Classification: Linear Discriminant Functions

CE-725: Statistical Pattern Recognition
Sharif University of Technology
Spring 2013

Soleymani

Outline

- Discriminant functions
  - Linear Discriminant functions + BW: nonlinear discriminant functions
- Linear Discriminant Function
  - Least Mean Squared Error Method LMS alg.
  - Sum of Squared Error Method SSE
  - Perceptron Error Correction method
- Multi-class problems
  - Linear machine
  - Completely Linearly Separable
  - Pairwise Linearly Separable
- Generalized LDFs

Classification Problem

- Given: Training set
  - labeled set of N input-output pairs $D = \left\{\left(\boldsymbol{x}^{(i)}, y^{(i)}\right)\right\}_{i=1}^{N}$
  - $y \in \{1, \dots, c\}$
- Goal: Given an input $\boldsymbol{x}$, assign it to one of c classes

Types of Classifiers

- Probabilistic classification approaches (previous lectures):
  - Generative
  - Discriminative

- Discriminant function
  - Various procedures for determining discriminant functions (some of them are statistical)
    - However, they don't require knowledge of the forms of underlying probability distributions

Discriminant Functions

- **Discriminant functions**: A popular way of representing a classifier
  - A discriminant function $g_i(\boldsymbol{x})$ for each class ω_i $(i = 1, \dots, c)$:
    /omega
    - $\boldsymbol{x}$ is assigned to class ω_i if:

$$g_i(\boldsymbol{x}) > g_j(\boldsymbol{x}) \quad \forall j \neq i$$

- **Decision surfaces** (boundaries) can also be found using discriminant functions
  - Boundary of the $\mathcal{R}_i$ and $\mathcal{R}_j$: $\forall \boldsymbol{x}, g_i(\boldsymbol{x}) = g_j(\boldsymbol{x})$

Probabilistic Discriminant Functions

Prior probability: p(w_i)

- Maximum likelihood
  - $g_i(\boldsymbol{x}) = p(\boldsymbol{x}|\omega_i)$ a.k.a. conditional probability: p(xlw_i),
    e.g. If you're positive(w_i), and you tested (x) positive.
- Bayesian Classifier
  - $g_i(\boldsymbol{x}) = P(\omega_i|\boldsymbol{x})$ a.k.a. Posterior probability: p(w_ilx),
    e.g. If you tested (x) positive, and you're ACTUALLY positive(w_i).
- Expected Loss (Conditional Risk)
  - $g_i(\boldsymbol{x}) = -R(a_i|\boldsymbol{x})$ +BW: a_i: treat as actions

Discriminant Functions: Two-Category

- For two-category problem, we can only find a function $g : \mathbb{R}^{\mathrm{d}} \to \mathbb{R}$
  - $g_1(\boldsymbol{x}) = g(\boldsymbol{x})$
  - $g_2(\boldsymbol{x}) = -g(\boldsymbol{x})$

H

- Decision surface: $g(\boldsymbol{x}) = 0$

- First, we explain two-category classification problem and then discuss the multi-category problems.

Linear Discriminant Functions

- Linear Discriminant Functions (LDFs)
  - assumption Decision boundaries are linear in $\boldsymbol{x}$, or linear in some given set of functions of $\boldsymbol{x}$

- Why LDFs?
  - They can be optimal for some problems
    - E.g., if the underlying distributions $p(\boldsymbol{x}|\omega_j)$ are gaussians having equal covariance
  - Even when they are not optimal, we can use their simplicity
    - LDFs are relatively easy to compute
    - In the absence of information suggesting otherwise, linear classifiers are an attractive candidates for initial, trial classifiers.

LDFs: Two Category

- $g(\boldsymbol{x};\boldsymbol{w}) = \boldsymbol{w}^T\boldsymbol{x} = w_0 + w_1x_1 + \ldots w_dx_d$ vs BW: high order poly forms
  - $\boldsymbol{x} = [1\ x_1\ x_2\ \ldots x_d]$ $\longrightarrow$ Augmented ,which is to add one dim, i.e. +1
  - $\boldsymbol{w} = [w_0\ w_1\ w_2\ \ldots w_d]$
    - w_0: bias
    - $\boldsymbol{w}$ contains the parameters we need to set

if $g(\boldsymbol{x};\boldsymbol{w}) = \boldsymbol{w}^T\boldsymbol{x} \geq 0$ then ω_1

else ω_2

- Decision surface (boundary): H $g(\boldsymbol{x};\boldsymbol{w}) = 0$
  - The equation $g(\boldsymbol{x};\boldsymbol{w}) = 0$ defines the decision surface separating samples of the two categories
    - When $g(\boldsymbol{x};\boldsymbol{w})$ is linear, the decision surface is a hyperplane. ???

LDFs : Two Category

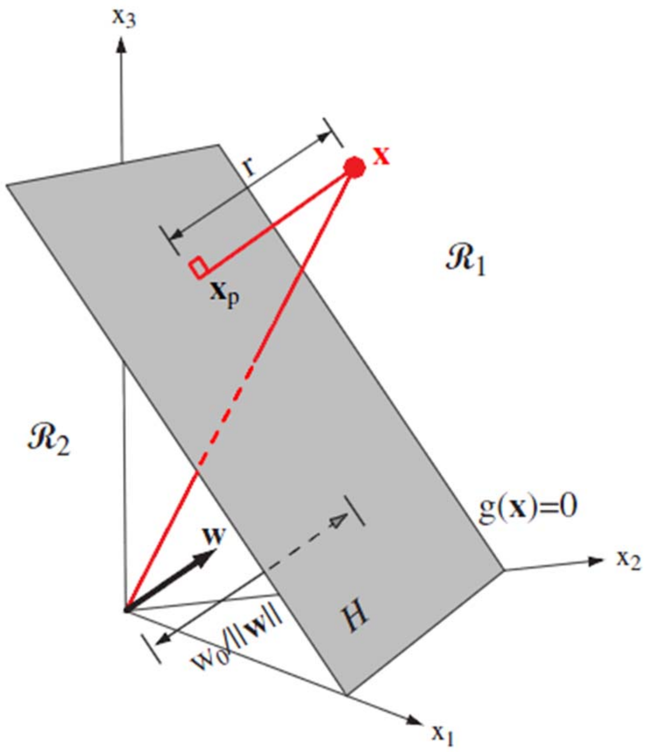
- Decision boundary is a $(d-1)$-dimensional hyperplane H in the d-dimensional feature space
  - The orientation of H is determined by the normal vector $[w_1, \ldots, w_d]$
  - w_0 is the location of the surface is determined by the bias.

+BW:笔记里有推导

$$\boldsymbol{x} = \boldsymbol{x}_p + r\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}$$

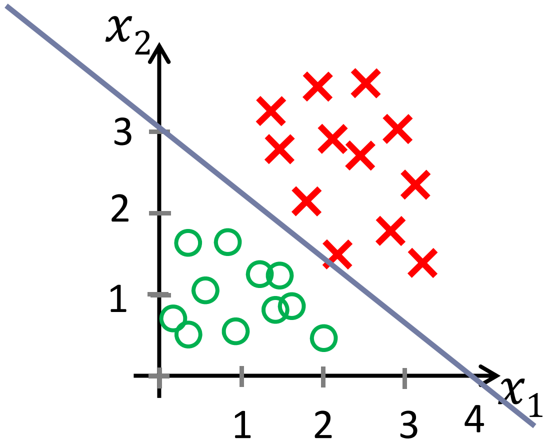
$$g(x) = r\|\boldsymbol{w}\| \Rightarrow r = \frac{g(\boldsymbol{x})}{\|\boldsymbol{w}\|}$$

$g(\boldsymbol{x})$ is proportional to the signed distance from $\boldsymbol{x}$ to H



LDFs: Two Category

$$3 + \frac{3}{4}x_1 + x_2 = 0$$



if $\boldsymbol{w}^T\boldsymbol{x} \geq 0$ ω_1
else ω_2

$$\boldsymbol{w} = [3, 0.75, 1]$$

$$\boldsymbol{x} = [1, x_1, x_2]$$

LDFs: Cost Function

- Finding LDFs is formulated as an optimization problem
  - A cost function is needed and a procedure is used to solve it.

