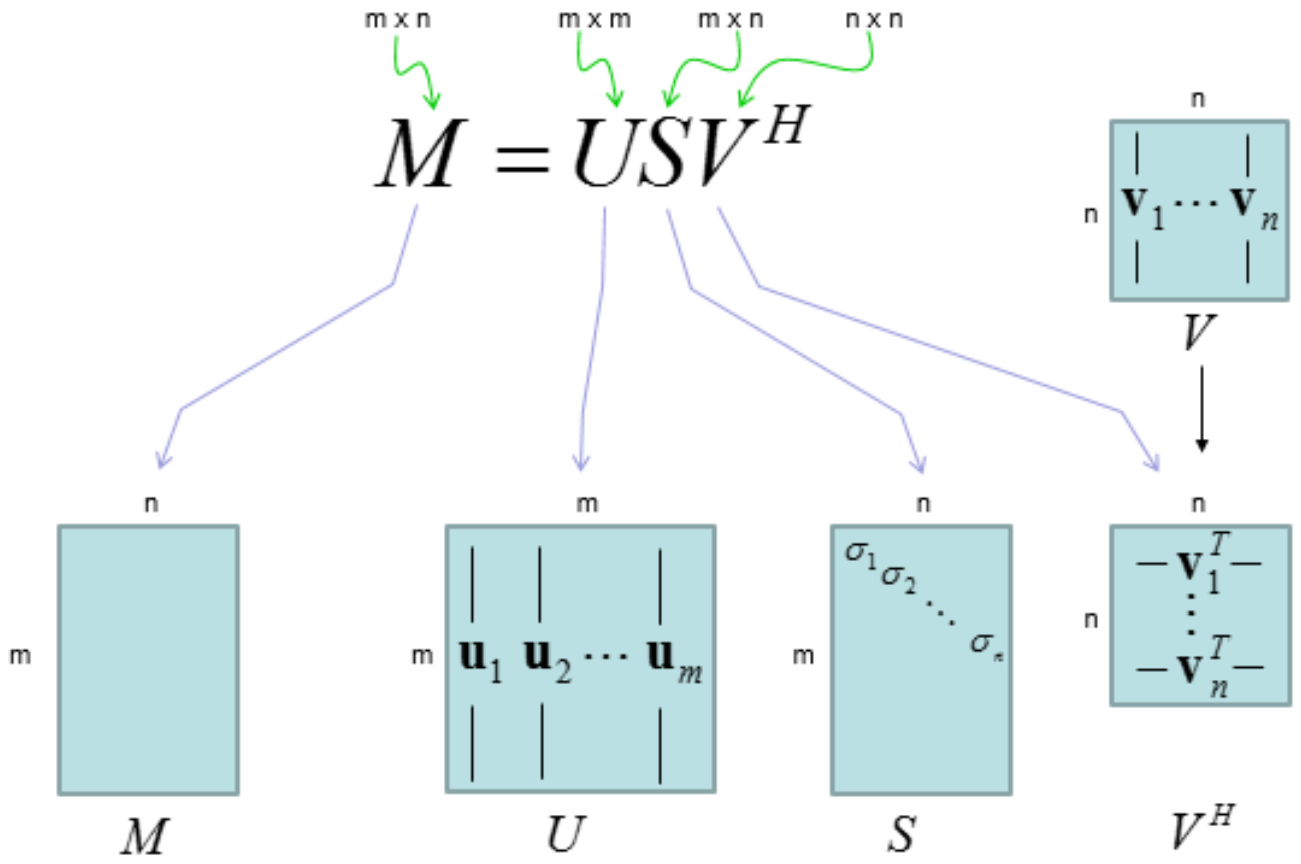


PCT vs. SVD

M – data matrix

C = *M'M* – covariance matrix

SVD of *M* is the same as diagonalization of *C*

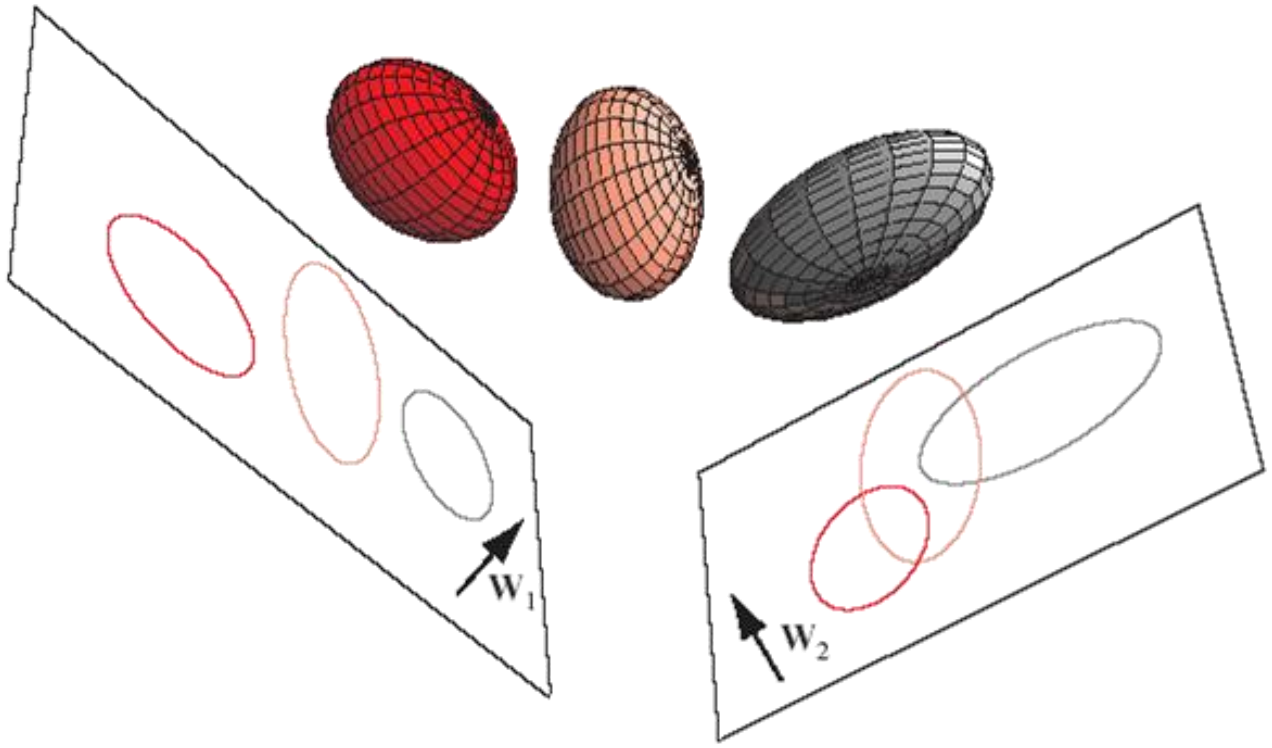


Multi-class problem

- Training sets for each class are available
- Dimensionality reduction methods for classification purposes must consider the discrimination power (separability) of individual features.
- The goal is to maximize the “distance” between the classes.

