

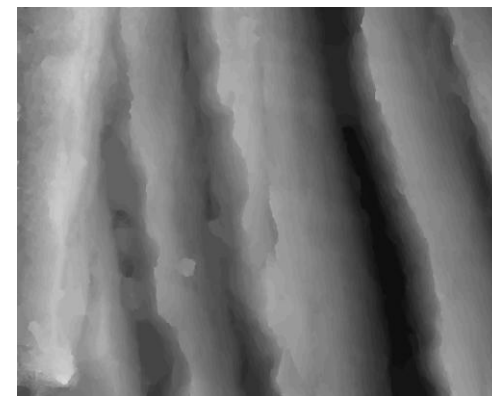
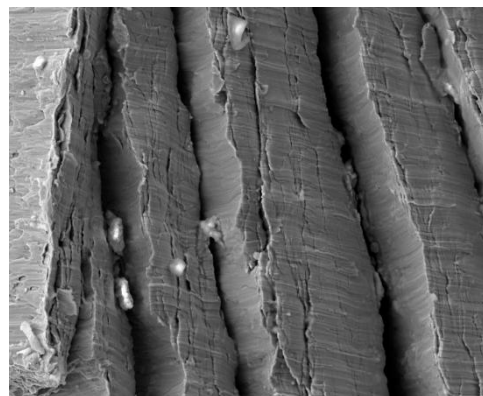
Bildverarbeitung und Mustererkennung

Image Processing and Pattern Recognition

710.080 2VO

710.081 1KU

$$\int_{\Omega} f(x, u(x), \nabla u(x)) dx \Leftrightarrow \sup_{\phi \in K} \int_{\Omega \times R} \phi \cdot D\mathbf{1}_u$$





Bildverarbeitung und Mustererkennung

Image Processing and Pattern Recognition

- **Supervised Classification**
 - Bayesian Decision Theory
 - Parametric vs. Non-Parametric Classification
 - Support Vector Machine
- **Unsupervised Classification**

Classification

- Task: Given a feature vector, provide a class label according to a (learned) mapping

$$\Theta : X \rightarrow \Omega$$

$$X = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{pmatrix} \rightarrow \begin{pmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_c \end{pmatrix} = \Omega$$



Supervised Classification

- Data vectors

$$X = \{\vec{x}_k \in \mathbb{R}^d \mid k = 1, 2, \dots, m\}$$

- Known class labels

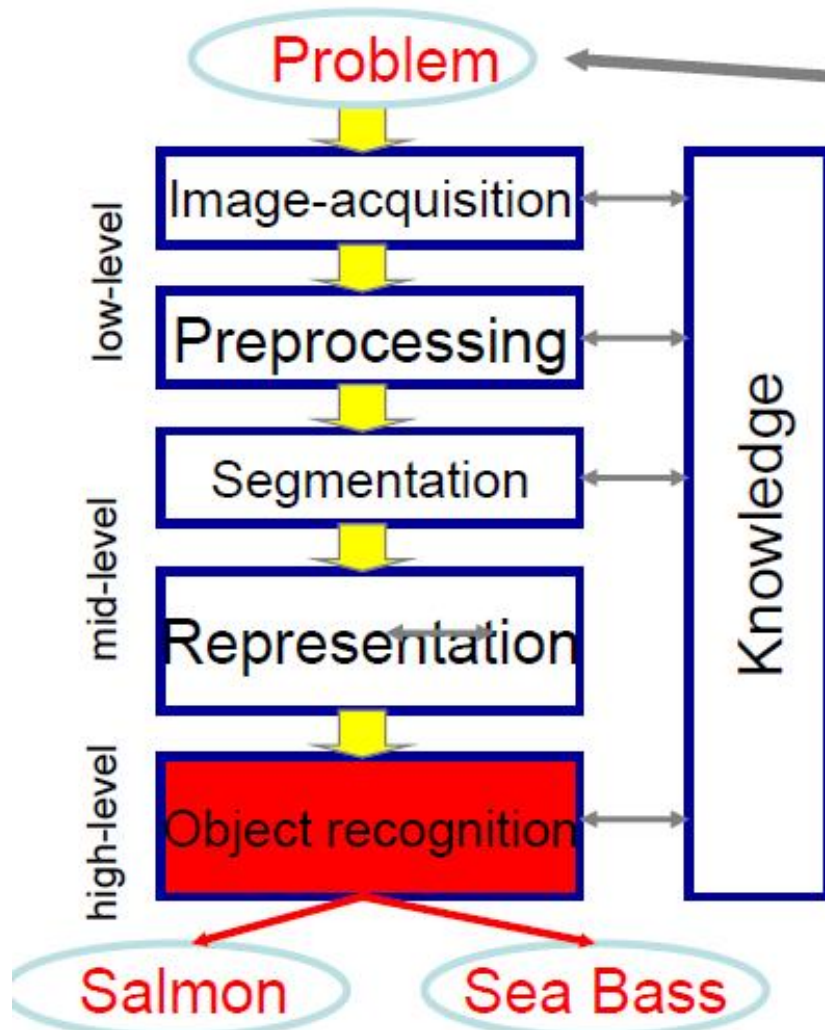
$$\Omega = \{\omega_i \mid i = 1, 2, \dots, c\} \quad c \in \mathbb{N}, c \geq 2$$

- Learn a classifier

$$\Theta : X \rightarrow \Omega$$

- Training data consists of an input vector and a class label
- Training data should reflect the structure and the properties of the entire (possible) input data, i.e. it should generalize well!

Example: Classification of Fish



A priori probability

- $P(\omega_1), P(\omega_2) \dots$ a priori probabilities

sea bass salmon $P(\omega_1) + P(\omega_2) = 1$

(Assumption: we only look at the classes ω_1 und ω_2)

- Very basic classification rule:

$$S = \begin{cases} \omega_1 & \text{if } P(\omega_1) > P(\omega_2) \\ \omega_2 & \text{else} \end{cases}$$

Decision without
using any features!



Conditional probability

- Conditional probability

$$P(A | B)$$

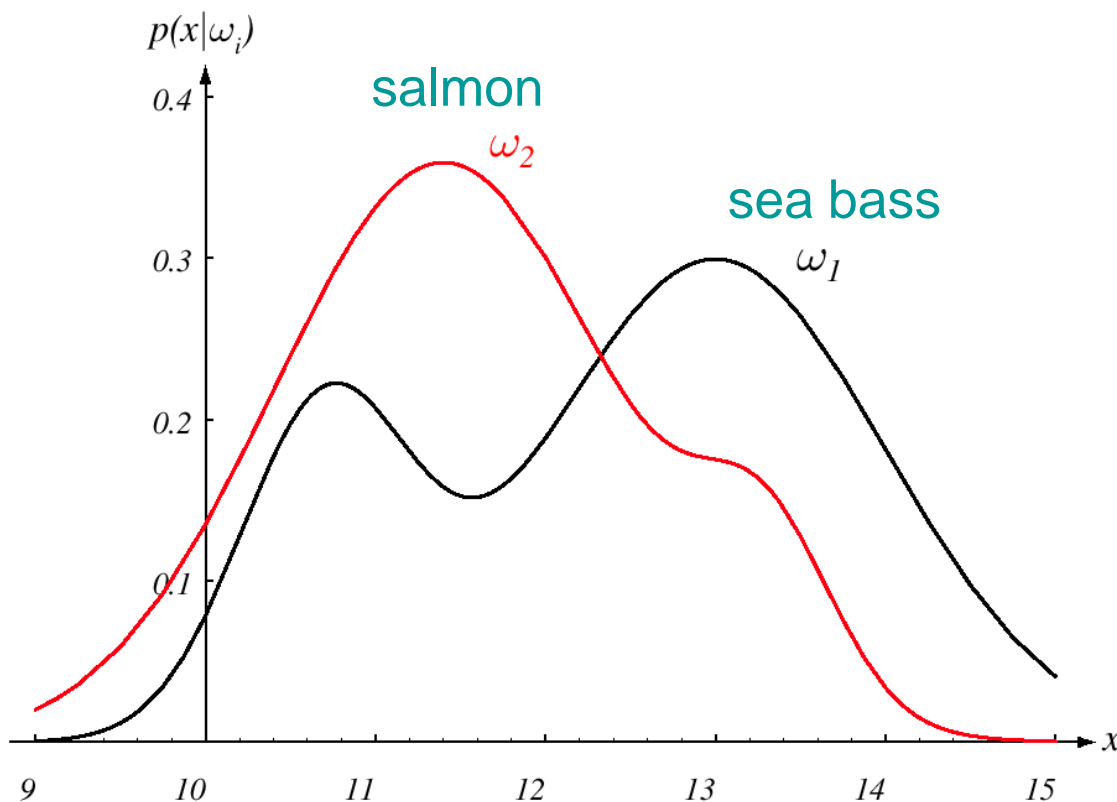
- ... is the probability of A if B is true.

$$P(A, B) = P(A | B)P(B) = P(B | A)P(A)$$

- A and B independent:

$$P(A, B) = P(A)P(B)$$

Class-conditional probability density (Likelihood)



$$\int_{-\infty}^{\infty} p(x|\omega_1) dx = 1$$

$$\int_{-\infty}^{\infty} p(x|\omega_2) dx = 1$$

Feature
(e.g. Lightness)

Posterior Probability

- Bayes Theorem:

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

$$\text{with } p(\vec{x}) = \sum_{j=1}^c p(\vec{x}|\omega_j)P(\omega_j)$$

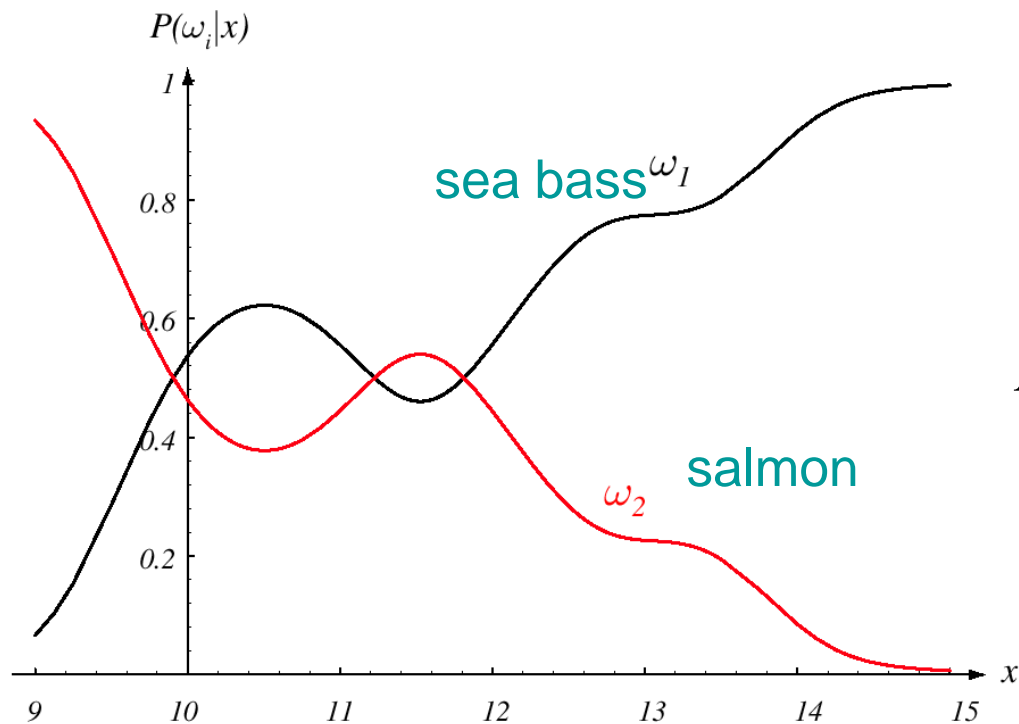
Classification:

$$S = \begin{cases} \omega_1 & \text{if } P(\omega_1|\vec{x}) > P(\omega_2|\vec{x}) \\ \omega_2 & \text{else} \end{cases}$$

Error Probability:

$$P(\text{error}|\vec{x}) = \begin{cases} P(\omega_1|\vec{x}) & \text{if } S = \omega_2 \\ P(\omega_2|\vec{x}) & \text{if } S = \omega_1 \end{cases}$$

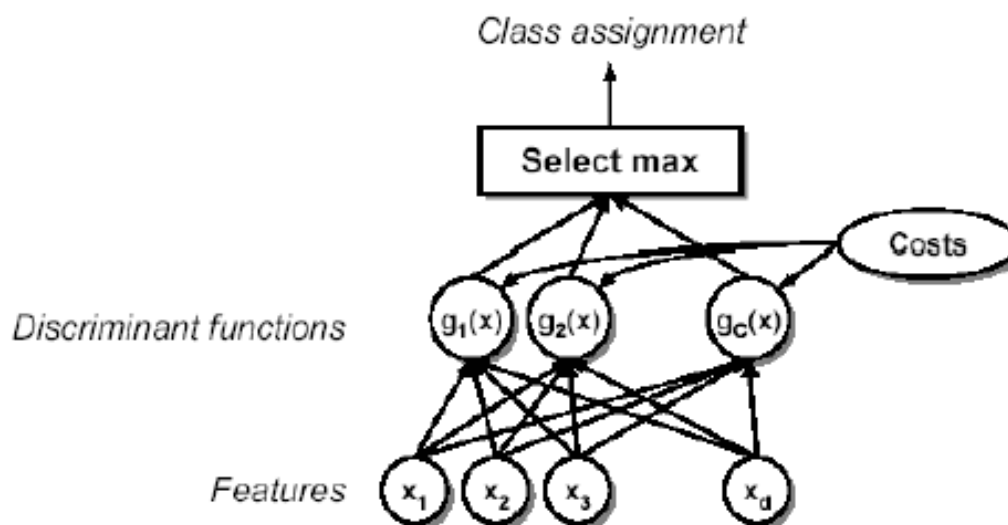
Posterior Probability



$$P(\omega_1|\vec{x}) + P(\omega_2|\vec{x}) = 1$$

Discriminant Functions

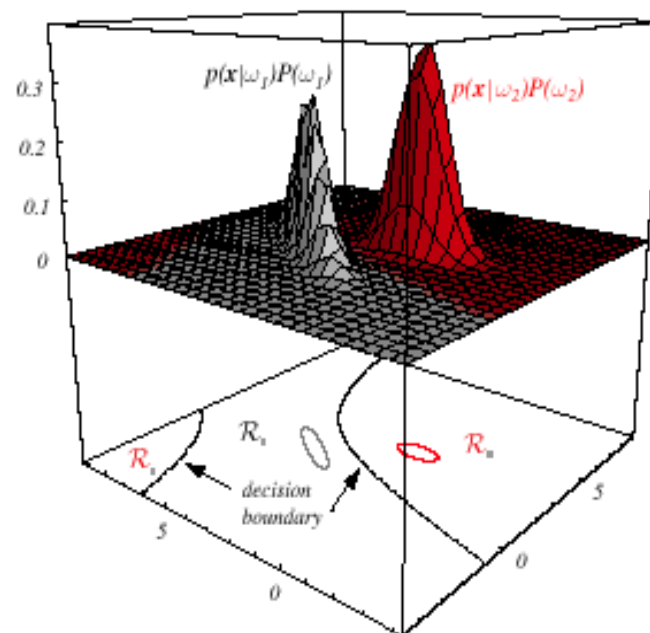
- A classifier can be seen as a set of several so-called discriminant functions $g_i(x)$
- A classifier assigns the class label ω_i to a feature vector x if $g_i(x) > g_j(x), \forall i \neq j$



Discriminant Functions

- A discriminant function is assigned to every class, and g_i decides for class ω_i ($i=1 \dots c$).
- Every discriminant function has a d -dimensional feature vector \mathbf{x} as input.
- The feature space is therefore separated in c decision regions $R_1 \dots R_c$ given by:

if $g_i(\mathbf{x}) > g_j(\mathbf{x})$ for all $j \neq i$,
then \mathbf{x} is in R_i





Bayesian Classifier

Decision function = a posteriori probability

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

Replace $g(x)$ by $f(g(x))$: e.g. $f = \ln$

$$g_i(x) = \ln(P(\omega_i | x)) = \ln(p(x | \omega_i)) + \ln(P(\omega_i)) - \ln\left(\sum_{j=1}^c p(x | \omega_j)P(\omega_j)\right)$$

Neglect constant terms

$$g_i(x) = \ln(p(x | \omega_i)) + \ln(P(\omega_i))$$



Bayesian Classifier - Discussion

- The Bayesian Classifier delivers the optimal decision.
- However, one needs to know:
 - The class conditional probability densities $p(x|\omega_i)$
 - The a priori probabilities $P(\omega_i)$
- How can we identify these?
 1. Parametric methods:
Assume that the probabilities follow a certain probabilistic model (e.g. normal distribution)
 2. Non-Parametric methods:
Parzen windows (method underlying mean shift), or k_n -nearest neighbors

Multivariate Normal Distribution

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

d Dimension of feature space

μ Mean vector

Σ $d \times d$ Covariance matrix (symmetric and pos. semidefinite)

$|\Sigma|$ Determinant of the covariance matrix

$$\vec{\mu} = E[\vec{x}] \quad \Sigma = E[(\vec{x} - \vec{\mu})(\vec{x} - \vec{\mu})^T]$$

$\Sigma = \{\sigma_{ij}\}$ $\sigma_{ij} = E[(x_i - \mu_i)(x_j - \mu_j)]$ with $\mu_i = E[x_i]$... element-wise

