Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

MIN-Fakultät
Fachbereich Informatik
Arbeitsbereich SAV/BV (KOGS)

Image Processing 1 (IP1)
Bildverarbeitung 1

Lecture 15 – Pattern Recognition

Winter Semester 2015/16

Slides: Prof. Bernd Neumann
Slightly revised by: Dr. Benjamin Seppke & Prof. Siegfried Stiehl

What is "Pattern Recognition"?

The term "Pattern Recognition" ("Mustererkennung") is used for

Methods for classifying unknown objects based on feature vectors
(narrow sense meaning of Pattern Recognition)

Methods or analyzing signals and recognizing interesting patterns
(wide sense meaning of Pattern Recognition)

Pattern recognition can be applied to all kinds of signals, e.g.

- images
- acoustic signals
- seismographic signals
- tomographic data etc.

The following section deals with Pattern Recognition in the narrow sense.

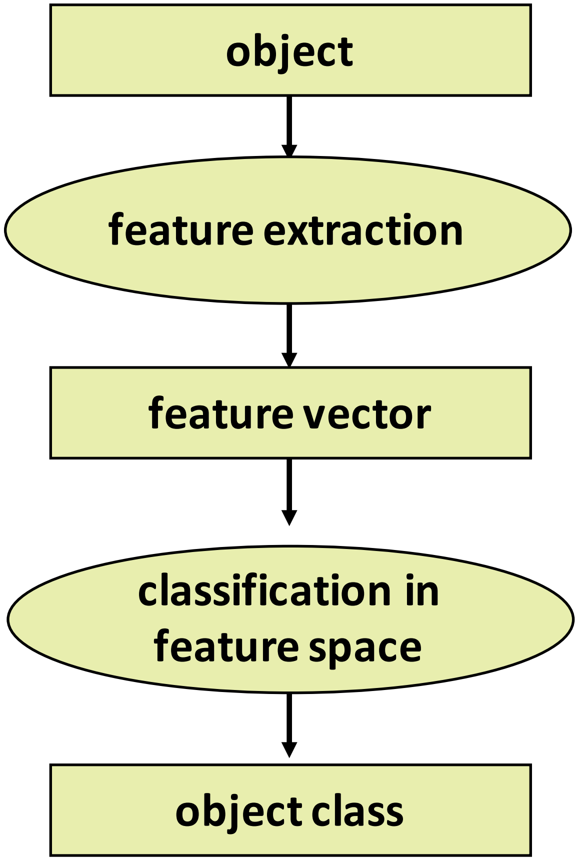
(see Duda and Hart, Pattern Classification and Scene Analysis, Wiley 73)

Introductory Example: Where is Wally?

(c) Martin Handford, Walker Books

(c) Martin Handford, Walker Books

Basic Terminology for Pattern Recognition



K classes $\omega_1 \ldots \omega_K$

N dimension of feature space

$\vec{x}^T = (x_1 \; x_2 \; \ldots \; x_N)$ feature vector

$\vec{y}^T = (y_1 \; y_2 \; \ldots \; y_N)$ prototype (feature vector with known class membership)

$\vec{y}_i^{(k)}$ i-th prototyp of class k

M_k number of prototypes for class k

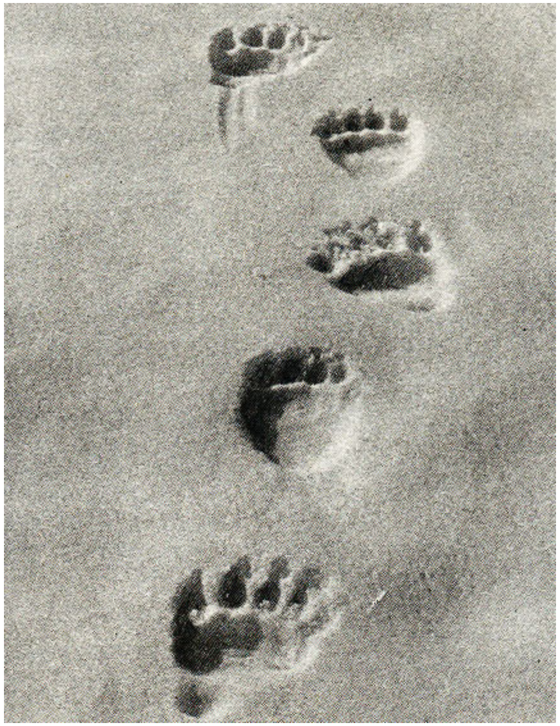
$g_k(\vec{x})$ discriminant function for class k

Problem: Determine $g_k(\vec{x})$ such that

$$\underset{\vec{x}\in\omega_k}{\forall} \; \underset{k\neq j}{\forall} \; g_k(\vec{x}) > g_j(\vec{x})$$

Example: Animal Footprints

Bear

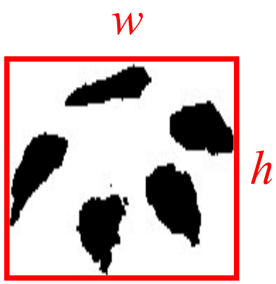


Hare

Wolf

What features can be used to distinguish the 3 footprint classes?

