# Linear Discriminant Analysis

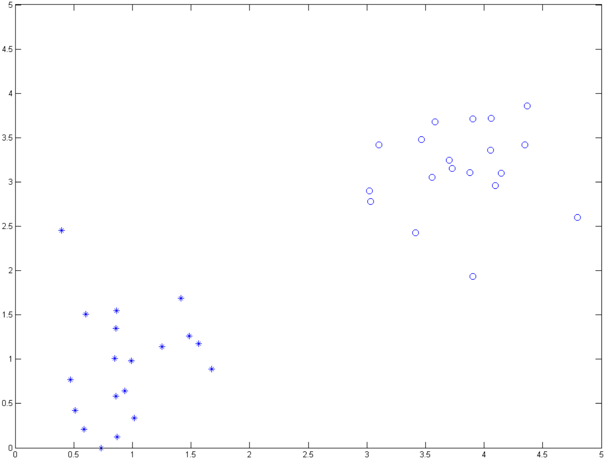
Tomasz Przybyła

30.03.2011

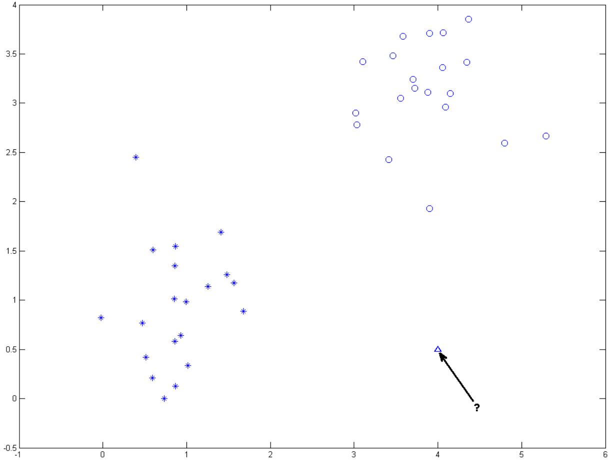
# Agenda

1. **Introduction**
   - Discriminant analysis
   - Discriminant function
   - Linear discriminant function
   - Linear machine
   - Generalised linear discrimination function
2. **Two–class algorithms**
   - General ideas
   - Perceptron criterion
   - Relaxation algorithm
   - Fisher's criterion
3. **Numerical fun**
   - Data set

# A simple picture



# A simple picture



# Discriminant analysis

## What is the discriminant analysis?

Main goal of the discriminant analysis is a formulation criteria for class separability.
This is a general description of the pattern recognition task – classification problem.

# Discriminant function

In discriminant analysis we seek a function that allows class separability. Such function is termed discriminant function.

## Discriminant function

In two–class problem, a discriminant function $h(\mathbf{x})$ is a function for which

$$h(\mathbf{x}) \begin{cases} > k, & \Rightarrow \mathbf{x} \in \omega_1 \\ < k, & \Rightarrow \mathbf{x} \in \omega_2 \end{cases}, \qquad (1)$$

for constant $k$, and the pattern $\mathbf{x}$.
In the case of equality ($h(\mathbf{x} = k)$, the pattern $\mathbf{x}$ may be assigned arbitrarily to one of the two classes.

# Discriminant function

Discriminant functions are not unique.

## Monotonic function

If function $f(\cdot)$ is a monotonic function then

$$g(\mathbf{x}) = f\left(h(\mathbf{x})\right) \begin{cases} > k', & \Rightarrow \mathbf{x} \in \omega_1 \\ < k', & \Rightarrow \mathbf{x} \in \omega_2 \end{cases} .$$

where $k' = f(k)$.

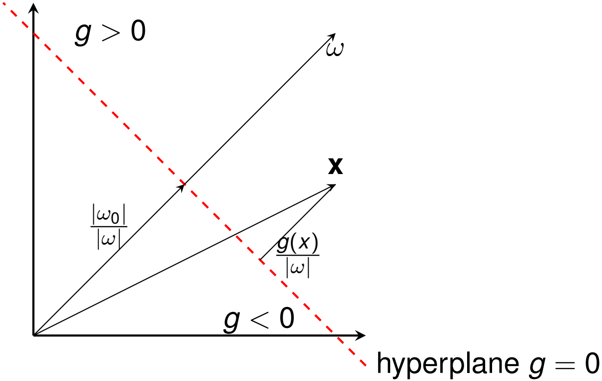
# Linear discriminant function

Let us consider the family of discrimiant functions that are linear combinations of the components of $\mathbf{x} = [x_1, x_2, \ldots, x_p]^T$, i.e.