$\sigma_{ii} = \sigma_i$... Variance of feature i

σ_{ij} ... Covariance of feature i and j

Example: Bivariate Normal Distribution

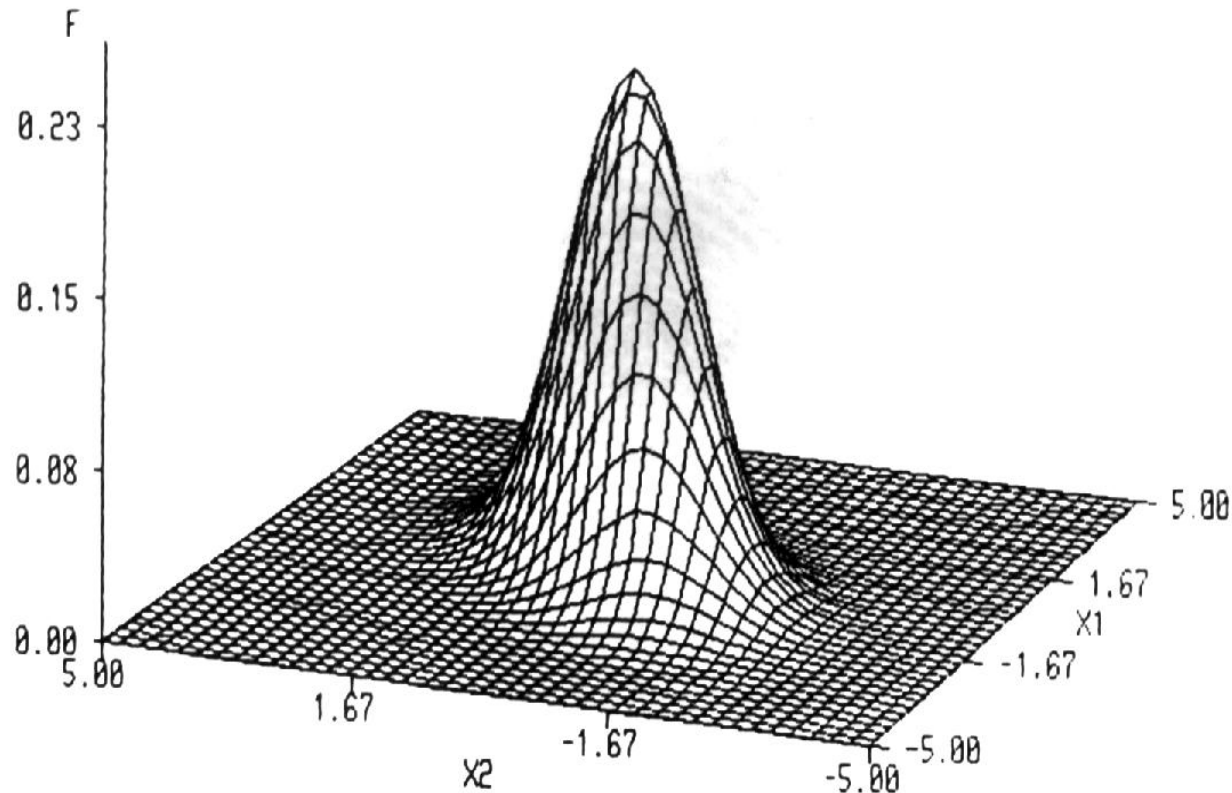


FIG. 7.3. The bivariate normal distribution.



Maximum Likelihood Classifier

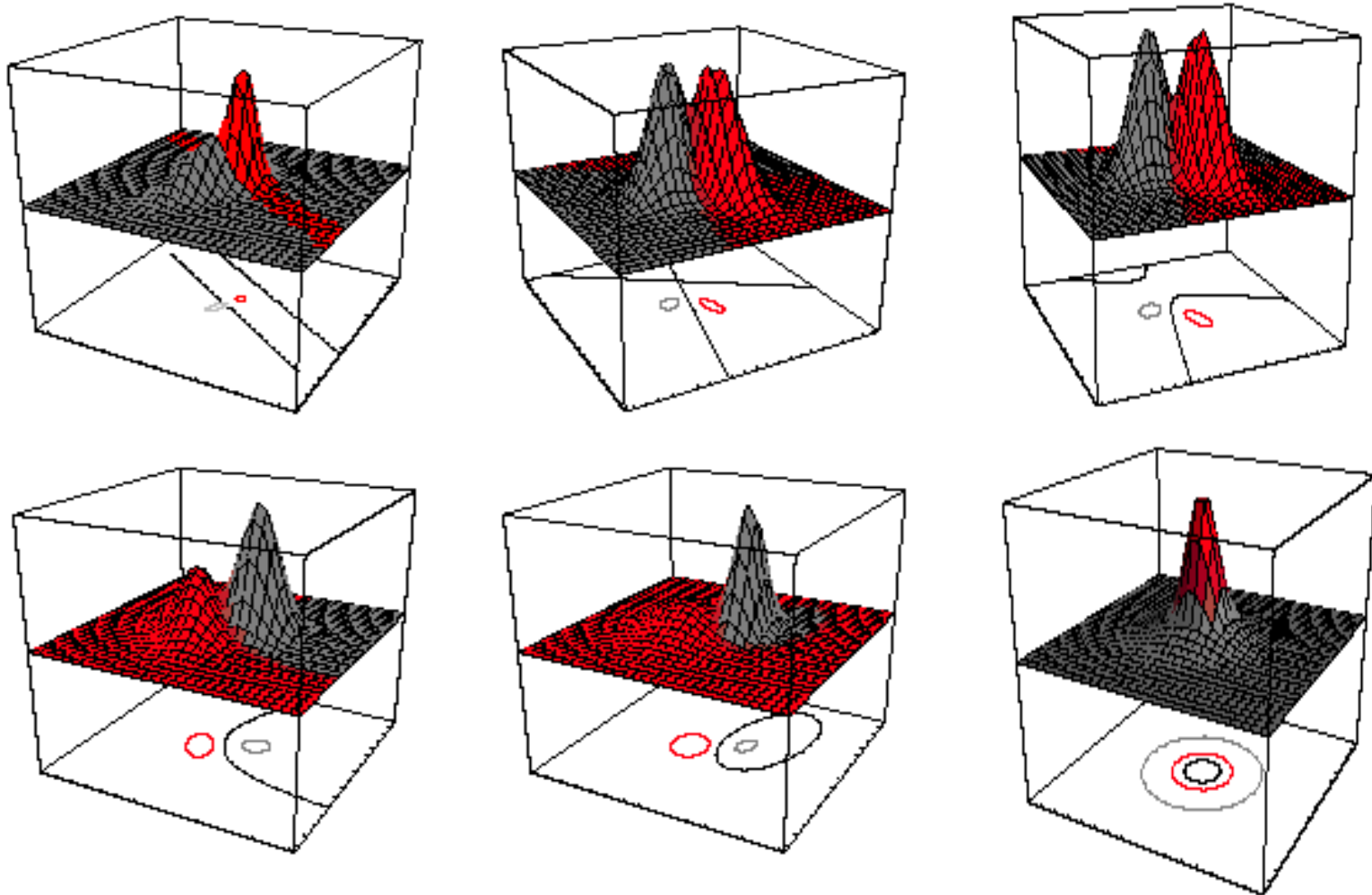
Maximum Likelihood Classifier

- Three cases can be distinguished:
 - $\Sigma = \sigma^2 I$... Minimum Distance Classifier (MD / MDs)
 - $\Sigma_i = \Sigma$... Mahalanobis Classifier (MA)
 - $\Sigma_i = \text{arbitrary}$... Hyperquadratic Classifier
- Estimate unknown parameters $\vec{\mu}_i$ and Σ_i from training data as

$$\vec{\mu}_i^* = \frac{1}{n_i} \sum_{j=1}^{n_i} \vec{x}_j$$

$$\Sigma_i^* = \frac{1}{n_i - 1} \sum_{j=1}^{n_i} (\vec{x}_j - \vec{\mu}_i^*) (\vec{x}_j - \vec{\mu}_i^*)^T$$

Maximum Likelihood Classifier

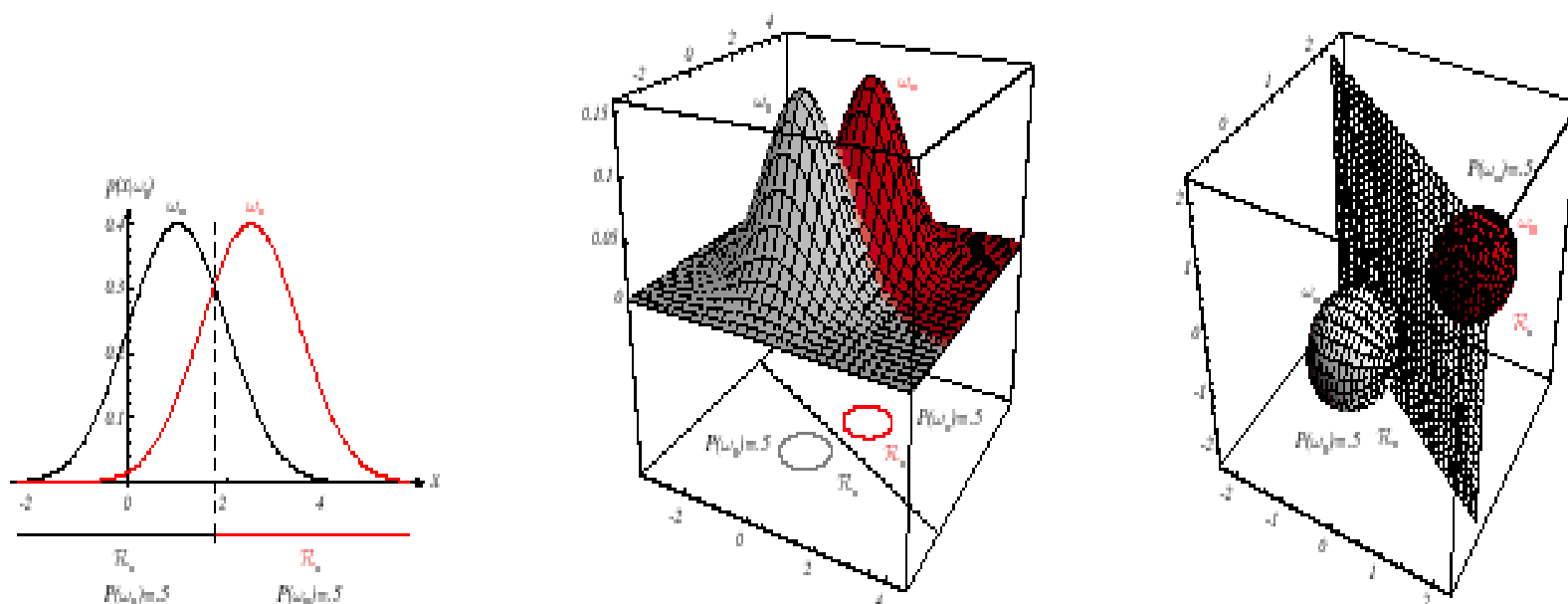




Minimum Distance Classifier (MD)

MD Classifier - Visualization

Two distributions with equal covariance proportional to the identity matrix are linearly separable!