A Feature Space for Footprints



ω_1 = **wolf**

ω_2 = **bear**

ω_3 = **hare**

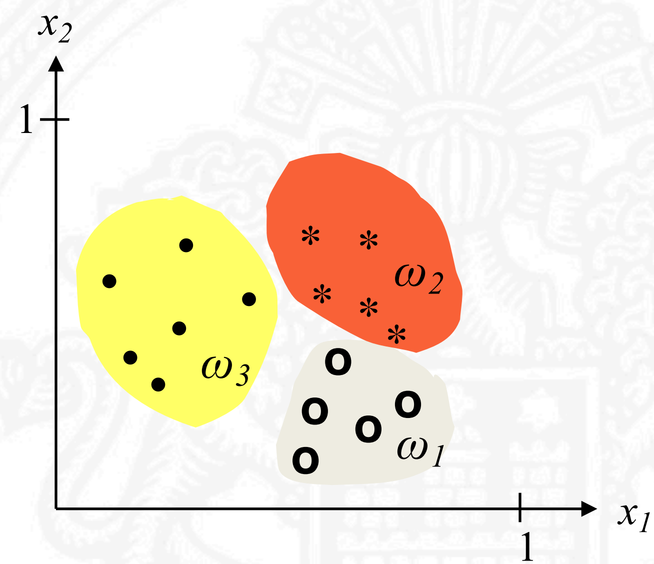
c = „circumference" = $2(w+h)$

a = „area" = $w \times h$

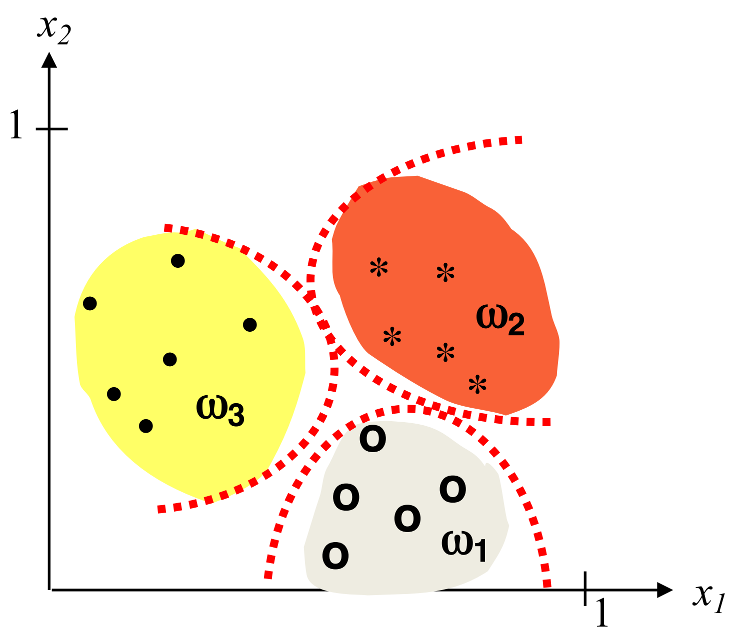
pa = „print area" = $card(\{\, g(x,y) > 0 \})$