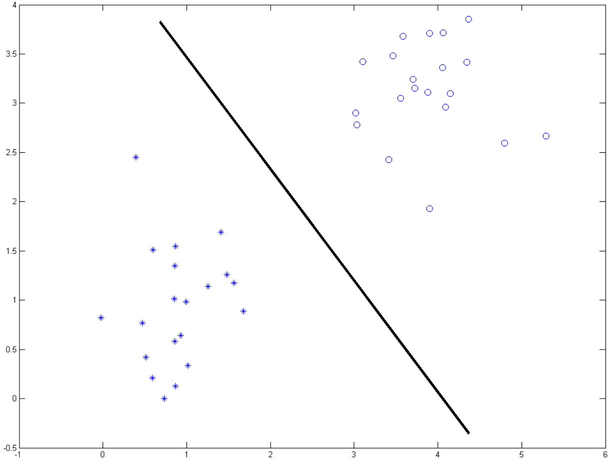
$$g(\mathbf{x}) = \omega^T \mathbf{x} + \omega_0 = \sum_{i=1}^{p} \omega_i x_i + \omega_0. \tag{2}$$

A complete specification of a linear discriminant function is achieved by prescribing thw weight wector $\omega$ and the treshold weight $\omega_0$.

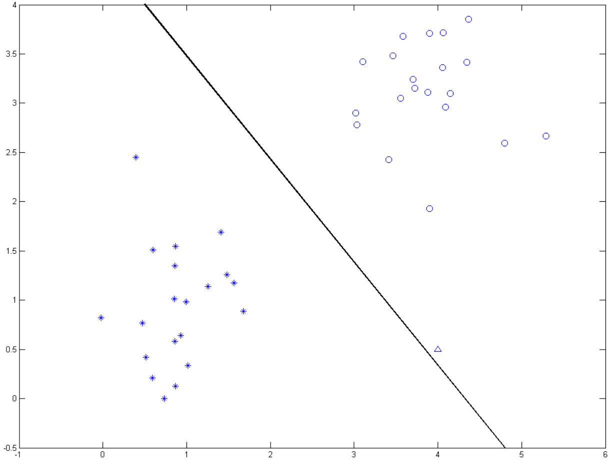
## Geometry of linear discrimination function



# Geometry of linear discrimination function



# Geometry of linear discrimination function



# Linear machine

## Linear machine

A pattern classifier employing linear discriminant function is termed a linear machine.

Suppose we are given a set of prototype points (vectors) $\mathbf{p}_1, \ldots, \mathbf{p}_c$, one for each of the $c$ classes $\omega_1, \ldots, \omega_c$. The minimum–distance classifier assigns a pattern $\mathbf{x}$ to the class $\omega_k$ associated with the nearest point $\mathbf{p}_k$.
For each point, the squared Euclidean distance is

$$d_k = |\mathbf{x} - \mathbf{p}_k|^2$$

# Linear machine

Let us consider the squared Euclidean distance

$$d_k = |\mathbf{x} - \mathbf{p}_k|^2 = \mathbf{x}^T\mathbf{x} - 2\mathbf{x}^T\mathbf{p}_k + \mathbf{p}_k^T\mathbf{p}_k.$$

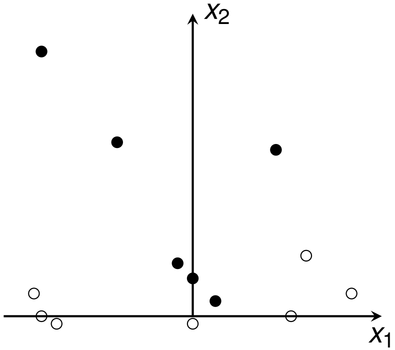
Minimum–distance classification is achieved by comparing the expressions $\mathbf{x}^T\mathbf{p}_k - 1/2\mathbf{p}_k^T\mathbf{p}_k$ and selecting the largest value. Thus the linear discriminant function is

