Nonparametric classification techniques

Taxonomy

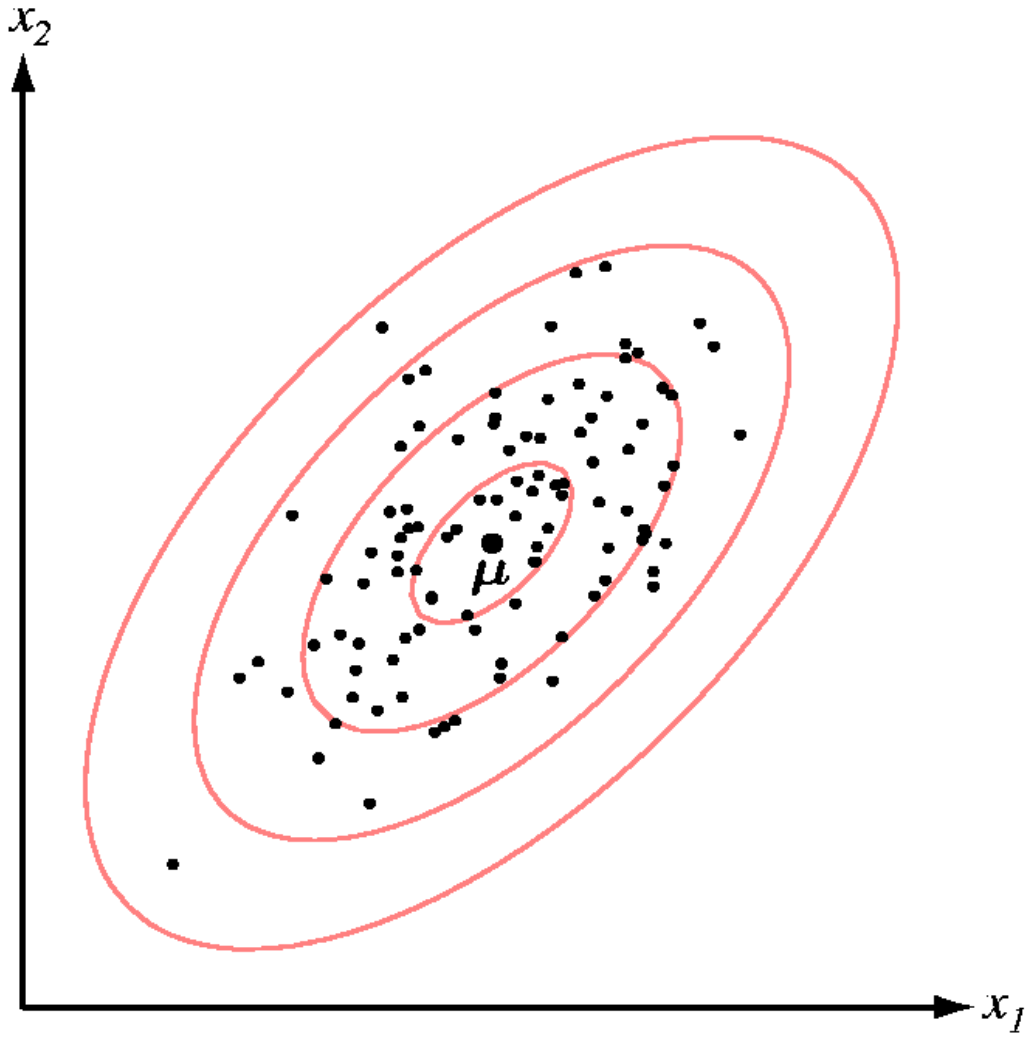
Parametric classifier – the form of the underlying density is known or assumed to be of a certain type, e.g. normal distribution which has certain parameters (mean and covariance matrix)

Nonparametric classifier – no (explicit) assumption (e.g. 'normal') is made about the underlying probability density; hence, there are no parameters

Possible approaches to nonparametric classification

- estimating the density functions $p(\mathrm{x} \mid \omega_j)$ from training data
- estimating directly the posterior probabilities $P(\omega_j \mid \mathrm{x})$

Density estimation



from Duda, Hart, Stork (2001) Pattern classification

Density estimation

A good estimate for *p*(x) is:

$$p_n(\mathrm{x}) = \frac{k_n/n}{V_n}$$

where:

- n - the total number of samples
- V_n - a volume around **x**
- k_n - the number of samples observed in that volume

Density estimation

Difficulties:

- An accurate estimate of p(x) is obtained if V is small (close to 0). However, the number of samples is limited and the volume cannot be allowed to become arbitrarily small.
- Convergence $p_n(\mathrm{x}) \rightarrow p(\mathrm{x})$ is achieved only if k_n is large;

Convergence conditions

Given:

$$p_n(\mathrm{x}) = \frac{\mathrm{k}_n/\mathrm{n}}{\mathrm{V}_n}$$

Convergence $p_n(\mathrm{x}) \rightarrow p(\mathrm{x})$ is achieved if:

(a) $\lim_{n\to\infty} V_n = 0$

(b) $\lim_{n\to\infty} k_n = \infty$

(c) $\lim_{n\to\infty} k_n / n = 0$

Approaches to density estimation

Parzen windows – choose fixed V_n (as a function of the total number of samples n), e.g.

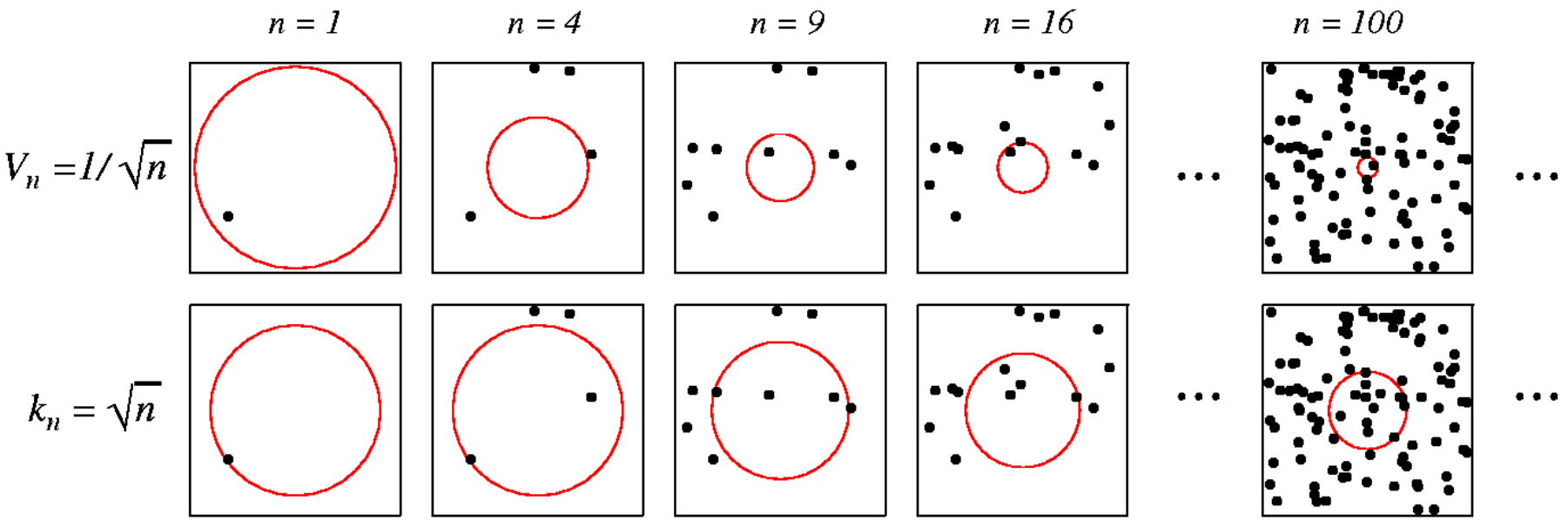
$$V_n = 1/\sqrt{n}$$

k_n nearest-neighbors – grow V_n till it includes a fixed number of neighbors k_n, e.g.

$$k_n = \sqrt{n}$$

Approaches to density estimation

Parzen windows



k_n nearest-neighbors

from Duda, Hart, Stork (2001) Pattern classification

Parzen windows

Consider a hypercube of volume $V_n = h_n^d$, with h_n the length of an edge. For estimating $p_n(\mathrm{x})$, we simply count the number of samples k_n inside the hypercube.

More generally, this can be expressed analytically by defining the *window function:*

$$\varphi(\mathrm{u}) = \begin{cases} 1, & |\mathrm{u}_j| \leq 1/2,\ j = 1 \ldots d \\ 0, & \text{otherwise} \end{cases}$$

Parzen windows

Number of samples:

$$k_n = \sum_{i=1}^{n} \varphi\left(\frac{\mathrm{x} - \mathrm{x}_i}{h_n} \right)$$

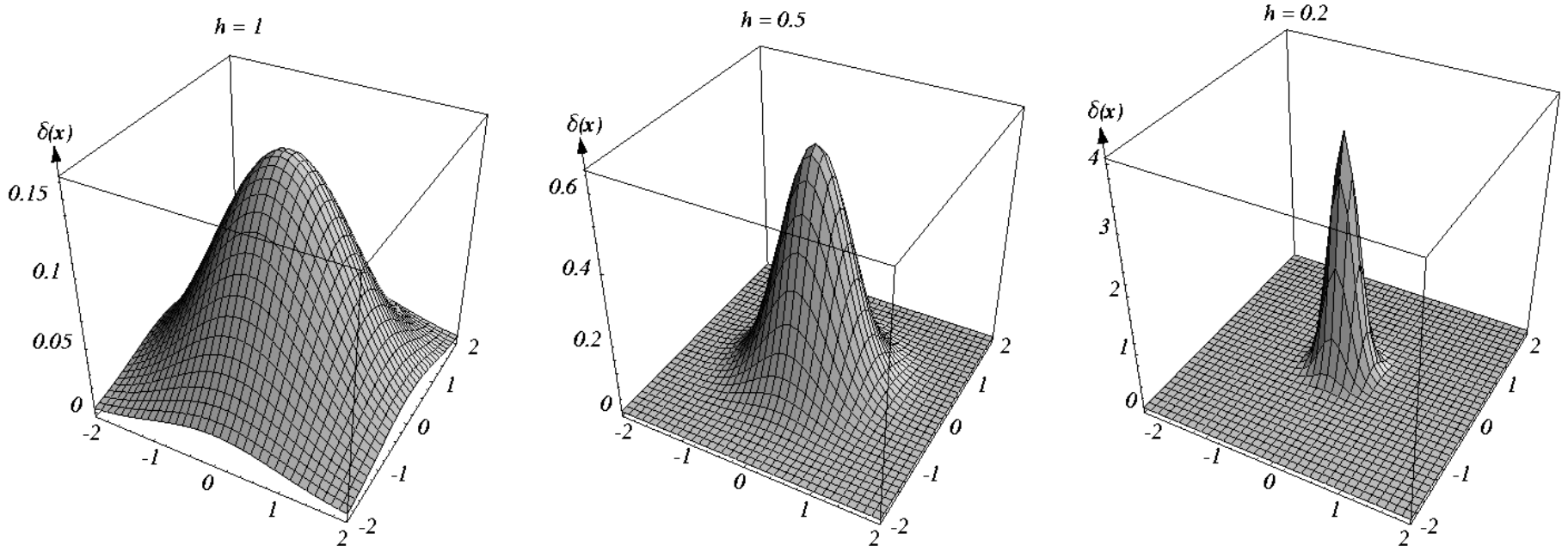
Estimate of $p(\mathrm{x})$:

$$p_n(\mathrm{x}) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{V_n} \varphi\left(\frac{\mathrm{x} - \mathrm{x}_i}{h_n} \right)$$

Parzen windows

General window function $\varphi(\mathrm{u}) \geq 0, \int \varphi(\mathrm{u}) d\mathrm{u} = 1$