Example

3 classes, 3D feature space, reduction to 2D



High discriminability

Low discriminability

Feature selection

Two things needed:

- **Discriminability measure** which we want to maximize

- **Selection strategy** (optimization method)

  Feature selection → optimization problem

Measures of discriminability between classes

Analogous to those used in clustering but here “clusters” – training sets – are fixed, while the features are subject to selection.

Ward criterion doesn’t work.

$$J = \sum_{i=1}^{N} \sum_{\mathbf{x} \in C_i} \|\mathbf{x} - \mathbf{m}_i\|^2$$

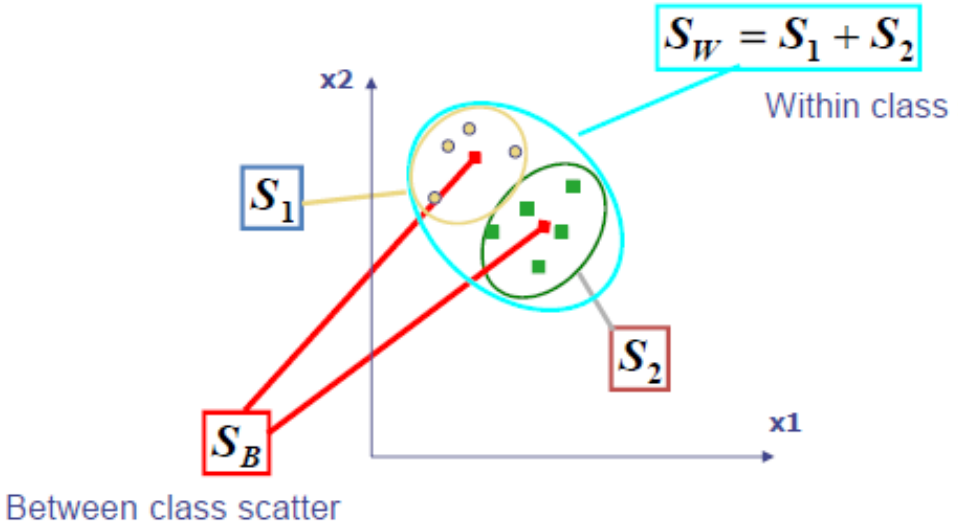


Scatter matrices

- between cluster matrix $$B = \sum_{i=1}^{N} n_i(\mathbf{m}_i - \mathbf{m})(\mathbf{m}_i - \mathbf{m})^t$$

- within cluster matrix $$W = \sum_{i=1}^{N} W_i$$

$$W_i = \sum_{\mathbf{x} \in C_i} (\mathbf{x} - \mathbf{m}_i)(\mathbf{x} - \mathbf{m}_i)^t$$



The most common discriminability measure

$$\mathrm{tr}(W^{-1}B)$$

If N=2 and both training sets have the same number of elements, then

$$\max \mathrm{tr}(W^{-1}B) \sim \max(\mathbf{m}_1 - \mathbf{m}_2)^t (C_1 + C_2)^{-1} (\mathbf{m}_1 - \mathbf{m}_2)$$

Mahalanobis distance

Bhattacharyya distance

Generalization of the M.D., depends more on the class shapes

$$B = \frac{1}{2}M + \ln \frac{\left|\frac{1}{2}(C_1 + C_2)\right|}{\sqrt{|C_1| \cdot |C_2|}}$$

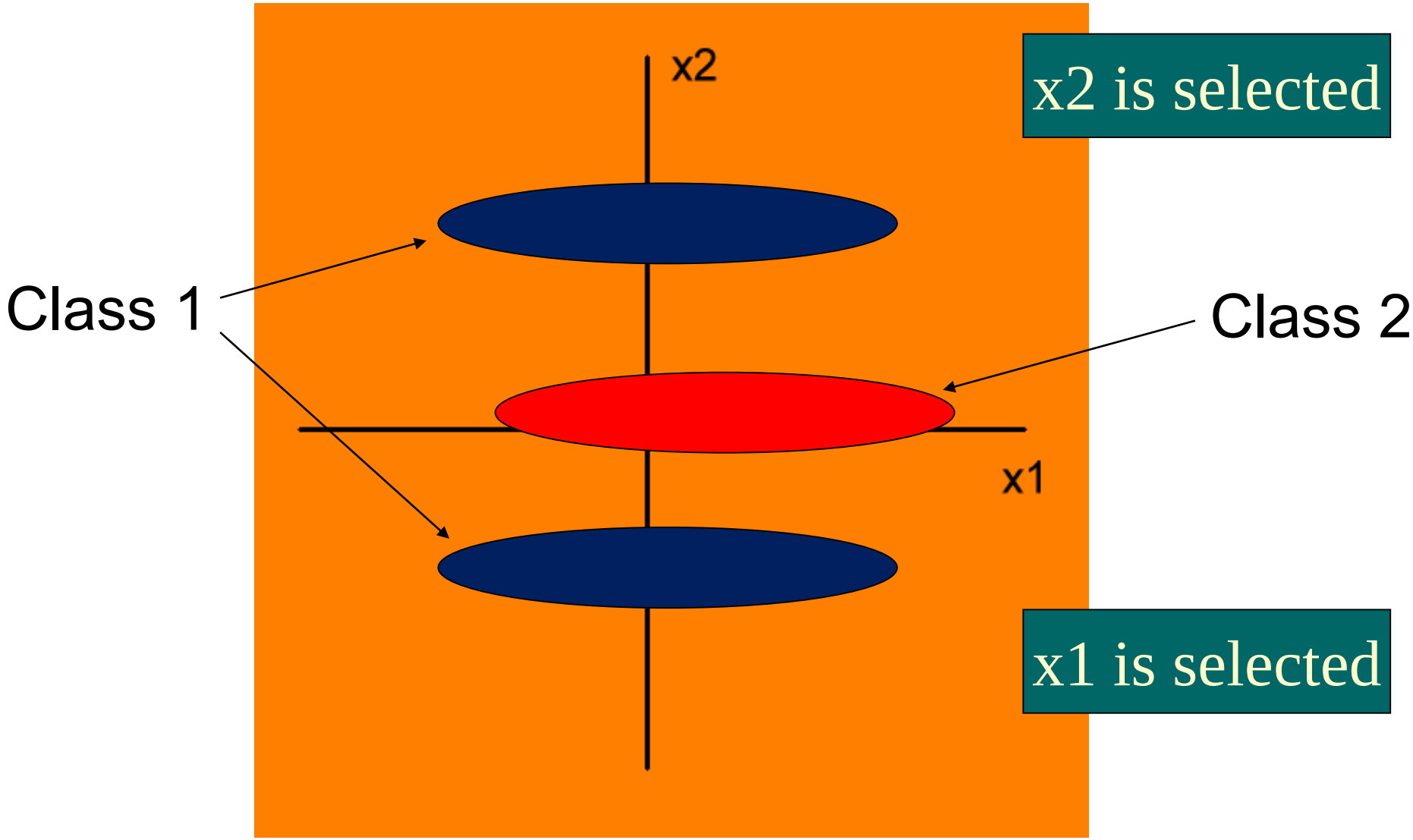
Both *M* and *B* are often used pair-wise
also in a multi-class case.