$$g_k(\mathbf{x}) = \omega_k^T\mathbf{x} + \omega_{k,0},$$

where

$$\omega_k = \mathbf{p}_k$$
$$\omega_{k,0} = \frac{1}{2}|\mathbf{p}_k|^2$$

# Generalised linear discriminant function



# Generalised linear discriminant function

A generalised linear discriminant function, also termed a phi machine is a discriminant function of the form

$$g(\mathbf{x}) = \omega^T \phi + \omega_0,$$

where

$$\phi = \begin{bmatrix} \phi_1(\mathbf{x}) \\ \phi_2(\mathbf{x}) \\ \vdots \\ \phi_D(\mathbf{x}) \end{bmatrix}$$

is a vector function of $\mathbf{x}$.
If $D = p$, and $\phi_i(\mathbf{x}) = x_i$, then we have a linear discriminant function.

# Generalised linear discriminant function

**Generalised ...**

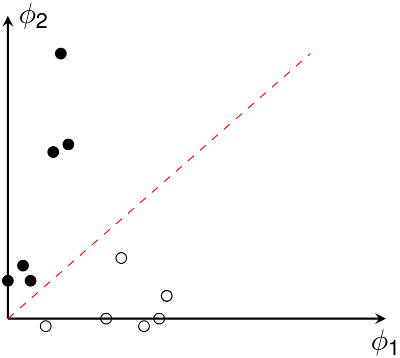
The discriminant function is linear in the functions $\phi_i$, not in the original features $x_i$.

If we make the transformation

$$\phi_1(\mathbf{x}) = x_1^2$$
$$\phi_2(\mathbf{x}) = x_2$$

then the classes in previous example can be separated in the $\phi$-space by a straight line.

# Generalised linear discriminant function



# General ideas

Suppose we have a set of training patters $\mathbf{x}_1, \ldots, \mathbf{x}_N$, each of which is assigned to one of two classes $\omega_1$ or $\omega_2$. Using this set we seek a weight vector $\omega$ and a treshold $\omega_0$ such that

$$\omega^T \mathbf{x} + \omega_0 \begin{cases} > 0, \\ < 0, \end{cases} \Rightarrow \mathbf{x} \in \begin{cases} \omega_1 \\ \omega_2 \end{cases},$$

## General ideas

A discriminant function can be defined in the following way

$$\mathbf{v}^T\mathbf{z}\begin{cases} > 0, \\ < 0, \end{cases} \quad \Rightarrow \mathbf{x} \in \begin{cases} \omega_1 \\ \omega_2 \end{cases} \quad ,$$

where

$$\mathbf{z} = [1, x_1, \ldots, x_p]^T \ ,$$

and

$$\mathbf{v} = [\omega_0, \omega_1, \ldots, \omega_p]^T \ .$$

$\mathbf{z}$ could also be

$$\mathbf{z} = [1, \phi_1(\mathbf{x}), \ldots, \phi_D(\mathbf{x})]^T \ ,$$

with $\mathbf{v}$ a $(D+1)$-dimensional vector of weights.

## General ideas

A sample in class $\omega_2$ is classified correctly, if

$$\mathbf{v}^T\mathbf{z} < 0.$$

If we were to redefine all samples in class $\omega_2$ in the design set by their negative values and denote these redefined samples by $\mathbf{y}$, then we seek a value for $\mathbf{v}$ which satisfies

$$\mathbf{v}^T\mathbf{y} > 0,$$

for all $\mathbf{y}_i$ corresponding to $\mathbf{x}_i$ in the design set.

$$\mathbf{y}_i^T = \begin{cases} \left[1, \mathbf{x}_i^T\right]^T, & \mathbf{x}_i \in \omega_1 \\ \left[-1, -\mathbf{x}_i^T\right]^T, & \mathbf{x}_i \in \omega_2 \end{cases}$$

# Perceptron criterion

The perceptron criterion function is defined as follows

$$J_P(\mathbf{v}) = \sum_{\mathbf{y}_i \in Y} \left( -\mathbf{v}^T \mathbf{y}_i \right),$$

where

$$Y = \left\{ \mathbf{y}_l | \mathbf{v}^T \mathbf{y}_l < 0 \right\}.$$

$J_P$ is proportional to the sum of the distances of the misclassified samples to the decision boundary.

