



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

Escola Tècnica Superior d'Enginyeria Informàtica



Tema 6. Representación basada en Kernels y LDA

Percepción (PER)

Curso 2020/2021

Departamento de Sistemas Informáticos y Computación

Índice

- 1 Introducción ▷ 3
- 2 Clasificación binaria y kernels ▷ 6
- 3 Aprendizaje - Kernel Perceptron ▷ 12
- 4 Tipos de Kernel ▷ 16
- 5 Kernels generalizados ▷ 21
- 6 *Linear Discriminant Analysis* (LDA) ▷ 25

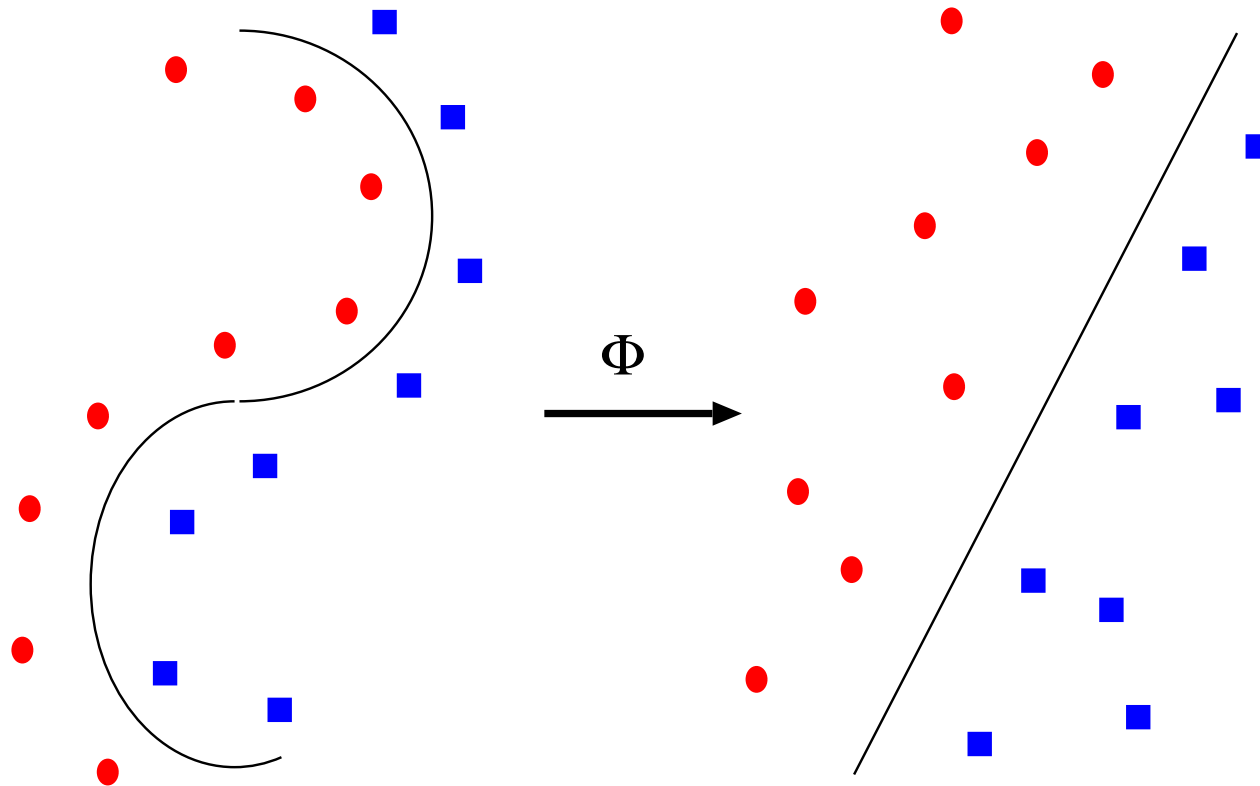
Índice

- 1 *Introducción* ▷ 3
- 2 Clasificación binaria y kernels ▷ 6
- 3 Aprendizaje - Kernel Perceptron ▷ 12
- 4 Tipos de Kernel ▷ 16
- 5 Kernels generalizados ▷ 21
- 6 *Linear Discriminant Analysis* (LDA) ▷ 25

Introducción

Objetivo principal de la representación basada en kernels:

Cambio de espacio de representación para obtener separabilidad lineal



Introducción

Problema principal de la reducción por PCA: *no supervisada*

La técnica de LDA sí tiene en cuenta las etiquetas de clase

LDA se basa en vectores propios generalizados:

- Definición: dadas dos matrices W y V encontrar aquellos vectores y escalares que son solución de la siguiente expresión:

$$W^t \mathbf{x} = \lambda V^t \mathbf{x}$$

- Los posibles valores propios deben de satisfacer la ecuación:

$$\det(W^t - \lambda V^t) = 0$$

- El problema original se podría reescribir como un problema de vectores propios usual:

$$(V^t)^{-1} W^t \mathbf{x} = \lambda \mathbf{x}$$

- En la práctica, por estabilidad numérica, se resuelve el problema de vectores propios generalizados en lugar de invertir la matriz V^t

Índice

- 1 Introducción ▷ 3
- 2 *Clasificación binaria y kernels* ▷ 6
- 3 Aprendizaje - Kernel Perceptron ▷ 12
- 4 Tipos de Kernel ▷ 16
- 5 Kernels generalizados ▷ 21
- 6 *Linear Discriminant Analysis* (LDA) ▷ 25

Clasificación binaria y kernels

- Usualmente se estudian los métodos kernel en problemas de clasificación binarios
- Conjunto de entrenamiento

$$X = \{(\mathbf{x}_1, c_1), (\mathbf{x}_2, c_2), \dots, (\mathbf{x}_N, c_N)\} \quad \text{con} \quad c_i \in \{-1, +1\}$$

- La clasificación de una nueva muestra \mathbf{x} se realiza por el signo de una función discriminante $g(\mathbf{x})$:

$$c(\mathbf{x}) = \begin{cases} +1 & \text{si } g(\mathbf{x}) \geq 0 \\ -1 & \text{si } g(\mathbf{x}) < 0 \end{cases}$$

- El algoritmo Perceptron se puede reescribir para la clasificación en dos clases

Aprendizaje - Perceptron de 2 clases

- Entrada: $X = \{(\mathbf{x}_1, c_1), (\mathbf{x}_2, c_2), \dots, (\mathbf{x}_N, c_N)\}$ y factor de aprendizaje α
- Salida: \mathbf{w} y w_0 // vector de pesos entrenados y término independiente
- Algoritmo:

$\mathbf{w} = \mathbf{0}$; $w_0 = 0$ // vector de pesos iniciales y peso umbral nulos

do

$m = 0$; // número de muestras bien clasificadas

for ($n = 1$; $n \leq N$; $n++$)

$g(\mathbf{x}_n) = \mathbf{w}^t \cdot \mathbf{x}_n + w_0$

if $c_n \cdot g(\mathbf{x}_n) \leq 0$ **then** // Si hay un error de clasificación

$\mathbf{w} = \mathbf{w} + \alpha c_n \mathbf{x}_n$; $w_0 = w_0 + \alpha c_n$

else

$m = m + 1$

while ($m < N$)

Aprendizaje - Perceptron de 2 clases

- Entrada: $X = \{(\mathbf{x}_1, c_1), (\mathbf{x}_2, c_2), \dots, (\mathbf{x}_N, c_N)\}$,
factor de aprendizaje $\alpha \in \mathbb{R}^N \wedge \alpha_i = \alpha_j \forall i, j$
- Salida: $g(\mathbf{x})$ // función de clasificación
- Algoritmo:

$g(\mathbf{x}) = 0$

do

$m = 0$; // número de muestras bien clasificadas

for ($n = 1$; $n \leq N$; $n++$)

if $c_n \cdot g(\mathbf{x}_n) \leq 0$ **then** // Si hay un error de clasificación

$g(\mathbf{x}) = g(\mathbf{x}) + \alpha_n c_n (\mathbf{x}_n^t \cdot \mathbf{x}) + \alpha_n c_n$

else

$m = m + 1$;

while ($m < N$)

Clasificación binaria y kernels

- $g(\mathbf{x}) = \mathbf{w}^t \cdot \mathbf{x} + w_0$ es un clasificador con vector de pesos \mathbf{w} y peso umbral w_0 :

$$\mathbf{w} = \sum_{n=1}^N \alpha_n c_n \mathbf{x}_n \quad w_0 = \sum_{n=1}^N \alpha_n c_n$$

- $g(\mathbf{x})$ relaciona \mathbf{x} con algunas muestras de entrenamiento por el producto escalar y α_i pasa de factor de aprendizaje al peso de cada muestra:

$$g(\mathbf{x}) = \sum_{n=1}^N \alpha_n c_n (\mathbf{x}_n^t \cdot \mathbf{x}) + \alpha_n c_n$$

- Generalizar producto escalar para resolver tareas no linealmente separables
- Cambio por una función **kernel** $K(\mathbf{x}_n, \mathbf{x})$: proyecta a un espacio donde las muestras son linealmente separables y realiza el producto escalar
- La proyección es **implícita** y se obtiene al calcular la función kernel:

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

Clasificación binaria y kernels

- **Función kernel**: función que dado un par de objetos del espacio de representación original nos devuelve un valor real:

$$K : E \times E \rightarrow \mathbb{R}$$

- Usualmente la representación es vectorial, entonces:

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

- Dicho valor real modela el producto escalar de esos dos objetos en un nuevo espacio de representación:

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

- **La representación alternativa no se llega a producir**, sólo se necesita el **resultado** del producto escalar en esa representación para usarlo en un clasificador lineal

Índice

- 1 Introducción ▷ 3
- 2 Clasificación binaria y kernels ▷ 6
- 3 *Aprendizaje - Kernel Perceptron* ▷ 12
- 4 Tipos de Kernel ▷ 16
- 5 Kernels generalizados ▷ 21
- 6 *Linear Discriminant Analysis (LDA)* ▷ 25

Aprendizaje - Kernel Perceptron

- El algoritmo Kernel Perceptron aprende la siguiente función:

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

- Es decir, **una función lineal en un espacio de representación alternativo:**

$$g(\mathbf{x}) = \mathbf{w} \Phi(\mathbf{x}) + w_0$$

con

$$\mathbf{w} = \sum_{n=1}^N \alpha_n c_n \Phi(\mathbf{x}_n) \quad w_0 = \sum_{n=1}^N \alpha_n c_n$$

- Los únicos parámetros a aprender son los α_n
- En fase de aprendizaje la función kernel se representa por una matriz $\mathbf{K} \in \mathbb{R}^{N \times N}$ tal que $\mathbf{K}_{i,j} = K(\mathbf{x}_i, \mathbf{x}_j)$ (**matriz Gramm**)

Aprendizaje - Kernel Perceptron

Desde el punto de vista de la función a aprender:

- Entrada: $X = \{(\mathbf{x}_1, c_1), (\mathbf{x}_2, c_2), \dots, (\mathbf{x}_N, c_N)\}$
- Salida: $g(\mathbf{x})$
- Algoritmo:

$g(\mathbf{x}) = 0;$

do

$m = 0;$ // número de muestras bien clasificadas

for ($n = 1; n \leq N; n++$)

if $c_n \cdot g(\mathbf{x}_n) \leq 0$ **then** // Si hay un error de clasificación

$g(\mathbf{x}) = g(\mathbf{x}) + c_n K(\mathbf{x}_n, \mathbf{x}) + c_i$

else

$m = m + 1$

while ($m < N$)

Aprendizaje - Kernel Perceptron

Desde el punto de vista de los parámetros α :

- Entrada: $X = \{(\mathbf{x}_1, c_1), (\mathbf{x}_2, c_2), \dots, (\mathbf{x}_N, c_N)\}$
- Salida: $\alpha \in \mathbb{R}^N$
- Algoritmo:

$\alpha = \mathbf{0}$;

do

$m = 0$; // número de muestras bien clasificadas

for ($n = 1$; $n \leq N$; $n++$)

if $c_n \cdot g(\mathbf{x}_n) \leq 0$ **then** // Si hay un error de clasificación

$\alpha_n = \alpha_n + 1$

else

$m = m + 1$

while ($m < N$)

Índice

- 1 Introducción ▷ 3
- 2 Clasificación binaria y kernels ▷ 6
- 3 Aprendizaje - Kernel Perceptron ▷ 12
- 4 *Tipos de Kernel* ▷ 16
- 5 Kernels generalizados ▷ 21
- 6 *Linear Discriminant Analysis* (LDA) ▷ 25

Tipos de Kernel

Entre los más usados están el *polinomial* y el *gaussiano* (o radial)

- Kernel polinomial:

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^t \cdot \mathbf{y} + c)^d \quad \text{con } c, d > 0$$

- Ejemplo $d = 2$

$$K(\mathbf{x}, \mathbf{y}) = \left(\begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} + c \right)^2 = (x_1 y_1 + x_2 y_2 + c) (x_1 y_1 + x_2 y_2 + c)$$