Prior knowledge in feature selection

- The above discriminability measures require **normally distributed** classes (approximation for some **unimodal** classes).
  They are misleading and inapplicable otherwise.

- The normality should be tested (Pearson's test) before.

A two-class counter-example



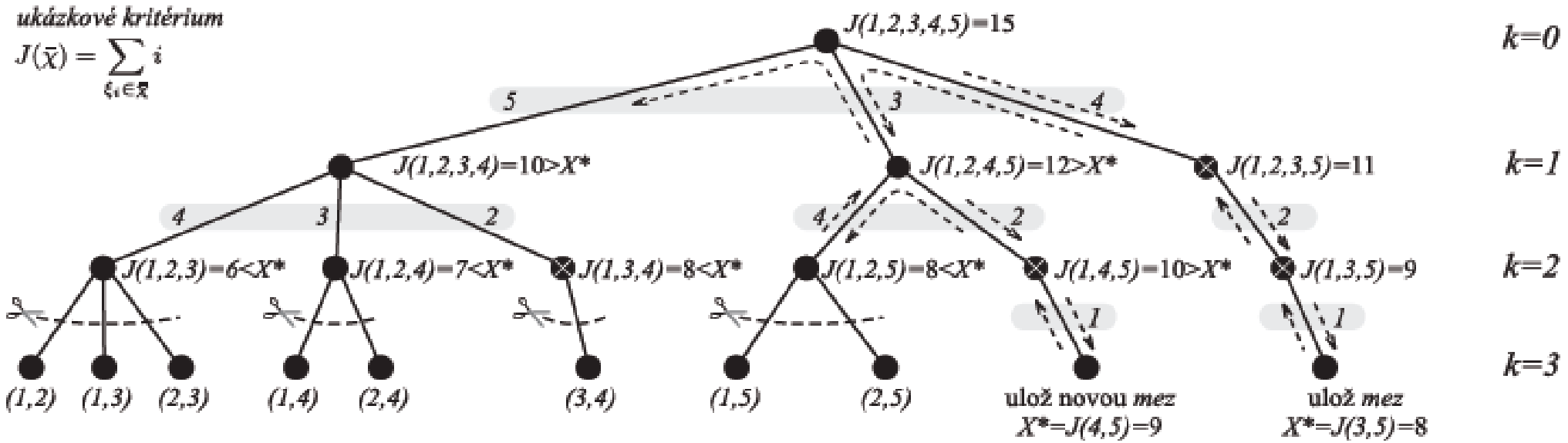
Feature selection algorithms

- **Optimal methods**
  - full search and its modifications
  - complexity *D!/(D-n)!n!*
  - guarantee the global optimum

- **Sub-optimal methods**
  - much faster
  - do not guarantee the global optimum

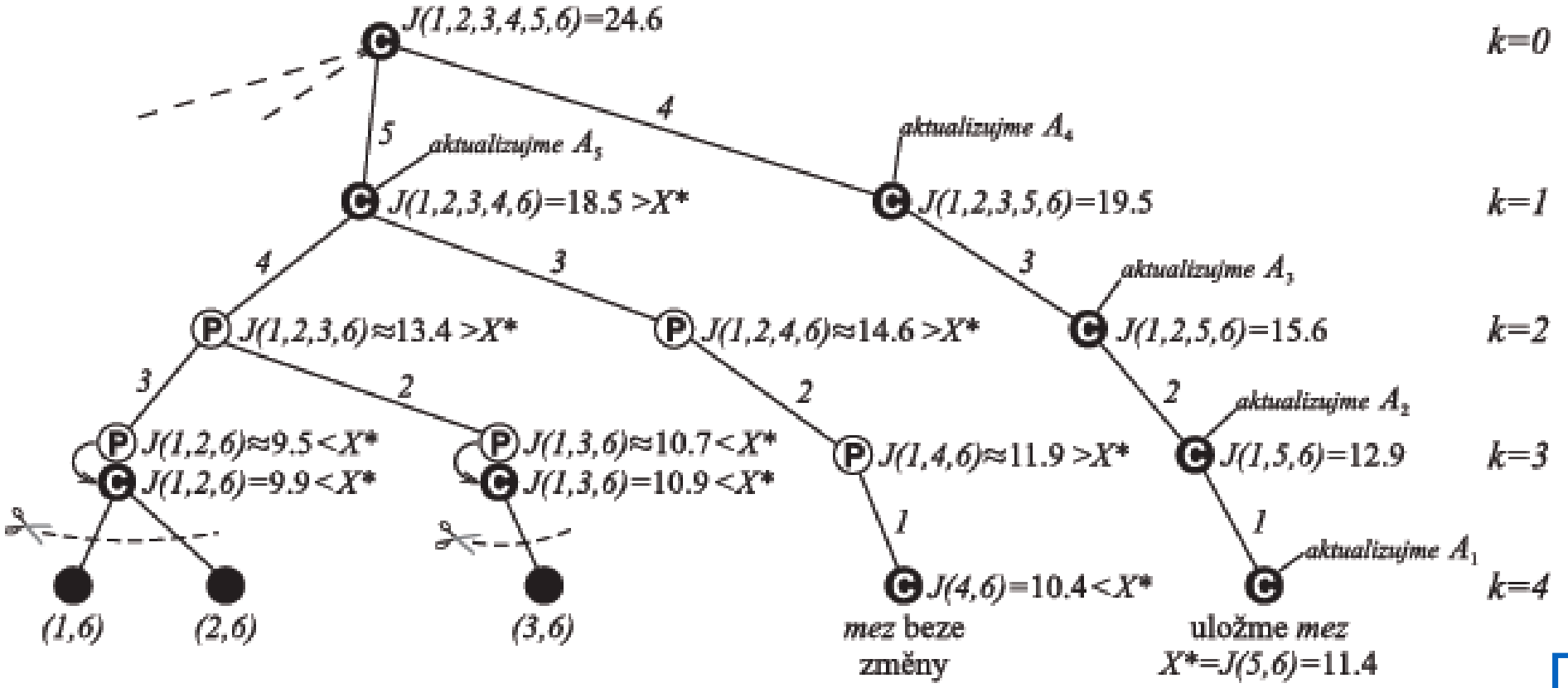
Optimal methods

- Standard full search
- Branch & bound (requires monotonic criteria)