# Error–correction procedure

Since the criterion function $J_P$ is continuous, we can use a gradient–based procedure, such as the method of steepest descent, to determine its minimum

$$\frac{\partial J_P}{\partial \mathbf{v}} = \sum_{\mathbf{y}_i \in Y} (-\mathbf{y}_i)$$

which is the sum of misclassified patterns.
The update rule is given by

$$\mathbf{v}_{k+1} = \mathbf{v}_k + \rho_k \sum_{\mathbf{y}_i \in Y} \mathbf{y}_i$$

where $\rho_k$ is the scale parameter that determine the step size.

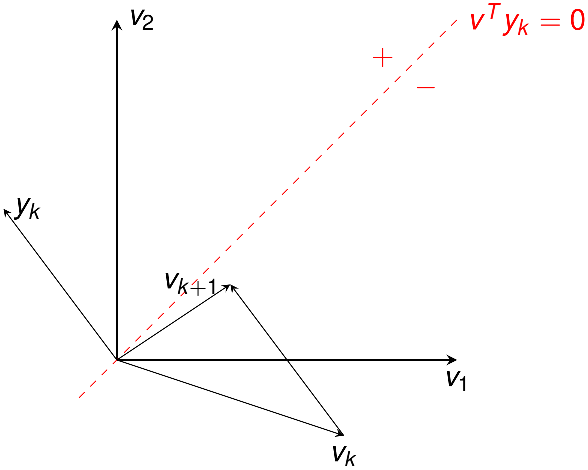
## Error–correction procedure

The mentioned algorithm is sometimes referred to as *many–pattern adaptation* or *batch–update* since all given pattern samples are used in the update $\mathbf{v}$.
The corresponding *single–pattern adaptation* scheme is

$$\mathbf{v}_{k+1} = \mathbf{v}_k + \rho_k \mathbf{y}_i.$$

where $\mathbf{y}_i$ is a training sample that has been misclassified by $\mathbf{v}_k$.

# Error–correction procedure



# Variants

## $\rho$ parameter

What is the best value of the scale parameter $\rho$?

## Constant

For the constant $\rho$ we obtain the *fixed–increment* rule. BTW, is the simplest algorithm for solving systems of linear inequalities.

# Variants

## $\rho$ parameter

What is the best value of the scale parameter $\rho$?

## Absolute correction rule

Choose the value of $\rho$ so that the value of

$$\mathbf{v}_{k+1}^T \mathbf{y}_i,$$

is positive.
Thus

$$\rho > \frac{\left|\mathbf{v}_{k+1}^T \mathbf{y}_i\right|}{|\mathbf{y}_i|^2}$$

where $\mathbf{y}_i$ is misclassified pattern presented at the $k$-th step.

# Variants

## Margin $b$

A margin, $b > 0$, is introduced and the weight vector is updated whenever

$$\mathbf{v}^T \mathbf{y}_i \leq b.$$

Thus, the solution vector $\mathbf{v}$ must lie at a distance greater than $\frac{b}{|\mathbf{y}_i|}$ from each hyperplane $\mathbf{v}^T \mathbf{y}_i = 0$.

# Relaxation algorithm

## Relaxation algorithm

The *relaxation algorithm* or *Agmon–Mays algorithm* minimises the criterion

$$J_R = \frac{1}{2} \sum_{\mathbf{y}_i \in Y} \frac{(\mathbf{v}^T \mathbf{y}_i - b)^2}{|\mathbf{y}_i|^2}$$

where $Y$ is $\{\mathbf{y}_i | \mathbf{v}^T \mathbf{y}_i \leq b\}$.

# Relaxation algorithm

## Batch update

The basic update formula is

$$\mathbf{v}_{k+1} = \mathbf{v}_k + \rho_k \sum_{\mathbf{y}_i \in Y} \frac{b - \mathbf{v}^T \mathbf{y}_i}{|\mathbf{y}_i|^2} \mathbf{y}_i$$

## Single–pattern scheme

Single–pattern scheme is defined as follows

$$\mathbf{v}_{k+1} = \mathbf{v}_k + \rho_k \frac{b - \mathbf{v}^T \mathbf{y}_i}{|\mathbf{y}_i|^2} \mathbf{y}_i$$

# Fisher's criterion

## Fisher's approach

The approach adopted by Fisher was to find a linear combination of the variables that separates the two classes as much as possible. That is, we seek the direction along which the two classes are best separated (in some sense).