- Criterion or cost functions for classification:
  - Average training error or loss incurred in classifying training samples
    - A small training error does not guarantee a small test error
  - We will investigate several cost functions for the classification problem

LDFs: Methods

- Many classification methods are based on LDFs:
  - Mean Squared Error i.e. MSE
  - Sum of Squared Error i.e. SSE
  - Perceptron i.e. error correction method
  - Fisher Linear Discriminant Analysis (LDA) [next lectures]
  - SVM [next lecture]

Main Steps in Methods based on LDFs

- We have specified the class of discriminant functions as linear

- Select how to measure the prediction loss
  - Based on the training set $D = \left\{\left(\boldsymbol{x}^{(i)}, y^{(i)}\right)\right\}_{i=1}^{n}$, a cost function $J(\boldsymbol{w})$ is defined (e.g., SSE) that is a function of the classifier parameters

- Solve the resulting optimization problem to find parameters:
  - Find optimal $\hat{g}(\boldsymbol{x}) = g(\boldsymbol{x}; \widehat{\boldsymbol{w}})$ where $\widehat{\boldsymbol{w}} = \underset{\boldsymbol{w}}{\operatorname{argmin}}\, J(\boldsymbol{w})$

Bayes vs. LDFs' Cost Function

- Bayes minimum error classifier:

$$\min_{\alpha(.)} E_{\boldsymbol{x},y}[L_{0-1}(\alpha(\boldsymbol{x}), y)]$$

  - If we know the probabilities in advance then the above optimization problem will be solved easily.
    - $\alpha(\boldsymbol{x}) = \underset{y}{\operatorname{argmax}}\, P(y|\boldsymbol{x})$

- We only have a set of training samples $\mathcal{D}$ instead of $p(\boldsymbol{x}, y)$ and usually optimize the following problem:

$$\min_{\alpha(.)} \frac{1}{N} \sum_{i=1}^{N} L(\alpha(\boldsymbol{x}^{(i)}), y^{(i)})$$

Mean Squared Error

Two-category: $y \in \{-1,1\}$
$y = -1$ for ω_2, $y = 1$ for ω_1

Squared Loss $\mathrm{Loss}(\alpha(\boldsymbol{x}), y) = (y - \alpha(\boldsymbol{x}))^2$

$$J(\boldsymbol{w}) = E_{\boldsymbol{x}y}\left[\left(y - g(\boldsymbol{x};\boldsymbol{w})\right)^2\right] = E[(y - \boldsymbol{w}^T\boldsymbol{x})^2]$$

E?

Gradient Descent

$$\nabla_{\boldsymbol{w}} J(\boldsymbol{w}) = \boldsymbol{0} \Rightarrow 2E[\boldsymbol{x}(y - \boldsymbol{w}^T\boldsymbol{x})] = \boldsymbol{0}$$

$$\Rightarrow E[\boldsymbol{x}y] = E[\boldsymbol{x}\boldsymbol{x}^T]\boldsymbol{w} \Rightarrow \widehat{\boldsymbol{w}} = \frac{E[\boldsymbol{x}y]}{E[\boldsymbol{x}\boldsymbol{x}^T]} = \boldsymbol{R}_{\boldsymbol{x}}^{-1}\boldsymbol{R}_{\boldsymbol{x}y}$$

$$\boldsymbol{R}_{\boldsymbol{x}} = \begin{bmatrix} E[x_1x_1] & \cdots & E[x_1x_d] \\ \vdots & \ddots & \vdots \\ E[x_dx_1] & \cdots & E[x_dx_d] \end{bmatrix} \quad \boldsymbol{R}_{\boldsymbol{x}y} = \begin{bmatrix} E[x_1y] \\ \vdots \\ E[x_dy] \end{bmatrix}$$

Sum of Squared Error (SSE)

- Cost function: Prediction errors on the training set:
  - empirical loss on N training samples

$$
\begin{aligned}
J(\boldsymbol{w}) &= \frac{1}{N}\sum_{i=1}^{N} \mathrm{Loss}\left(y^{(i)}, g\left(\boldsymbol{x}^{(i)}; \boldsymbol{w}\right)\right) \\
&= \frac{1}{N}\sum_{i=1}^{N} \left(y^{(i)} - g\left(\boldsymbol{x}^{(i)}; \boldsymbol{w}\right)\right)^2 \\
&= \frac{1}{N}\sum_{i=1}^{N} \left(y^{(i)} - \boldsymbol{w}^T \boldsymbol{x}^{(i)}\right)^2
\end{aligned}
$$

Sum of Squared Error (SSE)

$$J(\boldsymbol{w}) = \|\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}\|^2$$

unknown (i.e. w) is marked at the latter place

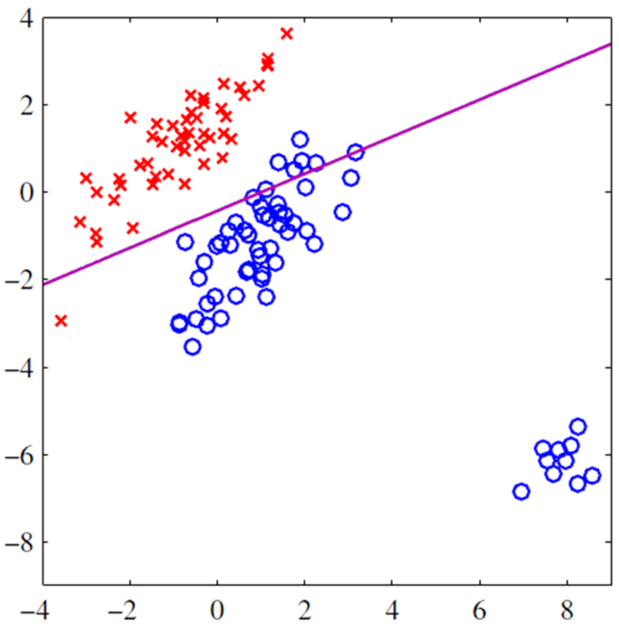
$$\underset{\text{Nx(d+1)}}{\boldsymbol{X}} = \begin{bmatrix} 1 & x_1^{(1)} & \cdots & x_d^{(1)} \\ 1 & x_1^{(2)} & \cdots & x_d^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(N)} & \cdots & x_d^{(N)} \end{bmatrix} \quad \boldsymbol{w} = \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_d \end{bmatrix} \quad \boldsymbol{y} = \begin{bmatrix} y^{(1)} \\ \vdots \\ y^{(N)} \end{bmatrix}$$