Optimal methods

- Predictive Branch & bound

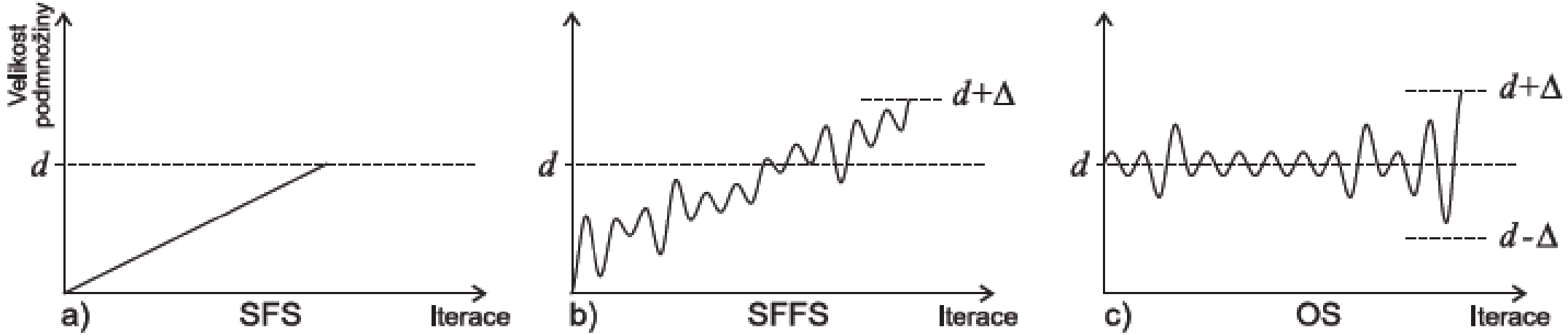


Demos

Sub-optimal methods

- Best individual features
  - optimal for uncorrelated data
- Sequential forward/backward selection (nesting effect!)
- “Plus *k* minus *m*”, $k > m$ (eliminates nesting)
- Floating search
- Oscillating search

Sub-optimal methods



Filters versus Wrappers

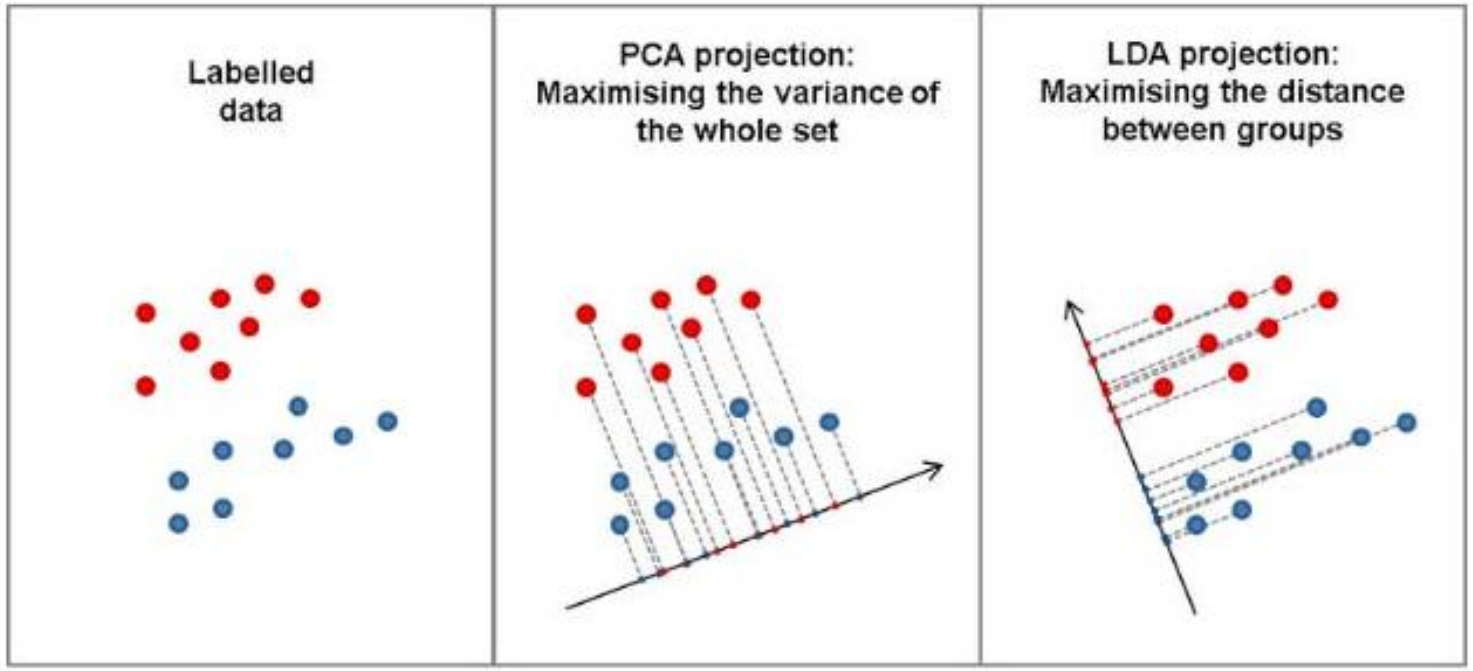
- **Filters** optimize the separability of the training set

Filters versus Wrappers

- **Filters** optimize the separability of the training set
- **Wrappers** optimize the performance of the particular classifier on the given test set (any selection method can be used, usually much slower)
- Neither filters nor wrappers guarantee optimal performance on independent data but wrappers are believed to be better in this sense

Linear discriminant analysis (LDA)

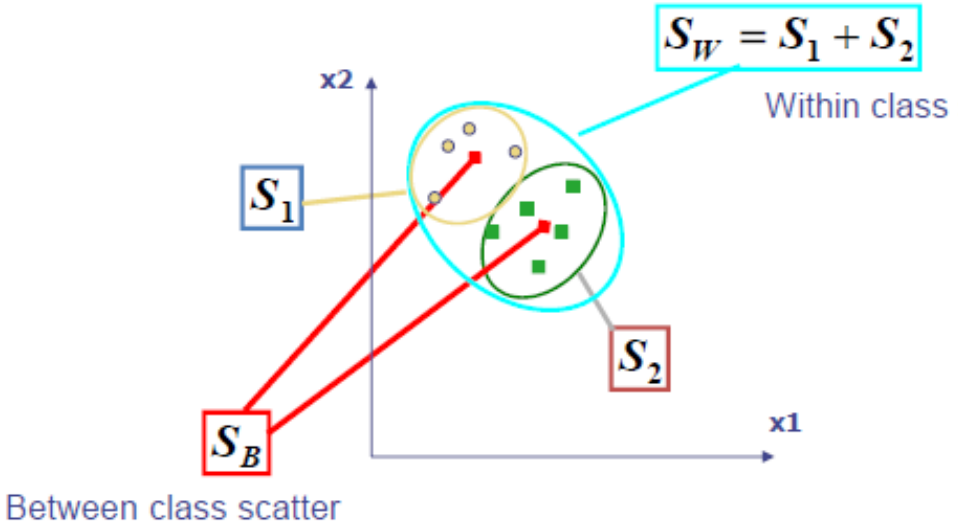
- Selection of the “best” feature/direction