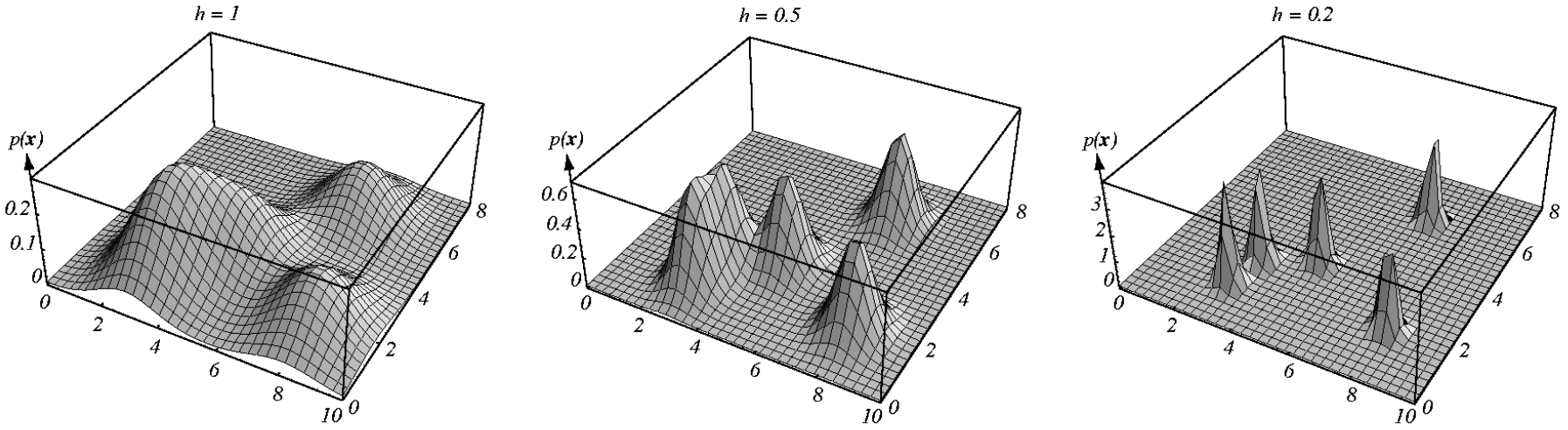
Each sample contributes differently to the estimate according to its distance to the center of the hypercube



from Duda, Hart, Stork (2001) Pattern classification

Parzen windows

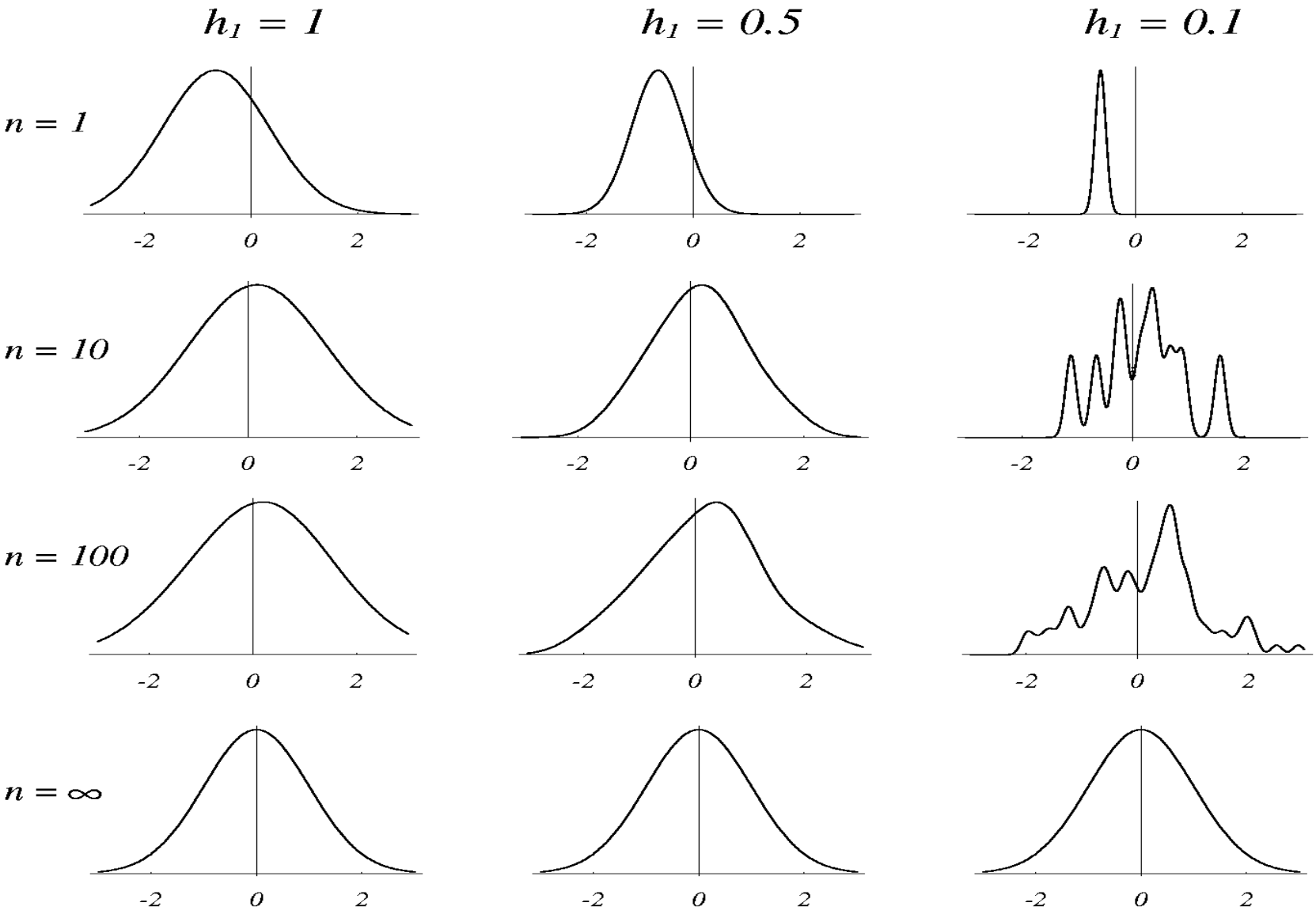
The choice of the size of V_n may have an important effect on the estimate $p_n(\mathrm{x})$. Large size leads to low resolution; small size leads to statistical variability.



Three density estimates obtained with different window sizes.

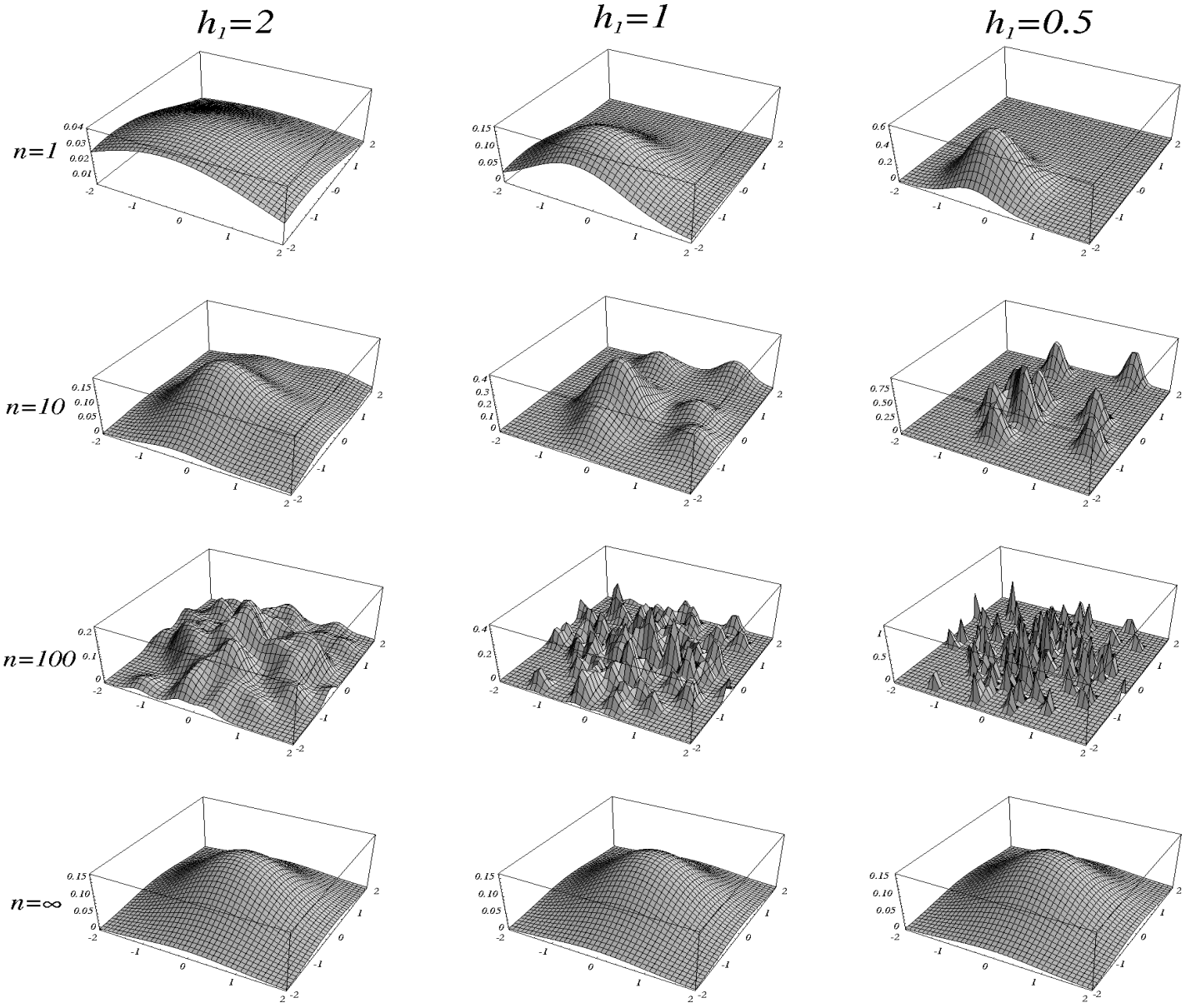
from Duda, Hart, Stork (2001) Pattern classification

