Chapter 1. Introduction: Linear and non-linear models

Neural Networks

2023/2024

Máster Universitario en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital

Departamento de Sistemas Informáticos y Computación

Index

1 Introduction ▷ 3

2 Classifiers and discriminant functions ▷ 8

3 Linear discriminant functions ▷ 14

4 Generalized linear discriminant functions and kernels ▷ 28

5 Maximum margin classifiers ▷ 45

6 Bibliography ▷ 54

Index

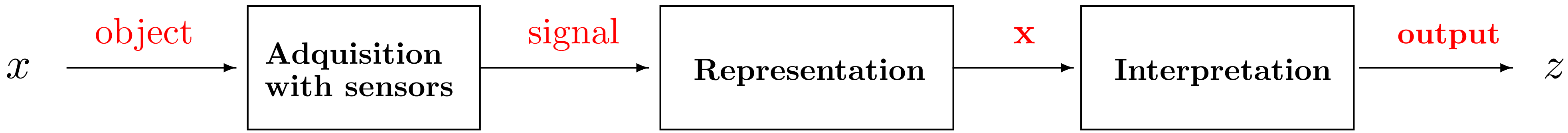
- 1 *Introduction* ▷ 3
- 2 Classifiers and discriminant functions ▷ 8
- 3 Linear discriminant functions ▷ 14
- 4 Generalized linear discriminant functions and kernels ▷ 28
- 5 Maximum margin classifiers ▷ 45
- 6 Bibliography ▷ 54

Neural networks

- **Goal:** Introduce the main techniques based on neural networks that deal with problems related to pattern recognition and machine learning.
- Focus on *automatic learning from training data and recognition*.
- Neural networks are inspired in the human ability to recognize patterns.
- **A densely interconnected set of elemental processors**.
- Other known names:
  - A connectionist models.
  - Artificial neuronal networks.
  - Distributed and parallel processing.
- Problem: Interpretability (Tools: SHAP, ...)

Adquisition, representation and recognition

- Perception entails a process of: information acquisition, representation and recognition:



- Sensors: Camera, microphone, ...
- Signals: Image file, data streaming, ...
- Representations ($\mathbf{x}$): Feature vectors, string of symbols, graphs,...
- Interpretation: class, string of symbols, real number, vector, ...

Recognition and interpretation

- Usually, pattern recognition is considered as the process to assign a **class** label to the objects that are properly acquired and represented

- However, pattern recognition can be a more general topic where the result of the process is not a class label but a vector (regression) or something more structured, e.g. a sentence, a graph, ...:

- But in this subject we will refer normally to the **classification** problem

Tools

- TensorFlow (An end-to-end open source machine learning platform) https://www.tensorflow.org/
- CUDA (A development environment for creating high performance GPU-accelerated (NVIDIA) applications) https://developer.nvidia.com/cuda-toolkit
- Keras (running on top of TensorFlow, CNTK, or Theano.) https://pypi.python.org/pypi/Keras
- PyTorch (A scientific computing framework with wide support for machine learning algorithms that puts GPUs first. Previously Torch) https://pytorch.org/
- Caffe (A deep learning framework) http://caffe.berkeleyvision.org/

Index

1 Introduction ▷ 3

○ 2 *Classifiers and discriminant functions* ▷ 8

3 Linear discriminant functions ▷ 14

4 Generalized linear discriminant functions and kernels ▷ 28

5 Maximum margin classifiers ▷ 45

6 Bibliography ▷ 54

Classes, representation space and classifiers

- **Objects & classes**: U and $\mathbb{C} = \{1, 2, \ldots, C\}$
  - Each *object* (or its signal) is showed in a *Primary Space* or "Universe", U
  - Let's assume that each object $x \in U$ belongs to a unique *class* $c(x) \in \mathbb{C}$
  - $\mathbb{C}$ is the set of all possible *identifiers* or *class labels*
  - Generalization of $\mathbb{C}$: to $\mathbb{R}$ (regression) or to a set of strings, graphs, ... (interpretation)

- **Representation Space**: E, generally $E = \mathbb{R}^D$
  - Let $\mathbf{x} = y(x)$ be the result of the preprocessing and feature extraction process applied to an object $x \in U$
  - E encloses all the possible results: $\{\mathbf{x} \, : \, \mathbf{x} = y(x), \, x \in U\} \subset E$
  - Given that two different objects from U can have the same representation in E, it is not guaranteed that each point in E belongs to a unique class.
  - Extension of E to sets of vector sequences or strings.

Classifiers and discriminant functions

Classifier: $G_{\mathbf{w}} : E \to \mathbb{C}$

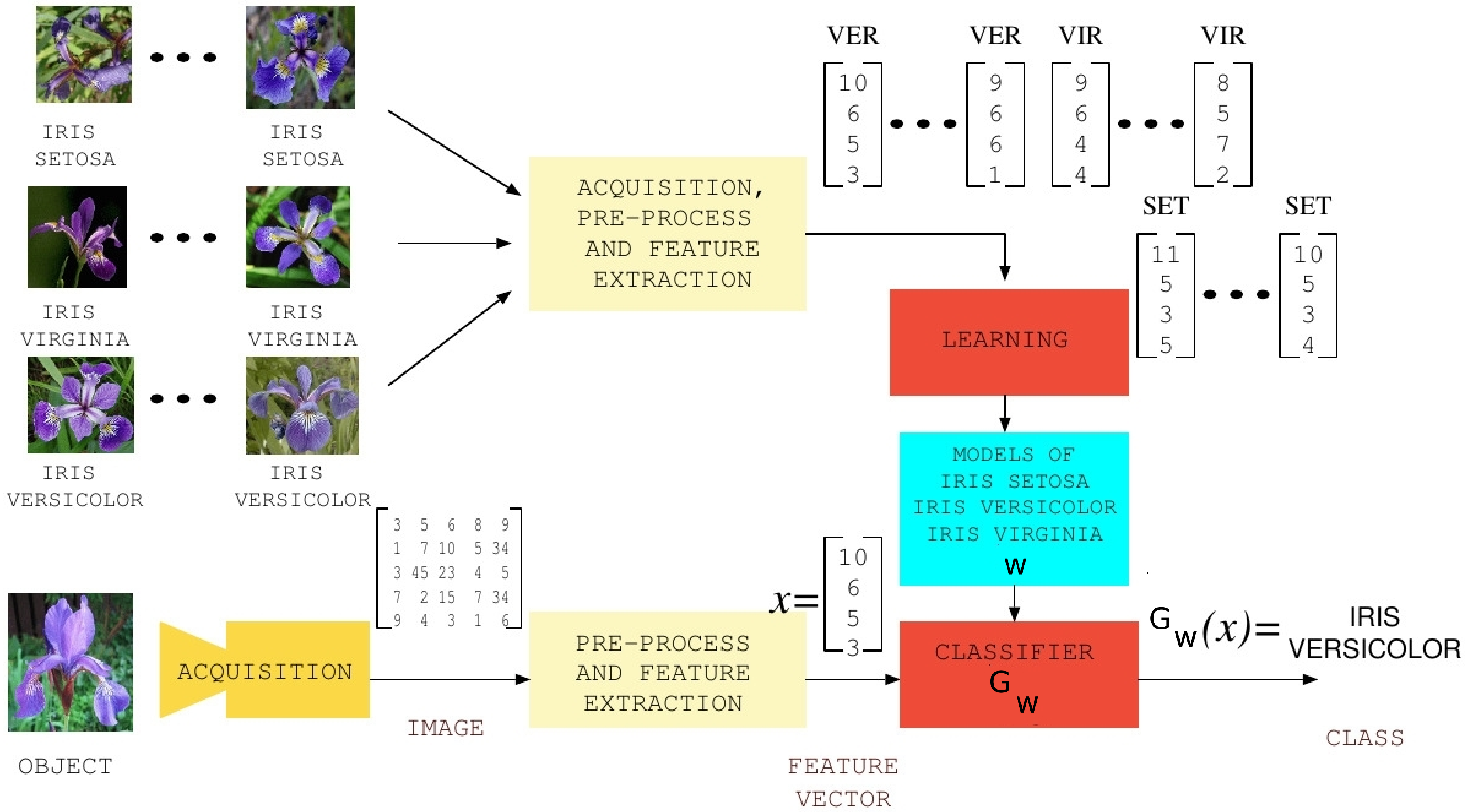
- $G_{\mathbf{w}}$ is learnt with N *labelled samples* $(\mathbf{x}_1, c_1), \ldots, (\mathbf{x}_N, c_N) \in E \times \mathbb{C}$
- The goal is to obtain the correct class for a new object $x \in U$:
$$G_{\mathbf{w}}(y(x)) = G_{\mathbf{w}}(\mathbf{x}) \stackrel{!}{=} c(x)$$
. . . the maximum number of times as possible (how to reach a 100%?)