column vectors

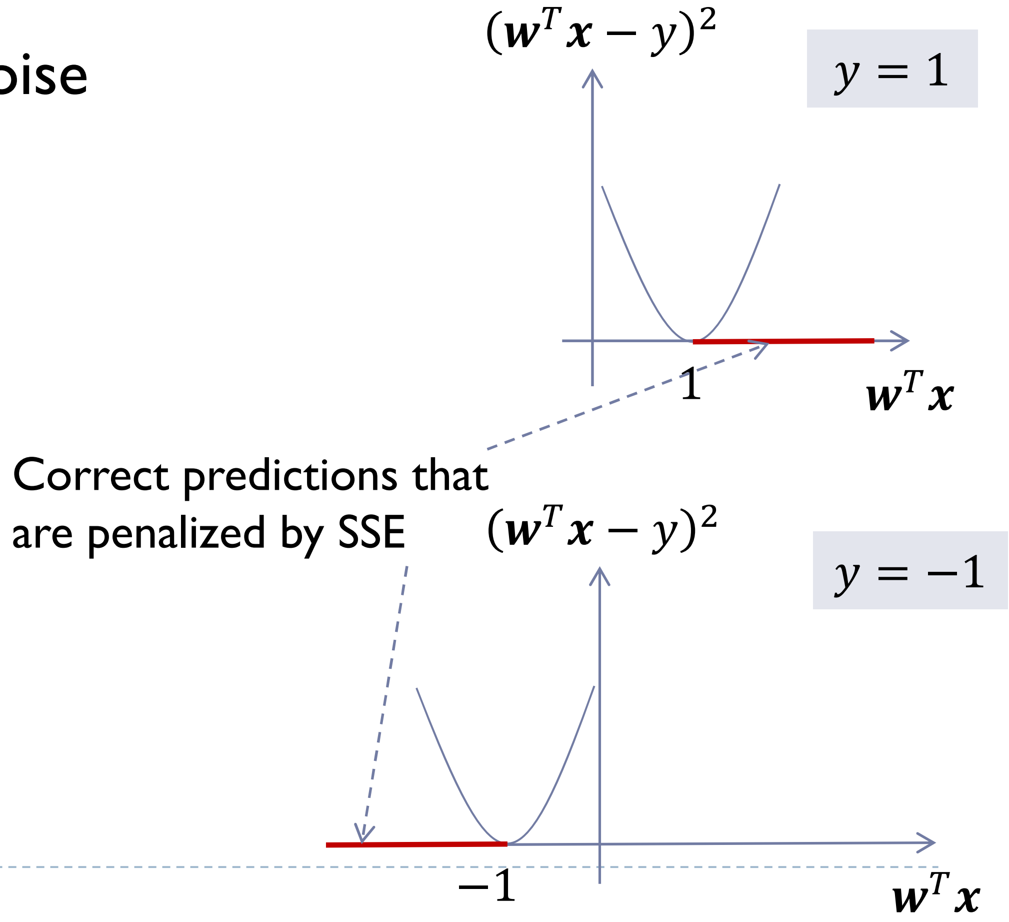
$$\nabla_{\boldsymbol{w}} J(\boldsymbol{w}) = \boldsymbol{0} \Rightarrow 2\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{w} - \boldsymbol{y}) = \boldsymbol{0} \Rightarrow \boldsymbol{X}^T\boldsymbol{X}\boldsymbol{w} = \boldsymbol{X}^T\boldsymbol{y} \Rightarrow \widehat{\boldsymbol{w}}$$
$$= \underbrace{(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T}_{\text{Pseudo Inverse of } X}\boldsymbol{y}$$

Sum of Squared Error (SSE)

- SSE penalizes 'too correct' predictions
  - 'too correct' predictions: samples lie a long way on the correct side of the decision
- It also lack robustness to noise



[Bishop]



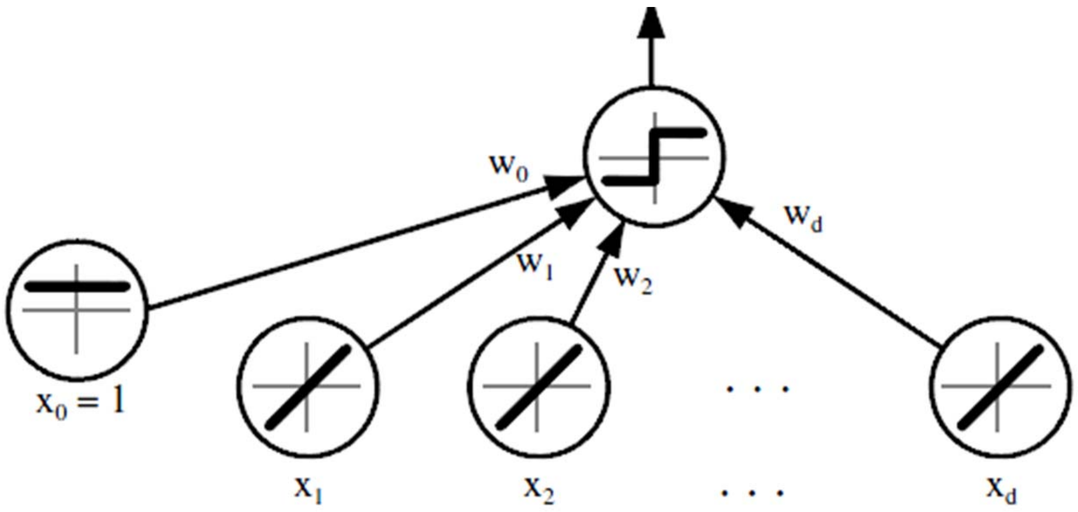
Perceptron a.k.a. Error Correction method

- Two-category: $y \in \{-1,1\}$
  - $y = -1$ for ω_2, $y = 1$ for ω_1
    /omega

- Goal: $\forall i, \boldsymbol{x}^{(i)} \in \omega_1 \Rightarrow \boldsymbol{w}^T\boldsymbol{x}^{(i)} > 0$

  $\forall i, \boldsymbol{x}^{(i)} \in \omega_2 \Rightarrow \boldsymbol{w}^T\boldsymbol{x}^{(i)} < 0$

$$f(\boldsymbol{x}) = \begin{cases} -1, & \boldsymbol{w}^T\boldsymbol{x} < 0 \\ 1, & \boldsymbol{w}^T\boldsymbol{x} \geq 0 \end{cases}$$



Perceptron Criterion

- Only misclassified training samples affect the discriminant functions:

$$J_P(\boldsymbol{w}) = -\sum_{i \in \mathcal{M}} \boldsymbol{w}^T \boldsymbol{x}^{(i)} y^{(i)}$$

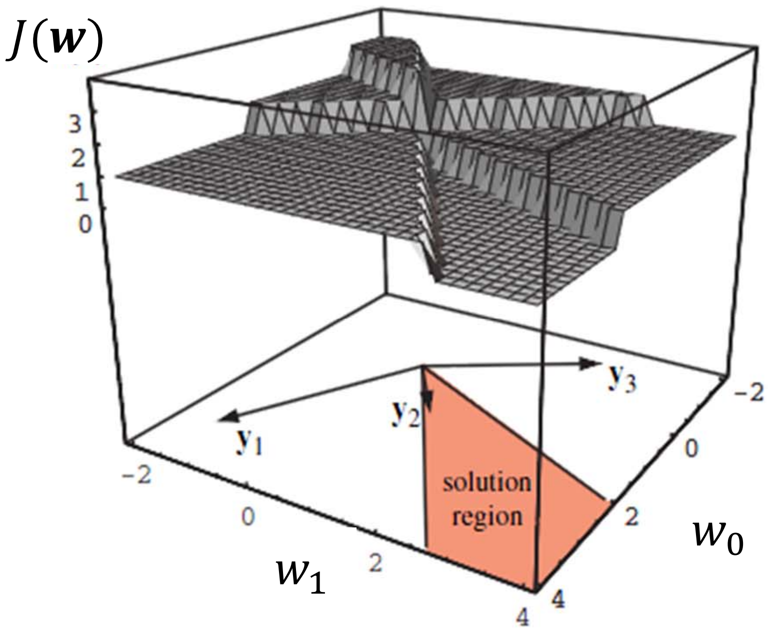
$\mathcal{M}$: subset of training data that are misclassified

- Many solutions? Which solution among them?