Example: 1D Normal density and Gaussian window function, $h_n = h_1 / \sqrt{n}$



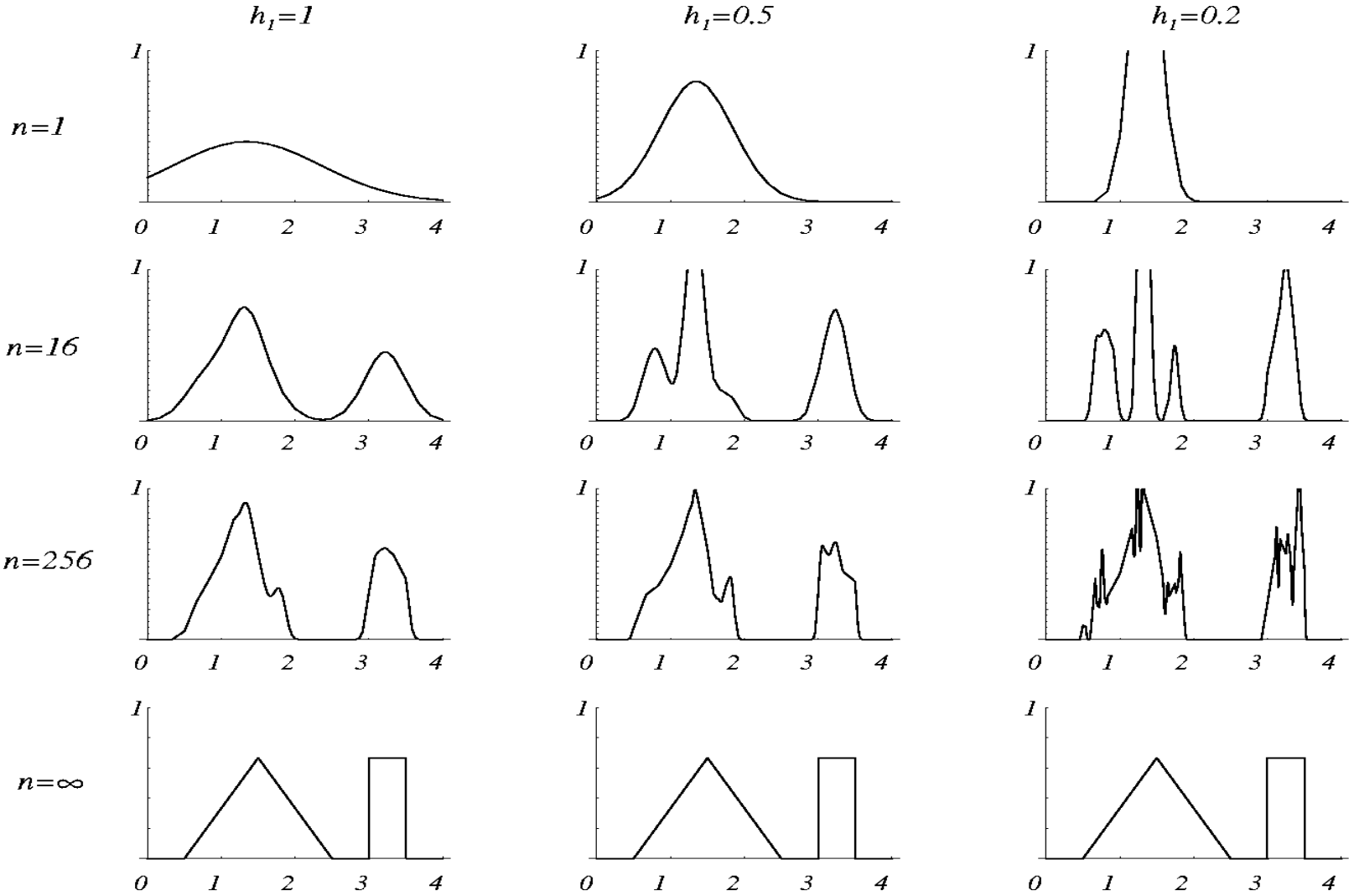
from Duda, Hart, Stork (2001) Pattern classification

Example: 2D normal density and Gaussian window function.



from Duda, Hart, Stork (2001) Pattern classification

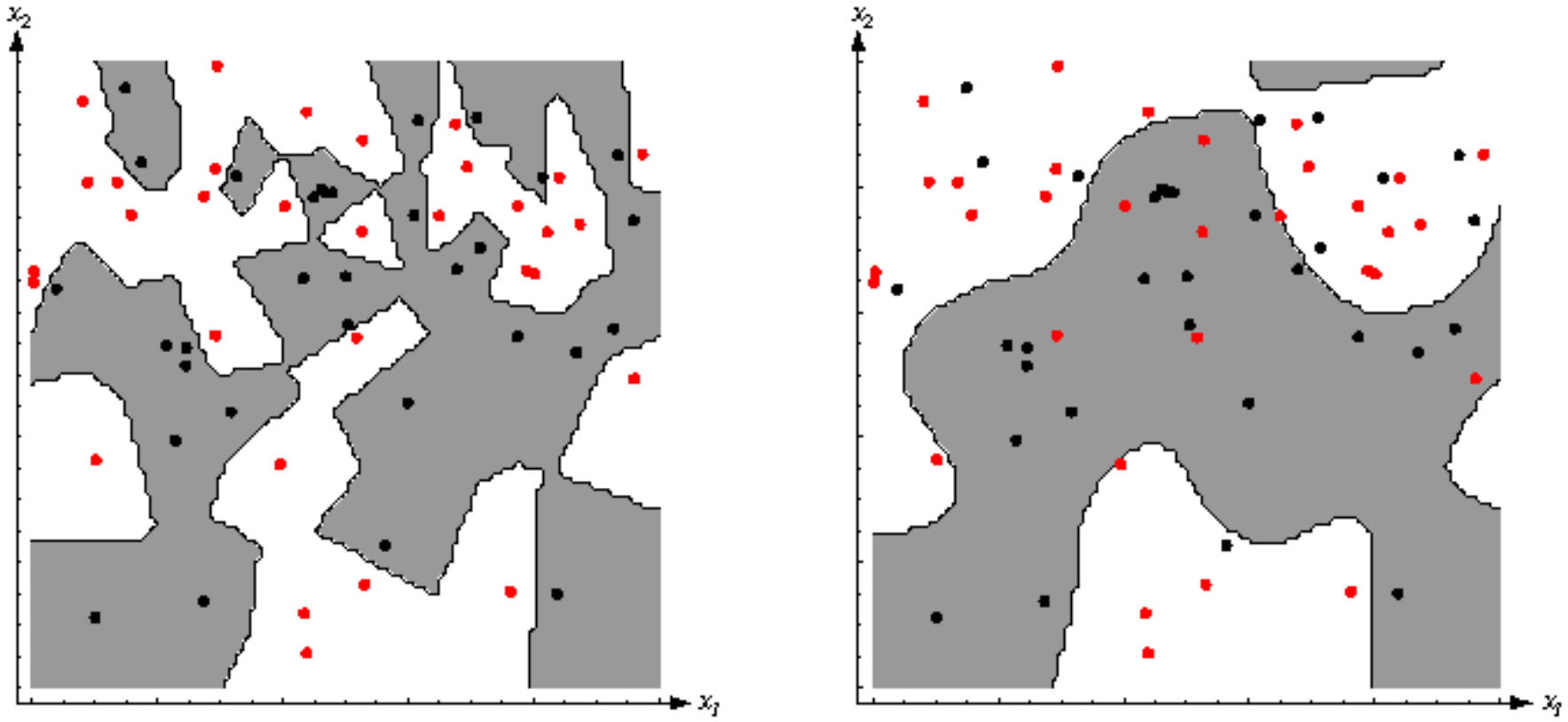
Example: 1D bimodal distribution and Gaussian window



Conclusion: good estimates if the number of samples is large

from Duda, Hart, Stork (2001) Pattern classification

Classification based on Parzen window estimation



Class regions and decision boundaries obtained with (left) small and (right) large window

from Duda, Hart, Stork (2001) Pattern classification

k_n-nearest-neighbor estimation

$$p_n(\mathrm{x}) = \frac{\mathrm{k}_n/\mathrm{n}}{\mathrm{V}_n}$$

The cell volume is a function of the available *training data.*

The cell is expanded until it encompasses k_n samples.

Similarity of k-nn and Parzen windows methods

In $$V_n(\mathrm{x}) = \frac{k_n/n}{p_n(\mathrm{x})}$$

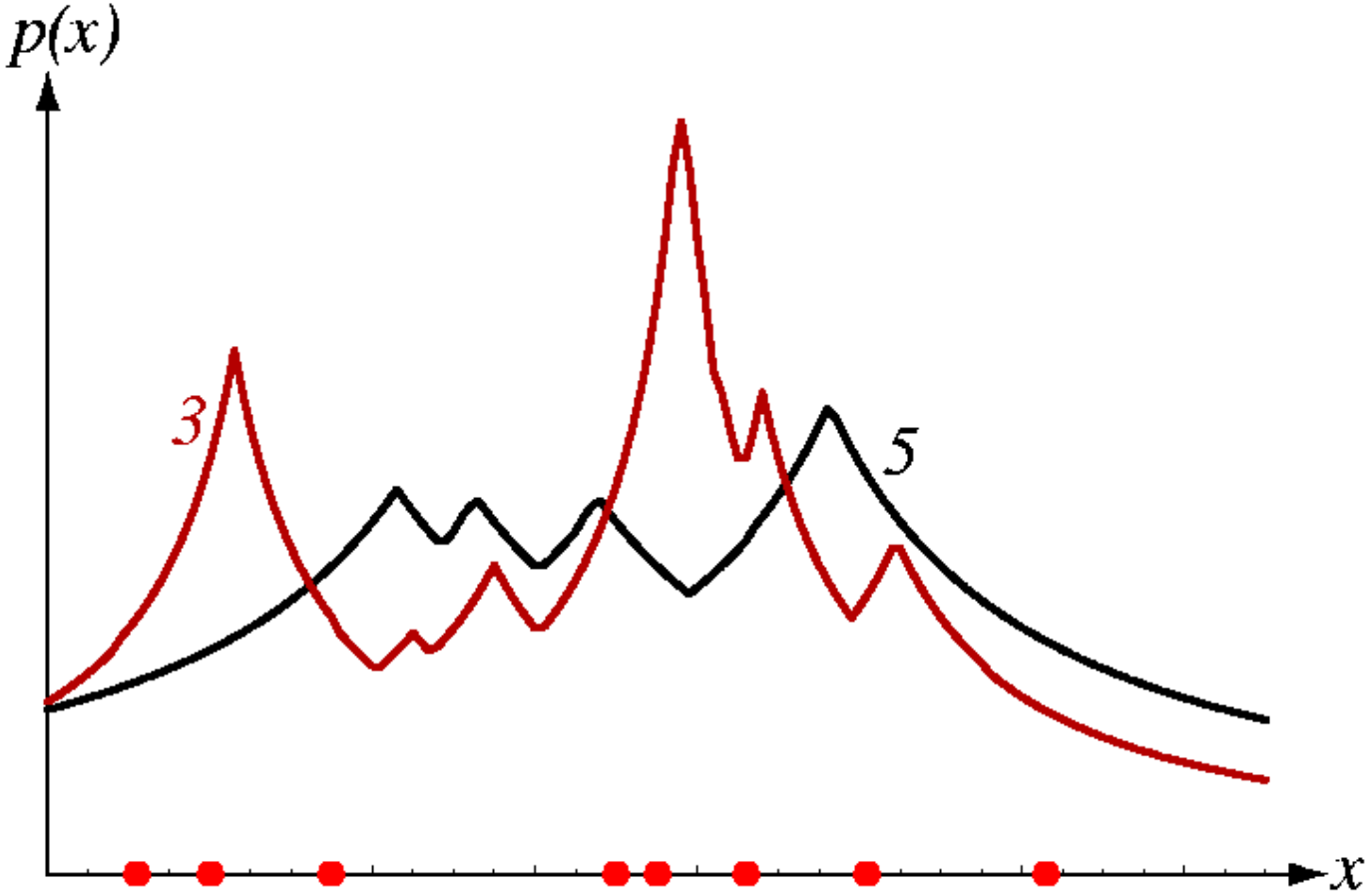
we substitute $$k_n = \sqrt{n}$$

Result: $$V_n = \frac{1}{\sqrt{n}\ p_n(\mathrm{x})} \cong \frac{1}{\sqrt{n}\ p(\mathrm{x})}$$

or $$V_n \cong V_1 / \sqrt{n}$$

(similar to Parzen window but V_1 is determined by the data)