Every classifier $G_{\mathbf{w}}$ into C classes can be stated in terms of C **discriminant functions** $g_c : E \to \mathbb{R}, \quad 1 \leq c \leq C,$ and the corresponding *classification rule*:

$$\hat{c} = G_{\mathbf{w}}(\mathbf{x}) \equiv \underset{1 \leq c \leq C}{\text{argmax}} \; g_c(\mathbf{x}; \mathbf{w})$$

Notation: $G \equiv G_{\mathbf{w}}$ and $g_c(\mathbf{x}) \equiv g_c(\mathbf{x}; \mathbf{w})$

Classes, representation space and classifiers



Decision or classification boundaires

- **Decision regions**: Any classifier partitions the representation space into C decision regions, $R_1, \ldots, R_C$:

$$R_j = \{\mathbf{x} \in E : \quad g_j(\mathbf{x}) > g_i(\mathbf{x}) \quad i \neq j,\ 1 \leq i \leq C\} \text{ for } 1 \leq j \leq C$$

- **Decision boundary between two classes i, j for $1 \leq i, j \leq C$:**
  Set of points $\mathbf{x} \in E$ for which $g_i(\mathbf{x}) = g_j(\mathbf{x})$

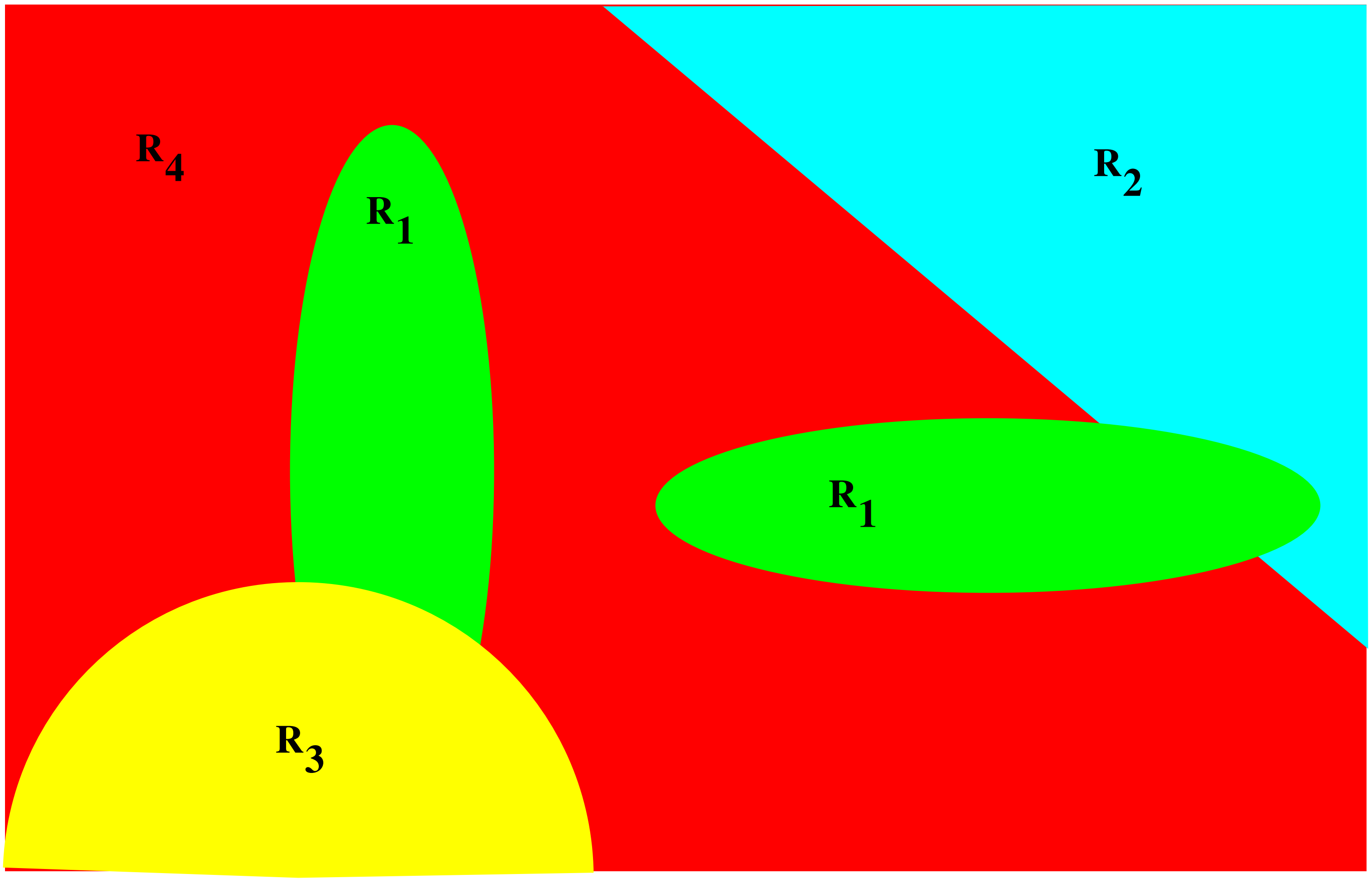
  In general they are *hypersurfaces* defined by the equations:

$$g_i(\mathbf{x}) - g_j(\mathbf{x}) = 0 \qquad i \neq j,\ \ 1 \leq i, j \leq C$$

  - If $E \equiv \mathbb{R}^3$, the boundaries are surfaces (ej. *planes*)
  - If $E \equiv \mathbb{R}^2$, the boundaries are lines (ej. *straight lines*)
  - If $E \equiv \mathbb{R}$, the boundaries are points

- **Decision boundary of a single class i for $1 \leq i \leq C$**:
  Set of points $\mathbf{x} \in E$ for which $g_i(\mathbf{x}) \ = \ \max\limits_{j \neq i} \ g_j(\mathbf{x})$

Regions and decision surfaces



Index

1 Introduction ▷ 3

2 Classifiers and discriminant functions ▷ 8

3 *Linear discriminant functions* ▷ 14

4 Generalized linear discriminant functions and kernels ▷ 28

5 Maximum margin classifiers ▷ 45

6 Bibliography ▷ 54

Linear discriminant functions (LDFs)

- A classifier is *linear* if its discriminant functions are *linear functions* of the vectors of E. Let be $\mathbf{x} \in E \equiv \mathbb{R}^D$ the representation of any object

$$g_c(\mathbf{x}) \;=\; \sum_{j=1}^{D} w_{cj} \cdot x_j + w_{c0} \;=\; \mathbf{w}_c^t \mathbf{x} + w_{c0}, \quad 1 \leq c \leq C$$

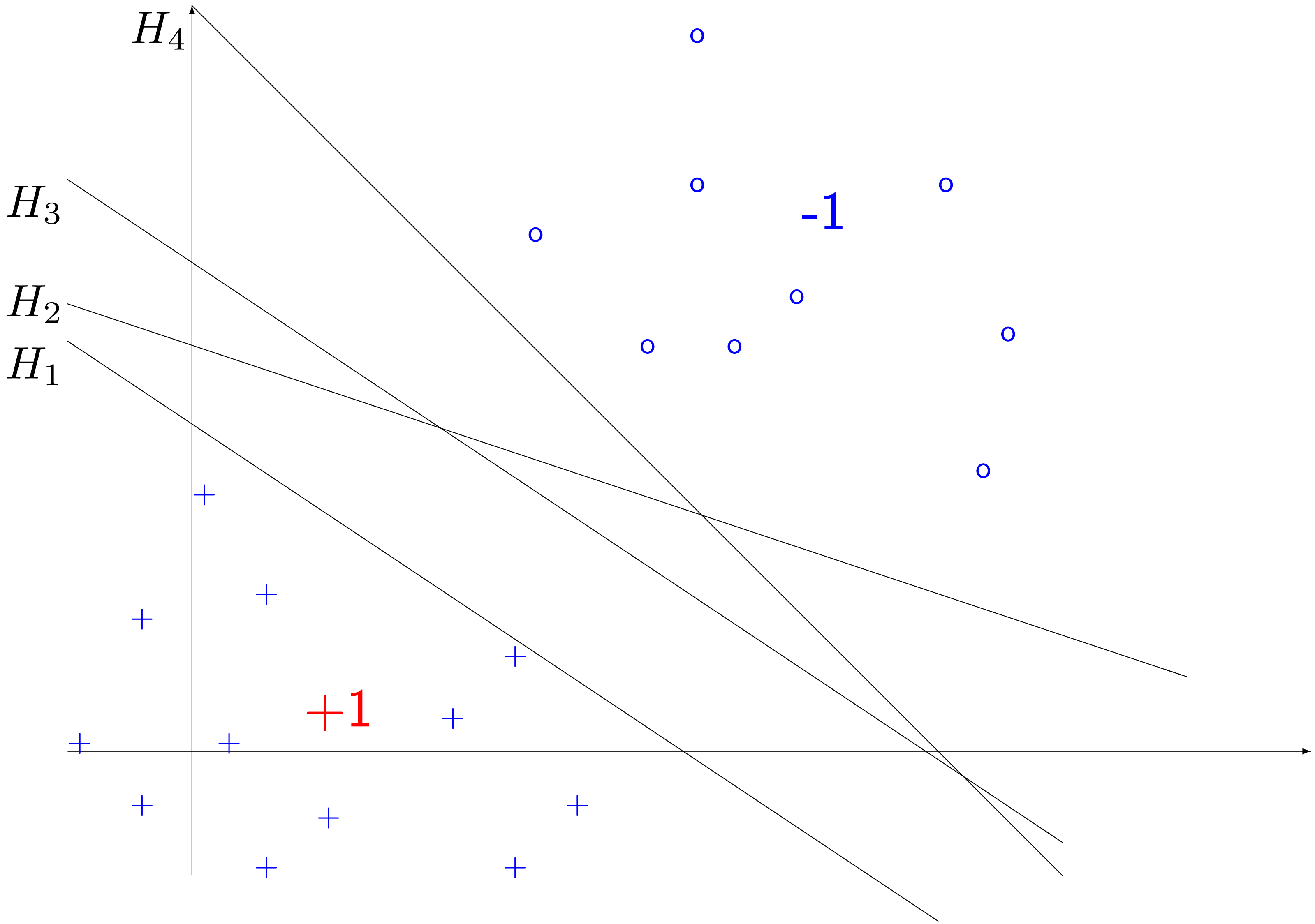
- **Classification rule**: $\hat{c} = G(\mathbf{x}) \equiv \underset{1 \leq c \leq C}{\text{argmax}} \;\; \mathbf{w}_c^t \mathbf{x} + w_{c0}$

- The *decision boundary* H_{ij} between any pair of classes i, j is:

$$H_{ij} \;=\; \{\mathbf{x} \in \mathbb{R}^D : \mathbf{w}_i^t \mathbf{x} + w_{i0} \;=\; \mathbf{w}_j^t \mathbf{x} + w_{j0}\}$$

- *Linear boundaries* or *hyperplanes* with dimension D (just lines if $D = 2$).