Recalling scatter matrices

- between cluster matrix $$B = \sum_{i=1}^{N} n_i(\mathbf{m}_i - \mathbf{m})(\mathbf{m}_i - \mathbf{m})^t$$

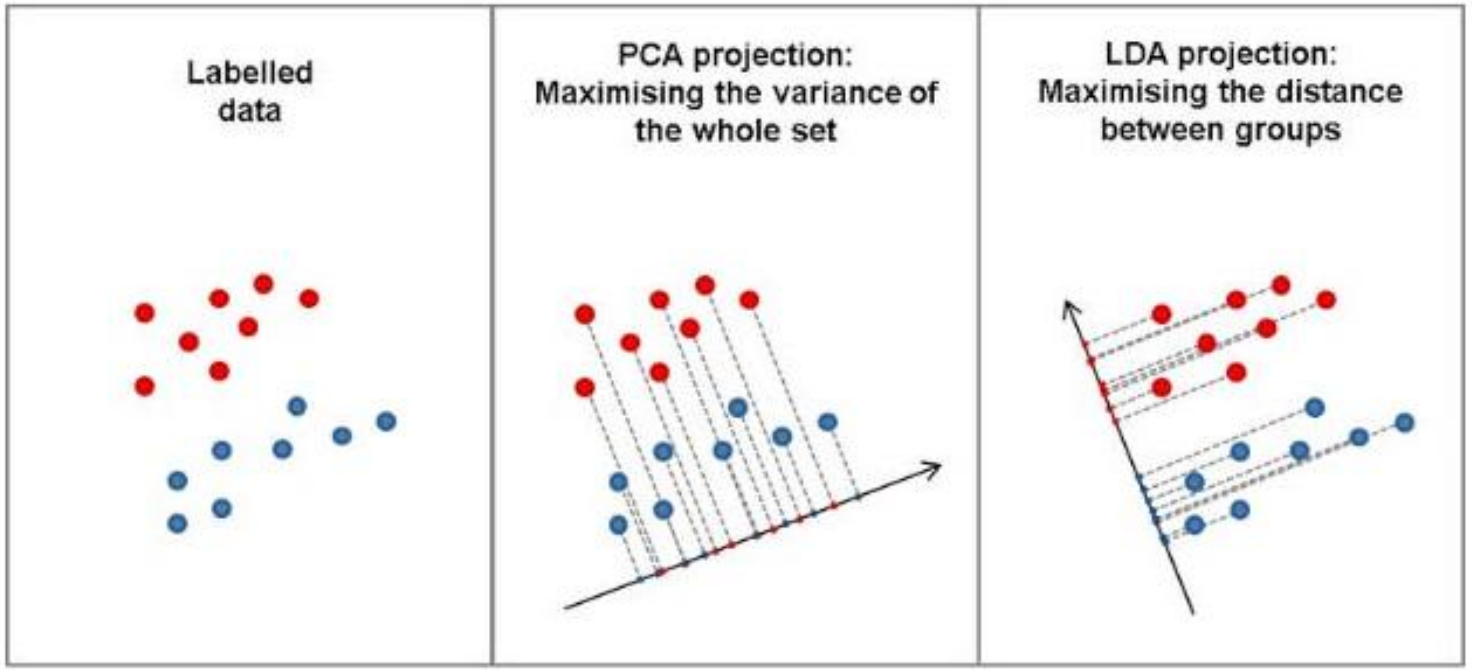
- within cluster matrix $$W = \sum_{i=1}^{N} W_i$$



$$W_i = \sum_{\mathbf{x} \in C_i} (\mathbf{x} - \mathbf{m}_i)(\mathbf{x} - \mathbf{m}_i)^t$$

Linear discriminant analysis (LDA)

- Selection of the “best” feature/direction
- Optimal direction is given by the principal eigenvector of $(W^{-1}B)$

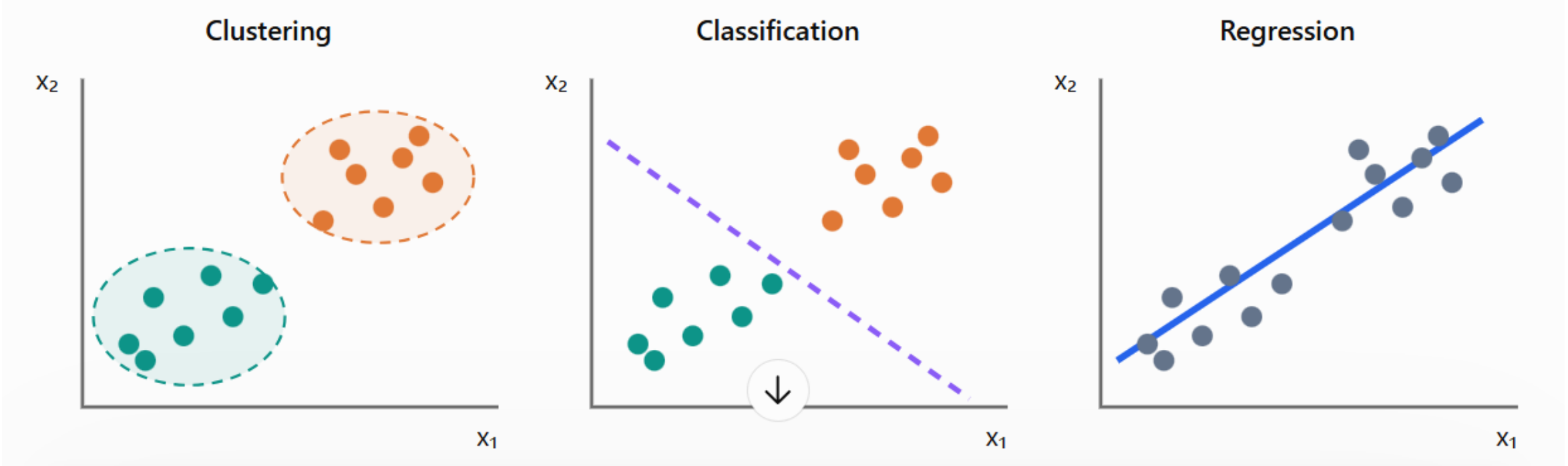


Linear Regression

- **Regression:**
  - data fitting, prediction, trend estimation
  - geometric interpretation