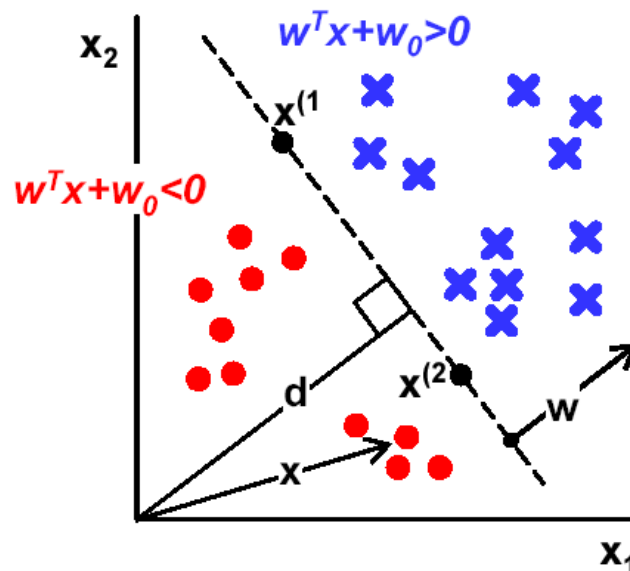
Linear discriminant functions

Linear discriminant functions

- Linear discriminant functions are of the form

$$g(x) = w^T x + w_0 \Leftrightarrow \begin{cases} g(x) > 0 & x \in \omega_1 \\ g(x) < 0 & x \in \omega_2 \end{cases}$$

- We therefore compute a hyperplane which separates the data
- w is a weight vector, and w_0 a threshold weight (bias)



$$d = \frac{w_0}{\|w\|}$$



Minimum Squared Error Solution

Comparison of the 3 Cases

- Minimum Distance simple Decision Function:

$$g_{MDs_i} = -(\vec{x} - \vec{\mu}_i)^t (\vec{x} - \vec{\mu}_i)$$

- Mahalanobis Distance Decision Function:

$$g_{MA_i} = -\frac{1}{2} (\vec{x} - \vec{\mu}_i)^t \Sigma^{-1} (\vec{x} - \vec{\mu}_i) + \ln P(\omega_i)$$

- Maximum Likelihood Decision Function:

$$g_{ML_i} = -\frac{1}{2} \ln |\Sigma_i| - \frac{1}{2} (\vec{x} - \vec{\mu}_i)^t \Sigma_i^{-1} (\vec{x} - \vec{\mu}_i) + \ln P(\omega_i)$$



Non-Parametric Classification

- Problems with the Bayesian approach:
 - Computation is complication for non-normally distributed data
 - A priori probability is hard to estimate
 - Once the model is setup, changes are hard to integrate
- Non-Parametric Methods
 - Parzen Windowing (underlying mean shift)
 - K Nearest Neighbors

k_n – Nearest Neighbors Classifier

- In contrast to parzen windowing, the kernel structure and size is estimated by evaluating the training data

- Idea: Enlarge the kernel until k_n training samples are covered

$$P^*(\omega_i | \vec{x}) = \frac{k_i}{k}$$

k_i ... Number of training samples of class i within the k nearest neighbors of data sample \mathbf{x}

- Decision function of the classifier:

$$g_{kNN_i} = k_i$$

- In other words, the Nearest – Neighbor – Rule is:

The class label which is most frequent among the k_n nearest neighbors of data point \mathbf{x} is assigned.



kNN - Discussion

- The kNN Classifier delivers in general worse results than Bayesian decision theory.
- However, if n is large the error is very small.
- Training is very fast: save all prototypes
- Evaluation is computationally intensive:
 - Comparison to all prototypes!
 - Workaround: Just store values close to the decision boundary
- Approximate nearest neighbor classifiers exist (e.g. FLANN in OpenCV)

kNN - Example

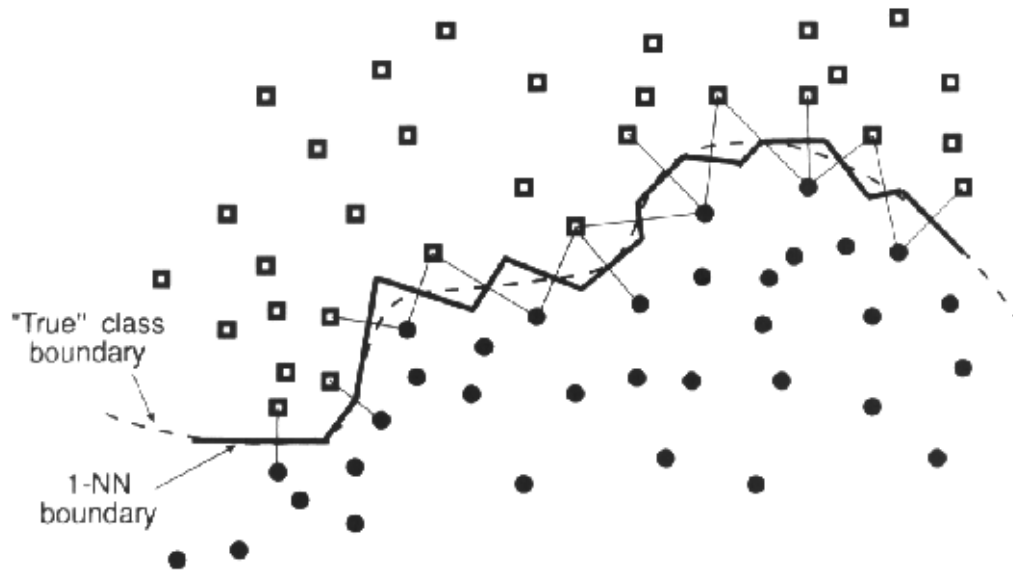


FIG. 8.1. 1 - NN classification.