Linear discriminant functions



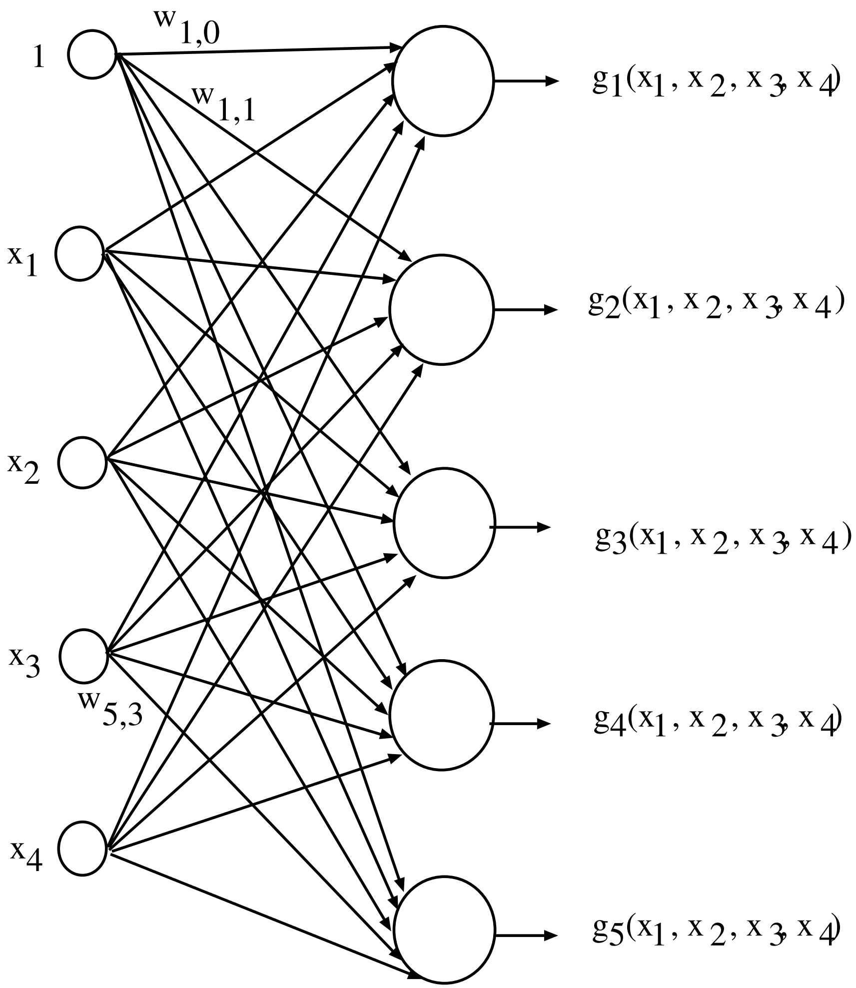
Linear Discriminant Functions

- Graphical representation of a LDF. Given an input $\mathbf{x} \in \mathbb{R}^D$

$$\hat{c} = G(\mathbf{x}) \equiv \underset{1 \leq c \leq C}{\text{argmax}} \quad g_c(\mathbf{x})$$

$$g_c(\mathbf{x}) = \mathbf{w}_c^t \mathbf{x} + w_{c0}$$

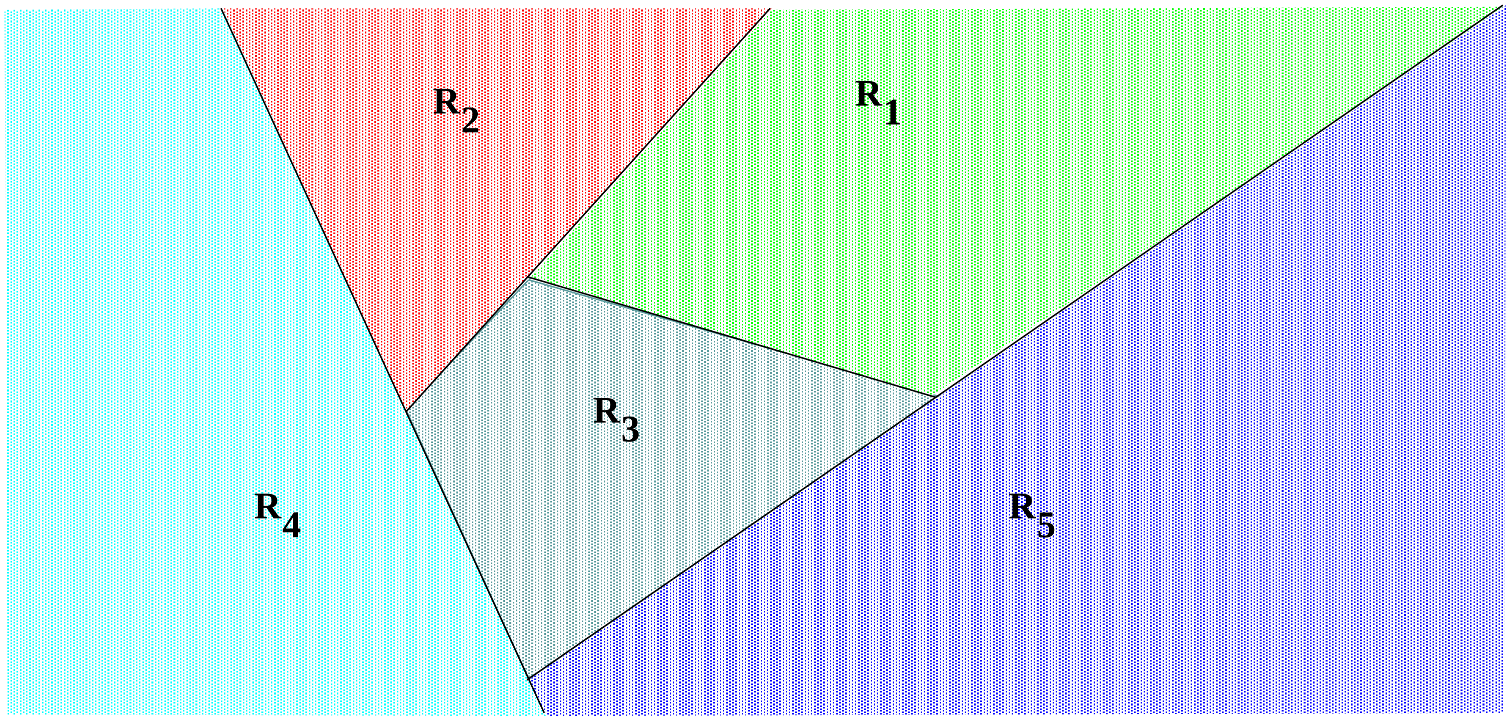
  – $E = \mathbb{R}^4$,
  – Classes$=\{1, 2, 3, 4, 5\}$



Decision surface of a LDF: Hyperplanes

$$H_{ij} = \{\mathbf{x} \mid g_i(\mathbf{x}) = g_j(\mathbf{x})\}$$

$$\mathbf{x} : \sum_{k=1}^{D} w_{ik}\ x_k + w_{i0} = \sum_{k=1}^{D} w_{jk}\ x_k + w_{j0} \Rightarrow \sum_{k=1}^{D} (w_{ik} - w_{jk})\ x_k + (w_{i0} - w_{j0}) = 0$$



The problem of two classes

$$g_1, g_2 : \mathbb{R}^D \to \mathbb{R}$$

$$G(\mathbf{x}) = \begin{cases} 1 & \text{if } g_1(\mathbf{x}) > g_2(\mathbf{x}) \\ 2 & \text{if } g_1(\mathbf{x}) < g_2(\mathbf{x}) \end{cases}$$

Simplification:

$$g(\mathbf{x}) = g_1(\mathbf{x}) - g_2(\mathbf{x})$$

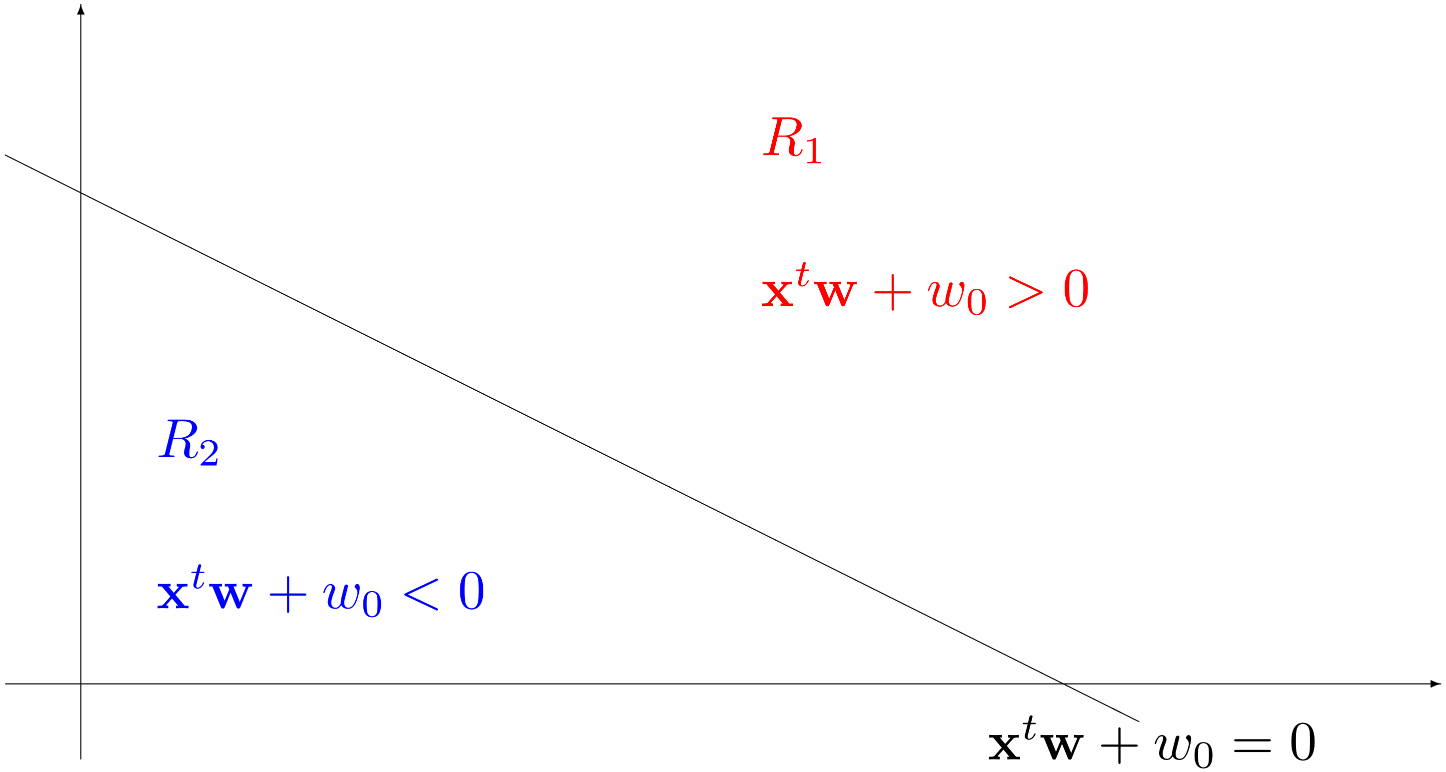
$$G(\mathbf{x}) = \begin{cases} 1\ (+1) & \text{if } g(\mathbf{x}) > 0 \\ 2\ (-1) & \text{if } g(\mathbf{x}) < 0 \end{cases}$$

$$g(\mathbf{x}) = \mathbf{x}^t\,(\mathbf{w}_1 - \mathbf{w}_2) + (w_{10} - w_{20}) \;=\; \mathbf{x}^t\mathbf{w} + w_0$$

Properties of the linear discriminant functions (2 classes)

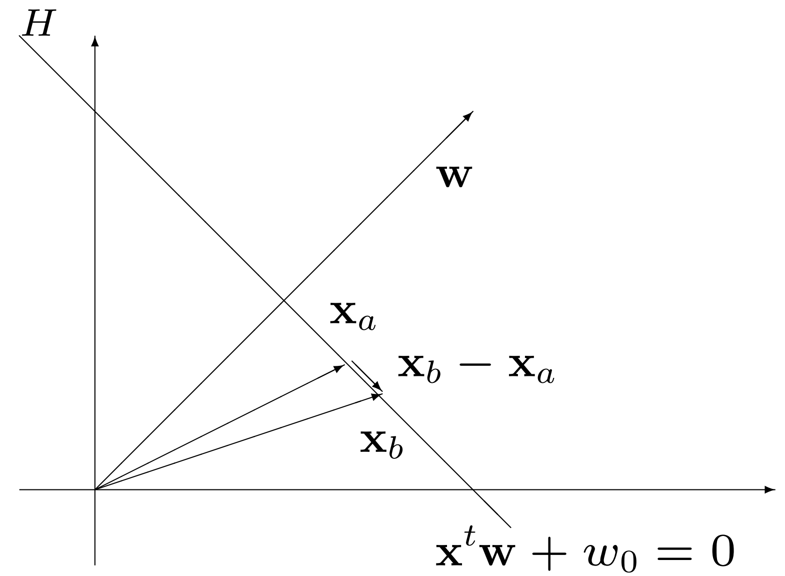
1. A LDF divides $\mathbb{R}^D$ in two semi-planes.
2. If $H = \{\mathbf{x} \mid g(\mathbf{x}) = 0\}$, H is orthogonal to $\mathbf{w}$.
3. If $w_0 > 0$, then the origin is in the positive part of H. If $w_0 < 0$, then the origin is in the negative part of H. If $w_0 = 0$, then H passes through the origin.
4. $r_{\mathbf{x}} = \dfrac{g(\mathbf{x})}{||\mathbf{w}||}$ is the distance of $\mathbf{x}$ to H.
5. The distance from the origin of coordinates to H is $w_0/||\mathbf{w}||$.
6. If $\gamma \in \mathbb{R}^+$, then $\gamma\ g(\cdot)$ and $g(\cdot)$ represents the same hyperplane of decision.

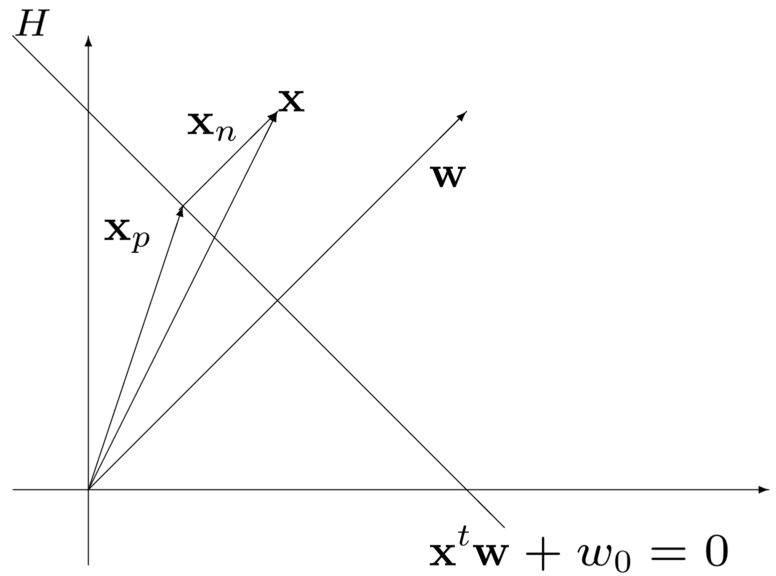
The problem of two classes



Properties of the linear discriminant functions

If $H = \{\mathbf{x} \mid g(\mathbf{x}) = 0\}$, H is orthogonal to $\mathbf{w}$ and
$g(\mathbf{x}) = r_{\mathbf{x}} ||\mathbf{w}||$, where $r_{\mathbf{x}}$ is the distance of $\mathbf{x}$ to H.





$$
\begin{aligned}
\mathbf{x}_a, \mathbf{x}_b \in H \quad &\Rightarrow \quad g(\mathbf{x}_a) = g(\mathbf{x}_b) \\
&\Rightarrow \quad \mathbf{w}^t(\mathbf{x}_b - \mathbf{x}_a) = 0 \\
&\Rightarrow \quad \mathbf{w} \perp H
\end{aligned}
$$

$$
\begin{aligned}
\mathbf{x} = \mathbf{x}_p + \mathbf{x}_n \quad &\Rightarrow \quad \mathbf{x} = \mathbf{x}_p + r\,\frac{\mathbf{w}}{||\mathbf{w}||} \\
&\Rightarrow \quad g(\mathbf{x}) = w_0 + \mathbf{w}^t\mathbf{x} \\
&\qquad = w_0 + \mathbf{w}^t\mathbf{x}_p + r\,\frac{||\mathbf{w}||^2}{||\mathbf{w}||} \\
&\qquad = r\,||\mathbf{w}||
\end{aligned}
$$

Properties of the linear discriminant functions

If $\gamma \in \mathbb{R}^+$, then $\gamma\ g$ and g represents the same decision hyperplane.

- $\gamma\ \mathbf{w}$ and $\mathbf{w}$ is in the same direction
- $H_{\gamma\, g}$ and H_g are at the same distance of the origin of coordinates:

$$\frac{\gamma\ w_0}{||\gamma\ \mathbf{w}||} = \frac{w_0}{||\mathbf{w}||}$$

LDFs learning, problems of two classes

- Let $X = \{(\mathbf{x}_1, c_1), \ldots, (\mathbf{x}_N, c_N)\}$, $\mathbf{x}_n \in \mathbb{R}^D$, $c_n \in \{+1, -1\}$ be a training set. X is linear separable (LS) if $\exists\, \mathbf{w} \in \mathbb{R}^D$ and $w_0 \in \mathbb{R}$ such that:

$$\forall n,\ 1 \leq n \leq N, \quad \mathbf{w}^t\mathbf{x}_n + w_0 \begin{cases} \geq 0 & \text{if } c_n = +1 \\ < 0 & \text{if } c_n = -1 \end{cases}; \quad \text{that is,} \quad c_n\,(\mathbf{w}^t\mathbf{x}_n + w_0) \geq 0$$

- Learning: Given X, search for $\hat{\mathbf{w}}$ solves the system of N inequalities:

$$c_n\,(\mathbf{w}^t\mathbf{x}_n + w_0) \geq 0, \quad 1 \leq n \leq N$$

- Alternatively: Given X and a margin b, solves the system of N inequalities:

$$c_n\,(\mathbf{w}^t\mathbf{x}_n + w_0) \geq b, \quad 1 \leq n \leq N$$

- Equivalently: Given X and a margin b, search for $\hat{\mathbf{w}}$ minimizes:

$$q_X(\mathbf{w}, w_0) \;=\; \sum_{\substack{(\mathbf{x},c) \in X \\ c\,(\mathbf{w}^t\mathbf{x} + w_0) < b}} -\,c\,(\mathbf{w}^t\mathbf{x} + w_0)$$