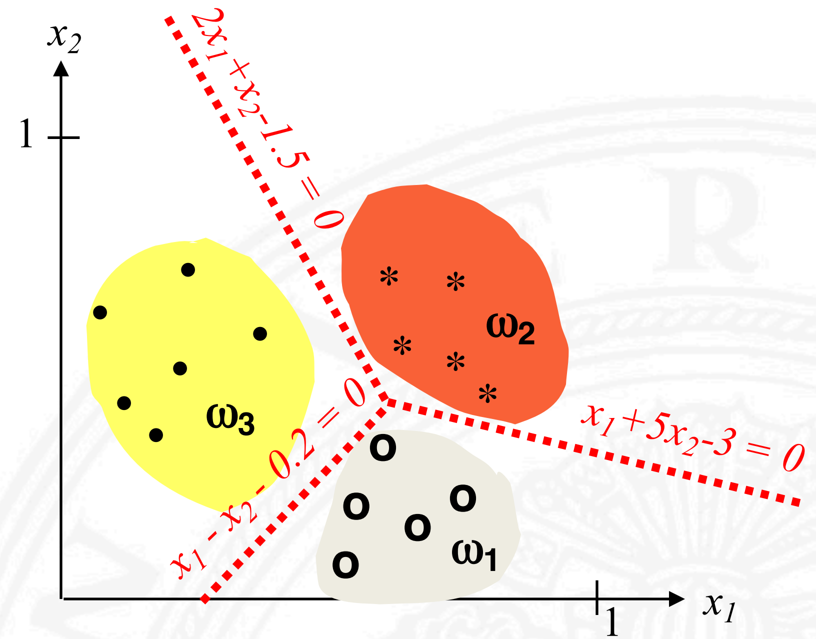
x_1 = "squareness" = $\dfrac{a}{c^2}$

x_2 = "solidness" = $\dfrac{pa}{c}$



Discriminant Functions for Footprints





Quadratic discriminant functions:

$g_1 = -9x_1^2 + 10.8x_1 - x_2 - 2.84$

$g_2 = x_1 + 20x_2^2 - 28x_2 + 9.4$

$g_3 = -x_1 + 5.6x_2^2 - 5.6x_2 - 1$

Piecewise linear discriminant functions:

$g_1 = 1$ if $(x_1-x_2-0.2 > 0) \wedge (x_1+5x_2-3 < 0)$ else 0

$g_2 = 1$ if $(x_1+5x_2-3 > 0) \wedge (2x_1+x_2-1.5 > 0)$ else 0

$g_3 = 1$ if $(2x_1+x_2-1.5 < 0) \wedge (x_1-x_2-0.2 < 0)$ else 0

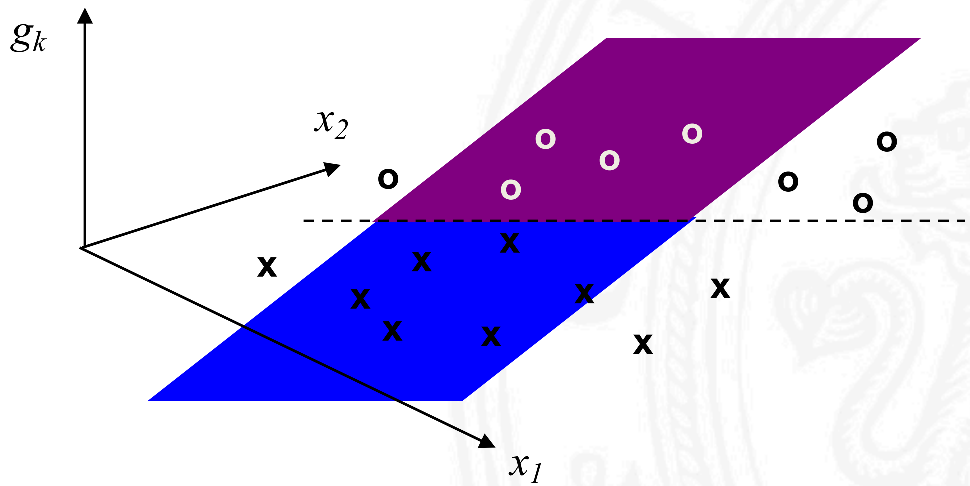
Linear Discriminant Functions

Linear discriminant functions are attractive because they can be

- **easily determined from prototypes**
- **easily analyzed**
- **easily evaluated**

Basic form of linear discriminant function:

$$g_k(\vec{x}) = (\vec{w}_k)^T \vec{x} + w_{k_0}$$



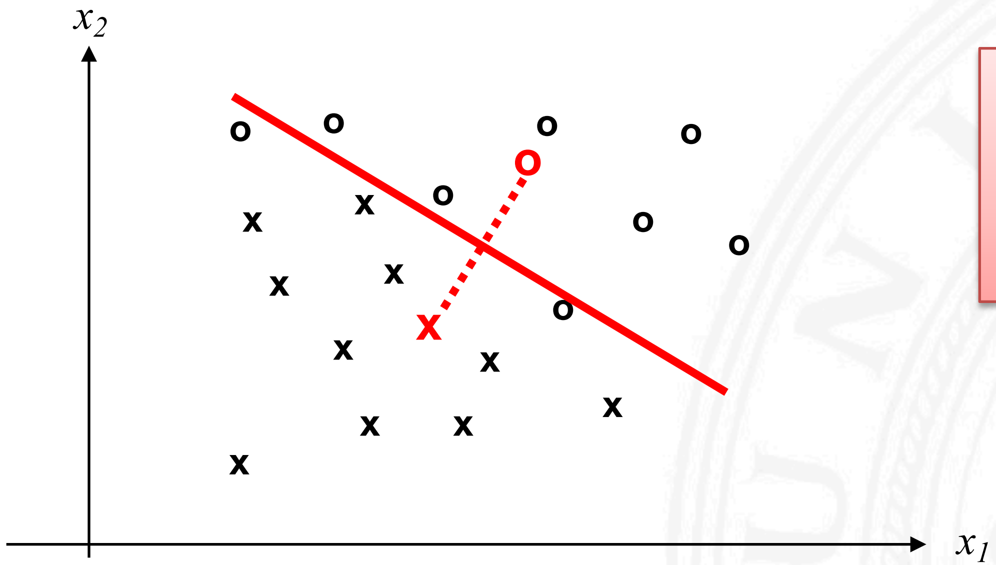
For $N=2$ the discriminant function is a 3D plane

boundary line:

$$g_k(\vec{x}) = (\vec{w}_k)^T \vec{x} + w_{k_0} = 0$$

Class Average
Minimal Distance Classification

- Represent prototypes by class averages
- Assign object to class with minimum distance between object and class average

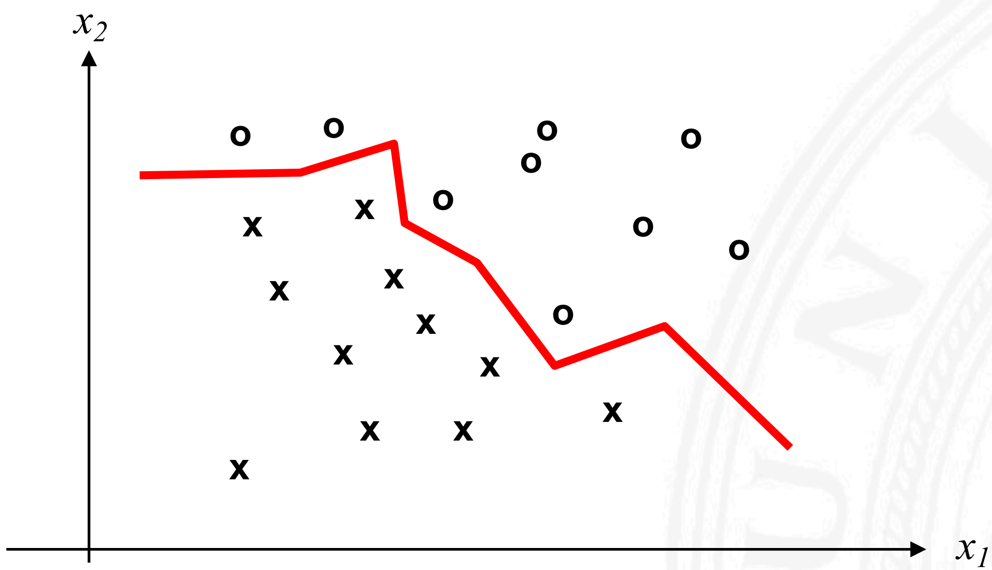


For a 2-class problem, the minimal distance criterion always results in a linear discriminant function!

Class average minimal distance classification may not separate prototypes even if they are linearly separable!

Nearest Neighbour Classification

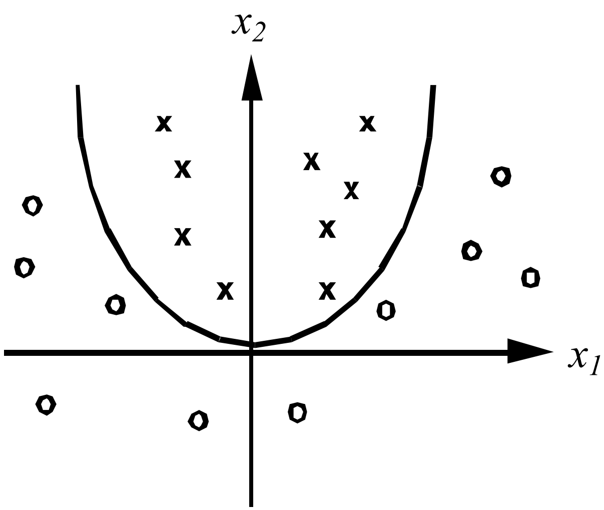
Assign object to class with nearest prototype



Piece-wise linear discriminant function

The nearest neighbour criterion classifies all prototypes correctly (except equal prototypes of different classes). The decision regions are not necessarily coherent.