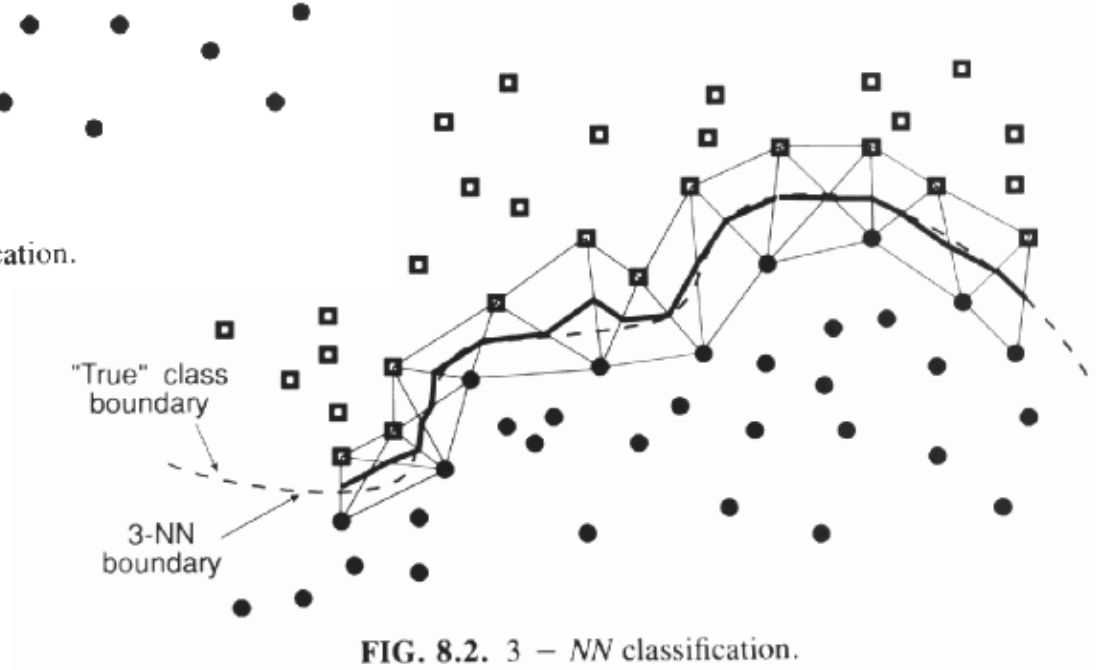
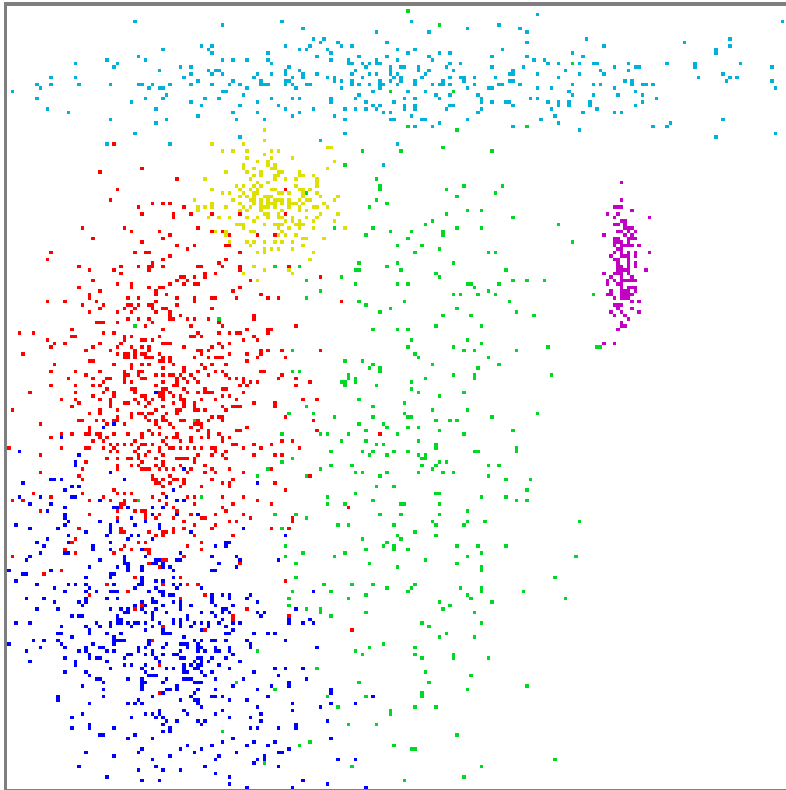
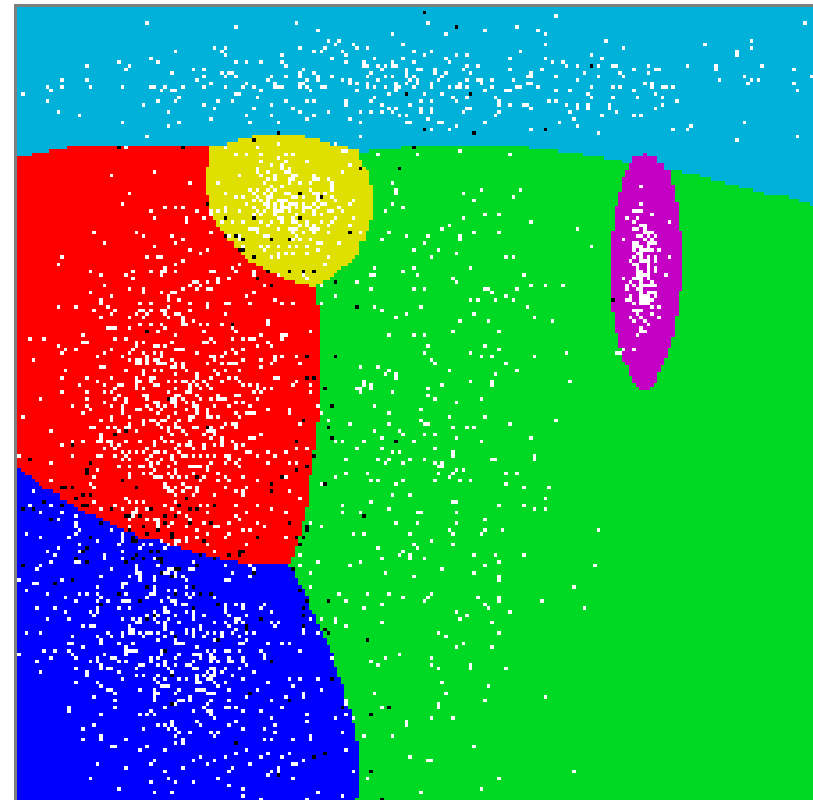


FIG. 8.2. 3 - NN classification.