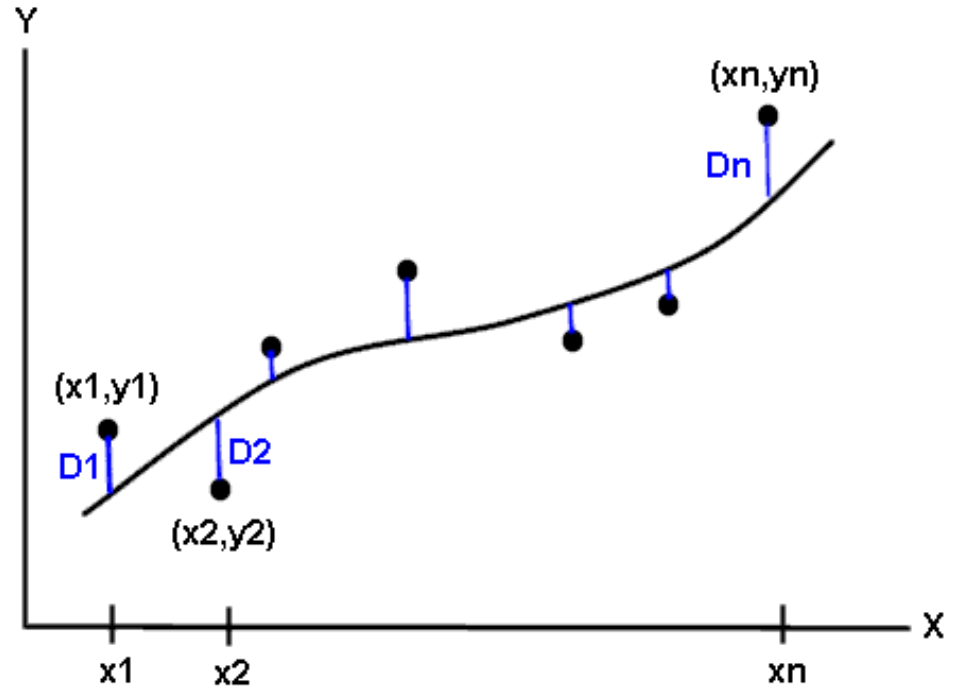
- **Linearity:**
  - regression function is linear
  - regression coefficients are roots of a linear system

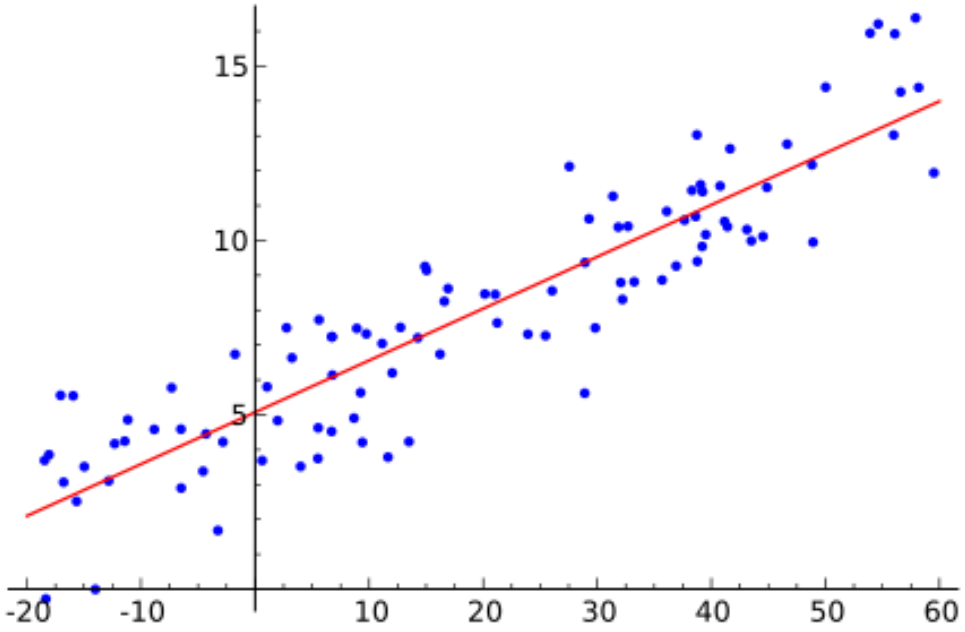
Single dataset, three tasks



Regression as data fitting

- Data approximation
- Solving over-determined systems

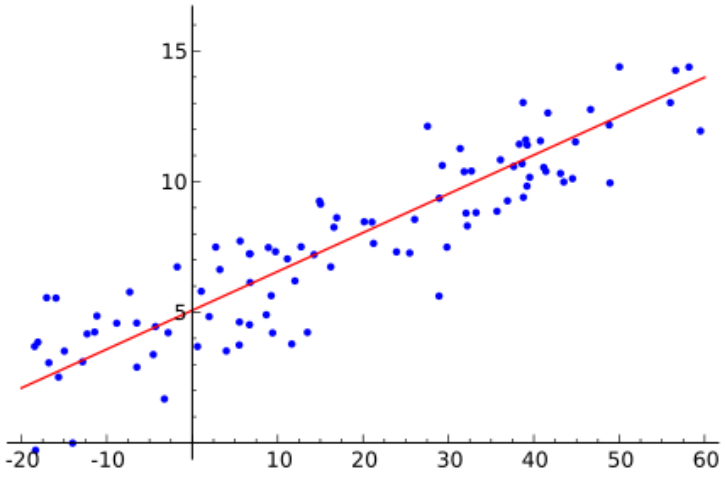




Least-square fit

- Assumption: errors are only in *y* and are normal
- Minimization of a sum of *squared* errors (why?)
- Maximum likelihood estimation