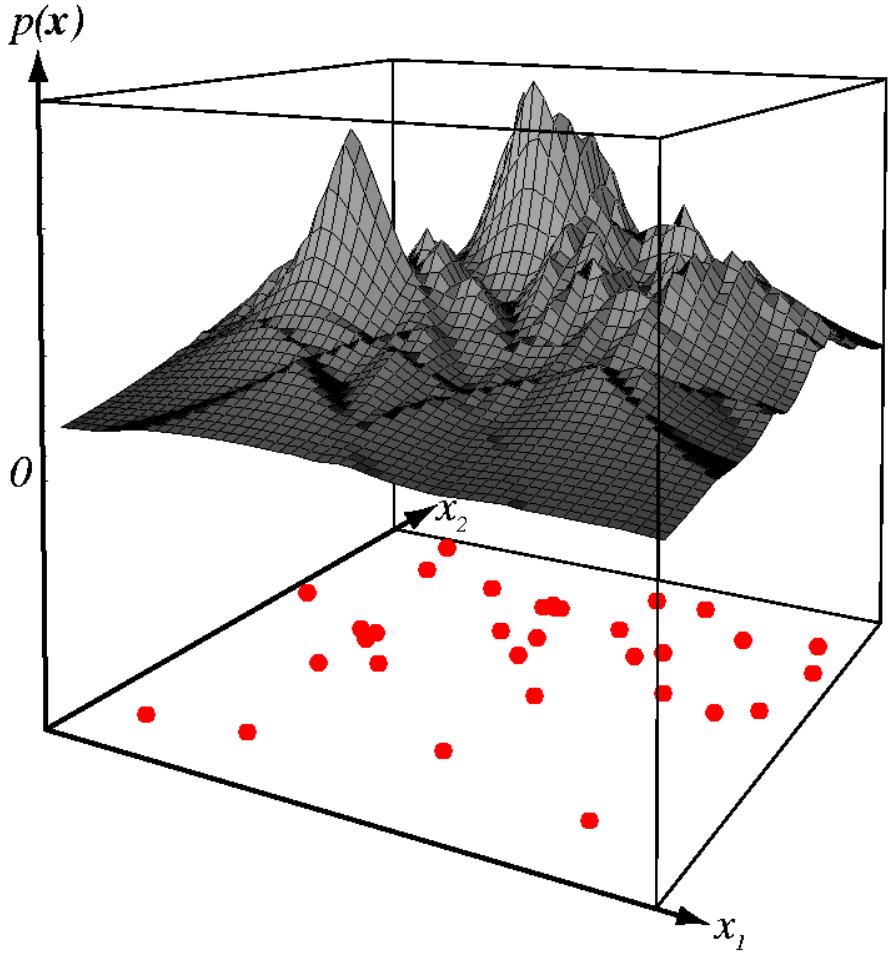
Examples of *k*-nn estimates



k-nearest-neighbor estimates for *k*=3, *k*=5

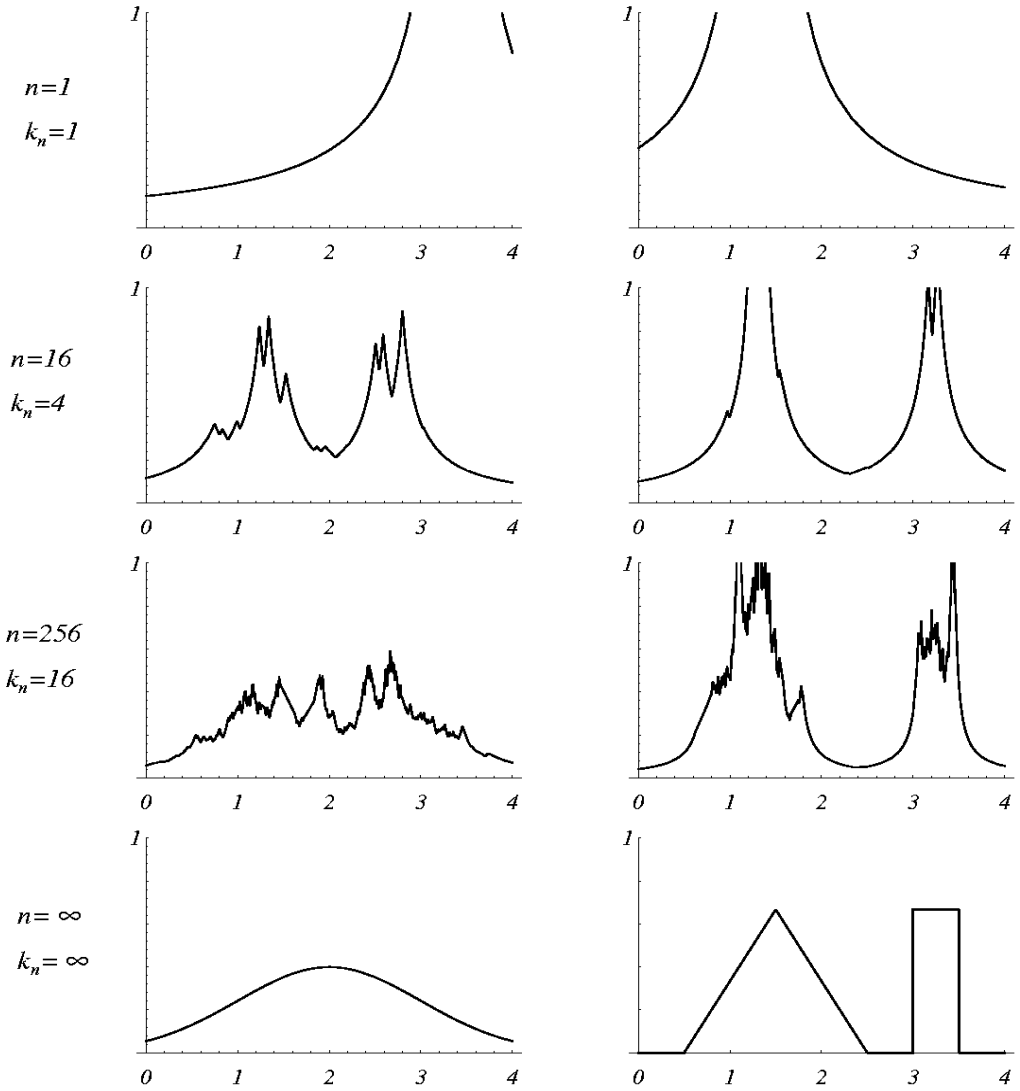
from Duda, Hart, Stork (2001) Pattern classification

Examples of *k*-nn estimates



k=5 nearest neighbor estimate in two dimensions

from Duda, Hart, Stork (2001) Pattern classification



Dependence of *k*-nn estimate of *p*(x) on *n*: (-) infinity for small *n*, (+) never becomes zero, good for large *n*

from Duda, Hart, Stork (2001) Pattern classification

Estimation of posterior probabilities

Suppose that a cell (=volume) V around **x** contains k samples of which k_i from class ω_i

An estimate for $p(\mathrm{x} \mid \omega_i)P(\omega_i)$

$$p_n(\mathrm{x} \mid \omega_i)P_n(\omega_i) = \frac{k_i/n_i}{V}\frac{n_i}{n} = \frac{k_i/n}{V}$$

Thus:

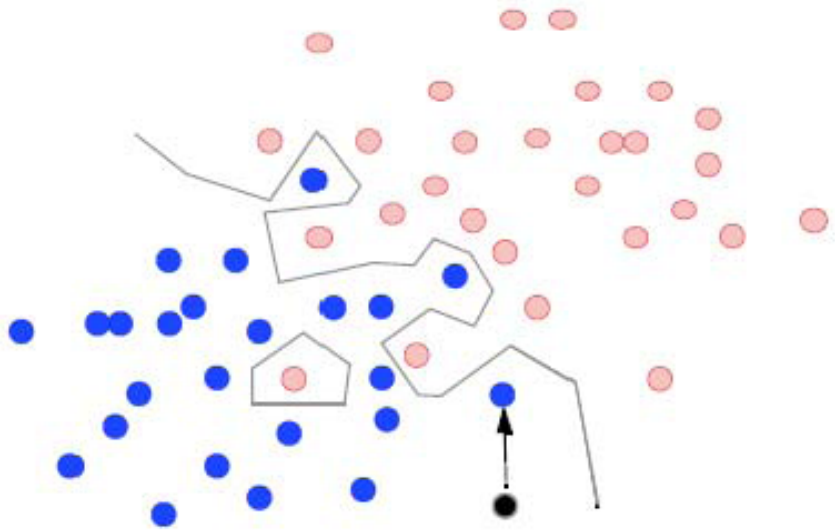
$$P_n(\omega_i \mid \mathrm{x}) = \frac{p_n(\mathrm{x}, \omega_i)P(\omega_i)}{\sum_{j=1}^{c} p_n(\mathrm{x}, \omega_j)P(\omega_j)} = \frac{k_i}{k}$$

The estimate of the posterior probability that a new vector **x** belongs to a given class ω_i is the fraction k_i/k of training vectors from that class within the concerned cell.

k-nn classification rule (for minimum error rate):
select the class that is most frequently among the k nearest neighbors.

Nearest neighbor classification (1-NN rule)

Classification rule: Assign a new feature vector $\mathbf{x}$ to class ω_i if the closest prototype belongs to ω_i



Often relies on the Euclidean distance.

Other distance measures can be used.

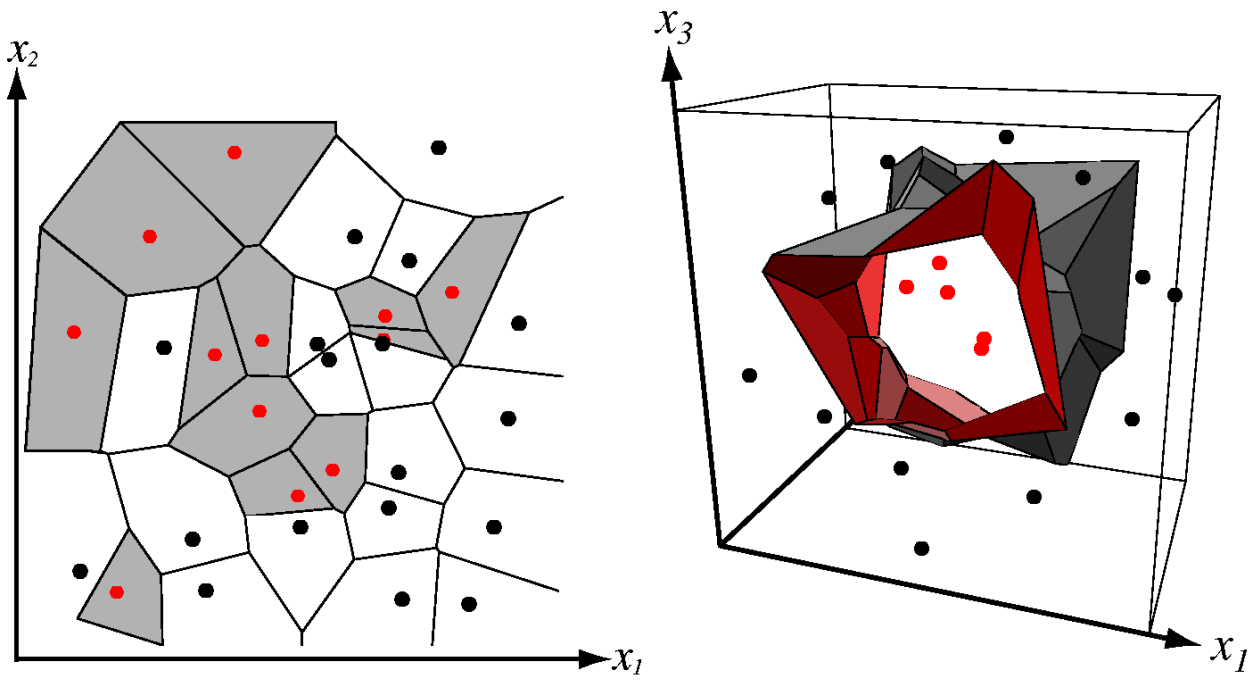
Insensitive to prior probabilities!

Scaling dependent. Features should be scaled properly.

There are no errors on the training set. The classifier is over-trained.