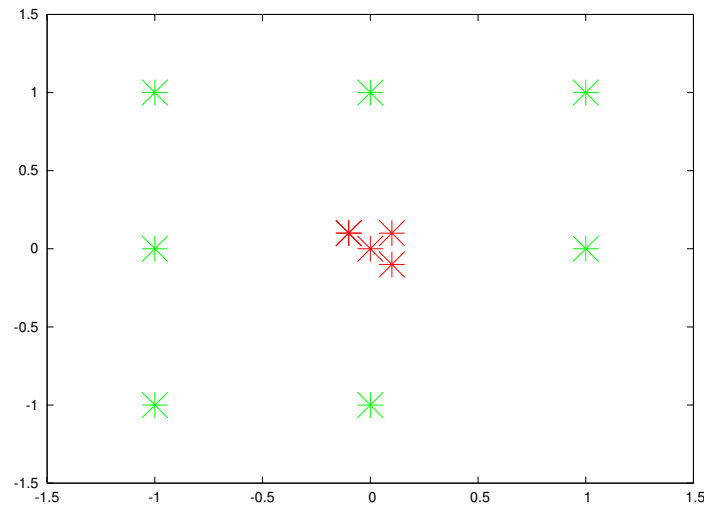
$$= x_1^2 y_1^2 + x_2^2 y_2^2 + 2 x_1 y_1 x_2 y_2 + 2 x_1 y_1 c + 2 x_2 y_2 c + c^2$$

$$= \begin{bmatrix} x_1^2 & x_2^2 & \sqrt{2} x_1 x_2 & \sqrt{2c} x_1 & \sqrt{2c} x_2 & c \end{bmatrix} \begin{bmatrix} y_1^2 \\ y_2^2 \\ \sqrt{2} y_1 y_2 \\ \sqrt{2c} y_1 \\ \sqrt{2c} y_2 \\ c \end{bmatrix}$$

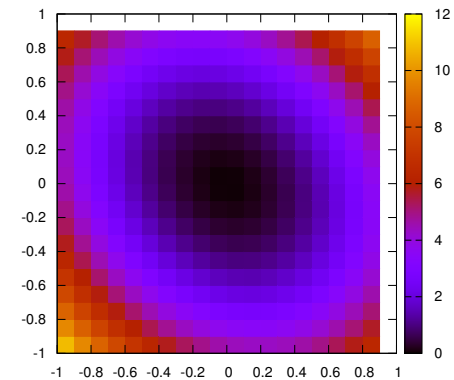
$$= \Phi(\mathbf{x}) \cdot \Phi(\mathbf{y})$$

Kernel polinomial

Conjunto de entrenamiento



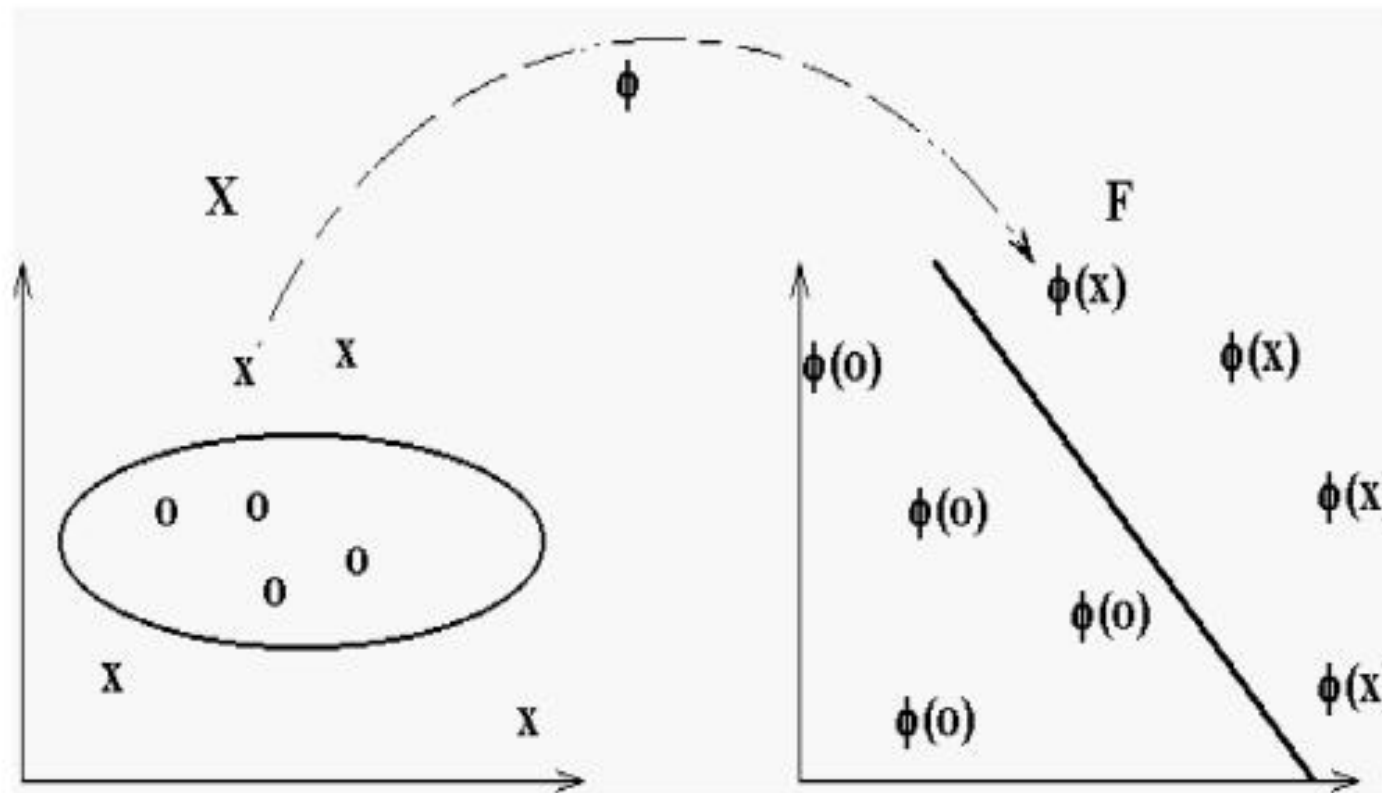
Representación de $g(\mathbf{x})$ con
 $\mathbf{x} \in [-1, 1]$ y $\alpha_n = 1, \forall n$