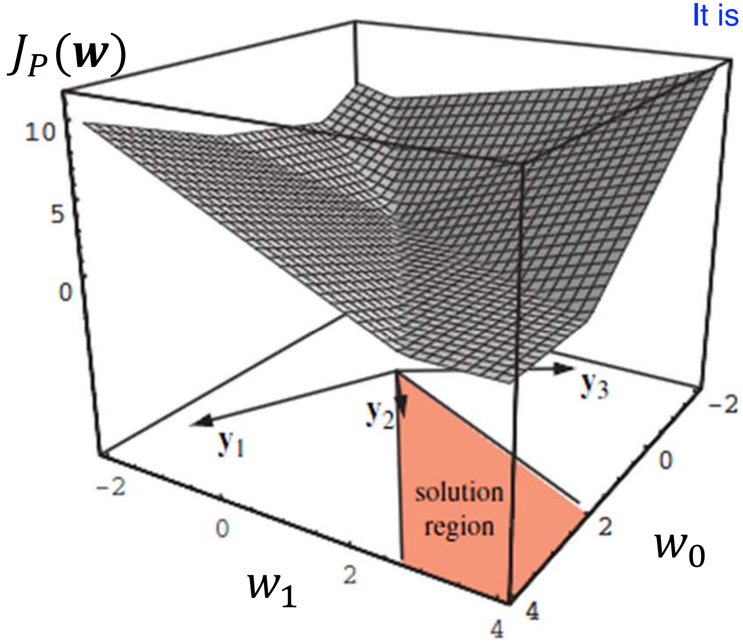
Perceptron vs. Other Criteria

Here, we are designing a better criteria or cost function.

Perceptron Jp(w) is piecewise linear.
It is not continuous.



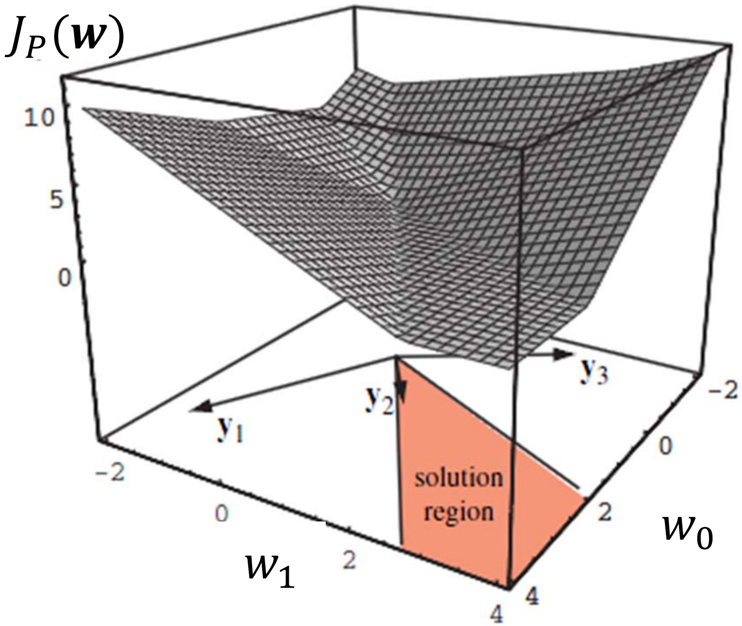
Misclassification



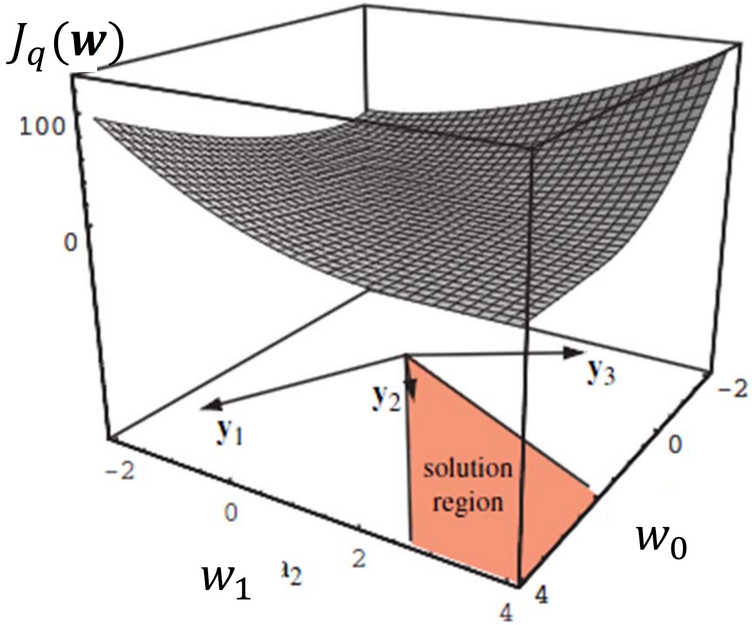
Perceptron

[Duda,Hart&Stork]

Some classification criteria



Perceptron



SSE

[Duda,Hart&Stork]

Batch Perceptron

"Gradient Descent" to solve the optimization problem:

/nabla

$$\boldsymbol{w}^{t+1} = \boldsymbol{w}^{t} - \eta \nabla_{\boldsymbol{w}} J_P(\boldsymbol{w}^{t})$$

/eta

$$\nabla_{\boldsymbol{w}} J_P(\boldsymbol{w}) = -\sum_{i \in \mathcal{M}} \boldsymbol{x}^{(i)} y^{(i)}$$

Batch Perceptron converges in finite number of steps for linearly separable data

Initialize $\boldsymbol{w}, t \leftarrow 0$
Repeat
$\quad \boldsymbol{w} = \boldsymbol{w} + \eta \sum_{i \in \mathcal{M}} \boldsymbol{x}^{(i)} y^{(i)}$
$\quad t \leftarrow t + 1$
Until $\eta \sum_{i \in \mathcal{M}} \boldsymbol{x}^{(i)} y^{(i)} < \theta$

Single-Sample Perceptron

- If $\boldsymbol{x}^{(i)}$ is misclassified:

$$\boldsymbol{w}^{t+1} = \boldsymbol{w}^{t} + \eta \boldsymbol{x}^{(i)} y^{(i)}$$