Nearest neighbor classification (k = 1)

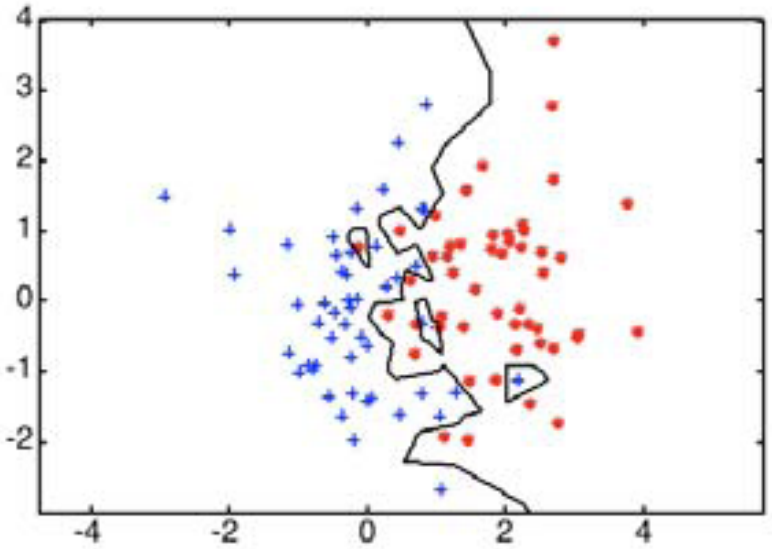
Assign a new feature vector x to the class of the closest object from the training data set

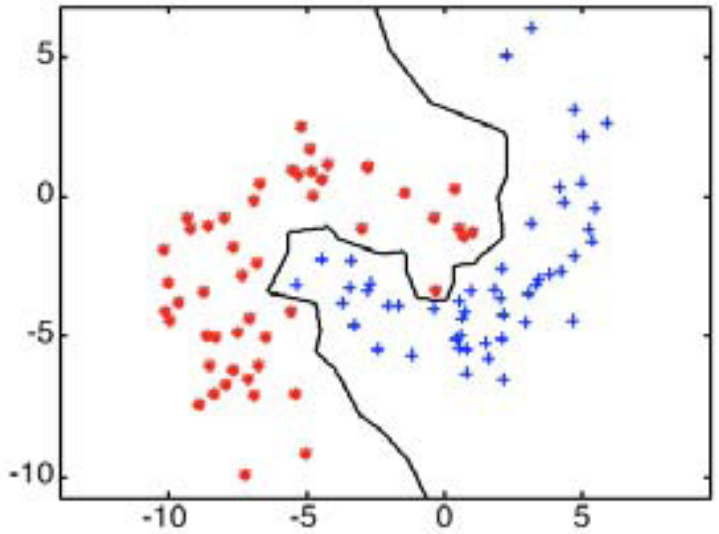


from Duda, Hart, Stork (2001) Pattern classification

Voronoi tessellation (diagram) of the feature space.
This is just an illustration of how irregular the decision boundary can be.
In practice, no such tessellation of the feature space is computed.

1-NN rule





Advantages:

- Simple.
- No training time.
- Works well for almost separable classes.
- Useful to shape non-linear decision boundaries.

Disadvantages:

- Long execution time.
- All data should be stored.

From course Advanced Pattern Recognition, TUD 2009

1-NN rule

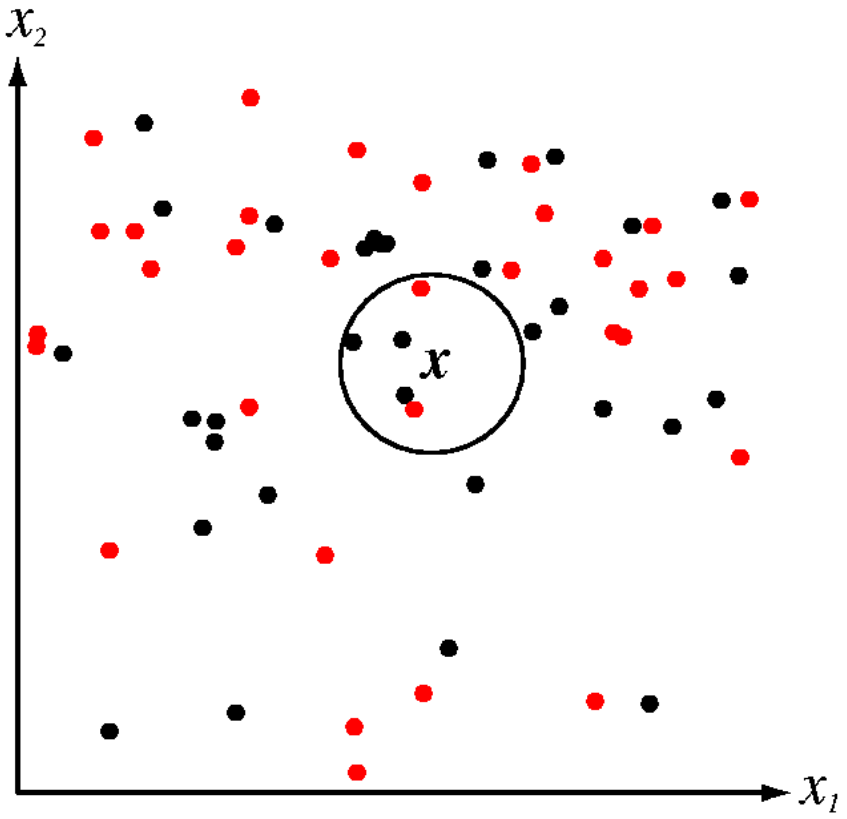
Asymptotically (for very large training sets), the error ε of the nearest neighbor rule will be less than twice the error ε^* of the best possible classifier:

$$\varepsilon^* \leq \varepsilon \leq 2\varepsilon^*(1- \varepsilon^*) \leq 2\varepsilon^*$$

1-NN is often a very good classifier!

k-nearest neighbor classification

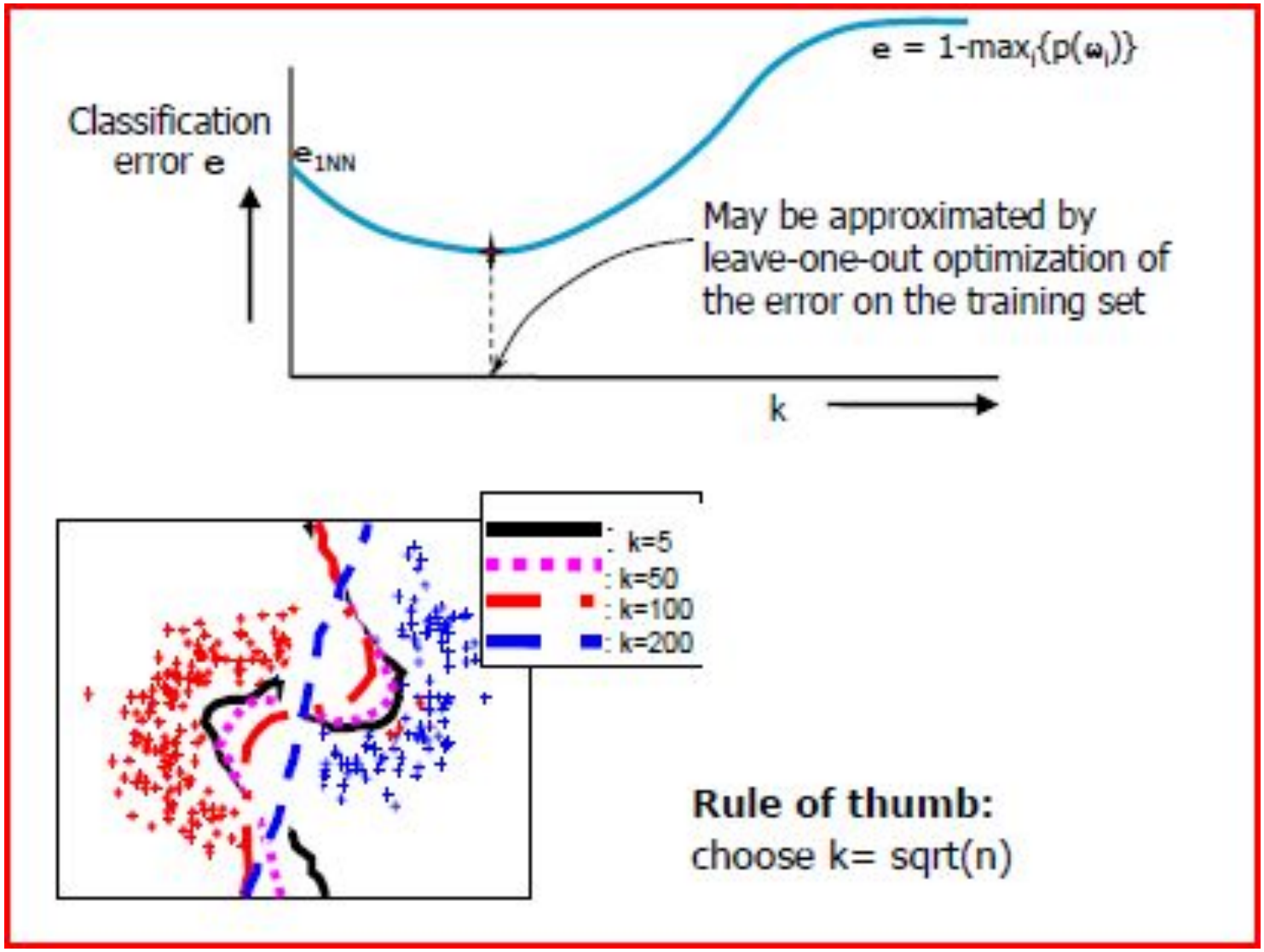
$$P_n(\omega_i \mid \mathrm{x}) = \frac{p_n(\mathrm{x}, \omega_i) P(\omega_i)}{\sum_{j=1}^{c} p_n(\mathrm{x}, \omega_j) P(\omega_j)} = \frac{k_i}{k}$$



Rule: Assign x to the class most frequently represented among its k nearest neighbors in the training set.

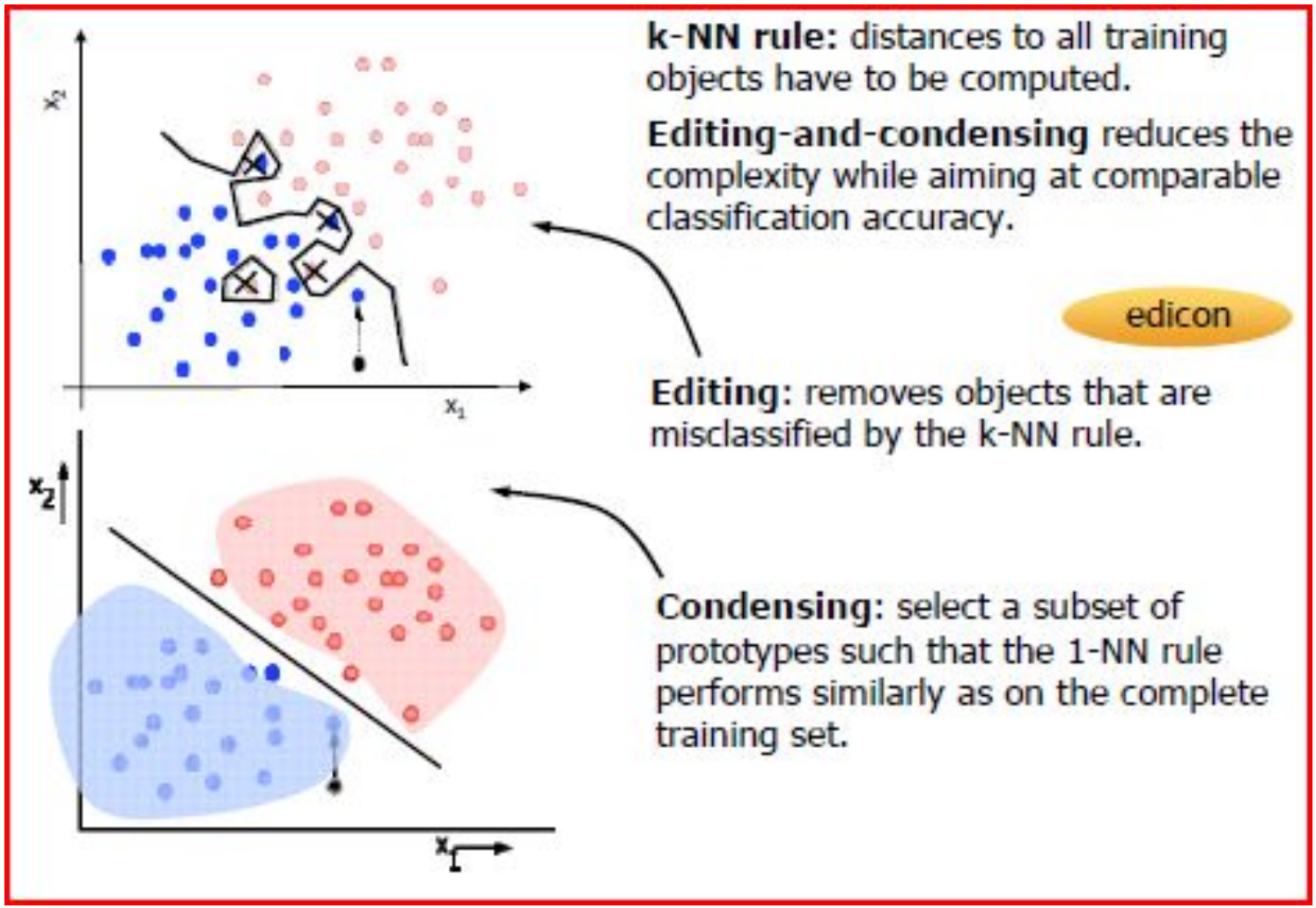
from Duda, Hart, Stork (2001) Pattern classification

k-NN decision boundaries: optimal k in the k-NN rule



From course Advanced Pattern Recognition, TUD 2009

Nearest prototype rule: editing and condensing



From course Advanced Pattern Recognition, TUD 2009

Before starting k-nn classification

A feature that has a wide range of possible values, much larger than the domains of other features, can be dominating in the computation of distances and render all other features useless.