Comparison of Classifiers

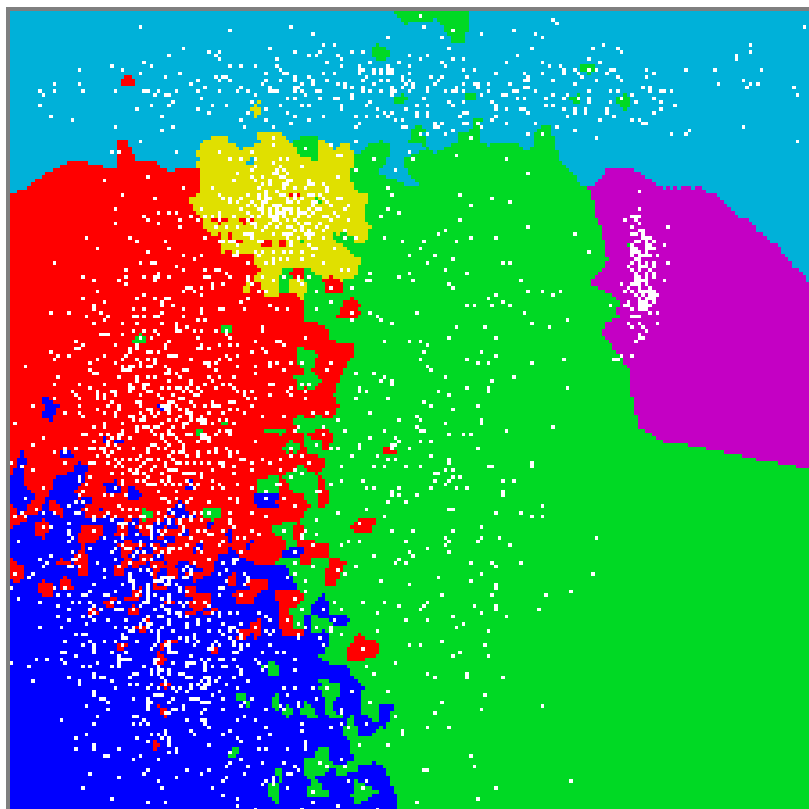


(a) Lerndaten

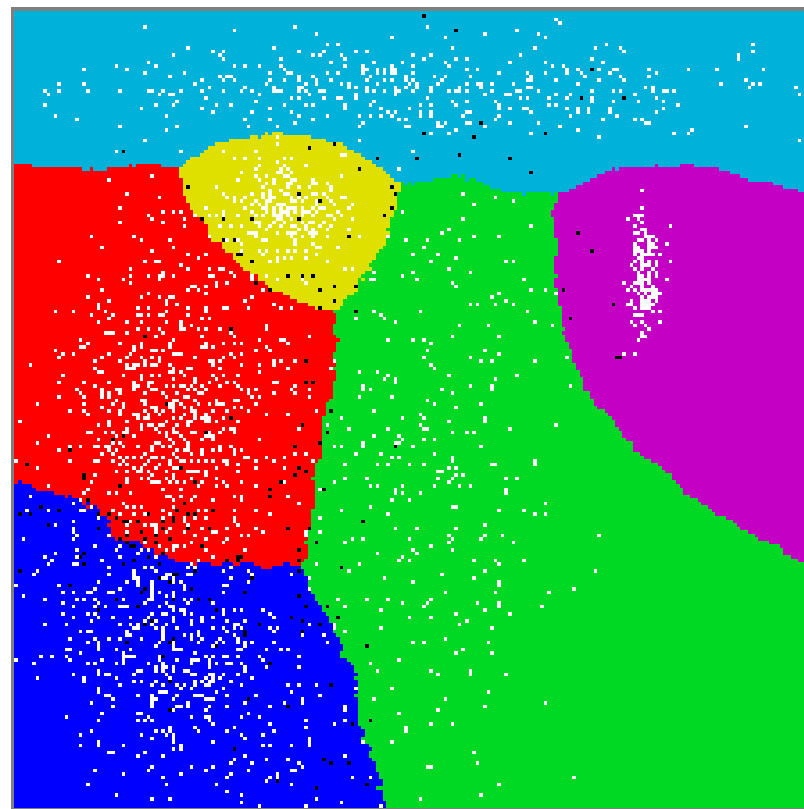


(b) ML Klassifikator

Comparison of Classifiers



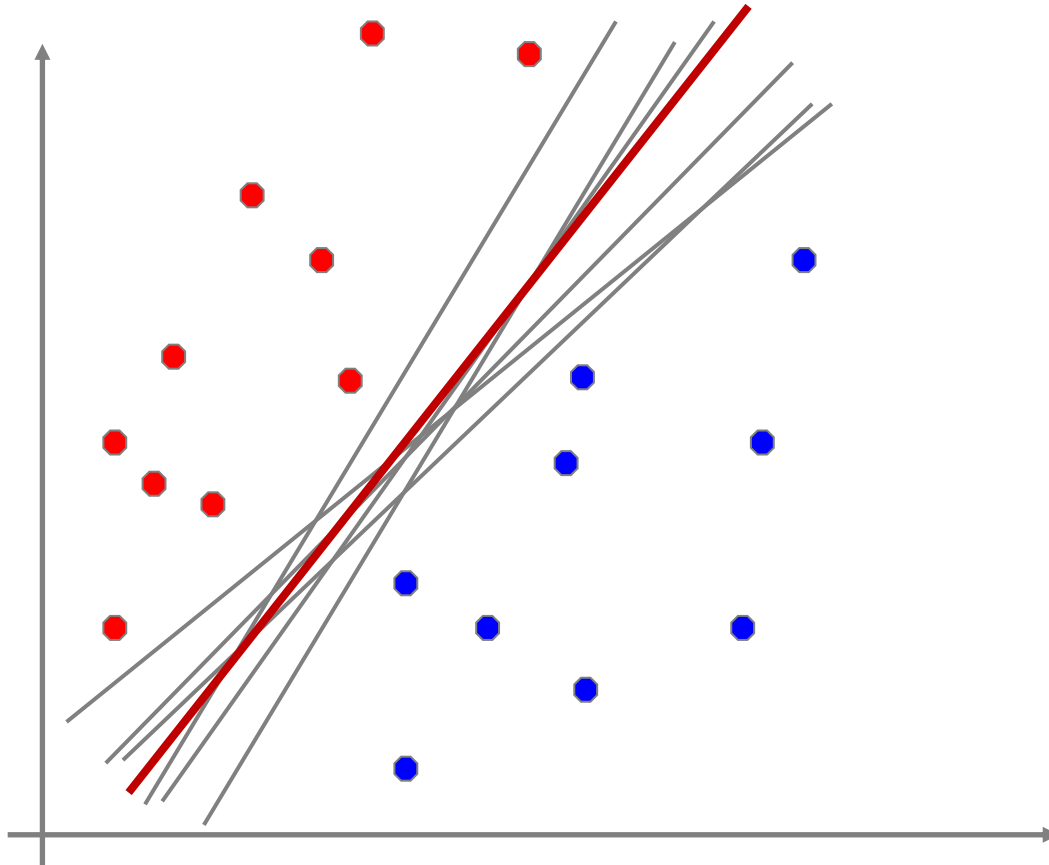
(a) k NN Klassifikator mit $k=1$



(b) k NN Klassifikator mit $k=70$

Support Vector Machine (SVM)

- Which decision boundary is optimal?





Support Vector Machine (SVM)

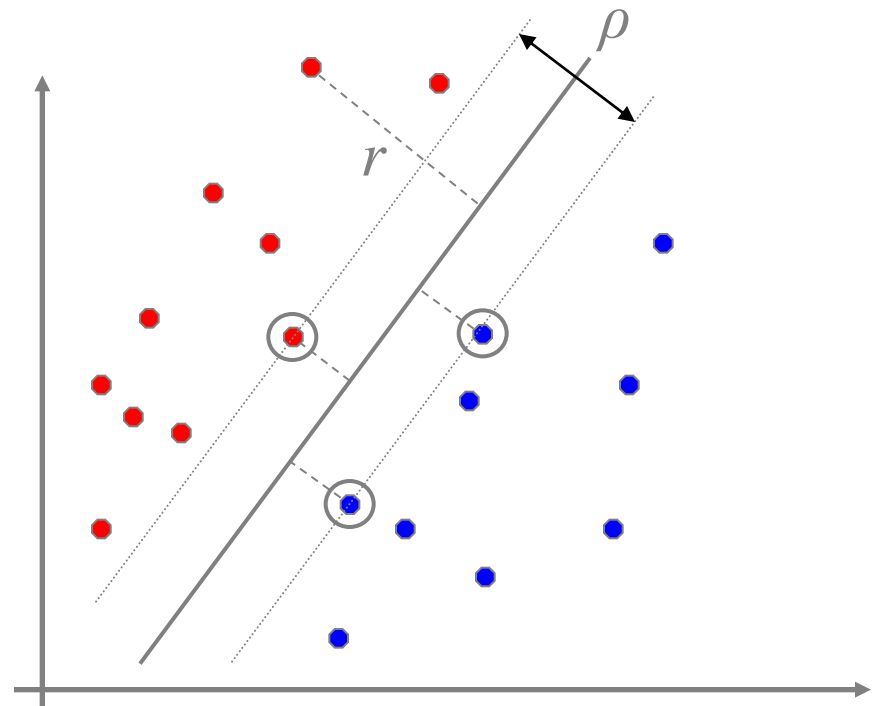
- The support vector machine is a linear separator.
- For the optimal solution of the linearly separable, binary pattern classification problem:
 - *The margin of separation between the two classes is the largest possible.*
 - *The data points drawn from the two classes, that are most difficult to separate from each other, define the support vectors.*
- In other words, the decision boundary is defined by the support vectors.

Classification Margin

- Distance between a feature vector and the decision boundary:

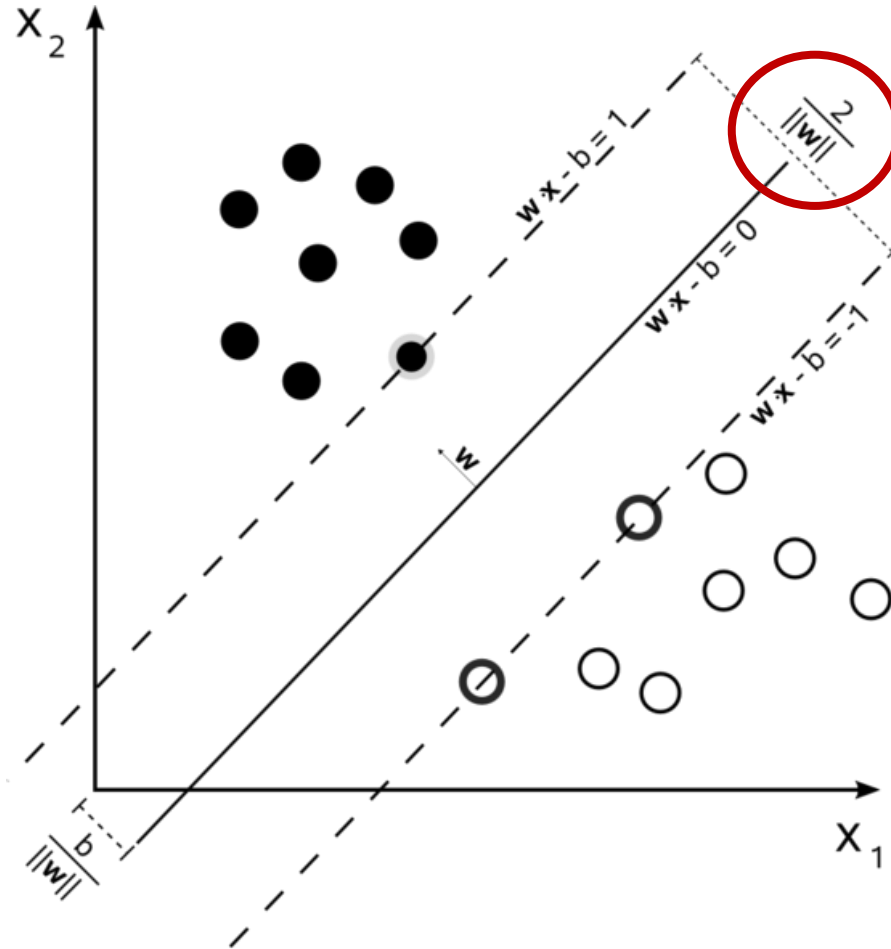
$$r = \frac{\mathbf{w}^T \mathbf{x} + b}{\|\mathbf{w}\|}$$

- Margin** ρ of the decision boundary is the distance between the classes



Maximum Margin Linear Classifier

- Decision boundary is set to maximize the margin



$$y = \mathbf{w}^T \mathbf{x} + b$$



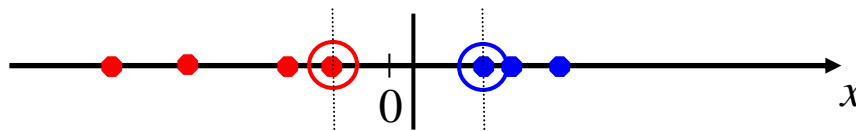
Maximum Margin Linear Classifier



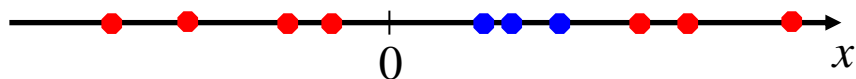
Nonseparable problems: Soft margin

Non-linear SVMs

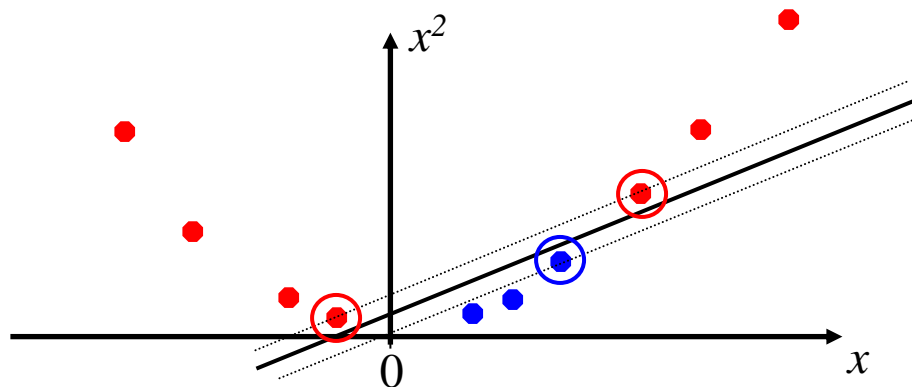
- SVM's work very well on linearly separable data:



- But what can we do if the data is “too difficult” for SVM's?



- Idea:** project into a higher dimensional space!