## Fisher's criterion

The criterion proposed by Fisher is the ratio between–class to within–class variances. We seek a direction **w** such that

$$J_F = \frac{\left|\mathbf{w}^T(\mathbf{m}_1 - \mathbf{m}_2)\right|^2}{\mathbf{w}^T S_W \mathbf{w}}$$

is a maximum.

# Fisher's criterion

List of parameters:

- $\mathbf{m}_1,\ \mathbf{m}_2$ are the group means,
- $S_W$ is the pooled within–class sample covariance matrix

$$S_W = \frac{1}{N-2}\left(N_1\hat{\Sigma}_1 + N_2\hat{\Sigma}_2\right)$$

- $\hat{\Sigma}_1,\ \hat{\Sigma}_2$ are the maximum likelihood estimates of the covariance matrices of classes $\omega_1$ and $\omega_2$, respectively,
- $N_i$ is number of samples in class $\omega_i$, $(N_1 + N_2 = N)$.

# Fisher's criterion

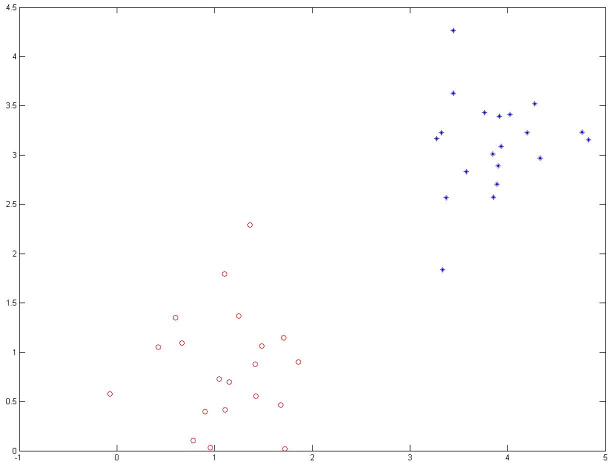
The solution that maximises $J_F$ can be obtained by differentiating $J_F$ with respect to $\mathbf{w}$ and equating to zero. After 25 years, 3 months and 23 days we can obtain the following equation

$$\mathbf{w} \propto S_W^{-1}\left(\mathbf{m}_1 - \mathbf{m}_2\right)$$

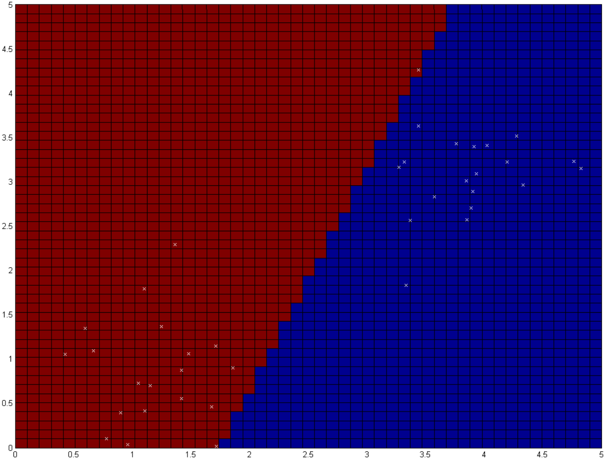
And the assign procedure of pattern $\mathbf{x}$ is given by

$$\left|\mathbf{w}^T\mathbf{x} - \mathbf{w}^T\mathbf{m}_1\right| < \left|\mathbf{w}^T\mathbf{x} - \mathbf{w}^T\mathbf{m}_2\right| .$$