Generalized Linear Discriminant Functions



Example:

Prototypes are not linearly separable

A quadratic discriminant function may work:

$$g_k(\vec{x}) = a_1 x_1 + a_1 x_1 + b_{11}(x_1)^2 + b_{22}(x_2)^2 + b_{12} x_1 x_2 + c$$

with $\vec{x}^T = (x_1 \;\; x_2)$

Transformation of prototypes into higher-dimensional feature space may allow linear discriminant functions.

Transformation for the example: $z_1 = x_1 \;,\; z_2 = x_2 \;,\; z_3 = (x_1)^2 \;,\; z_4 = (x_2)^2 \;,\; z_5 = x_1 x_2$

Linear discriminant function in z-space: $g_k(\vec{z}) = a_1 z_1 + a_2 z_2 + a_3 z_3 + a_4 z_4 + a_5 z_5 + c$

Advantage: Linear separation algorithms may be applied

Disadvantage: Dimensionality of feature space is drastically increased

Linear Discriminant Functions for 2-Class Problems

Normalize prototypes such that

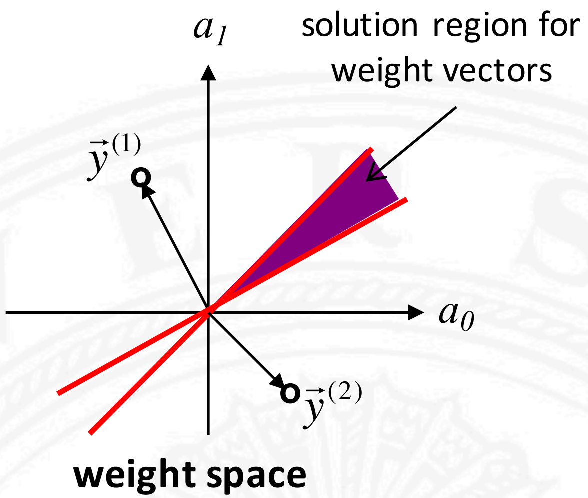
$$\vec{y}^T = \left(1 \;\; y_1 \;\; y_2 \;\; \ldots \;\; y_N\right)$$

Discriminant function g can be expressed as

$$g(\vec{x}) = \vec{a}^T \vec{x} \quad \text{with} \quad \vec{a}^T = \left(a_0 \;\; a_1 \;\; \ldots \;\; a_N\right)$$

Prototypes of class ω_2 are negated such that

$\vec{a}^T \vec{y} > 0$ → correct classification of both classes



Solution region in weight space (if it exists) is the space at the positive side of all hyperplanes $\vec{a}^T \vec{y} = 0$. Any weight vector $\vec{a}$ in this solution region gives a correct discriminant function.

Possible further constraints on solution vector a:

$$\|\vec{a}\| = 1 \;\; \wedge \;\; \forall_{\vec{y}} \, \vec{a}^T \vec{y} > b$$

b is "margin", i.e. minimal distance of a correctly classified point from the hyperplanes defined by the prototypes.

Perceptron Learning Rule

A solution vector a can be determined iteratively by minimizing a criterion function $J(\vec{a})$ by gradient descent.

Perceptron criterion function:

$$J_p(\vec{a}) = \sum_{y \in B} \left(-\vec{a}^T \vec{y}\right)$$

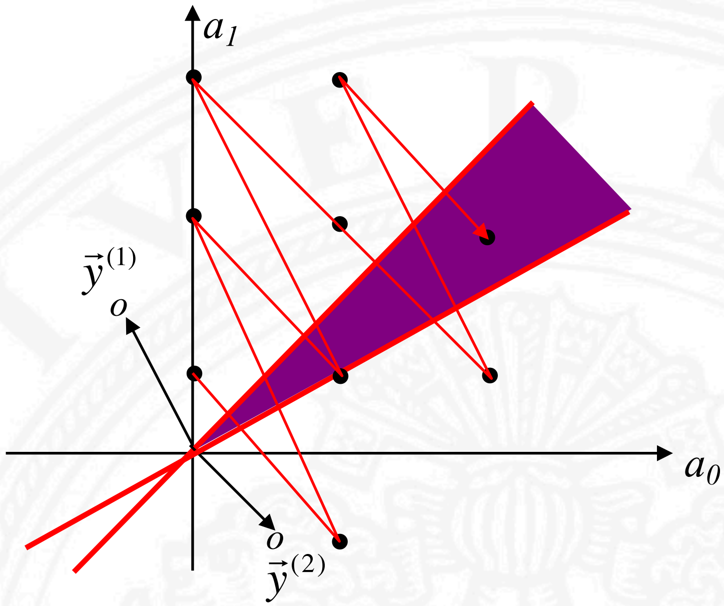
with B = {all misclassified prototypes}

Basic gradient descent algorithm:

Gradient: $$\nabla J_p(\vec{a}) = \sum_{y \in B} \left(-\vec{y}\right)$$

Step: $$\vec{a}_{k+1} = \vec{a}_k + \rho_k \sum_{y \in B} \left(\vec{y}\right)$$

Weight vector $\vec{a}$ is modified in negative gradient direction!



iterations viewed in weight space

Example (see illustration) with:

$\vec{y}_1 = \left(-1 \;\; 2\right)^T \;,\; \vec{y}_2 = \left(-1 \;\; 1\right)^T \;,\; \rho = 2$

k	0	1	2	3	4	5	6	7	8
$\vec{a}_k$	0 1	2 -1	0 3	2 1	0 5	2 3	4 1	2 5	4 3 (solution)

Minimizing the Discriminant Criterion

General form of gradient descent:

$$\vec{a}_{k+1} = \vec{a}_k - \rho_k \nabla J(\vec{a}_k) \quad \text{with} \quad \nabla J(\vec{a}_k)^T = \left(\frac{\partial J}{\partial a_0} \; \frac{\partial J}{\partial a_1} \; \dots \; \frac{\partial J}{\partial a_N} \right)$$

One can determine the optimal ρ_k which achieves the minimal $J(\vec{a}_{k+1})$ at the kth step by approximating $J(\vec{a})$ with a second-order Taylor series expansion:

$$J(\vec{a}_k) \approx J(\vec{a}_{k+1}) + \nabla^T J(\vec{a}_k)(\vec{a} - \vec{a}_k) + \tfrac{1}{2}(\vec{a} - \vec{a}_k)^T D(\vec{a}_k)(\vec{a} - \vec{a}_k)$$

where $D(\vec{a}_k)$ is the matrix of second derivatives $\frac{\partial^2 J}{a_i \partial a_j}$ evaluated at $\vec{a}_k$.

Using the iteration rule:

$$J(\vec{a}_{k+1}) \approx J(\vec{a}_k) - \rho_k \left\| \nabla^T J(\vec{a}_k) \right\|^2 + \tfrac{1}{2}\left(\rho_k\right)^2 \nabla J(\vec{a}_k)^T D(\vec{a}_k) \nabla J(\vec{a}_k)$$

The minimizing ρ_k is:

$$\rho_k = \frac{\left\| \nabla^T J(\vec{a}_k) \right\|^2}{\nabla J(\vec{a}_k)^T D(\vec{a}_k) \nabla J(\vec{a}_k)}$$

Newton´s algorithm is an alternative:
Choose $\vec{a}_{k+1}$ which minimizes $J(\vec{a})$ in the Taylor series approximation.

$$\vec{a}_{k+1} = a_k - D^{-1} \nabla J\left(a_k\right)$$

Quadratic Criterion Function

Quadratic criterion function:

$$J_q(\vec{a}) = \sum_{y \in B} \left(\vec{a}^T \vec{y}\right)^2$$

with $B = \{$all samples where $\vec{a}^T \vec{y} \leq 0\}$

Problems:

- slow convergence close to boundaries $\vec{a}^T \vec{y} \approx 0$
- dominated by long sample vectors $\vec{y}$

Normalized quadratic criterion function:

$$J_r(\vec{a}) = \frac{1}{2} \sum_{y \in B} \frac{\left(\vec{a}^T \vec{y} - b\right)^2}{\|\vec{y}\|^2}$$

with $B = \{$all samples where $\vec{a}^T \vec{y} < b\}$

Gradient: $\nabla J_r(\vec{a}) = \sum_{y \in B} \frac{\vec{a}^T \vec{y} - b}{\|\vec{y}\|^2} \vec{y}$

Iteration rule: $\vec{a}_{k+1} = \vec{a}_k + \rho_k \sum_{y \in B} \frac{b - \vec{a}^T \vec{y}}{\|\vec{y}\|^2} \vec{y}$