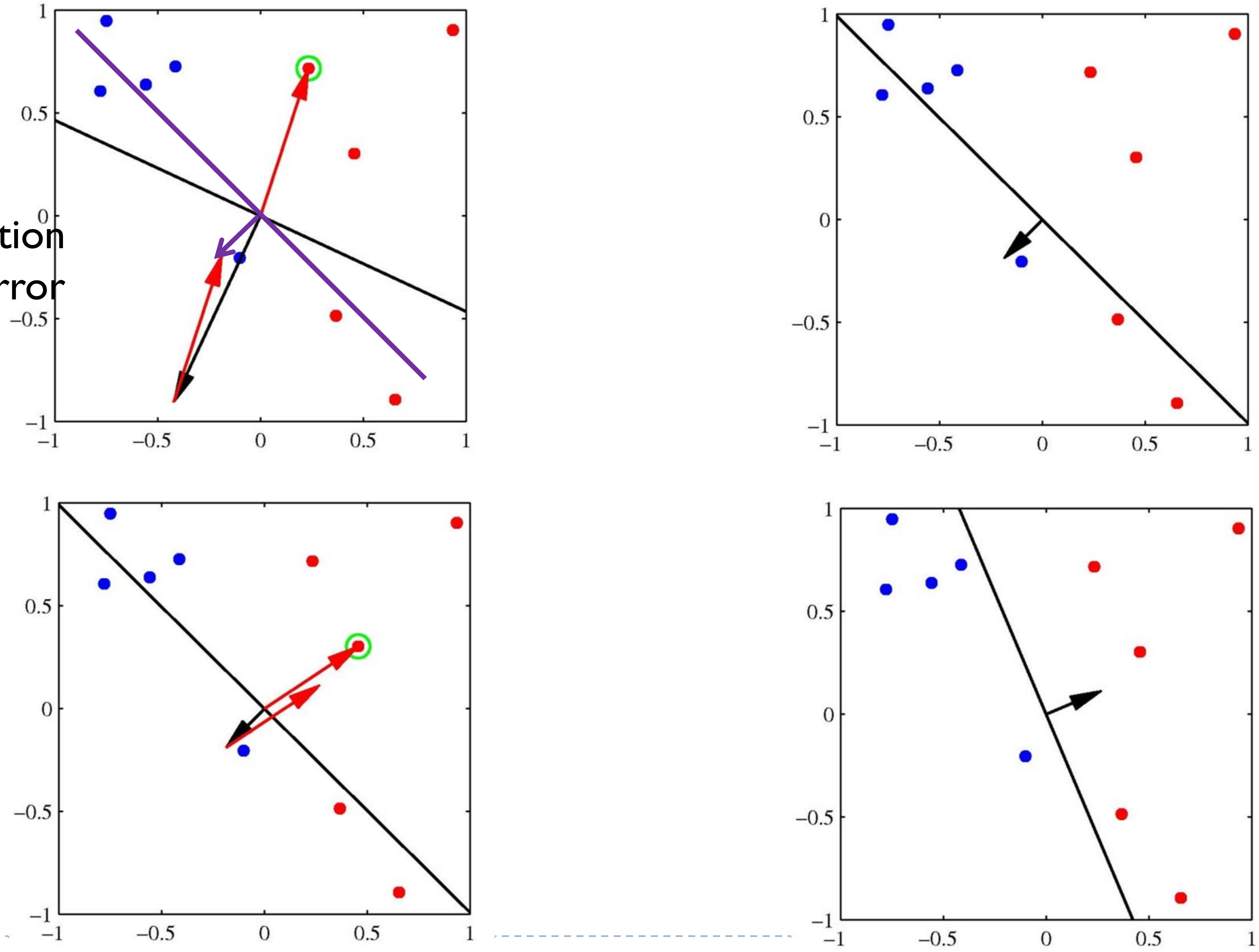
η can be set to 1

Fixed-Increment Single Sample Perceptron

Initialize $\boldsymbol{w}, k \leftarrow 0$
repeat
  $k \leftarrow (k+1) \bmod N$
  if $\boldsymbol{x}^{(i)}$ is misclassified then
    $\boldsymbol{w} = \boldsymbol{w} + \boldsymbol{x}^{(i)} y^{(i)}$
Until all patterns properly classified

Convergence of Perceptron

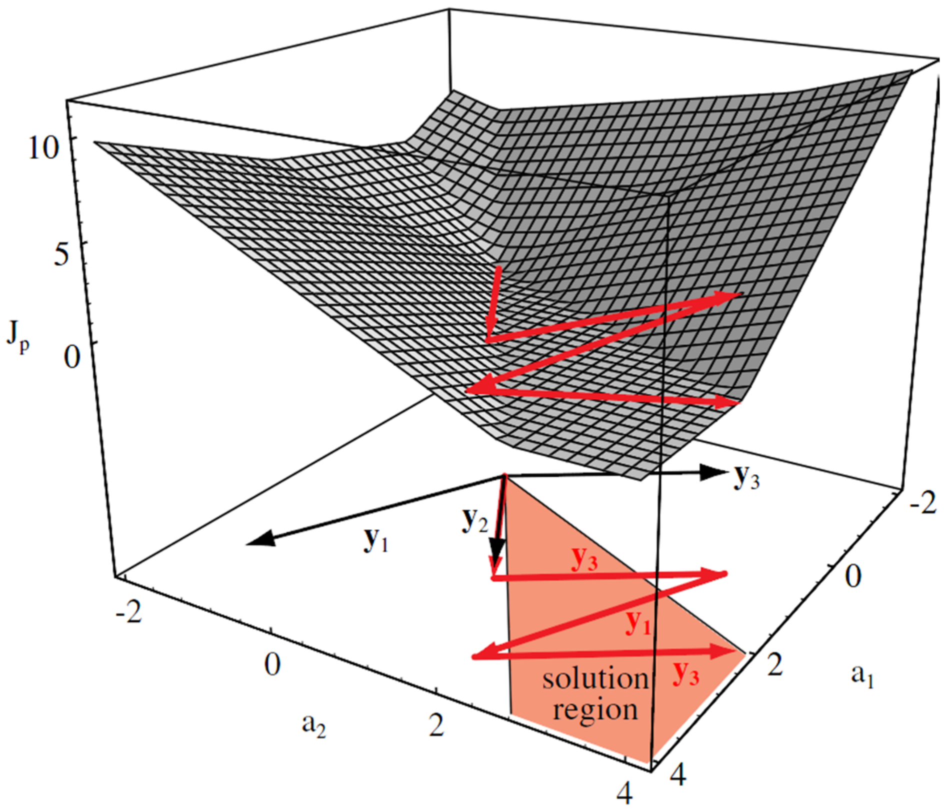
Change w in a direction that corrects the error



[Bishop]

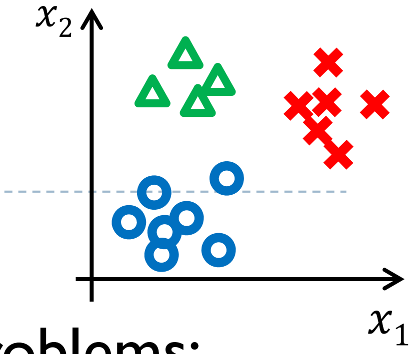
Convergence of Perceptron

- If the training data set is linearly separable, the single-sample perceptron algorithm is also guaranteed to find a solution in a finite number of steps



[Duda,Hart&Stork]