- Solution: Applying gradient descent to $q_X(\mathbf{w}, w_0)$

LDFs learning, Perceptron algorithm for problems of two classes

// Input: $\mathbf{w}$ a vector with initial weights, and w_0 an initial threshold
// $(\mathbf{x}_1, c_1), \ldots, (\mathbf{x}_N, c_N)$ with $\mathbf{x}_n \in \mathbb{R}^D$, $c_n \in \{-1, +1\}$ for $1 \leq n \leq N$;
// $\alpha \in \mathbb{R}^{>0}$, the "learning factor";
// $b \in \mathbb{R}^{>0}$, the "margin" (to control the convergence)
// Output: $\mathbf{w}$ and w_0 after the convergence
// Gradient descent on a function of the classification errors by $\mathbf{w}$ and w_0

repeat {
  error $= 0$;
  for all (n, $1 \leq n \leq N$ in a random way) {
    $g = c_n \ (\mathbf{w}^t \mathbf{x}_n + w_0)$;
    if $(g < b)$ {
      $\mathbf{w} = \mathbf{w} + \alpha\, c_n\, \mathbf{x}_n$; $w_0 = w_0 + \alpha\, c_n$;
      error $++$;
    }
  }
} **until** (error $= 0$)

The final weights and threshold are: $\widehat{\mathbf{w}} = \sum_{n=1}^{N} \beta_n \ c_n \ \mathbf{x}_n$ and $\widehat{w_0} = \sum_{n=1}^{N} \beta_n \ c_n$

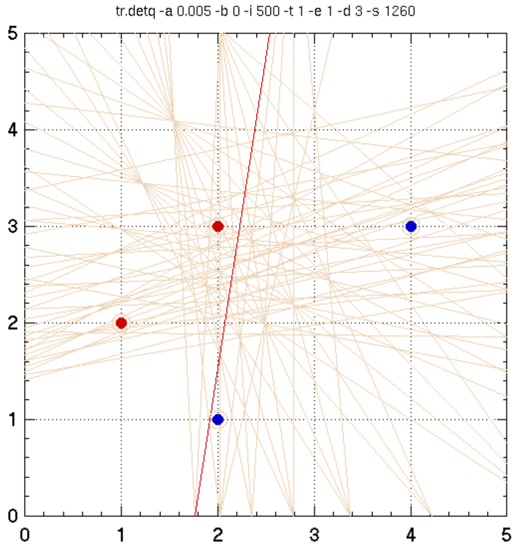
LDFs learning, Perceptron algorithm for C classes

// Let be: $\mathbf{w}_j,\ 1 \leq j \leq C,\ C$ vectors with initial weights;
// $(\mathbf{x}_1, c_1), \ldots, (\mathbf{x}_N, c_N),\ N$ learning samples;
// $\alpha \in \mathbb{R}^{>0}$, the "learning factor";
// $b \in \mathbb{R}$, the "margin" (to control the convergence)

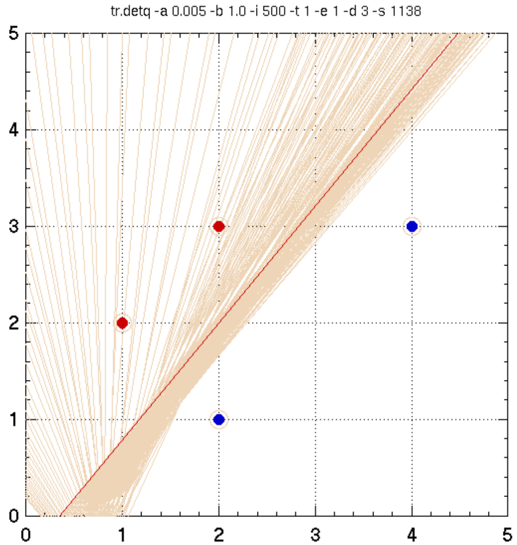
error $=$ *true*
while (error) {
  for $(n = 1;\ n \leq N;\ n{+}{+})$ {
    $i = c_n$; $g = \mathbf{w}_i^t \mathbf{x}_n + w_{i0}$; error=*false*
    for $(j = 1;\ j \leq C,\ j \neq i;\ j{+}{+})$ {
      if $(\mathbf{w}_j^t \mathbf{x}_n + w_{j0} + b > g)$ {
        $\mathbf{w}_j = \mathbf{w}_j - \alpha\, \mathbf{x}_n$; $w_{j0} = w_{j0} - \alpha$; error=*true*
      }
    if (error) $\mathbf{w}_i = \mathbf{w}_i + \alpha\, \mathbf{x}_n$; $w_{i0} = w_{i0} + \alpha$;
  }
}

In case of an error the algorithm modifies the weights of the correct class and the weights of the incorrect class/es: $(\mathbf{w}_j^t \mathbf{x}_n + w_{j0} + b > g)$.

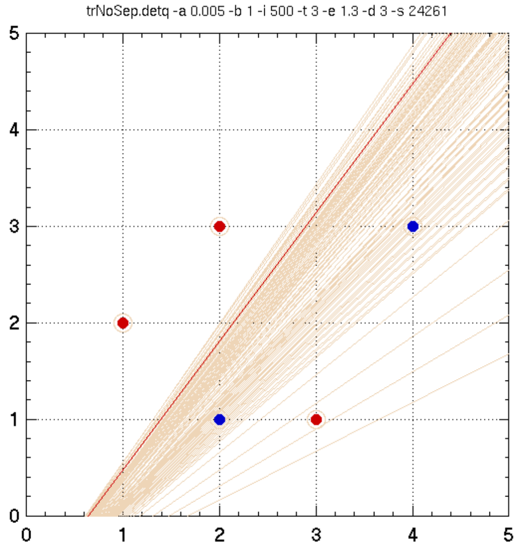
LDFs learning, Perceptron algorithm



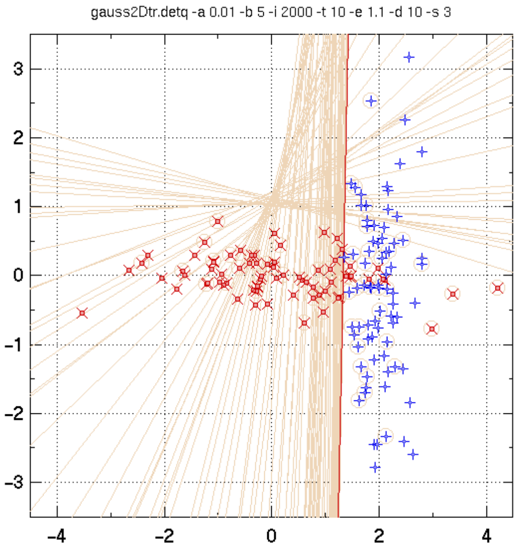
2 classes, margin 0



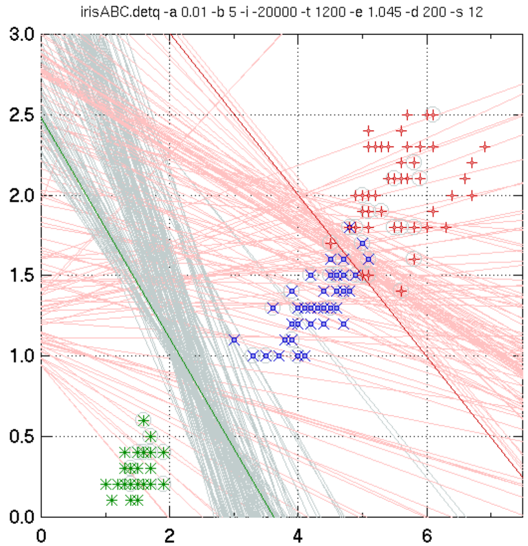
2 classes, margin > 0



2 classes, margin > 0 non-separable



2 gaussian classes



3 classes (iris)

Index

1 Introduction ▷ 3

2 Classifiers and discriminant functions ▷ 8

3 Linear discriminant functions ▷ 14

◦ 4 *Generalized linear discriminant functions and kernels* ▷ 28

5 Maximum margin classifiers ▷ 45

6 Bibliography ▷ 54

Introduction

- **Linear discriminant functions**. For $1 \leq c \leq C$:

$$g_c(\mathbf{x}) = \sum_{k=1}^{D} \mathbf{w}_{ck} x_k + \mathbf{w}_{c0}$$

- **Quadratic discriminant functions**. For $1 \leq c \leq C$:

$$g_c(\mathbf{x}) = \sum_{k_1=1}^{D} \sum_{k_2=1}^{D} \mathbf{w}_{ck_1k_2} x_{k_1} x_{k_2} + \sum_{k=1}^{D} \mathbf{w}_{ck} x_k + \mathbf{w}_{c0}$$

- **Polynomial discriminant functions**. For $1 \leq c \leq C$:

$$g_c(\mathbf{x}) = \sum_{k_1,\dots,k_p} \mathbf{w}_{ck_1\dots k_p} x_{k_1} \dots x_{k_P} + \dots + \sum_{k_1,k_2} \mathbf{w}_{ck_1k_2} x_{k_1} x_{k_2} + \sum_{k=1} \mathbf{w}_{ck} x_k + \mathbf{w}_{c0}$$

Generalized linear discriminant functions

- **Generalized linear discriminant functions** (GLDF) For $1 \leq c \leq C$

$$g_c(\mathbf{x}) = \sum_{k=1}^{D'} w_{ck}\Phi_k(\mathbf{x}) + w_{c0} = \mathbf{w}_c^t \mathbf{\Phi}(\mathbf{x}) + w_{c0}$$