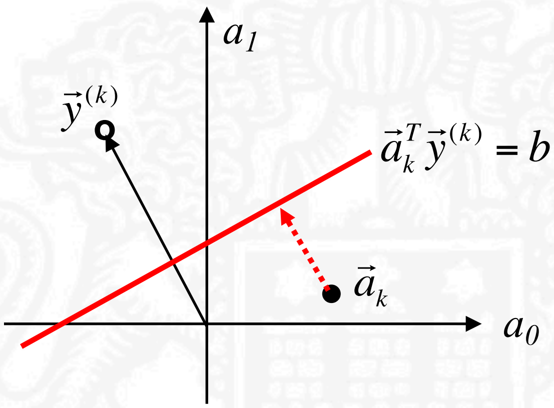
Relaxation Rule

If corrections based on the normalized quadratic criterion are performed for each single sample, one gets the "relaxation rule":

$$\vec{a}_{k+1} = \vec{a}_k + \rho \frac{b - \vec{a}_k^T \vec{y}^{(k)}}{\left\| \vec{y}^{(k)} \right\|^2} \vec{y}^{(k)} \qquad \text{where} \quad \forall_k \; \vec{a}_k^T \vec{y}^{(k)} < b$$

Distance from $\vec{a}_k$ to hyperplane $\vec{a}_k^T \vec{y}^{(k)} = b$ is: $\dfrac{b - \vec{a}_k^T \vec{y}^{(k)}}{\left\| \vec{y}^{(k)} \right\|^2}$

For $\rho = 1$, the iteration rule calls for moving $\vec{a}_k$ directly to the hyperplane
→"relaxation" of tension in inequality $\vec{a}_k^T \vec{y}^{(k)} < b$



Typical values: $0 < \rho < 2$
$\rho < 1$ "underrelaxation"
$\rho > 1$ "overrelaxation“

Minimum Squared Error

New criterion function for all samples:

Find $\vec{a}$ such that $\vec{a}^T \vec{y}_i = b_i$ with b_i = some positive constant

In matrix notation: $Y\vec{a} = \vec{b}$ with $Y = \begin{pmatrix} \vec{y}_1{}^T \\ \vec{y}_2{}^T \\ \vdots \\ \vec{y}_M{}^T \end{pmatrix}$ and $\vec{y}_i = \begin{pmatrix} y_{i_1} \\ y_{i_2} \\ \vdots \\ y_{i_{nN}} \end{pmatrix}$

In general, $M >> N$ and Y^{-1} does not exist, hence $\vec{a} = Y^{-1}\vec{b}$ is no solution.

Classical solution technique: Minimize squared error criterion:

$$J_s(\vec{a}) = \left\| Y\vec{a} - \vec{b} \right\|^2 = \sum \left(\vec{a}^T \vec{y}_i - b_i \right)^2$$

Closed-form solution by setting the gradient equal to 0.

$$\nabla J_s(\vec{a}) = 2Y^T \left(Y\vec{a} - \vec{b} \right) = 0 \;\Rightarrow\; \vec{a} = \left(Y^T Y \right)^{-1} Y^T \vec{b}$$

if $(Y^TY)^{-1}Y^T$ is nonsingular

pseudoinverse of Y

Ho-Kashyap Procedure

The MSE solution $\vec{a} = (Y^T Y)^{-1} Y^T \vec{b}$ does not necessarily provide a separating hyperplane if the classes are linearly separable, because $\vec{b}$ is chosen arbitrarily.

Ho-Kashyap algorithm searches for $\vec{a}$ and $\vec{b}$ such that $Y\vec{a} = \vec{b} > \vec{0}$ by minimizing J_s w.r.t. $\vec{a}$ and $\vec{b}$:

1. Iterate over $\vec{a}$ by choosing $\vec{a}_k = (Y^T Y)^{-1} Y^T \vec{b}_k$
2. Iterate over $\vec{b}$ by choosing $\vec{b}_1 > \vec{0}$:

$$\vec{b}_{k+1} = b_k + 2\rho \vec{e}_k^{\,+} \qquad 0 < \rho < 1$$

with

$$\vec{e}_k = Y\vec{a}_k - \vec{b}_k \qquad \text{error vector}$$

$$\vec{e}_k^{\,+} = \tfrac{1}{2}\left(\vec{e}_k + |\vec{e}_k|\right) \qquad \text{positive part of } \vec{e}_k$$

Ho-Kashyap iteration over $\vec{b}$ generates sequence of margin vectors $\vec{b}$ which

- minimizes squared error criterion
- gives only positive margins $\vec{b} > \vec{0}$

For linearly separable classes and $0 < \rho < 1$, the Ho-Kashyap algorithm will converge in a finite number of steps.