LDFs: Multi-Category

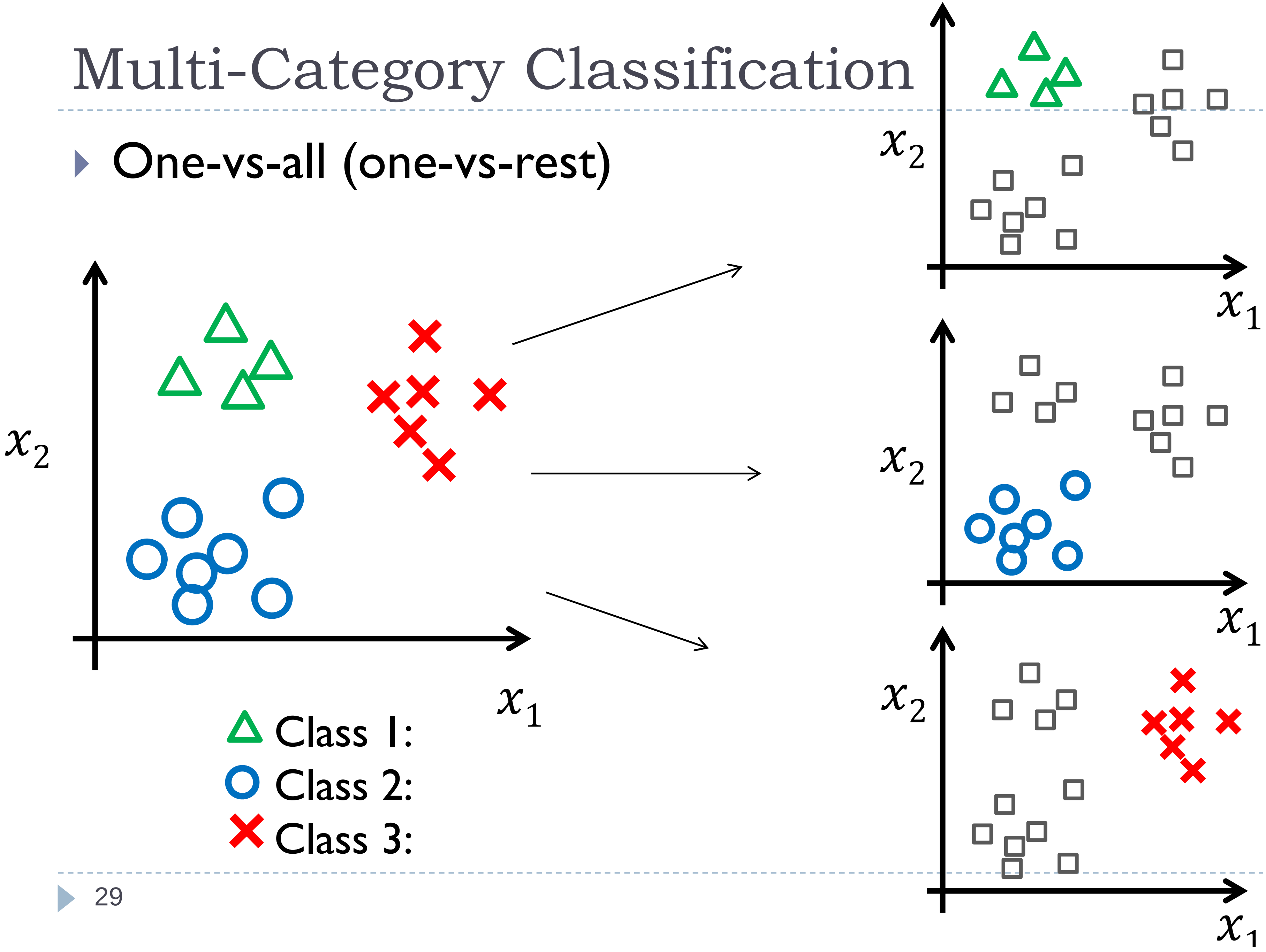


- Linear discriminant functions for multi-category problems:
  - Linear machine :
    - A discriminant function $g_i(\boldsymbol{x})$ for each class i
  - Converting the problem to a set of two-class problems:
    - "one versus rest" or "one against all"
      - For each class ω_i, an LDF separates samples of ω_i from all the other samples.
      - Totally linearly separable
    - "one versus one"
      - $c(c-1)/2$ LDFs are used, one to separate samples of a pair of classes.
      - Pairwise linearly separable

- Converting the problem to a set of two-class problems can lead to **regions in which the classification is undefined**.

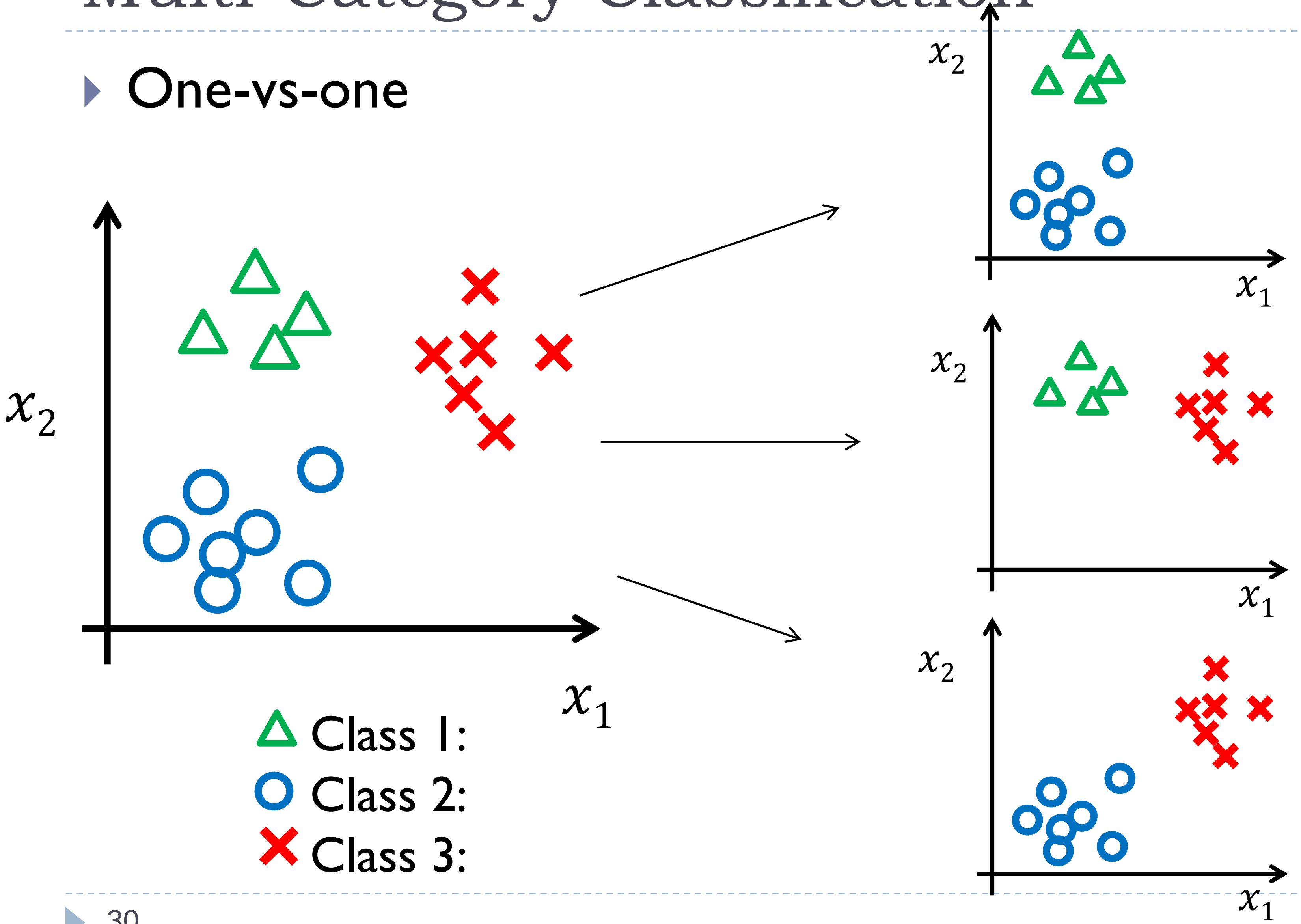
Multi-Category Classification

- One-vs-all (one-vs-rest)