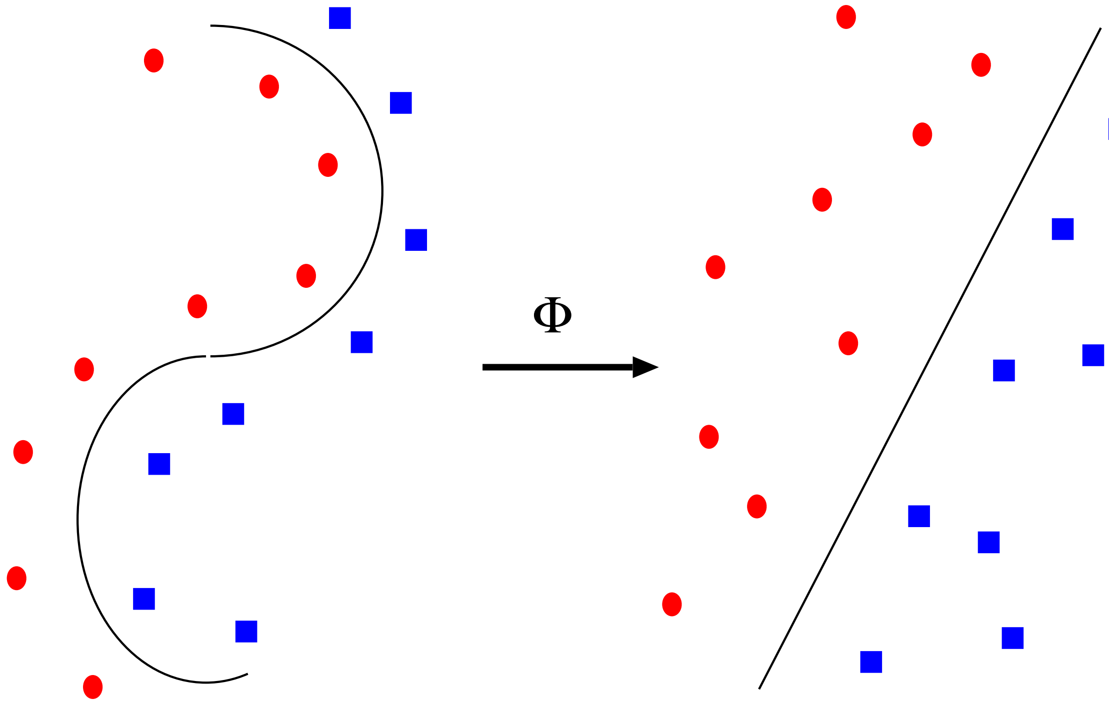
- For simplification, $\Phi_0(\mathbf{x}) = 1$, therefore $g_c(\mathbf{x};\theta) = \sum_{k=0}^{D'} w_{ck}\Phi_k(\mathbf{x})$

- Feature space mapping through a non-linear function $\mathbf{\Phi} : \mathbb{R}^D \rightarrow \mathbb{R}^{\hat{D}}$ $(\hat{D} = D' + 1)$

- Typically $\hat{D} \geq D$: Dimensionality problem.

Generalized linear discriminant functions

Transformation from a non-linear separability to a linear separability

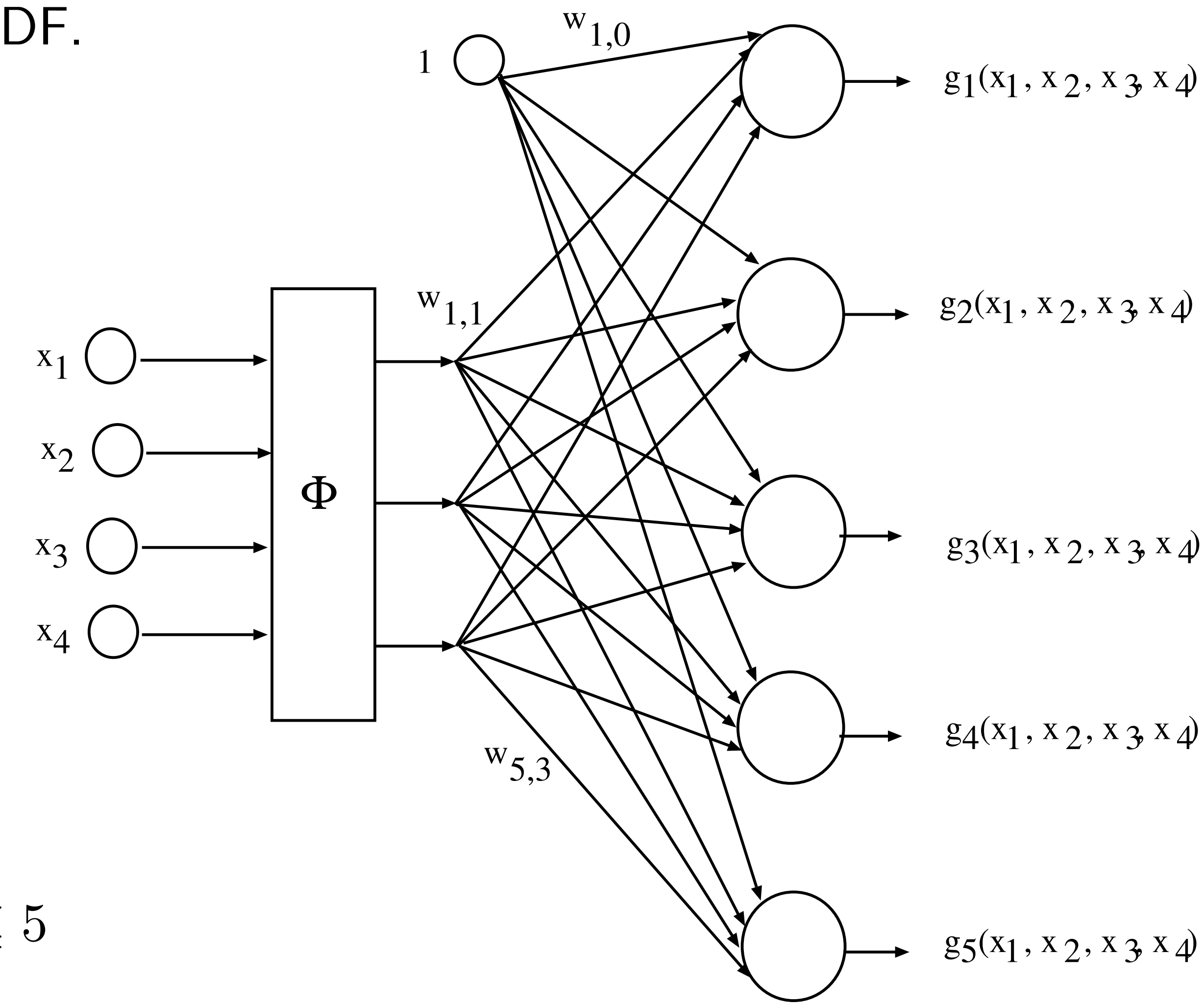


Generalized linear discriminant functions

- Graphical representation of a GLDF.
  Given an input $\mathbf{x} \in \mathbb{R}^D$

$$\hat{c} = G(\mathbf{x}) \equiv \underset{1 \leq c \leq C}{\text{argmax}} \quad g_c(\mathbf{x})$$

$$g_c(\mathbf{x}) = \mathbf{w}_c^t \Phi(\mathbf{x}) + w_{c0}$$



  - $E = \mathbb{R}^4$,
  - $\Phi : E \to \mathbb{R}^3$
  - Classes$=\{1, 2, 3, 4, 5\}$
  - $\mathbf{w}_c \in \mathbb{R}^3$, $w_{c0} \in \mathbb{R}$ for $1 \leq c \leq 5$

Binary classification and kernels

- The Perceptron algorithm (and others) defines a LDF

$$g(\mathbf{x}) = \mathbf{w}^t\mathbf{x} + w_0 = \sum_{n=1}^{N} \beta_n \; c_n \; \mathbf{x}_n^t\mathbf{x} + \sum_{n=1}^{N} \beta_n \; c_n$$

- If the training set is not linearly separable, we need to change the representation space to obtain linear separability.

- We obtain an implicit change of the representation space using the *kernel trick*:

$$g(\mathbf{x}) = \sum_{n=1}^{N} \beta_n \; c_n \; \Phi(\mathbf{x}_n)^t\Phi(\mathbf{x}) + \sum_{n=1}^{N} \beta_n \; c_n = \sum_{n=1}^{N} \beta_n \; c_n \; K(\mathbf{x}_n, \mathbf{x}) + \sum_{n=1}^{N} \beta_n \; c_n$$

Binary classification and Kernels

- We define a kernel function as:

$$K : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$$

Such that:

$$K(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x})^t \Phi(\mathbf{y})$$

- But this alternative representation Φ is not performed, we just need to know the scalar product on this new representation space.