Discrimination with Potential Functions

Idea: Electrostatic potential centered at each prototype may sum up to a useful discriminant function

Example:

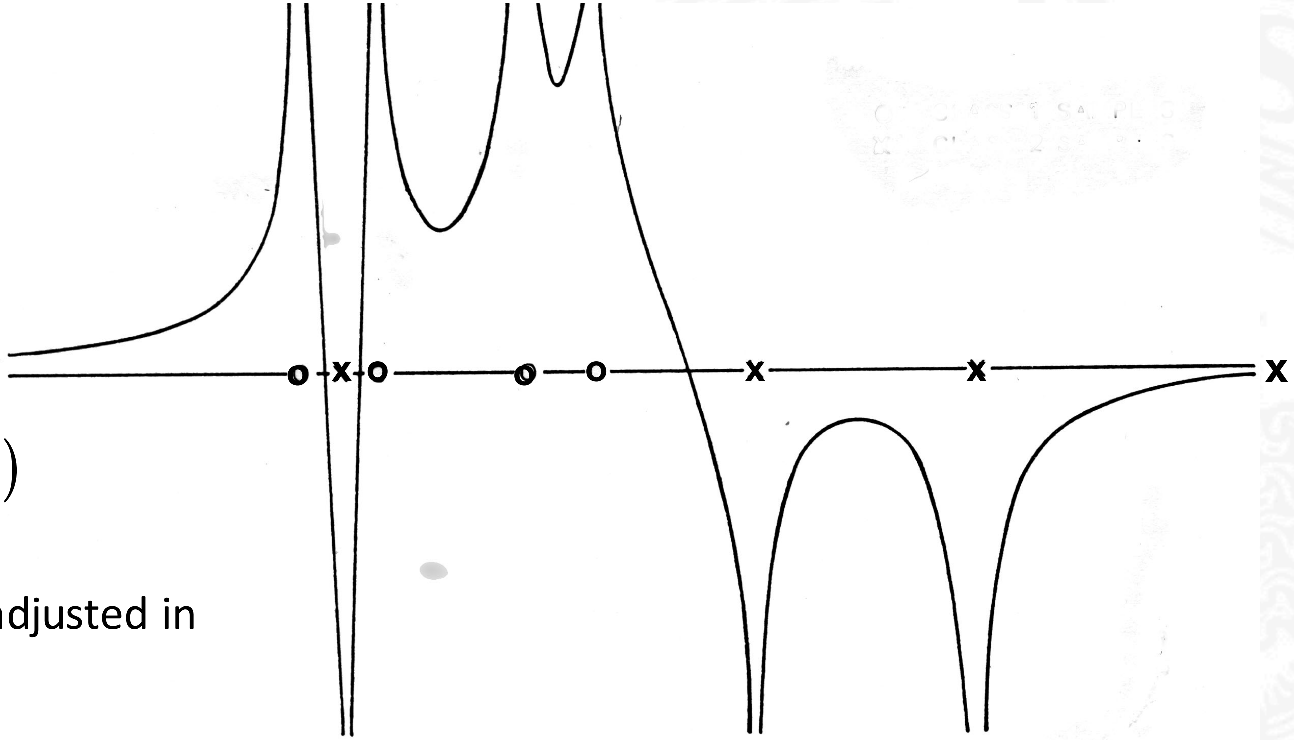
potential function

$$K\left(\vec{x}, \vec{x}_i\right) = \frac{1}{\left\|\vec{x} - \vec{x}_i\right\|^2}$$

discriminant function

$$g\left(\vec{x}\right) = \sum_i q_i K\left(\vec{x}, \vec{x}_i\right)$$

"charges" q_i may be adjusted in learning procedure



Construction of Discriminant Functions Based on Potential Functions

Different choices for potential functions are possible, for example:

$$K(\vec{x}, \vec{x}_k) = \frac{\sigma^2}{\sigma^2 + \left\| \vec{x} - \vec{x}_k \right\|^2}$$

$$K(\vec{x}, \vec{x}_k) = e^{-\frac{1}{2\sigma^2}\left\| \vec{x} - \vec{x}_k \right\|^2}$$

Potential functions must be tuned to provide the right kind of interpolation between samples!

Iterative construction:

$$g'(\vec{x}) = \begin{cases} g(\vec{x}) + K(\vec{x}, \vec{x}_k) & \text{if } \vec{x}_k \text{ is of class 1 and } g(\vec{x}_k) \le 0 \\ g(\vec{x}) - K(\vec{x}, \vec{x}_k) & \text{if } \vec{x}_k \text{ is of class 2 and } g(\vec{x}_k) \ge 0 \\ g(\vec{x}) & \textit{otherwise} \end{cases}$$