$$E(a,b) = \sum_{n=1}^{N} (y_n - (ax_n + b))^2$$

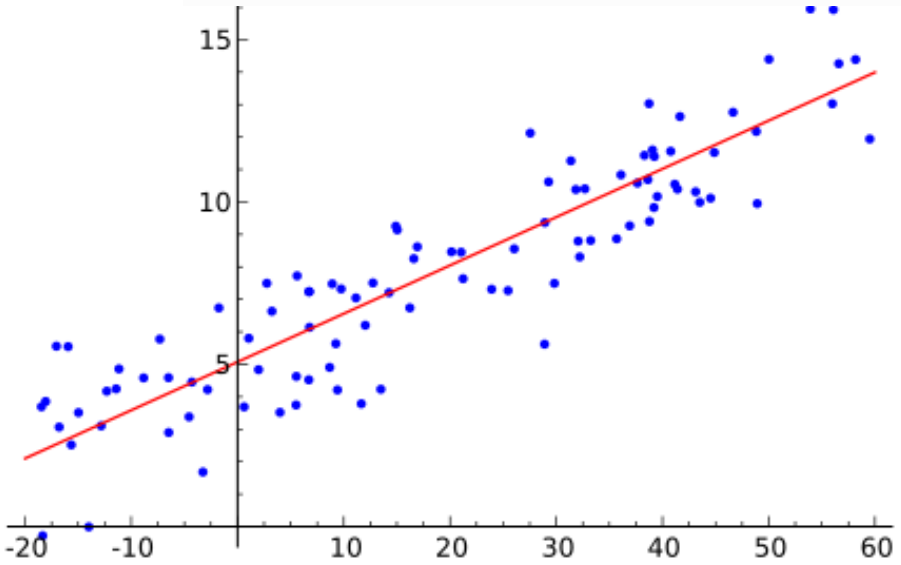


$$\frac{\partial E}{\partial a} = 0, \quad \frac{\partial E}{\partial b} = 0.$$

Normal equations

$$\begin{pmatrix} \sum_{n=1}^{N} x_n^2 & \sum_{n=1}^{N} x_n \\ \sum_{n=1}^{N} x_n & \sum_{n=1}^{N} 1 \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} \sum_{n=1}^{N} x_n y_n \\ \sum_{n=1}^{N} y_n \end{pmatrix}$$

$$\boxed{\mathbf{A}^T \mathbf{A} \hat{\boldsymbol{\beta}} = \mathbf{A}^T \mathbf{y}.}$$



Least Square Fit – Algebraic viewpoint

- **Over-determined system**

$$Ax = b$$

- not solvable in traditional sense

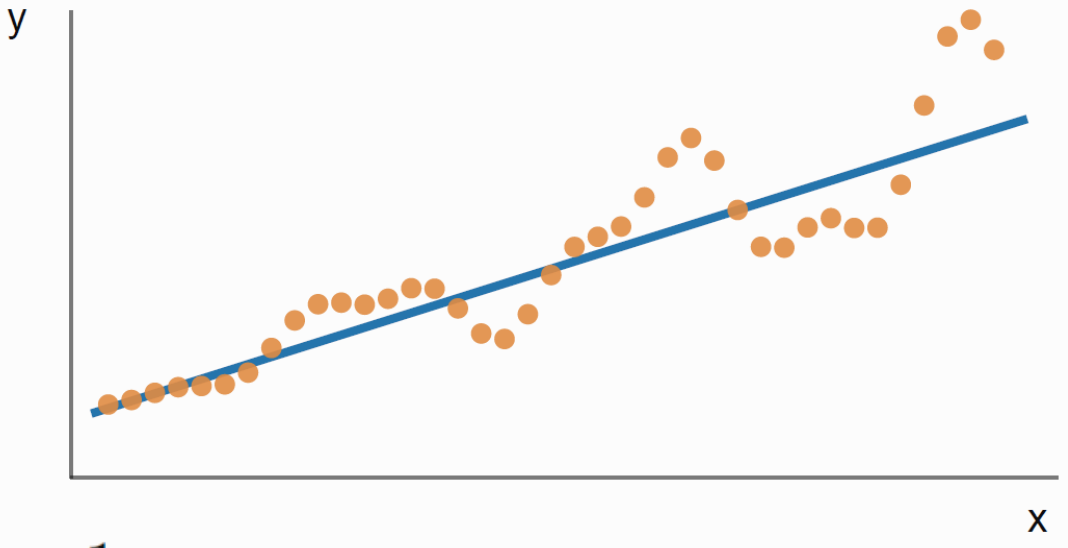
- **Least-square solution**

$$A'Ax = A'b$$

Generalized Least Square Fit

- Different variances of errors
- Correlated errors

- Incorporating covariance matrix -> Mahalanobis distance

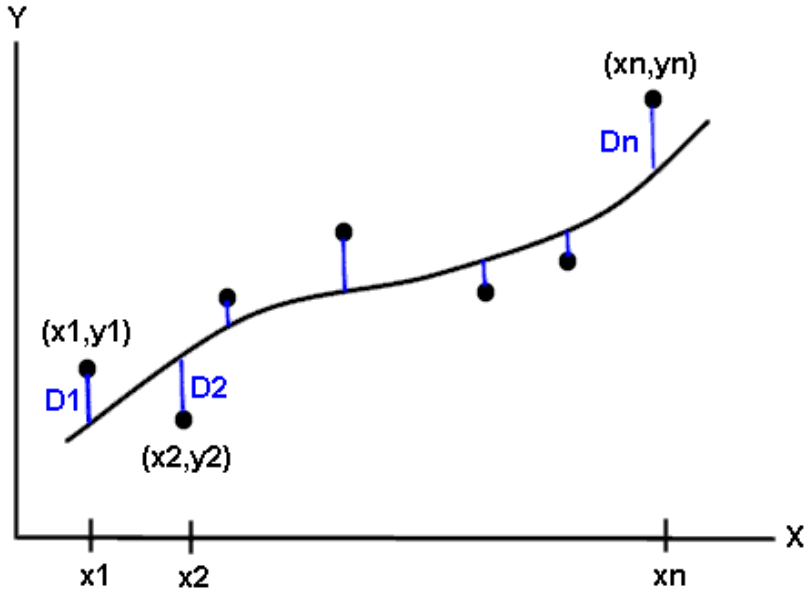


$$\min_{\beta}(A\beta - \mathbf{y})^T \Sigma^{-1}(A\beta - \mathbf{y})$$

$$A^T \Sigma^{-1} A\beta = A^T \Sigma^{-1} \mathbf{y}$$

Non-linear Least Square Fit

- Polynomial approximation always leads to linear system



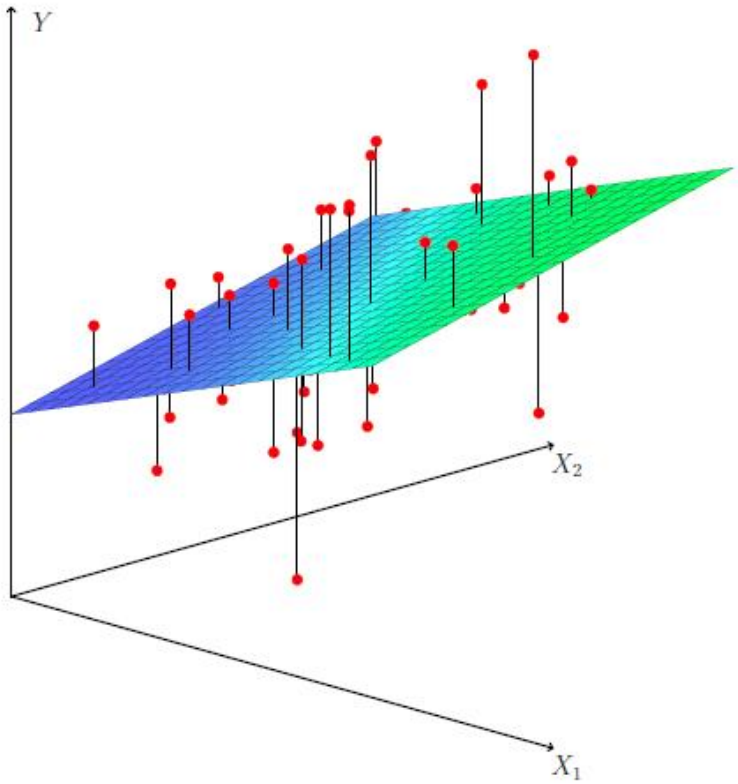
- Non-linear LS – numerical solution of normal equations