Kernels: an example

Let $\mathbf{x} = (x_1, x_2, x_3)^t$, $\mathbf{y} = (y_1, y_2, y_3)^t$ and $K : \mathbb{R}^3 \times \mathbb{R}^3 \rightarrow \mathbb{R}$ defined as:

$$K(\mathbf{x}, \mathbf{y}) \;=\; (x_1\ y_1 + x_2\ y_2 + x_3\ y_3)^2$$

Is $K(\mathbf{x}, \mathbf{y})$ a kernel? . . . *Yes*:

$$\begin{aligned} K(\mathbf{x}, \mathbf{y}) &= (x_1\ y_1 + x_2\ y_2 + x_3\ y_3)^2 \\ &= x_1^2\ y_1^2 + x_2^2\ y_2^2 + x_3^2\ y_3^2 + 2\ x_1\ y_1\ x_2\ y_2 + 2\ x_1\ y_1\ x_3\ y_3 + 2\ x_2\ y_2\ x_3\ y_3 \end{aligned}$$

$K(\mathbf{x}, \mathbf{y}) \;=\; \Phi(\mathbf{x})^t \Phi(\mathbf{y})$ si $\Phi : \mathbb{R}^3 \rightarrow \mathbb{R}^6$ is defined as:

$$\begin{aligned} \Phi(\mathbf{x}) &= (x_1^2,\ x_2^2,\ x_3^2,\ \sqrt{2}\ x_1 x_2,\ \sqrt{2}\ x_1 x_3,\ \sqrt{2}\ x_2 x_3)^t \\ \Phi(\mathbf{y}) &= (y_1^2,\ y_2^2,\ y_3^2,\ \sqrt{2}\, y_1 y_2,\ \sqrt{2}\ y_1 y_3,\ \sqrt{2}\ y_2 y_3)^t \end{aligned}$$

Alternatives for computing $K(\mathbf{x}, \mathbf{y})$:

- *Direct* in $\mathbb{R}^3$, using $K(\mathbf{x}, \mathbf{y})$: $3 + 2 + 1 = 6$ products + additions
- Compute first $\Phi(\mathbf{x})$, $\Phi(\mathbf{y})$ in $\mathbb{R}^6$ and compute next $\Phi(\mathbf{x})^t \Phi(\mathbf{y})$:
  $2 \cdot 6 + 6 + 5 = 23$ products + additions

Learning - Kernel Perceptron

- The Kernel Perceptron algorithm learns the following function:

$$g(\mathbf{x}) = \sum_{n=1}^{N} \beta_n \; c_n \; K(\mathbf{x}_n, \mathbf{x}) + \sum_{n=1}^{N} \beta_n \; c_n$$

- The parameters lo learn are β_n for $1 \leq n \leq N$

- In this learning stage the kernel function is just a matrix K with all the pairs $K(\mathbf{x}_i, \mathbf{x}_j)$

Learning - Kernel Perceptron

// Input: $(\mathbf{x}_1, c_1), \ldots, (\mathbf{x}_N, c_N)$ with $\mathbf{x}_n \in \mathbb{R}^D$, $c_n \in \{-1, +1\}$ for $1 \leq n \leq N$;
// Output: β_n for $1 \leq n \leq N$

$\beta_n = 0$ for $1 \leq n \leq N$**; ;**
repeat
  error $= 0$**;**
  for all $(n,\ 1 \leq n \leq N$ in a random way) **{**
    $g = c_n \ \left(\sum_{n'=1}^{N} \beta_{n'}\ c_{n'}\ K(\mathbf{x}_{n'}, \mathbf{x}_n) + \sum_{n'=1}^{N} \beta_{n'}\ c_{n'}\right)$**;**
    if $(g < b)$ **{**
      $\beta_n + +;$
      error $+ +$**;**
    }
  }
} **until** (error $= 0$)

Exercise

- Given the following Kernel matrix:

$$K = \begin{bmatrix} 1 & 1/3 & 1/3 & 1/3 & 1/5 \\ 1/3 & 1 & 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1 & 1/5 & 1/3 \\ 1/3 & 1/3 & 1/5 & 1 & 1/3 \\ 1/5 & 1/3 & 1/3 & 1/3 & 1 \end{bmatrix}$$

where $X = \{(\mathbf{x}_1, +1), (\mathbf{x}_2, -1), (\mathbf{x}_3, +1), (\mathbf{x}_4, -1), (\mathbf{x}_5, +1)\}$

- Obtain the β_i with the Kernel Perceptron algorithm and $b = 1$

Most used Kernels

- A polinomial kernel is: $K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + c)^d$

- The gaussian kernel is: $K(\mathbf{x}, \mathbf{y}) = \exp\left(\frac{-||\mathbf{x}-\mathbf{y}||^2}{2\sigma^2}\right)$

  In an infinity dimension space the training samples are linearly separables

Generalized kernels

- How to obtain new kernels functions

- It is necessary to demostrate that:

$$\exists \Phi : \mathbb{R}^D \to \mathbb{R}^{D'}\text{: } K(\mathbf{y}, \mathbf{x}) = \Phi(\mathbf{y})\Phi(\mathbf{x})$$

- **Mercer condition**: a necessary and suficient condition for K to be a valid kernel is that the Gramm matrix: $K(\mathbf{x}_i, \mathbf{x}_j)$ has to be positive-semidefinite for all possible pair of the training set $\{\mathbf{x}_1, \dots, \mathbf{x}_n\}$.

- A matrix K is positive-semidefinite if:

$$\mathbf{x}^t K \mathbf{x} \geq 0, \ \forall \mathbf{x} \in \mathbb{R}^D$$

Generalized Kernels

If K_1 and K_2 are kernels, then K is a kernel:

$$K(\mathbf{x},\mathbf{y}) = \begin{cases} c \cdot K_1(\mathbf{x},\mathbf{y}) & c > 0 \\ f(\mathbf{x}) \cdot K_1(\mathbf{x},\mathbf{y}) \cdot f(\mathbf{y}) & \text{for any function } f \\ q(K_1(\mathbf{x},\mathbf{y})) & q \text{ a polinomial with non-negative coeficients} \\ (c + K_1(\mathbf{x},\mathbf{y}))^d & d, c > 0 \\ K_1(\mathbf{x},\mathbf{y}) + K_2(\mathbf{x},\mathbf{y}) & \\ K_1(\mathbf{x},\mathbf{y}) \times K_2(\mathbf{x},\mathbf{y}) & \\ \exp\left(K_1(\mathbf{x},\mathbf{y})\right) & \\ \frac{K_1(\mathbf{x},\mathbf{y})}{\sqrt{K_2(\mathbf{x},\mathbf{y})}} & \end{cases}$$

Exercise

Given the following kernel function: $K(\mathbf{x}, \mathbf{y}) = (\mathbf{x} \cdot \mathbf{y} + 1)^2$ and the following training set $X = \{(\mathbf{x}_1, +1), (\mathbf{x}_2, +1), (\mathbf{x}_3, +1), (\mathbf{x}_4, -1), (\mathbf{x}_5, -1), (\mathbf{x}_6, -1)\}$ with:

$$\mathbf{x}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \mathbf{x}_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \mathbf{x}_3 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad \mathbf{x}_4 = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \quad \mathbf{x}_5 = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \quad \mathbf{x}_6 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

a) Compute the Kernel matrix K from the training set

b) Perform one iteration (from $\mathbf{x}_1$ to $\mathbf{x}_6$) of the Kernel Perceptron algorithm

c) Clasify the new test sample $\mathbf{x} = (0, 2)^t$ with the weights β obtained in the previous step

Radial basis function networks

- Given a vector $\boldsymbol{\mu}$, a **radial basis function** (RBF) is $\phi_{\boldsymbol{\mu}}(\mathbf{x}) = \phi(\| \mathbf{x} - \boldsymbol{\mu} \|)$

Some type of function:
  - Gaussian function: $\phi(r) = \exp\left(-\frac{r^2}{2\sigma^2}\right)$
  - Quadratic function: $\phi(r) = (r^2 + \sigma^2)^{1/2}$
  - "Thin-plane-spline" function: $\phi(r) = r^2 \log r$

- RBFs allow to define GLDF as: Given M vectors $\boldsymbol{\mu}_m \in \mathbb{R}^D$, $1 \leq m \leq M$

$$g_k(\mathbf{x}; \theta) = \sum_{m=1}^{M} w_{k,m}\ \phi(\| \mathbf{x} - \boldsymbol{\mu}_m \|) = \mathbf{w}_k^t \Phi(\mathbf{x}) \qquad 1 \leq k \leq C$$

- **Problem**: Given $A = \left\{(\mathbf{x}_n, \mathbf{t}_n) \mid \mathbf{x}_n \in \mathbb{R}^D, \mathbf{t}_n \in \mathbb{R}^C\right\}_{1 \leq n \leq N}$ search for vectors $\mathbf{w}_c \in \mathbb{R}^N$ $(1 \leq c \leq C)$ and vectors $\boldsymbol{\mu}_k \in \mathbb{R}^d$ $(1 \leq k \leq M)$ such that $\mathbf{g}(\mathbf{x}_n; \theta) = \mathbf{t}_n$ for $1 \leq n \leq N$.

Learning with radial basis function networks

Given $A = \{(\mathbf{x}_1, \mathbf{t}_1), \ldots, (\mathbf{x}_n, \mathbf{t}_n)\}$, with $\mathbf{x}_i \in \mathbb{R}^D$, $\mathbf{t}_i \in \mathbb{R}^C$,

- **Sequential learning** of the radial basis functions ($\boldsymbol{\mu}_k$ and $\boldsymbol{\sigma}_k$) and the weight ($\mathbf{w}_y$):
  1. Learning radial basis functions ϕ from $A' = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$,
     - using *clustering*
     - using *mixtures of gaussians*
     - using *autoorganizative maps*
  2. Learning weights $\mathbf{w}$ from $A" = \{(\phi(\mathbf{x}_1), \mathbf{t}_1), \ldots, (\phi(\mathbf{x}_N), \mathbf{t}_N)\}$,
     - Perceptron
     - Pocket Perceptron
     - Widrow-Hoff

- **Integrate learning** of radial basis function and the weights by minimizing the mean squared error.