Recordad $g(\mathbf{x}) = \sum_{n=1}^N \alpha_n c_n K(\mathbf{x}_n, \mathbf{x})$ siendo $K(\mathbf{x}_n, \mathbf{x})$ un kernel polinómico

Kernel polinomial

En el ejemplo previo hay una proyección implícita a un nuevo espacio donde las muestras de entrenamiento son linealmente separables:



http://upload.wikimedia.org/wikipedia/commons/b/b1/Svm_8_polinomial.JPG

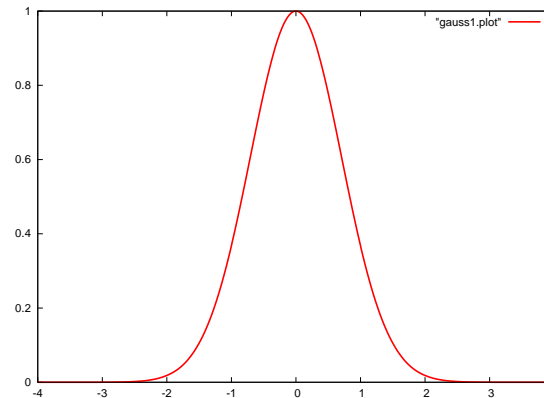
Kernel gaussiano

- Kernel Gaussiano:

Eso es distancia euclídea

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

- Representación gráfica de la gaussiana (unidimensional):



- Kernel muy empleado, pues asume que la función $\Phi(\cdot)$ implícitamente relacionada proyecta los puntos a un espacio de dimensionalidad infinita
- En dimensionalidad infinita los datos son *siempre* linealmente separables

Índice

- 1 Introducción ▷ 3
- 2 Clasificación binaria y kernels ▷ 6
- 3 Aprendizaje - Kernel Perceptron ▷ 12
- 4 Tipos de Kernel ▷ 16
- 5 *Kernels generalizados* ▷ 21
- 6 *Linear Discriminant Analysis* (LDA) ▷ 25

Kernels generalizados

- Objetivo: definir y evaluar diferentes funciones kernel
- La función kernel debe cumplir que:

$$\exists \Phi : \mathbb{R}^D \rightarrow \mathbb{R}^{D'} : K(\mathbf{x}, \mathbf{y}) = \Phi(\mathbf{x}^t) \Phi(\mathbf{y})$$

- **Mercer condition**: condición necesaria y suficiente para caracterizar que K sea un kernel válido:

“La matriz Gramm $\mathbf{K}_{i,j}$ definida para el conjunto de entrenamiento $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ es semidefinida positiva ($\mathbf{z}^t \mathbf{K} \mathbf{z} \geq 0, \forall \mathbf{z} \in \mathbb{R}^{N \times 1}, \mathbf{z} \neq \mathbf{0}$)”

- La matriz Gramm \mathbf{K} es semidefinida positiva si se cumple:

$$\sum_{i=1}^N \sum_{j=1}^N K(\mathbf{x}_i, \mathbf{x}_j) z_i z_j \geq 0 \quad \forall z_i, z_j \in \mathbb{R}$$