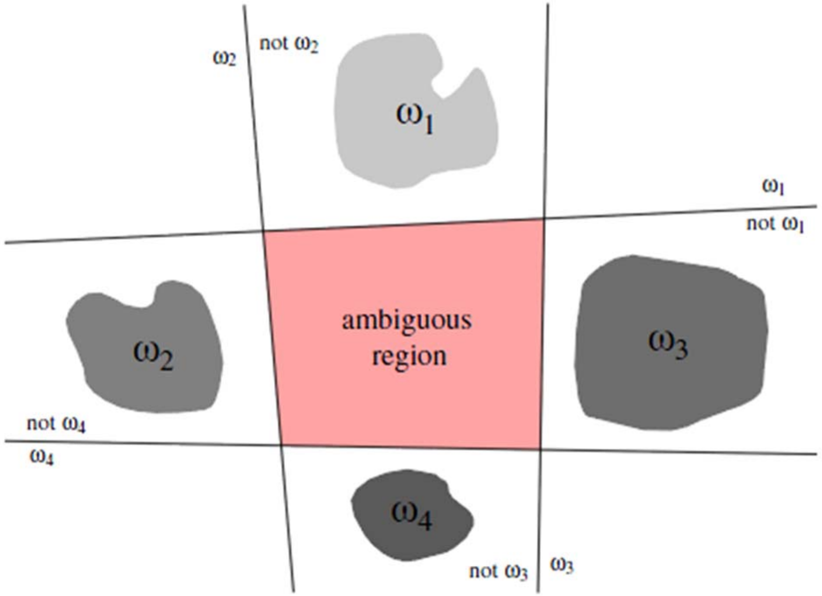


Multi-Category Classification

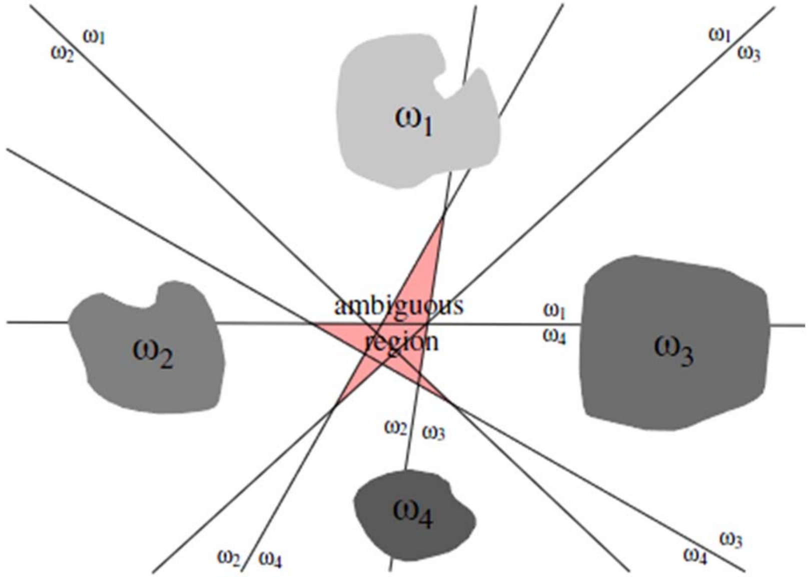
- One-vs-one



Multi-Category Classification: Ambiguity



one versus rest



one versus one

[Duda,Hart&Stork]

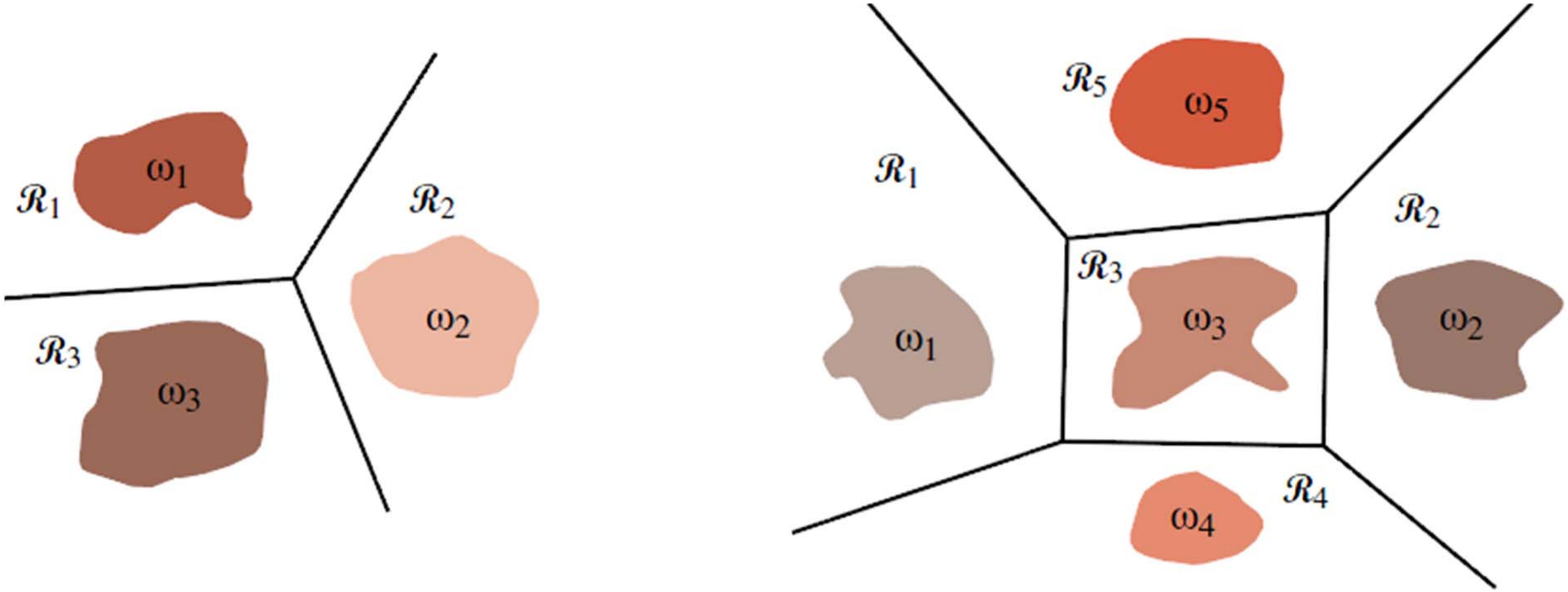
Multi-Category Classification: Linear Machine

- A discriminant function $g_i(\boldsymbol{x}) = \boldsymbol{w}_i^T \boldsymbol{x} + w_{i0}$ for each class ω_i $(i = 1, \dots, c)$:
  - $\boldsymbol{x}$ is assigned to class ω_i if:

$$g_i(\boldsymbol{x}) > g_j(\boldsymbol{x}) \quad \forall j \neq i$$

- **Decision surfaces** (boundaries) can also be found using discriminant functions
  - Boundary of the contiguous $\mathcal{R}_i$ and $\mathcal{R}_j$: $\forall \boldsymbol{x}, g_i(\boldsymbol{x}) = g_j(\boldsymbol{x})$
  - $(\boldsymbol{w}_i - \boldsymbol{w}_j)^T \boldsymbol{x} + (w_{i0} - w_{j0}) = 0$

Multi-Category Classification: Linear Machine



[Duda,Hart&Stork]

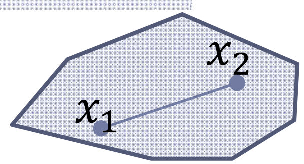
Multi-Category Classification: Linear Machine

- Decision regions are convex
  - Linear machines are most suitable for problems where $p(\boldsymbol{x}|\omega_i)$ are unimodal.

$$x_1, x_2 \in \mathcal{R}_i \Rightarrow \forall j \neq i, g_i(x_1) \geq g_j(x_1)$$
$$g_i(x_2) \geq g_j(x_2)$$

$$\Rightarrow \alpha g_i(x_1) + (1-\alpha) g_i(x_2) \geq \alpha g_j(x_1) + (1-\alpha) g_j(x_2)$$
g_i is linear $\Rightarrow g_i(\alpha x_1 + (1-\alpha)x_2) \geq g_j(\alpha x_1 + (1-\alpha)x_2)$
$$\Rightarrow \alpha x_1 + (1-\alpha)x_2 \in \mathcal{R}_i$$

Convex region definition: $\forall x_1, x_2 \in \mathcal{R},\ 0 \leq \alpha \leq 1 \Rightarrow \alpha x_1 + (1-\alpha)x_2 \in \mathcal{R}$