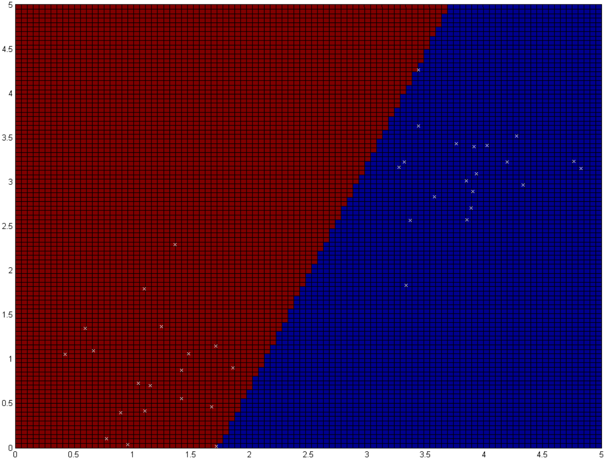
# Numerical fun



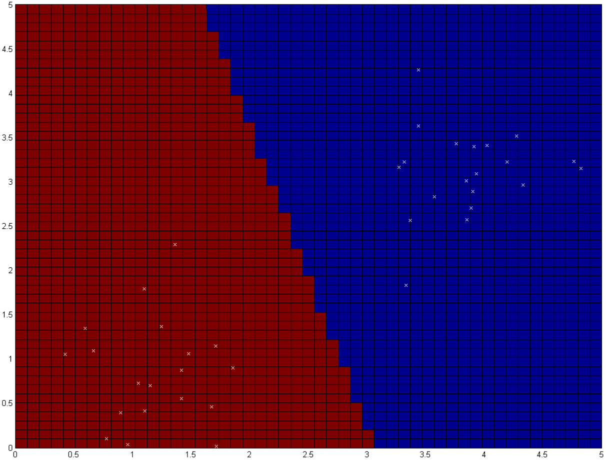
# Perceptron



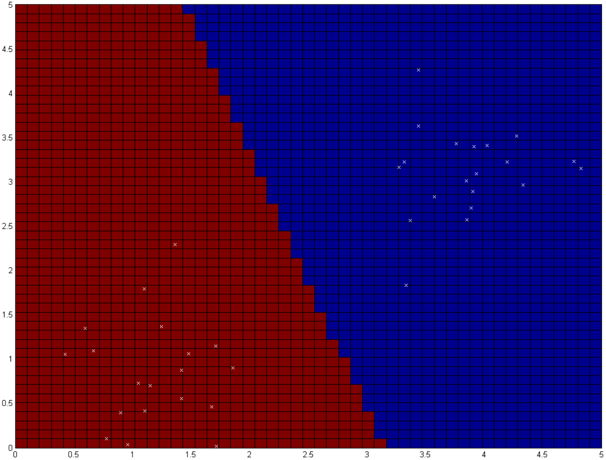
# Perceptron



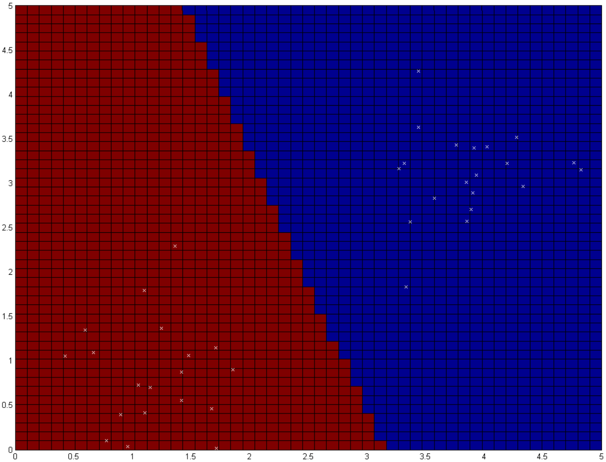
# Perceptron



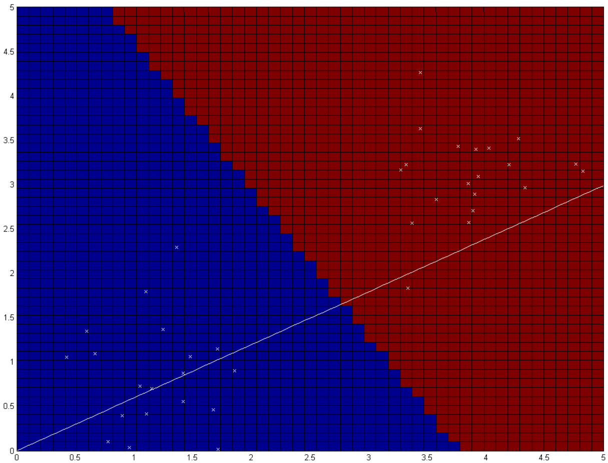
# Relaxation



# Relaxation



# Fisher



# Thanks 4 attention