- Usando esta propiedad podemos construir kernels desde kernels más simples

Kernels generalizados

Si K_1 y K_2 son kernels, entonces K es un kernel:

$$K(\mathbf{x}, \mathbf{y}) = \left\{ \begin{array}{ll} c \cdot K_1(\mathbf{x}, \mathbf{y}) & c > 0 \\ f(\mathbf{x}) \cdot K_1(\mathbf{x}, \mathbf{y}) \cdot f(\mathbf{y}) & \text{para cualquier función } f \\ q(K_1(\mathbf{x}, \mathbf{y})) & q \text{ polinomio con coeficientes no negativos} \\ (c + K_1(\mathbf{x}, \mathbf{y}))^d & d, c > 0 \\ K_1(\mathbf{x}, \mathbf{y}) + K_2(\mathbf{x}, \mathbf{y}) \\ K_1(\mathbf{x}, \mathbf{y}) \cdot K_2(\mathbf{x}, \mathbf{y}) \\ \exp(K_1(\mathbf{x}, \mathbf{y})) \\ \frac{K_1(\mathbf{x}, \mathbf{y})}{\sqrt{K_2(\mathbf{x}, \mathbf{y})}} \end{array} \right.$$

$\exp = e^k$

Kernels generalizados

- Sea $\mathcal{A} \in \mathbb{R}^{D \times D}$ una matriz semidefinida positiva, entonces K es un kernel:

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

- Sean $\mathbf{x}, \mathbf{y} \in \mathbb{R}^D$, tales que $\mathbf{x} = (\mathbf{x}_a, \mathbf{x}_b)$, $\mathbf{y} = (\mathbf{y}_a, \mathbf{y}_b)$, con:

- $\mathbf{x}_a, \mathbf{y}_a \in \mathbb{R}^{D_a}$
- $\mathbf{x}_b, \mathbf{y}_b \in \mathbb{R}^{D_b}$
- $D = D_a + D_b$

Si K_a y K_b son kernels en \mathbb{R}^{D_a} y \mathbb{R}^{D_b} , respectivamente, K es un kernel:

$$K(\mathbf{x}, \mathbf{y}) = \begin{cases} K_a(\mathbf{x}_a, \mathbf{y}_a) + K_b(\mathbf{x}_b, \mathbf{y}_b) \\ K_a(\mathbf{x}_a, \mathbf{y}_a) \cdot K_b(\mathbf{x}_b, \mathbf{y}_b) \end{cases}$$

Índice

- 1 Introducción ▷ 3
- 2 Clasificación binaria y kernels ▷ 6
- 3 Aprendizaje - Kernel Perceptron ▷ 12
- 4 Tipos de Kernel ▷ 16
- 5 Kernels generalizados ▷ 21
- 6 Linear Discriminant Analysis (*LDA*) ▷ 25

Introducción a LDA

- LDA: Linear Discriminant Analysis
- Técnica de reducción de dimensionalidad *supervisada*
- Se pretende que la proyección lineal:
 - Preserve la separación de las clases del espacio original
 - Que los puntos de una misma clase permanezcan cercanos entre ellos
- Estas dos propiedades se resumen en dos estadísticos:
 - La separación de las medias de las clases
 - La reducción de las covarianzas intra-clase
- Se basa en **vectores propios generalizados** ($W^t \mathbf{x} = \lambda V^t \mathbf{x}$)

Conceptos previos

- Recordemos que $\mathbf{x}' = W^t \mathbf{x}$, donde $W \in \mathbb{R}^{D \times k}$

- Bajo este supuesto tenemos:
 - $\bar{\mathbf{x}}' = W^t \bar{\mathbf{x}}$ (media de los puntos proyectados)
 - $\Sigma'_{\mathcal{X}} = W^t \Sigma_{\mathcal{X}} W$ (matriz de covarianzas de los puntos proyectados)

- Por lo tanto:
 - La media de los puntos en el espacio proyectado es la proyección de la media de los puntos en el espacio original
 - La matriz de covarianza en el espacio proyectado es la proyección (dos veces) de la matriz de covarianzas de los puntos en el espacio original

LDA

Definiciones:

- Conjunto de muestras etiquetadas: $\mathcal{X} = \{(\mathbf{x}_1, c_1), (\mathbf{x}_2, c_2) \cdots, (\mathbf{x}_N, c_N)\}$
- Conjunto de clases: $\mathbb{C} = \{1, 2, \dots, C\}$, $c_n \in \mathbb{C}$
- Media total: $\bar{\mathbf{x}} = \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n$
- Número de muestras de la clase c : N_c
- Media de clase c : $\bar{\mathbf{x}}_c = \frac{1}{N_c} \sum_{n:c_n=c}^N \mathbf{x}_n$
- Matriz de covarianzas de clase c : $\Sigma_c = \frac{1}{N_c} \sum_{n:c_n=c}^N (\mathbf{x}_n - \bar{\mathbf{x}}_c) (\mathbf{x}_n - \bar{\mathbf{x}}_c)^t$