Multi-Category Classification: Target Coding Scheme

- Target values:
  - Binary classification: a target variable $y \in \{0,1\}$

  - Multiple classes ($K > 2$):
    - TargetClass C_j: $y_j = 1$

      $\forall i \neq j \; y_i = 0$

SSE Cost Function: Multi-Class

$$J(\boldsymbol{W}) = \underset{???}{Tr}\{(\boldsymbol{X}\boldsymbol{W} - \boldsymbol{Y})^T(\boldsymbol{X}\boldsymbol{W} - \boldsymbol{Y})\}$$

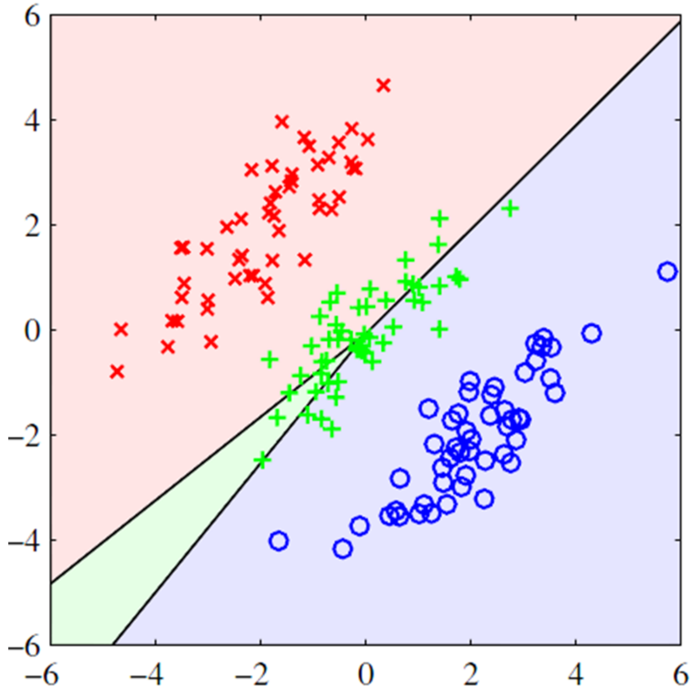
$$\boldsymbol{X} = \begin{bmatrix} 1 & x_1^{(1)} & \cdots & x_d^{(1)} \\ 1 & x_1^{(2)} & \cdots & x_d^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(n)} & \cdots & x_d^{(n)} \end{bmatrix} \quad \boldsymbol{W} = [\boldsymbol{w}_1 \quad \cdots \quad \boldsymbol{w}_c]$$

$$\boldsymbol{Y} = [\boldsymbol{y}_1 \quad \cdots \quad \boldsymbol{y}_c]$$

$$\nabla_{\boldsymbol{W}} J(\boldsymbol{W}) = \boldsymbol{0} \Rightarrow \widehat{\boldsymbol{W}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}\boldsymbol{Y}$$

SSE Cost Function: Multi-Class

- Low performance of the SSE cost function for the classification problem



[Bishop]

Perceptron: Multi-Class

$$\hat{y} = \underset{i=1,\dots,c}{\operatorname{argmax}} \boldsymbol{w}_i^T \boldsymbol{x}$$

$$J_P(\boldsymbol{W}) = -\sum_{i \in \mathcal{M}} \left(\boldsymbol{w}_{y^{(i)}} - \boldsymbol{w}_{\hat{y}^{(i)}} \right)^T \boldsymbol{x}^{(i)}$$

$\mathcal{M}$: subset of training data that are misclassified

$\mathcal{M} = \left\{ i | \hat{y}^{(i)} \neq y^{(i)} \right\}$

Initialize $\boldsymbol{W} = [\boldsymbol{w}_1, \dots, \boldsymbol{w}_c], k \leftarrow 0$
repeat
    $k \leftarrow (k+1) \bmod N$
    if $\boldsymbol{x}^{(i)}$ is misclassified then
        $\boldsymbol{w}_{\hat{y}^{(i)}} = \boldsymbol{w}_{\hat{y}^{(i)}} - \boldsymbol{x}^{(i)}$
        $\boldsymbol{w}_{y^{(i)}} = \boldsymbol{w}_{y^{(i)}} + \boldsymbol{x}^{(i)}$
Until all patterns properly classified

Generalized LDFs

- Linear combination of a fixed non-linear functions of the input vector

$$g(\boldsymbol{x};\boldsymbol{w}) = w_0 + w_1\phi_1(\boldsymbol{x}) + \dots w_m\phi_m(\boldsymbol{x})$$

/phi

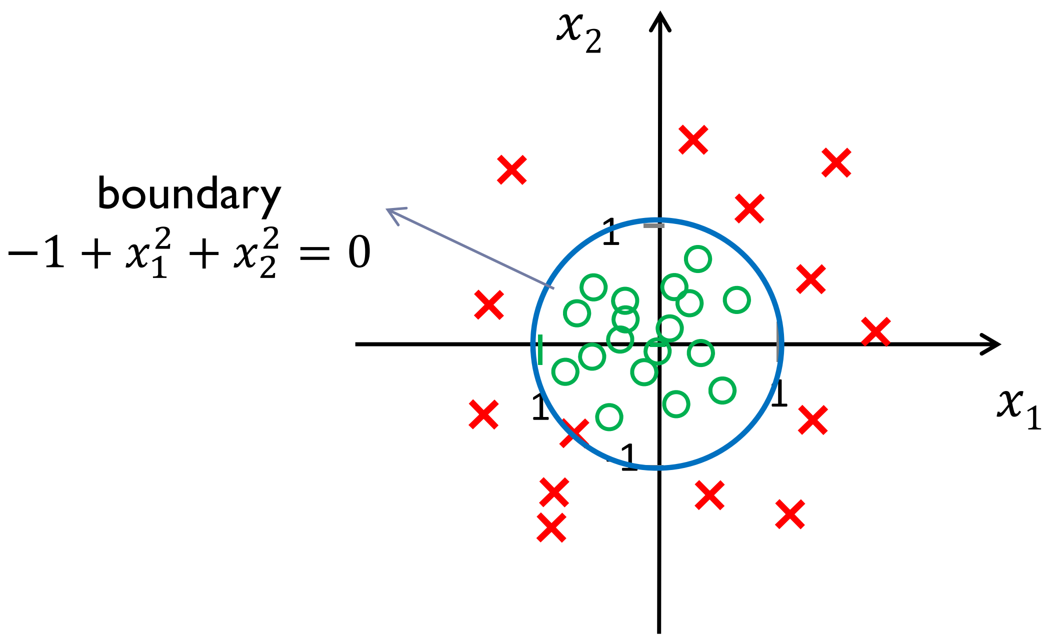
$\{\phi_1(\boldsymbol{x}), \dots, \phi_m(\boldsymbol{x})\}$: set of basis functions (or features)

$\phi_i(\boldsymbol{x}): \mathbb{R}^d \to \mathbb{R}$

We will discuss about them in the next lectures.

Generalized LDFs: Example

- Choose non-linear features
- Discriminant functions are still linear in parameters $\boldsymbol{w}$



$\boldsymbol{x} = [\boldsymbol{x}_1, \boldsymbol{x}_2]$

$\boldsymbol{\phi}(\boldsymbol{x}) = [1, \boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_1^2, \boldsymbol{x}_2^2, \boldsymbol{x}_1\boldsymbol{x}_2]$

$\boldsymbol{w} = [-1, 0, 0, 1, 1, 0]$

if $\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x}) \geq 0$ then $y = 1$
else $y = -1$

We will discuss about them in the next lectures.