Therefore, before starting to use the k-nn method (or any other method based on distances) it is customary to normalize features.

One possible way to do this is by substituting feature values by their Z-scores. The Z-score of a feature value is computed by computing the mean and the standard deviation of the set of observed values of a given feature across all classes. The Z-score of a given feature value is the deviation of that value from the mean divided by the standard deviation.

Parametric vs. non-parametric methods

Bayesian with e.g. normal distributions:

An assumption is made about the involved distributions (e.g. normal)

A distribution is characterized by some parameters, such as the mean and the covariance matrix. Therefore, such methods are called parametric methods.

k-NN:

No assumptions about the underlying distributions (e.g. Gaussian). No parameters of any distribution are involved. Therefore, such methods are called non-parametric methods.

k-nearest neighbor classification (comparison with parametric methods)

k-NN:

☺ No assumptions about the distributions (e.g. Gaussian)

☹ For the classification of a new point, distances need to be computed to all training data points. All training data needs to be stored

Bayesian with e.g. normal distribution:

☺ Once the parameters of the distributions are estimated, we know the classification result in every point of the feature space: evaluate discriminant functions

☹ Assumption about the distributions (e.g. Gaussian)

k-nearest neighbor classification (comparison with LVQ)

Both methods are based on distance computations

k-NN:

☺ Uses all training data

☺ Good theoretical foundation (related to Bayesian …)

☹ For the classification of a new point, distances need to be computed to all training data points. All training data need to be stored

LVQ:

☺ Less computations – compute distances only to prototypes

☹ Based on heuristics and experience (it works)

k-nearest neighbor classification (advantages and disadvantages)

Advantages:

☺ no assumptions about the distributions (e.g. Gaussian)

☺ uses all available training data (no information reduction)

☺ has good theoretical foundation (related to Bayes theory)

Disadvantages:

☹ Distances need to be computed to all training data points

☹ All training data needs to be stored

k-nearest neighbors for database retrieval

Given a query, retrieve from a database similar instances

Example: content base image retrieval

Image retrieval

http://images.google.com

Key word based image retrieval

http://images.google.com

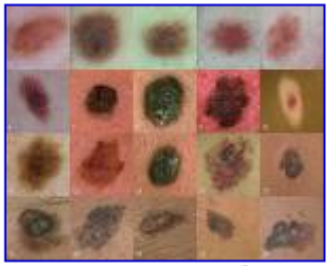
Het internet **Afbeeldingen** Maps Nieuws Video Gmail meer ▾ **sgorpi@gmail.com** | Mijn account | Afmelden

Google

skin lesion | Afbeeldingen zoeken | Het internet

Gemiddelde SafeSearch is aan

Afbeeldingen Weergeven: Alle afbeeldingsformaten Resultaten **1 - 20** van circa **45.200** voor **skin lesion** (0,13



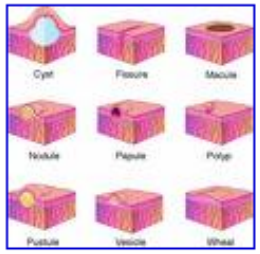
Twenty images of **skin lesions**.
1280 x 1024 - 274 kB - jpg
www.cs.wright.edu

... plaque-like **skin lesion**.
680 x 521 - 175 kB - jpg
www.ispub.com

Configuration of **Skin Lesions**
263 x 400 - 10 kB - jpg
www.med-ed.virginia.edu

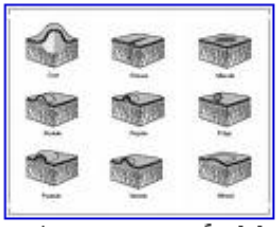
... a slowly progressive **skin lesion ...**
900 x 600 - 100 kB - jpg
www.afids.org

... infections on the **skin** do



Skin Lesions

... infections on the **skin** do



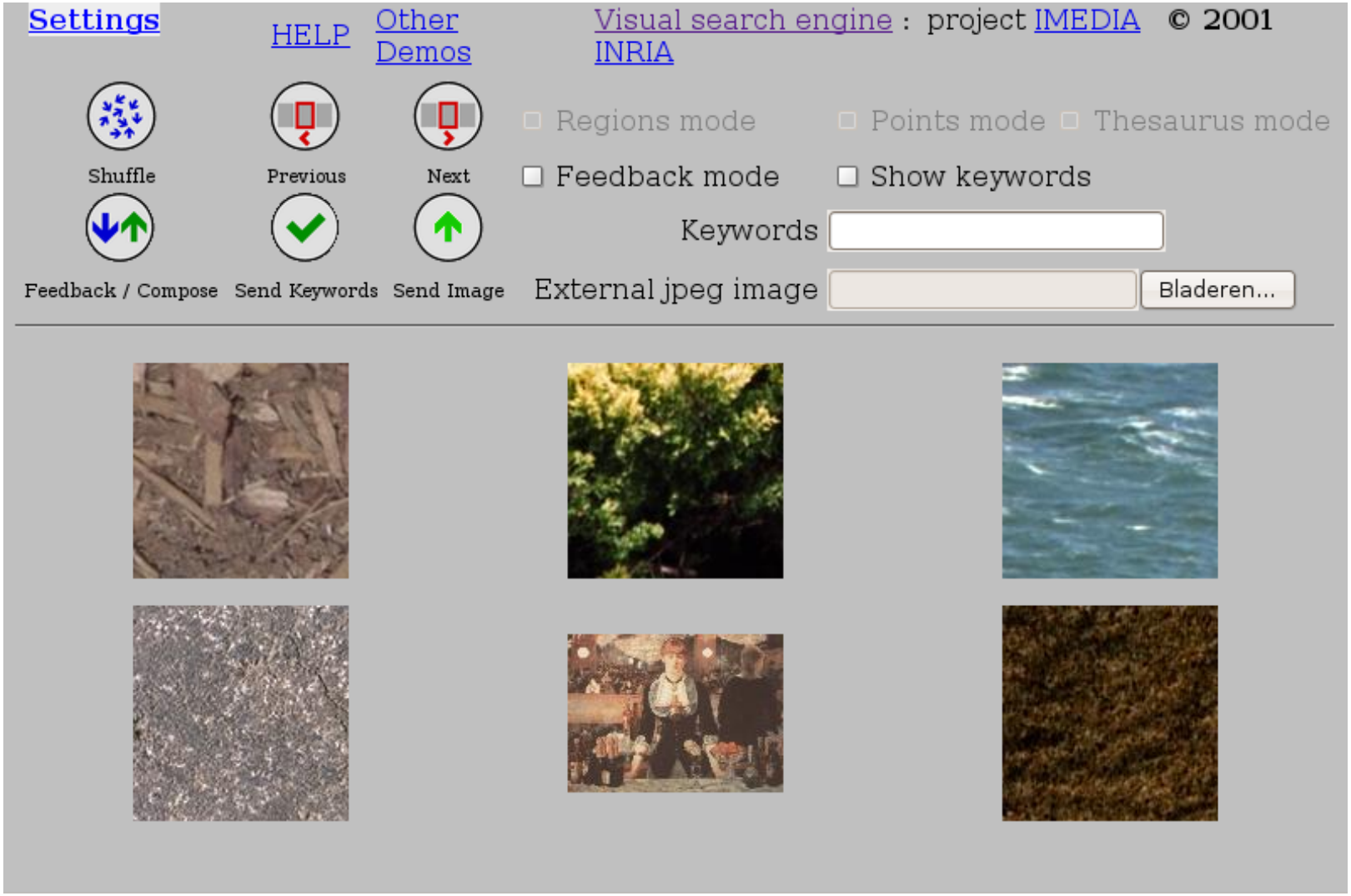
Various types of **skin**

Content based image retrieval

- Search for images that are similar to an input image using image content, such as:
  - color
  - texture
  - shape
  - text tags
  - etc...
- See http://en.wikipedia.org/wiki/CBIR

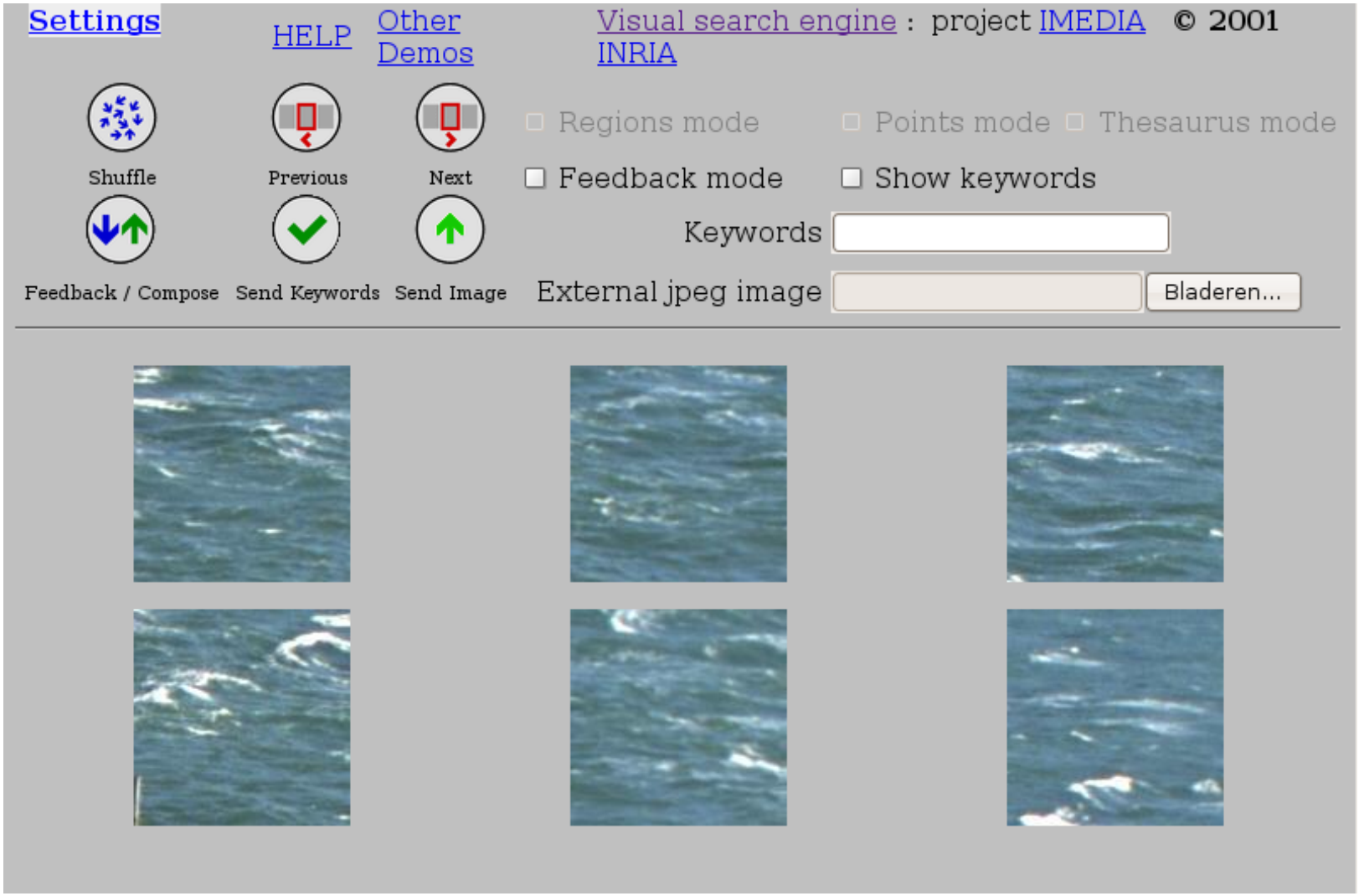
IMEDIA project – select query

http://www-rocq.inria.fr/imedia/



iMedia project – query results

http://www-rocq.inria.fr/cgi-bin/imedia/circario.cgi/v2std



GIFT – select query

http://viper.unige.ch/demo/php/demo.php

Random (fetch a random set of images) Query (launch the query) Clear (Clear the query)