LDA propone encontrar una matriz de proyección W que separe las medias $\bar{\mathbf{x}}_c$ pero reduzca las matrices de covarianzas Σ_c

LDA

- Se proponen dos tipos de matrices relacionadas con el anterior objetivo:

- La matriz *entre-clases*¹:

$$S_b = \sum_{c=1}^C N_c (\bar{\mathbf{x}}_c - \bar{\mathbf{x}})(\bar{\mathbf{x}}_c - \bar{\mathbf{x}})^t$$

- La matriz *intra-clases*²:

$$S_w = \sum_{c=1}^C \Sigma_c$$

- Una buena proyección lineal debería conseguir en el espacio proyectado:
 - S_b grande, y
 - S_w pequeño

¹between-class

²within-class

LDA

- En el espacio proyectado ambas matrices, S_b y S_w se pueden calcular como:

$$S'_b = W^t S_b W$$

$$S'_w = W^t S_w W$$

- Suponiendo W ortonormal, S'_b y S'_w son matrices diagonales, donde cada elemento es la varianza en la dimensión a la que se proyecta
- Varianza total: operador traza Tr (suma de elementos de la diagonal)
- Buscamos una proyección lineal W que:
 - Maximice la varianza entre-clase $Tr(S'_b)$, y
 - Minimice la varianza intra-clase $Tr(S'_w)$
- Se puede demostrar que es equivalente a la siguiente función objetivo:³

$$\widehat{W} = \operatorname{argmax}_W \frac{Tr(W^t S_b W)}{Tr(W^t S_w W)}$$

³K. Fukunaga, "Statistical Pattern Recognition", pp. 446-447

Problema de optimización

- Por simplicidad, optimizaremos la proyección a una única dimensión

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmax}} \frac{\mathbf{w}^t S_b \mathbf{w}}{\mathbf{w}^t S_w \mathbf{w}}$$

- Dado que la función objetivo es invariante al escalado de \mathbf{w} , simplificaremos el problema de optimización condicionándolo a que $\mathbf{w}^t S_w \mathbf{w} = 1$

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmax}} \mathbf{w}^t S_b \mathbf{w} \quad \text{sujeto a} \quad \mathbf{w}^t S_w \mathbf{w} = 1$$

- Expresado con el multiplicador de Lagrange correspondiente

$$\hat{\mathbf{w}} = \underset{\mathbf{w}}{\operatorname{argmax}} \underset{\lambda}{\operatorname{máx}} \mathbf{w}^t S_b \mathbf{w} + \lambda(1 - \mathbf{w}^t S_w \mathbf{w})$$

- Tras derivar respecto de \mathbf{w} y λ e igualar a cero, obtenemos

$$S_b \mathbf{w} = \lambda S_w \mathbf{w}$$

donde \mathbf{w} es el vector propio generalizado de S_b y S_w de mayor valor propio

Problema de optimización

- En el caso general se busca la matriz de proyección W

$$\widehat{W} = \underset{W}{\operatorname{argmax}} \operatorname{Tr}(W^t S_b W) \quad \text{sueto a} \quad \mathbf{w}_j^t S_w \mathbf{w}_j = 1 \quad \forall j$$

que se puede expresar mediante un sumatorio de multiplicadores de Lagrange

$$\widehat{W} = \underset{W}{\operatorname{argmax}} \underset{\Lambda}{\operatorname{máx}} \operatorname{Tr}(W^t S_b W) + \operatorname{Tr}(\Lambda \cdot (I - W^t S_w W))$$

donde Λ es una matriz diagonal con los multiplicadores de Lagrange en la diagonal, uno por cada vector de proyección; e I es la matriz identidad

- Derivando con respecto a W y Λ e igualando a 0 nos queda:

$$S_b W = \Lambda S_w W$$

donde W son los vectores propios *generalizados* de S_b y S_w

Problema de optimización

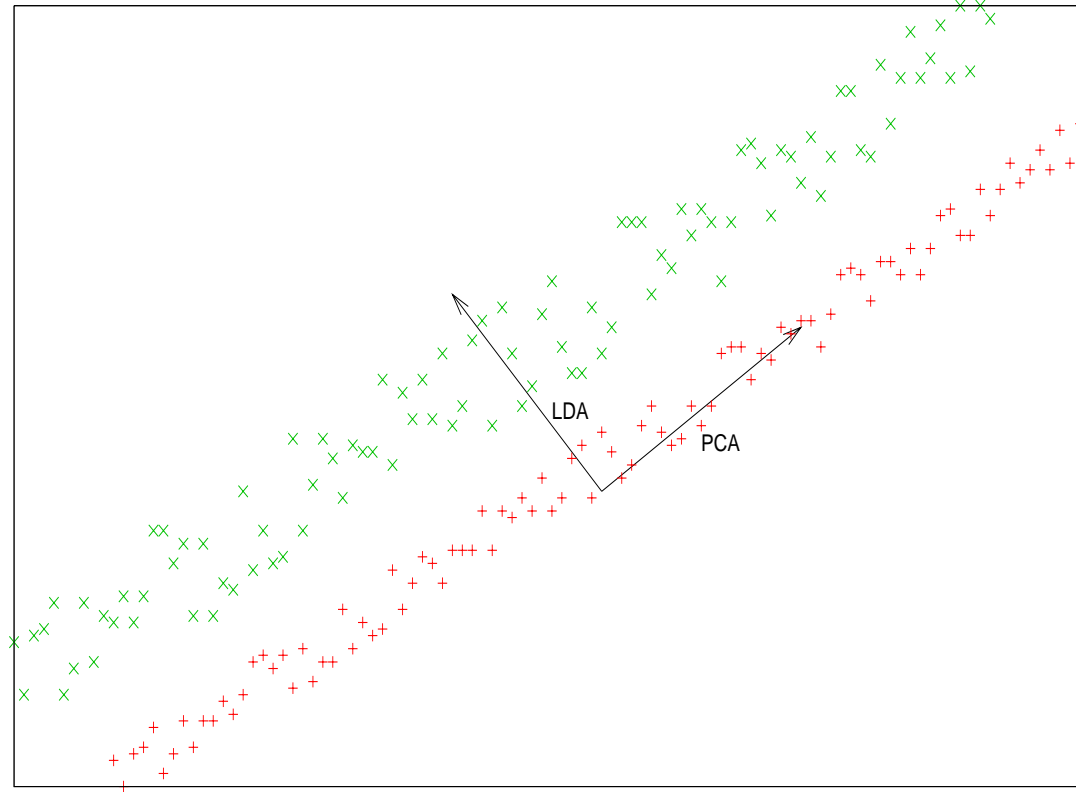
- Los vectores propios que maximizan la función objetivo original son aquellos con mayor valor propio asociado

$$W_{D \times k} = (\mathbf{w}_1 \quad \mathbf{w}_2 \quad \dots \quad \mathbf{w}_k), \quad \lambda_1 > \lambda_2 > \dots > \lambda_k$$

- La configuración de la matriz S_b hace que *no tengan más de $C - 1$ vectores propios linealmente independientes*
- Por ello no tiene sentido proyectar a una dimensionalidad $k > C - 1$

Problema de optimización

LDA vs. PCA



Algoritmo LDA

- Entrada: $N, D, k, \mathcal{X} = \{(\mathbf{x}_1, c_1), (\mathbf{x}_2, c_2), \dots, (\mathbf{x}_N, c_N)\}$
- Salida: W
- Algoritmo:
 1. Calcular la media de los datos: $\bar{\mathbf{x}} = \frac{1}{N} \sum_{n=1}^N \mathbf{x}_n$
 2. Calcular la media por clase: $\bar{\mathbf{x}}_c = \frac{1}{N_c} \sum_{n:c_n=c}^N \mathbf{x}_n$
 3. Calcular la matriz de covarianzas de clase: $\Sigma_c = \frac{1}{N_c} \sum_{n:c_n=c}^N (\mathbf{x}_n - \bar{\mathbf{x}}_c) (\mathbf{x}_n - \bar{\mathbf{x}}_c)^t$
 4. Calcular $S_w = \sum_{c=1}^C \Sigma_c$
 5. Calcular $S_b = \sum_{c=1}^C n_c (\bar{\mathbf{x}}_c - \bar{\mathbf{x}}) (\bar{\mathbf{x}}_c - \bar{\mathbf{x}})^t$
 6. Encontrar vectores propios generalizados de S_b y S_w
 7. Ordenarlos según los valores propios asociados
 8. Definir W como la matriz con los k primeros vectores propios

Consideraciones prácticas

- La inversión de la matriz S_w y la solución del problema de vectores propios generalizados pueden acarrear problemas numéricos
 - Habitual si $D \gg n$ (D dimensión espacio original, n número de muestras)
 - En estos casos la aplicación *directa* de LDA no es aconsejable
- En este caso, para hacer una proyección lineal discriminativa se aconseja:
 - Realizar una primera reducción de dimensionalidad mediante PCA
 - Realizar una segunda reducción de dimensionalidad mediante LDA
- La proyección quedaría como:

$$\mathbf{x}' = V^t W^t (\mathbf{x} - \bar{\mathbf{x}})$$

- W : matriz de proyección PCA
- V : matriz de proyección LDA