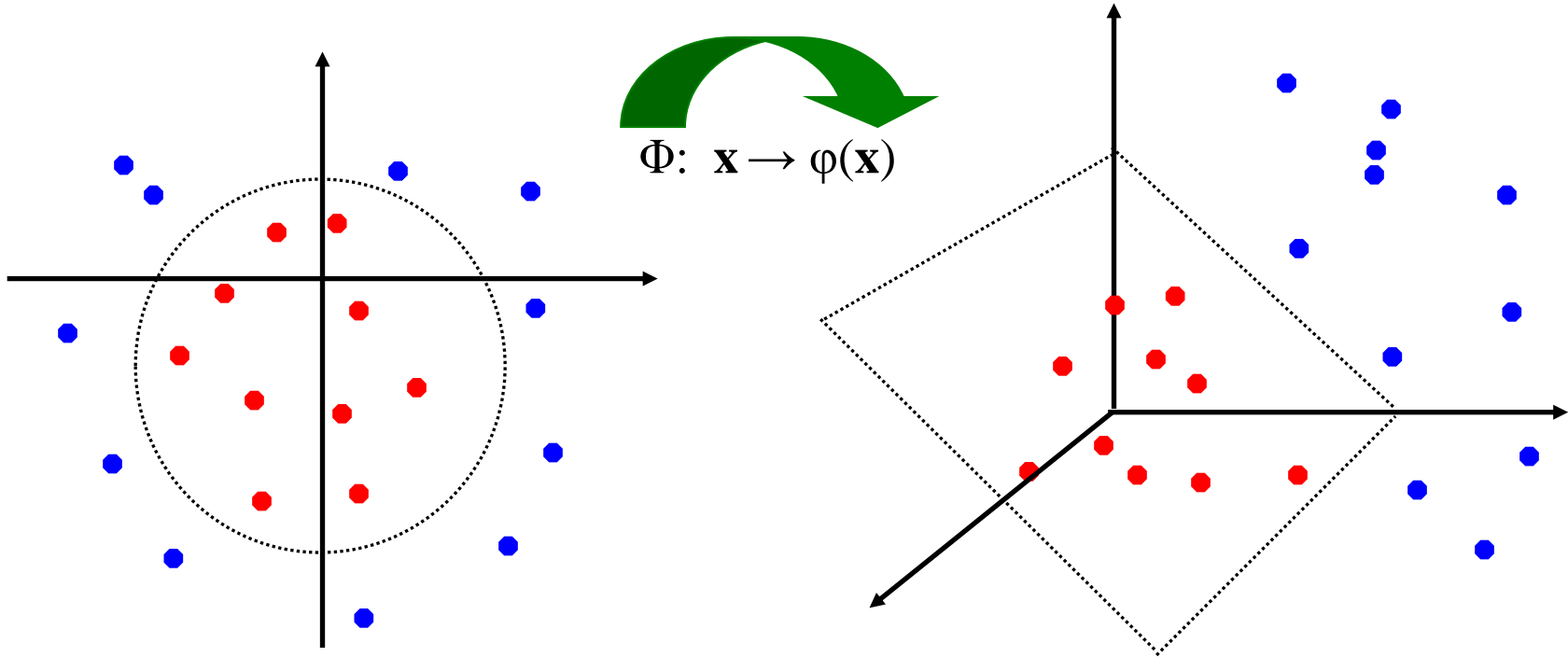


Cover's Theorem

Given a nonlinearly separable pattern classification problem, it can be transformed into a linearly separable classification problem provided two conditions are satisfied:

1. The transformation from the input data space into a hidden (feature) space **must** be nonlinear.
2. The dimensionality of the hidden space **should** be high compared to the input space.

Non-linear SVMs: Feature spaces



The Kernel Trick

- Original feature space

$$g(\mathbf{x}) = \sum_i a_i y_i \mathbf{s}_i \cdot \mathbf{x} + w_0$$

- High-dimensional feature space

$$g(\mathbf{x}) = \sum_i a_i y_i \phi(\mathbf{s}_i) \cdot \phi(\mathbf{x}) + w_0 = \sum_i a_i y_i K(\mathbf{s}_i, \mathbf{x}) + w_0$$

- **Inner products can be written as kernel operations!**

The Kernel Trick - Example

Gegeben zwei Expressionsprofile $p = (g_1, g_2)$ und $q = (h_1, h_2)$.

Wir wenden auf p und q die Abbildung $\Phi : (g_1, g_2) \mapsto (g_1^2, \sqrt{2}g_1g_2, g_2^2)$ an und dann berechnen wir das Skalarprodukt.

$$\begin{aligned}\langle \Phi(p), \Phi(q) \rangle &= (g_1^2, \sqrt{2}g_1g_2, g_2^2)(h_1^2, \sqrt{2}h_1h_2, h_2^2)^t \\ &= g_1^2h_1^2 + 2g_1h_1g_2h_2 + g_2^2h_2^2 \\ &= (g_1h_1 + g_2h_2)^2 \\ &= \langle p, q \rangle^2 =: \mathcal{K}(p, q)\end{aligned}$$

Wir können also das Skalarprodukt zwischen $\Phi(p)$ und $\Phi(q)$ ausrechnen, ohne die Funktion Φ anzuwenden. Es reicht, das Quadrat des Skalarproduktes von p und q im \mathbb{R}^2 zu berechnen.

[Credit goes to Florian Markowetz, MPI Berlin]

Examples of Kernel Functions

- Linear: $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$
- Polynomial of power p : $K(\mathbf{x}_i, \mathbf{x}_j) = (1 + \mathbf{x}_i^T \mathbf{x}_j)^p$
- Gaussian (radial-basis function network):

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}}$$

... projects into a Hilbert space with infinite dimensionality

- Two-layer perceptron: $K(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\beta_0 \mathbf{x}_i^T \mathbf{x}_j + \beta_1)$

SVM Classification Example

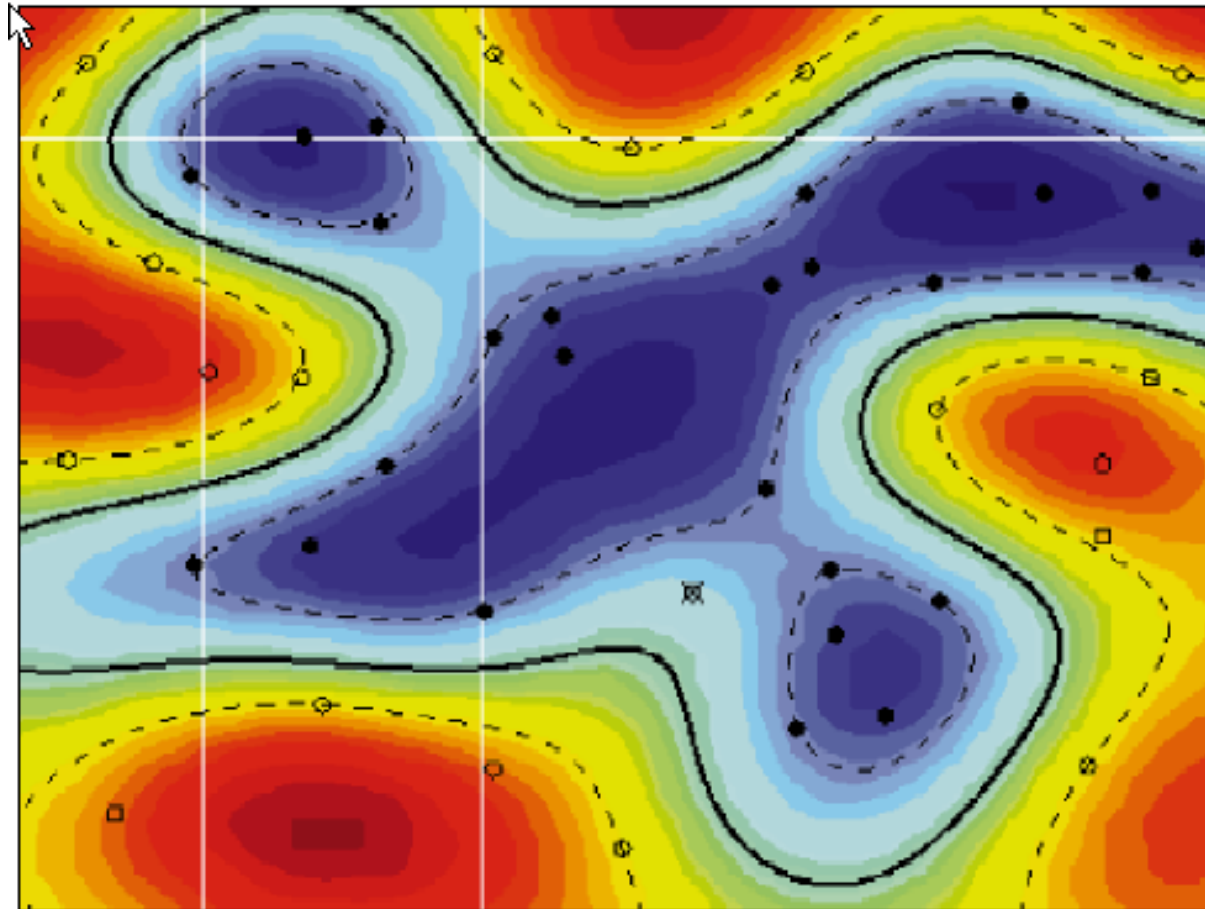
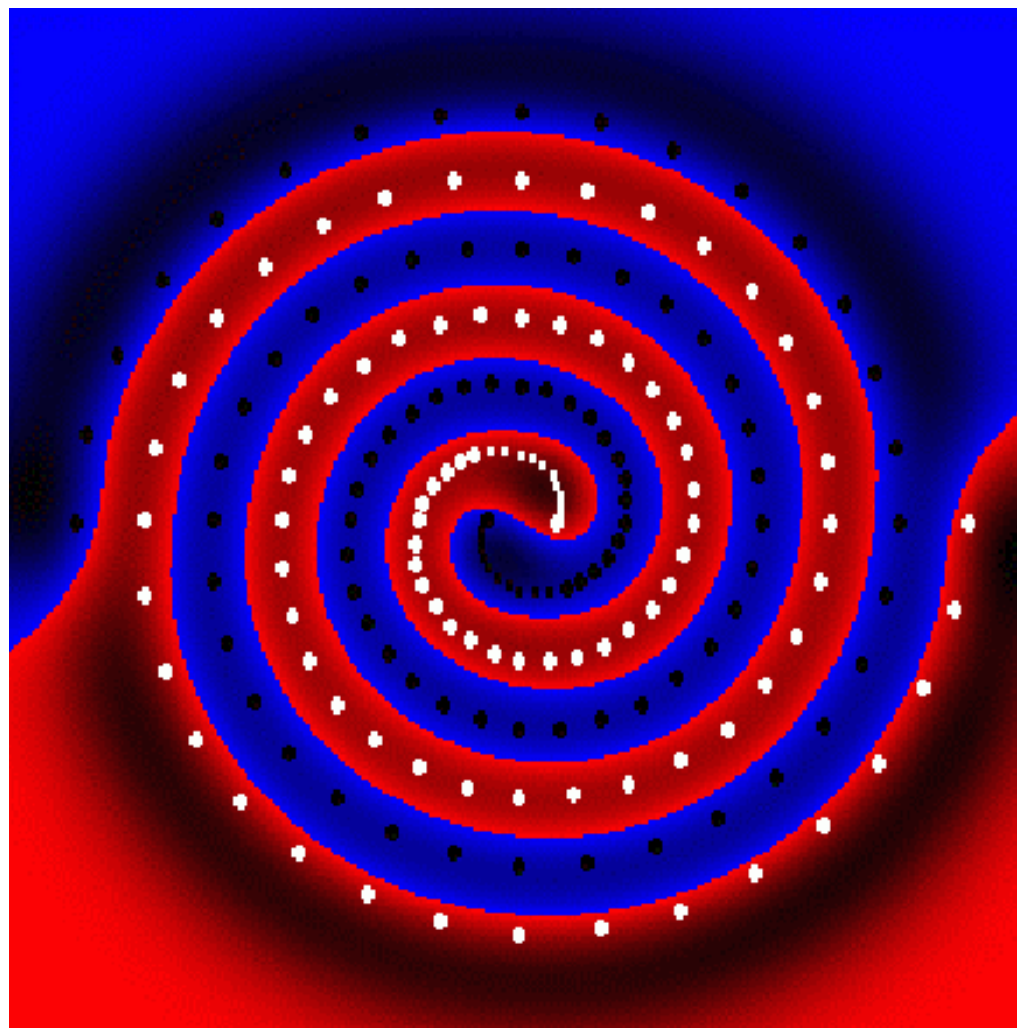