 Similarity: 1.000000 neutral top	 Similarity: 1.000000 rel top	 Similarity: 1.000000 neutral top	 Similarity: 1.000000 neutral top	 Similarity: 1.000000 neutral top
 Similarity: 1.000000 neutral top	 Similarity: 1.000000 neutral top	 Similarity: 1.000000 neutral top	 Similarity: 1.000000 neutral top	 Similarity: 1.000000 neutral top

GIFT – query result

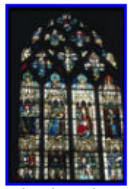
http://www.gnu.org/software/gift/gift.html

Random (fetch a random set of images) | Query (launch the query) | Clear (Clear the query)

Query:

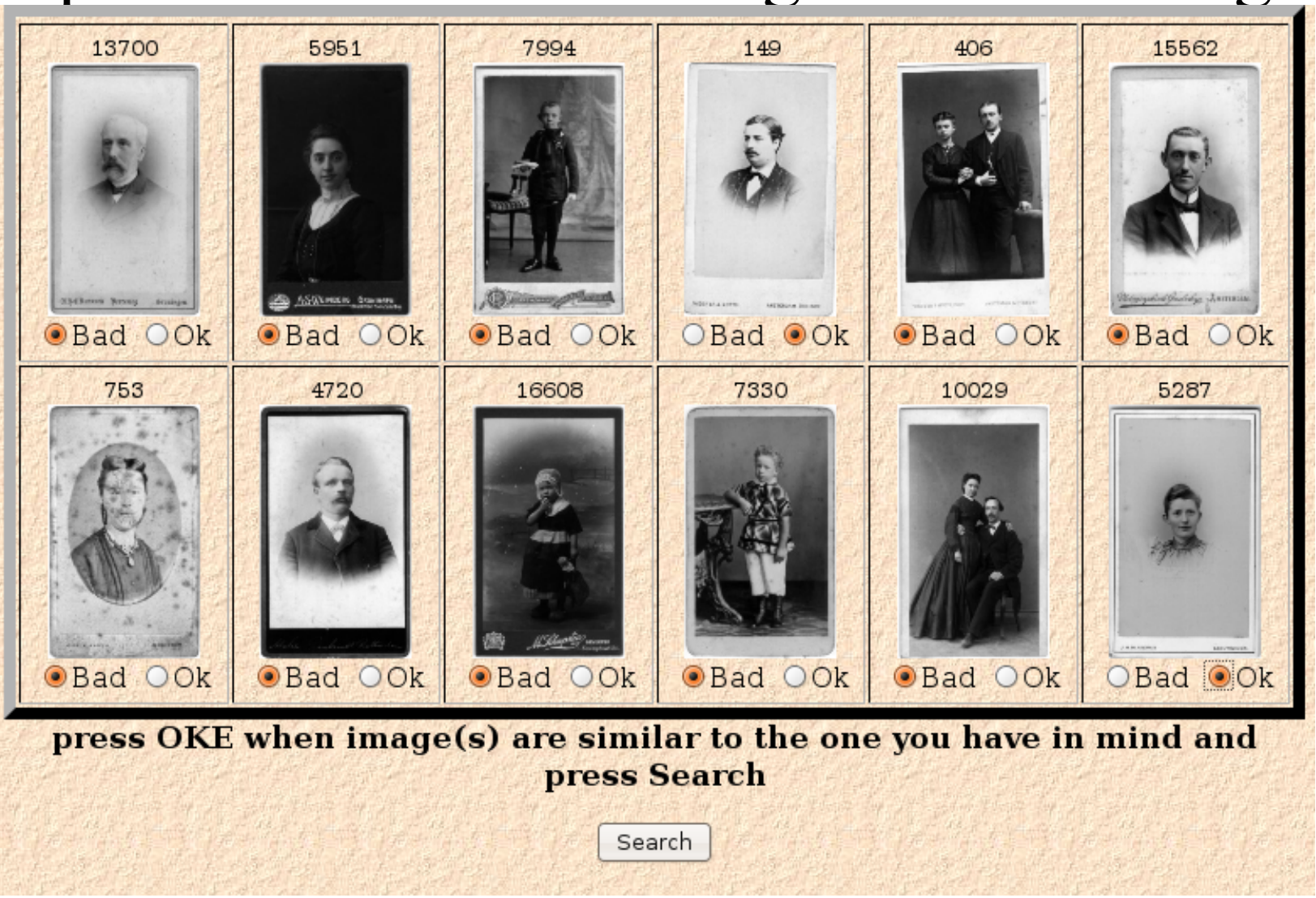
rel

Result:

 Similarity: 1.000000 Query Image top	 Similarity: 0.171830 neutral top	 Similarity: 0.160725 neutral top	 Similarity: 0.144877 neutral top	 Similarity: 0.141121 neutral top
 Similarity: 0.139827 neutral top	 Similarity: 0.137843 neutral top	 Similarity: 0.135897 neutral top	 Similarity: 0.135666 neutral top	 Similarity: 0.135661 neutral top

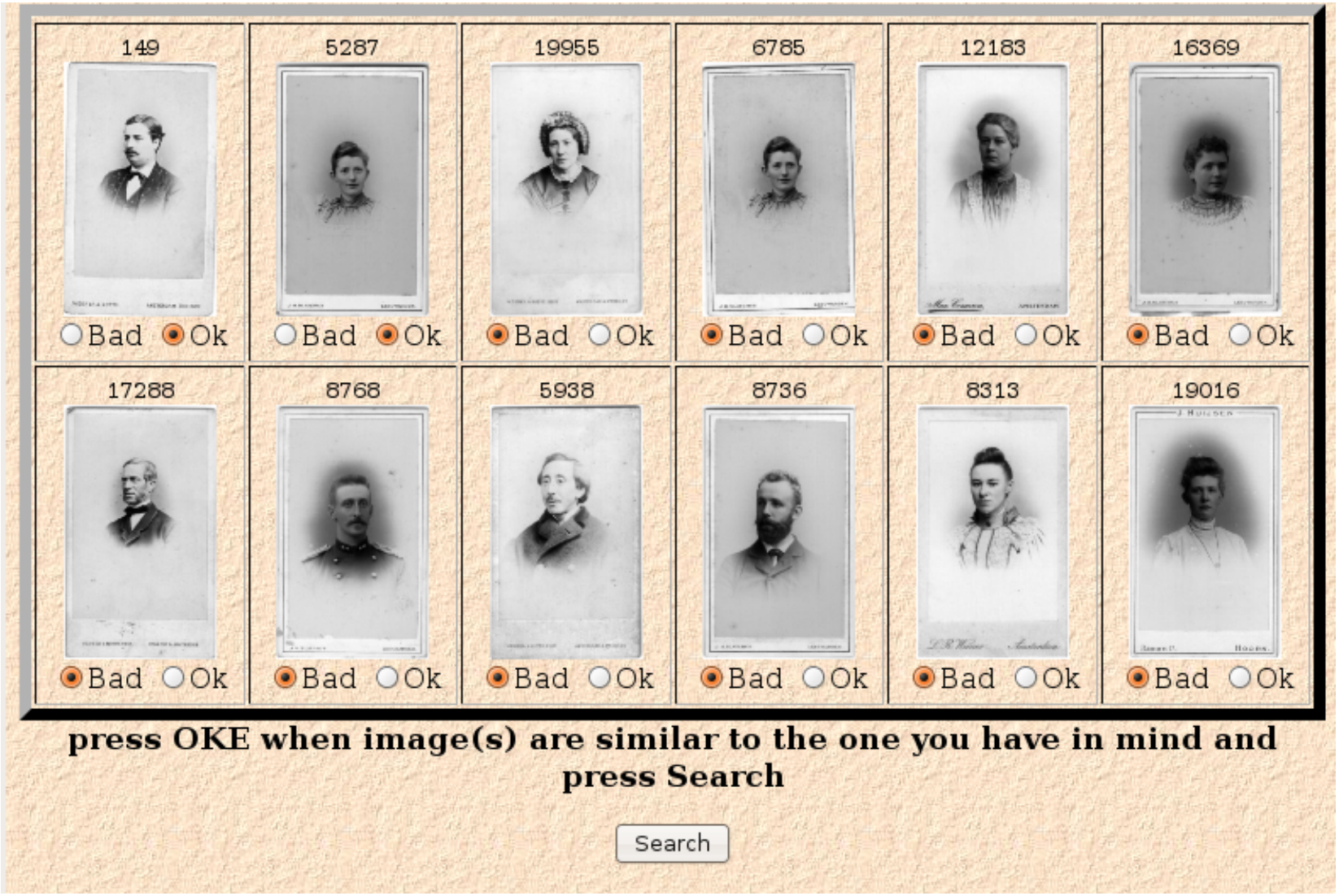
Leiden 19th-century portrait database

http://nies.liacs.nl:1860/cgi-bin/FindImage.pl

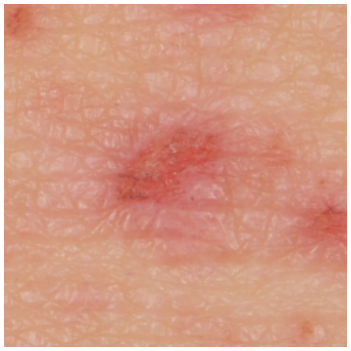


Leiden 19th-century portrait database

(result examples)