Multivariate regression

- Data fitting by a hyperplane

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon$$

$$RSS = \sum_{i=1}^{m} (y_i - \hat{\beta}_0 - \sum_{j=1}^{p} \hat{\beta}_j x_{ij})^2$$



$$X^T X \boldsymbol{\beta} = X^T \mathbf{y}$$

$$C_{XX} \boldsymbol{\beta} = \mathbf{c}_{Xy}$$

Feature selection

- Looking for the „best“ feature subset
- Criterion: minimization of RSS
- Traditional feature selection algorithms can be used

- Alternative: **feature shrinkage**. Regularization that enforces insignificant coefficients to zero
  - Ridge regression
  - Lasso regression

Regularized regression

- Ridge (does not actually perform FS but stabilizes the plane)

$$\sum_{i=1}^{m}(y_i - \hat{\beta}_0 - \sum_{j=1}^{p}\hat{\beta}_j x_{ij})^2 + \lambda \sum_{j=1}^{p}\hat{\beta}_j^2 = RSS + \lambda \sum_{j=1}^{p}\hat{\beta}_j^2$$

- Lasso (performs FS by forcing some betas to zero)

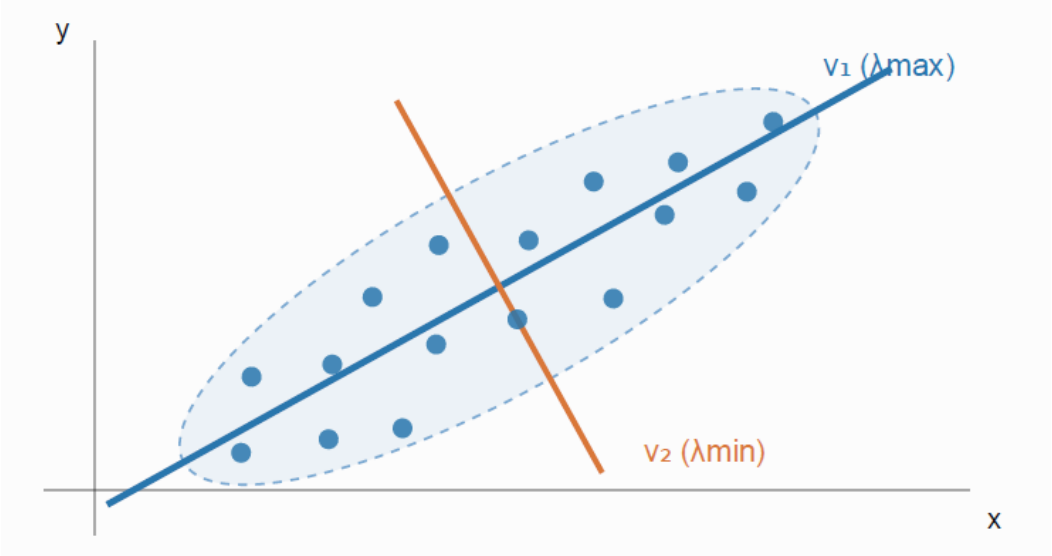
$$\sum_{i=1}^{m}(y_i - \hat{\beta}_0 - \sum_{j=1}^{p}\hat{\beta}_j x_{ij})^2 + \lambda \sum_{j=1}^{p}|\hat{\beta}_j| = RSS + \lambda \sum_{j=1}^{p}|\hat{\beta}_j|$$

- Selection of lambda by cross-validation

Total Least Squares

- Data fitting by hyperplane when errors are in all dimensions
- Criterion: minimization of sum of squared distances from the hyperplane

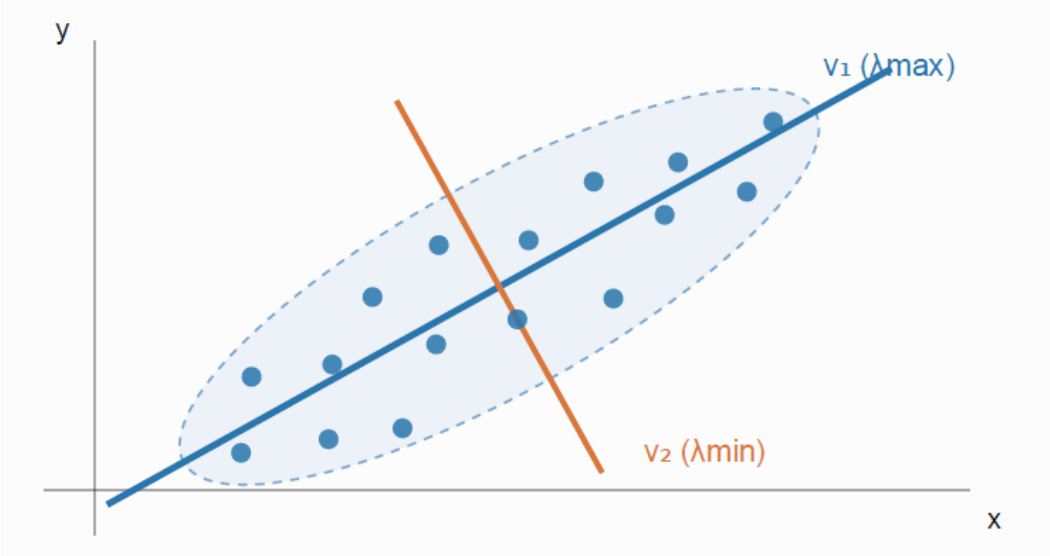
$$\min_{\|\mathbf{n}\|=1} \sum_{i=1}^{N} (\mathbf{n}^T \mathbf{z}_i)^2 = \min_{\|\mathbf{n}\|=1} N\mathbf{n}^T C\mathbf{n}$$



TLS and PCA

- Optimal hyperplane is generated by all eigenvectors of covariance matrix *C* **except the last one.**

- The last eigenvector is perpendicular to it and the eigenvalue determines the error



Thank you!

Any questions?