Figure 3. Example of an SV classifier found by using a radial basis function kernel (Equation 8). Circles and disks are two classes of training examples; the solid line is the decision surface; the support vectors found by the algorithm lie on, or between, the dashed lines. Colors code the modulus of the argument $\sum_i v_i \cdot k(\mathbf{x}, \mathbf{x}_i) + b$ of the decision function in Equation 10.

SVM Classification Example



Gaussian Kernel

SVM for N Classes

2 Strategies:

- **1 versus All:**

- Learn N SVM's, each having one class as positive training samples and all other classes as negative training samples

- **All versus All:**

- Learn $N(N-1)/2$ SVM's. Each SVM decides only between two classes.
- Majority voting using the pairwise results to find the most suitable class.

Structured SVMs / Structured Learning

”Normal” Machine Learning:

$$f : \mathcal{X} \rightarrow \mathbb{R}.$$

- ▶ inputs \mathcal{X} can be any kind of objects
 - ▶ images, text, audio, sequence of amino acids, ...
- ▶ output y is a real number
 - ▶ classification, regression, density estimation, ...

Structured Output Learning:

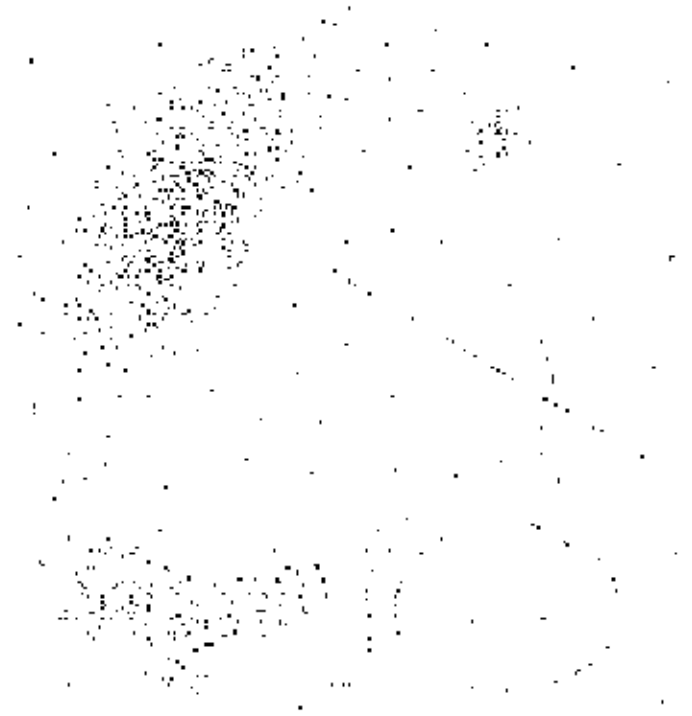
$$f : \mathcal{X} \rightarrow \mathcal{Y}.$$

- ▶ inputs \mathcal{X} can be any kind of objects
- ▶ outputs $y \in \mathcal{Y}$ are complex (structured) objects
 - ▶ images, parse trees, folds of a protein, ...

[Nowozin and Lampert, 2011]

Unsupervised Classification

- Important difference: Unsupervised classification does not make use of class labels during training!
- The goal is to develop algorithms which try to find structures (clusters) in the data





Why use Unsupervised Classification?

- Labelling data, especially correctly labelling data is a hard and tedious task (some researchers use Amazon mechanical turk)
- Unsupervised learning allows using much larger amounts of training data
- Can quickly adapt to changes in feature space
- Automatic detection of discriminative features



Clustering Methods

- Clustering is a method of statistical data analysis
- In general, it requires the definition of a distance measure between feature points
- 3 different classes:
 - Partitional Clustering
 - All clusters are estimated at the same time
 - Hierarchical Clustering
 - Creation of a hierarchy, bottom-up or top-down
 - Graph-theoretic Methods

Typical Methods

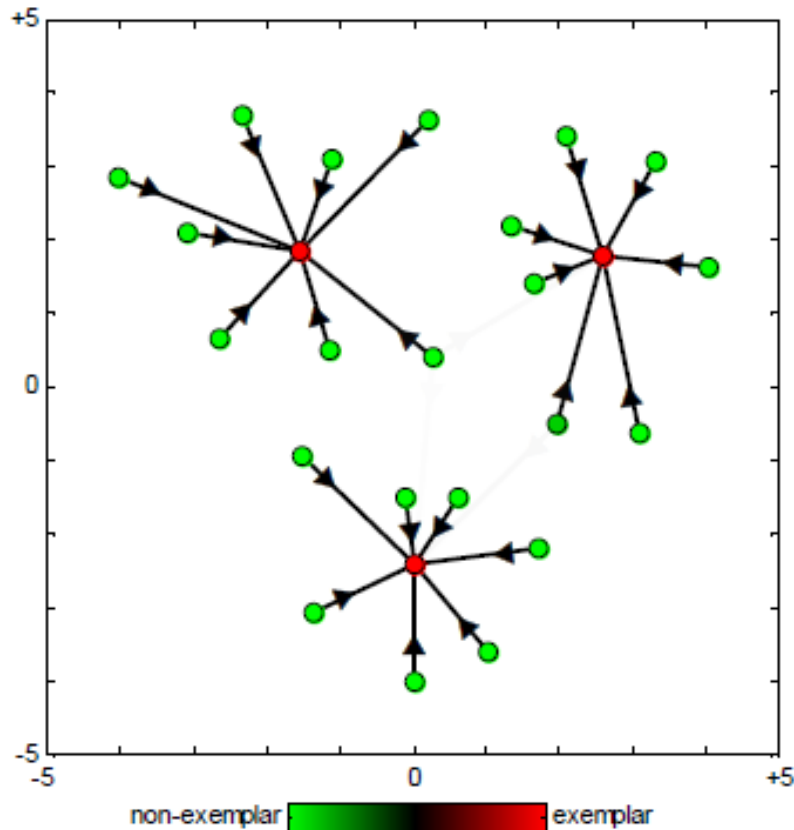
- Mixture Densities – Maximum Likelihood Estimation
Known distribution, known number of clusters, estimate best parameters (EM-algorithm)
- Simplification of the posterior: k-means algorithm

$$P(\omega_i | x_k, \hat{\mu}) = \begin{cases} 1 & \dots & x_k \text{ nearest to } \mu_i \\ 0 & \dots & \text{else} \end{cases}$$

- Also: k-medians algorithm
Cluster centers correspond to a trained data point
- Mean-Shift (as discussed before)

Advanced Methods

- Affinity Propagation



All data points are simultaneously considered as exemplars, but exchange deterministic messages until a good set of exemplars gradually emerges