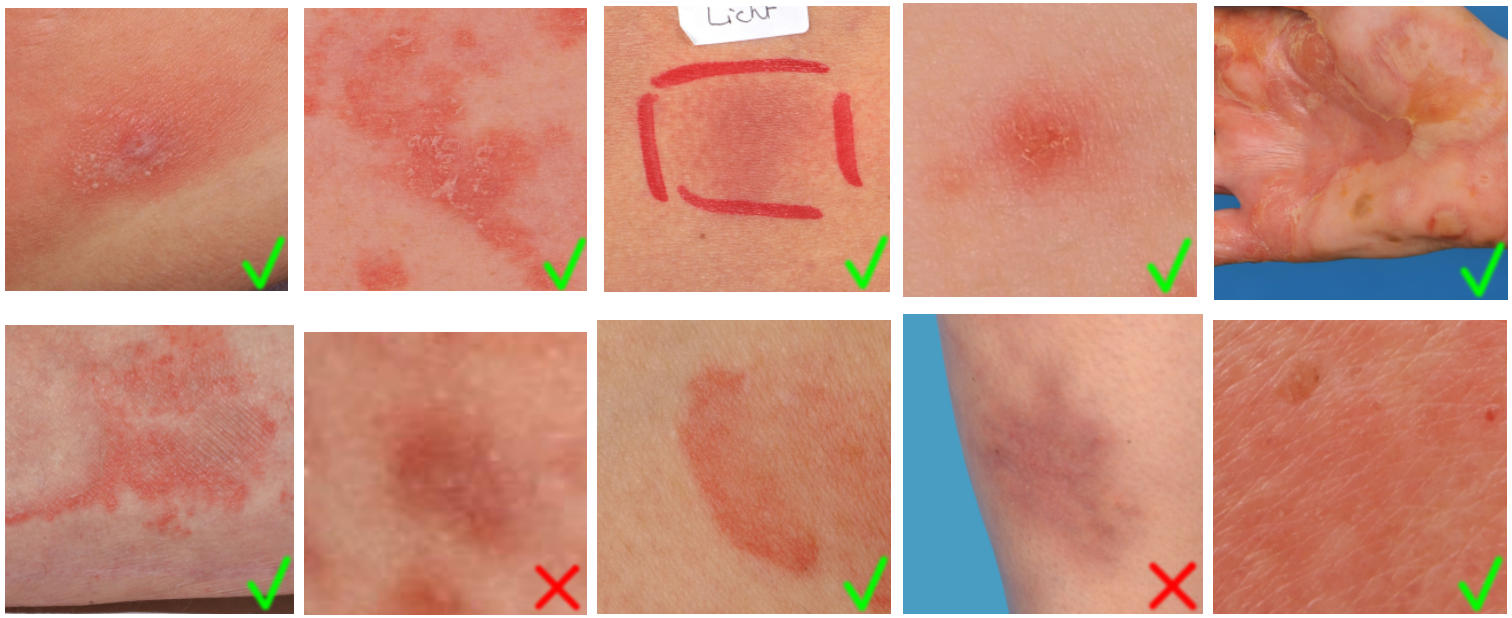


CBIR in dermatology



Query

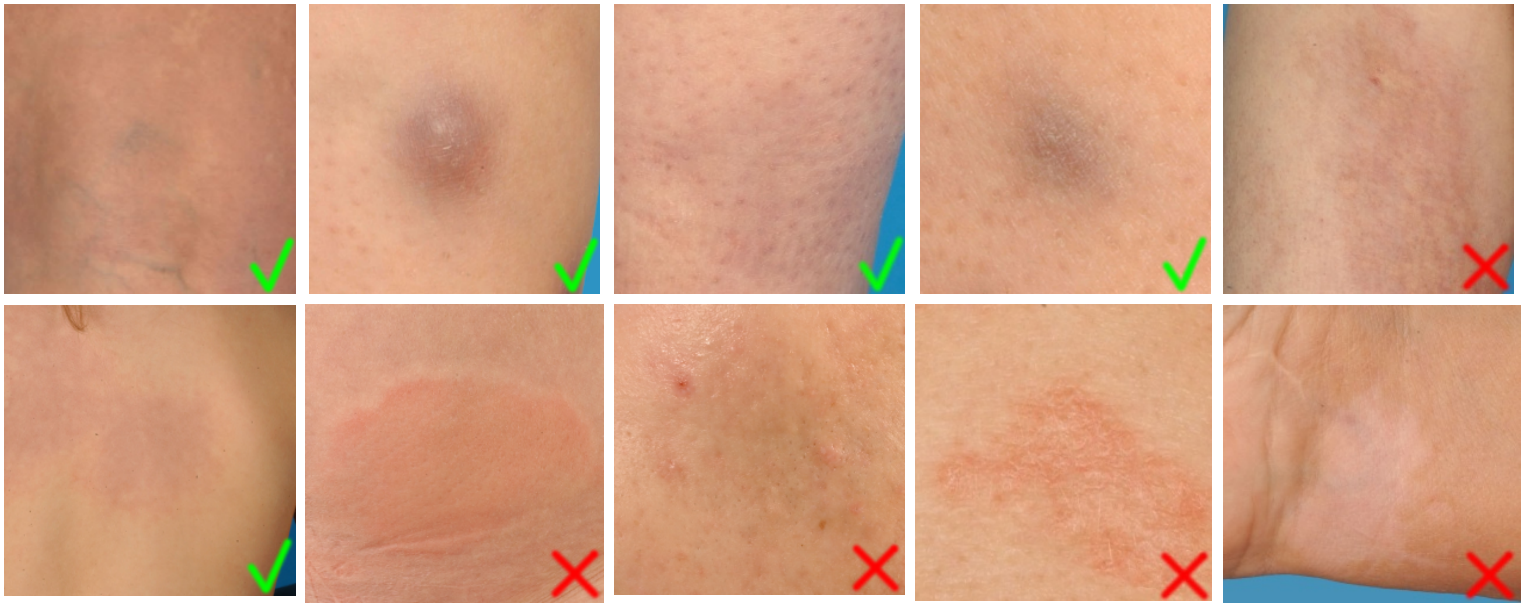
Results



CBIR in dermatology

Query

Results



k-nearest neighbors
(How CBIR is done)

Feature vector computation

Reduce the image content to some feature vector (descriptor)

Color feature vector representation of an image

$(R_h,G_h,B_h,R_l,G_l,B_l) = (89,110,41,50,172,30)$

Use of the colors of healthy and lesional skin

MPEG-7 image descriptors

- http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm

Computational complexity of k-nn

In the general *d*-dimensional case, a straightforward naive approach would be to compute the distance of a point **x** to all training set points x_i, $i = 1 \ldots n$.

Each distance calculation is $O(d)$, total $O(nd)$.

Reducing the computational complexity of k-nn

Approaches:

(a) computing partial distances;

(b) pre-structuring;

(c) editing stored prototypes.

Reducing computational complexity of k-nn - partial distances

(a) Partial distances:

Compute distances in a selected number of dimensions *r*

$$D_r(\mathrm{a},\mathrm{b}) = \left(\sum_{i=1}^{r} (a_i - b_i)^2 \right)^{1/2}$$

Do not further compute distances if the partial distance is greater than the full distance to the current nearest prototype.

Reducing computational complexity of k-nn – pre-structuring

(b) Pre-structuring:

Create an additional data structure, usually a *search tree* in which the prototypes (or only a subset of them) are linked according to some criteria.

Reducing computational complexity of k-nn – editing (for k=1)

(c) Editing:

Reduce the number of prototypes by eliminating the unnecessary ones (*pruning* or *condensing)*.

Example: eliminate the prototypes that are surrounded by training points of the same category (this procedure leaves the decision boundaries and errors unchanged).

Reducing computational complexity of k-nn – combined techniques

The three techniques can be combined:

1. edit the prototypes
2. construct a search tree
3. compute partial distances during classification

Metrics and nearest neighbor classification

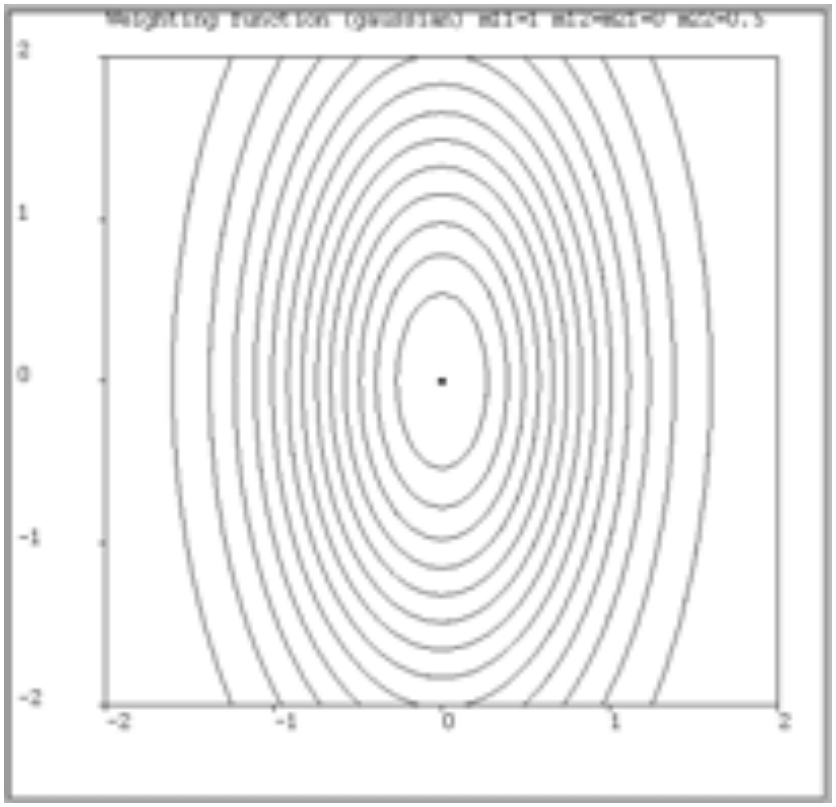
Minkowski metric $L_k(\mathrm{a},\mathrm{b}) = \left(\sum_{i=1}^{d} | a_i - b_i |^k \right)^{1/k}$

Special cases:

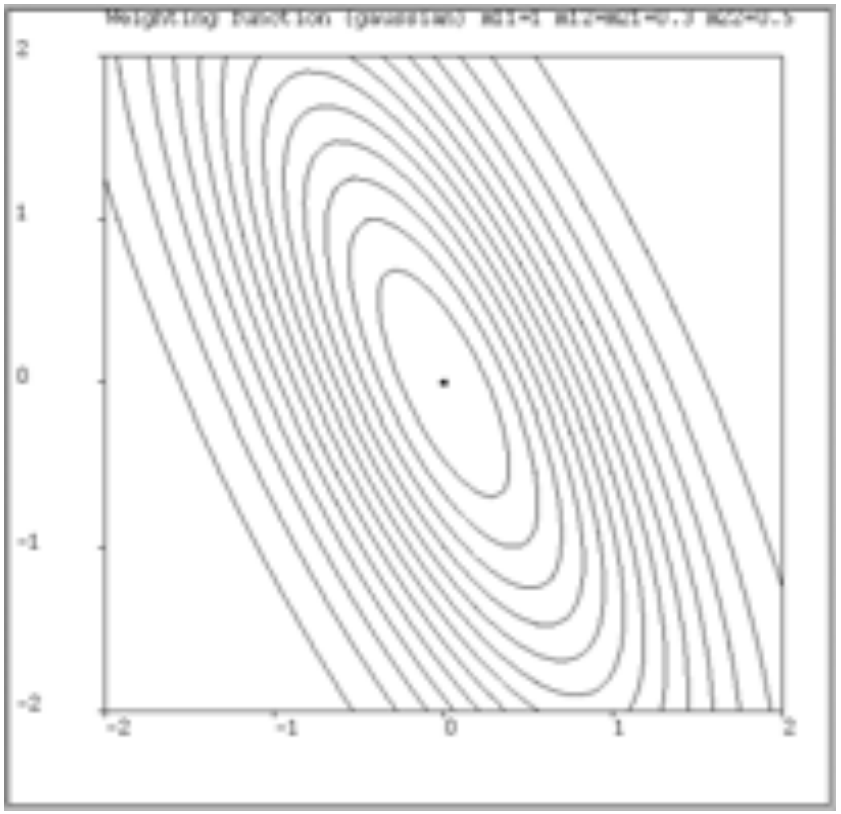
L_1 norm (Manhattan or city block distance) $L_1(\mathrm{a},\mathrm{b}) = \sum_{i=1}^{d} | a_i - b_i |$

L_2 norm (Euclidian distance) $L_2(\mathrm{a},\mathrm{b}) = \left(\sum_{i=1}^{d} (a_i - b_i)^2 \right)^{1/2}$

L_∞ norm $L_\infty(\mathrm{a},\mathrm{b}) = \max_{i=1}^{d} | a_i - b_i |$



Scaled Euclidian $(2x)^2+y^2$



Mahalanobis $(2x+y)^2+(0.5x-y)^2$

Cov 4.25 1.5
1.5 2

Nonparametric Techniques

Summary and conclusions:

- Techniques for density estimation and classification: Parzen windows, k-nearest neighbors
- Good theoretical basis and conceptual simplicity ☺
- Applied in situations in which the analytical form of the distribution of feature vectors is not known ☺
- Many samples needed for good results ☹
- All training data has to be kept (no data reduction) -> heavy requirements on storage and computation ☹
- Number of samples needed grows exponentially with the dimensionality of the feature space (curse of dimensionality) ☹