Index

1 Introduction ▷ 3

2 Classifiers and discriminant functions ▷ 8

3 Linear discriminant functions ▷ 14

4 Generalized linear discriminant functions and kernels ▷ 28

∘ 5 *Maximum margin classifiers* ▷ 45

6 Bibliography ▷ 54

Introduction

- Motivation: find linear classifiers with good generalization
- Lagrange optimization problem
- Convex optimization problem
- Extension to kernel to deal with non-linear boundaries
- Soft-constrainst to avoid complex boundaries

Maximum margin classifiers

A *canonical LDF* w.r.t. a set X of N samples is $\mathbf{w} \equiv (\mathbf{w}, w_0)$, such that

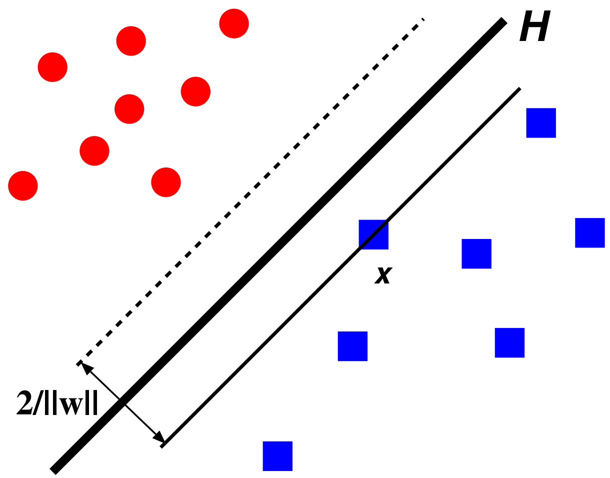
$$\min_{1 \leq n \leq N} |\, g(\mathbf{x}_n; \mathbf{w}) \,| \quad = \quad \min_{1 \leq n \leq N} |\mathbf{w}^t \mathbf{x}_n + w_0\,| \quad = \quad 1$$

The distance r of the nearest $\mathbf{x} \in X$ to H is:

$$r \;=\; \frac{|\,\mathbf{w}^t \mathbf{x} + w_0\,|}{\|\mathbf{w}\|} \;=\; \frac{1}{\|\mathbf{w}\|}$$

And the *margin* of H with respect to X is:

$$2r \;=\; \frac{2}{\|\mathbf{w}\|}$$



Given a (linearly separable) training data set: $X = \{(\mathbf{x}_1, c_1), \cdots, (\mathbf{x}_N, c_N)\}$, $\mathbf{x}_i \in \mathbb{R}^d$ and $c_i \in \{-1, +1\}$, find a canonical LDF w.r.t to X that classify correctly all the samples with maximum margin:

- **maximize $\frac{2}{||\mathbf{w}||}$**
- **subject to $c_i(\mathbf{w}\mathbf{x}_i + w_0) \geq 1,$ for all $1 \leq i \leq N$**

Support Vector Machines. Linear SVM

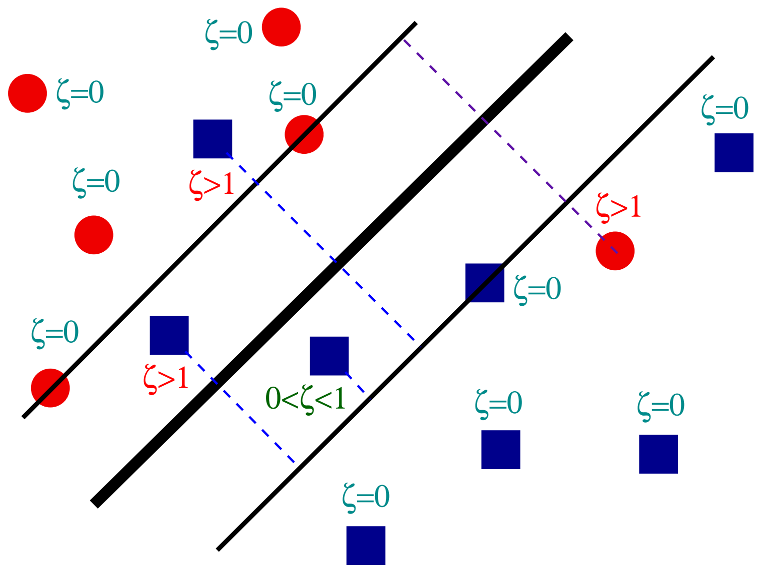
- Alternatively, we can formulated the previous optimization problem as: Given a LS set X of N samples:
  - minimize $\frac{1}{2}\|\mathbf{w}\|^2$
  - subject to $c_i(\mathbf{w}\mathbf{x}_i + w_0) \geq 1,$ for all $1 \leq i \leq N$
- By using the Lagrange multipliers method, it is necessary to solve the Dual Lagrange optimization problem (α_i for $1 \leq i \leq N$ are known as Langrange multipliers):
  - maximize $\Lambda_D(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j c_i c_j \mathbf{x}_i^T \mathbf{x}_j$
  - subject to $\sum_{i=1}^{N} \alpha_i c_i = 0$ and $\alpha_i \geq 0$ for $1 \leq i \leq N$

  This problem can now be solved by standard quadratic programming techniques.
- Support vectors $S \subset X$: The subset of samples $(\mathbf{x}_n, c_n) \in X$ s.t. $\alpha_n^* \neq 0$
- The solution is $g_c(\mathbf{x}) = \sum_{(\mathbf{x}_n, c_n) \in S} \alpha_n^* c_n \mathbf{x}_n^t \mathbf{x} + w_0$ with $w_0 = c_n - \mathbf{w}^{*t} \mathbf{x}_n$, for a n s.t. $0 < \alpha_n^*$

Support Vector Machines. Soft Margin

- Idea: to allow mislabeled examples
- Slack variables:

  The method introduces non-negative slack variables, ξ_i, which measure the degree of misclassification of the data x_i: $c_i(\mathbf{w}\mathbf{x}_i + w_0) \geq 1 - \xi_i$



- The objective function is then increased by a function which penalizes non-zero ξ_i:
  - minimize $\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i$
  - subject to $c_i(\mathbf{w}\mathbf{x}_i + w_0) \geq 1 - \xi_i$ and $\xi_i \geq 0$, for all $1 \leq i \leq N$

Support Vector Machines. Soft Margin

- By using the Lagrange multipliers method, it is necessary to solve the Dual Lagrange optimization problem (α_i for $1 \leq i \leq N$ are known as Langrange multipliers):
  - maximize $\Lambda_D(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j c_i c_j \mathbf{x}_i^T \mathbf{x}_j$
  - subject to $\sum_{i=1}^{N} \alpha_i c_i = 0$ and $C \geq \alpha_i \geq 0$ for $1 \leq i \leq N$

  This problem can now be solved by standard quadratic programming techniques.

- Support vectors $S \subset X$: The subset of samples $(\mathbf{x}_n, c_n) \in X$ s.t. $\alpha_n^* \neq 0$
- The solution is $g_c(\mathbf{x}) = \sum_{(\mathbf{x}_n, c_n) \in S} \alpha_n^* c_n \mathbf{x}_n^t \mathbf{x} + w_0$ with $w_0 = c_n - \mathbf{w}^{*t} \mathbf{x}_n$, for a n s.t. $0 < \alpha_n^* < C$

Exercise

- Given a training set:

$$S = \{((1,4),+1),((2,2),+1),((2,3),+1),((4,2),+1),((3,4),-1),((3,5),-1),$$
$$((5,4),-1),((5,6),-1),((4,4),+1),((4,3),-1)\},$$

the optimal Lagrange multipliers α_n^* with $C = 1000$ obtained are:

$$[250.87, 0.0, 0.0, 500.75, 751.62, 0.0, 0.0, 0.0, 1000.0, 1000.0]$$

Write the corresponding linear discriminant function, the slack variables ξ_n^* and classify the sample $(4,5)$.

Support Vector Machines. Kernel SVM

- Kernel extension:

  Maximize,

  $$\Lambda_D(\boldsymbol{\alpha}) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j c_i c_j K(\mathbf{x}_i, \mathbf{x}_j) \text{ s.t. } \textstyle\sum_{i=1}^{n} \alpha_i c_i = 0 \text{ and } \alpha_i \geq 0$$

  where $K(\cdot, \cdot)$ is a kernel function

- The solution is

  $$g_c(\mathbf{x}) = \sum_{(\mathbf{x}_n, c_n) \in S} \alpha_n^* c_n K(\mathbf{x}_n, \mathbf{x}) + w_0$$

  with $w_0 = c_n - \sum_{m=1}^{N} \alpha_m^* c_m K(\mathbf{x}_m, \mathbf{x}_n)$: for a n s.t. $0 < \alpha_n^* < C$

- The parameter selection of C and the Kernel function is carried out by cross-validation

Support Vector Machines for C classes

- One-against-one plus DAG: $C(C-1)/2$ classifiers; classification directed by a DAG (direct acyclic graph)

- One-against-rest: C discriminant functions.

- SVM$^{\text{multiC}}$: Direct optimization in C classes [Cramer & Singer, 01]

- Kesler construction: Transform a problem of C classes into another of 2 classes (with high dimension). [Duda & Hart, 73], [Franc & Hlaváč, 02]

Index

1 Introduction ▷ 3

2 Classifiers and discriminant functions ▷ 8

3 Linear discriminant functions ▷ 14

4 Generalized linear discriminant functions and kernels ▷ 28

5 Maximum margin classifiers ▷ 45

◦ 6 *Bibliography* ▷ 54

Recommended bibliography

- Christopher M. Bishop. 2006. Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag New York, Inc., Secaucus, NJ, USA.

- Christopher M. Bishop. 1995. Neural Networks for Pattern Recognition. Oxford University Press, Inc., New York, NY, USA.