UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escola Tècnica Superior d'Enginyeria **Informàtica**

etsinf

Tema 5. Distribuciones de probabilidad

Percepción (PER)

Curso 2019/2020

Departamento de Sistemas Informáticos y Computación

Índice

Índice

Introducción

Clasificador de Bayes:

$$c^*(x) = \underset{c=1,\ldots,C}{\text{argmax}}\ P(c \mid x) = \underset{c=1,\ldots,C}{\text{argmax}}\ \frac{P(c)\,p(x \mid c)}{p(x)} =$$

$$\underset{c=1,\ldots,C}{\text{argmax}}\ P(c)\,p(x \mid c) = \underset{c=1,\ldots,C}{\text{argmax}}\ \log P(c) + \log p(x \mid c)$$

- $P(c)$: probabilidad *a priori*
- $p(x|c)$: función de densidad (f.d.) de probabilidad condicional para clase c

En la práctica, se usan ***estimaciones*** de $P(c)$ y $p(x|c)$:

$$c^*(x) \approx \underset{c=1,\ldots,C}{\text{argmax}}\ \log \hat{P}(c) + \log \hat{p}(x \mid c)$$

Introducción

$\hat{P}(c)$ y $\hat{p}(x \mid c)$ se estiman desde N muestras etiquetadas, $(x_1, c_1), \ldots, (x_N, c_N)$, extraídas aleatoriamente de $p(x, c)$

Estimación de la probabilidad *a priori*:

$$\hat{P}(c) = \frac{N_c}{N} \qquad\qquad N_c = \sum_{n\,:\,c_n = c} 1$$

Estimación de la condicional $\hat{p}(x|c)$: a partir de las muestras x_n con $c_n = c$

Forma habitual: se asume un ***tipo de distribución*** sobre los datos de la clase y ***se estiman sus parámetros***

Introducción

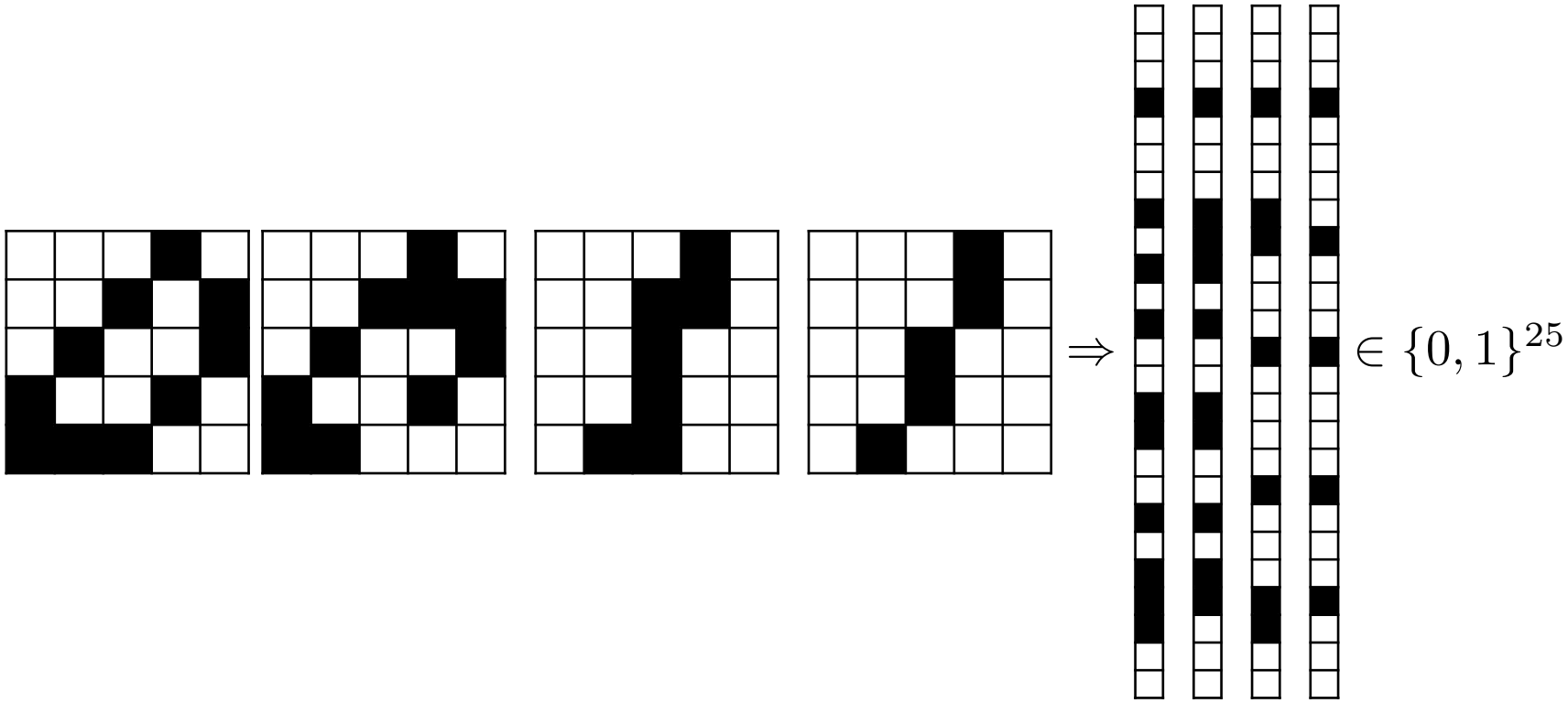
Estudiaremos tres tipos de distribución de probabilidad $p(x \mid c)$ que son aplicables en función de los datos a modelizar:

- Distribución de Bernoulli: datos binarios
- Distribución multinomial: datos que son contadores (enteros positivos)
- Distribución Gaussiana: datos que son números reales

Bernoulli: Motivación

Algunas tareas de RF conllevan objetos representados como un *vector de bits.*

Ejemplo: imágenes binarias de $5 \times 5 \rightarrow$ vectores de bits de 25 dimensiones

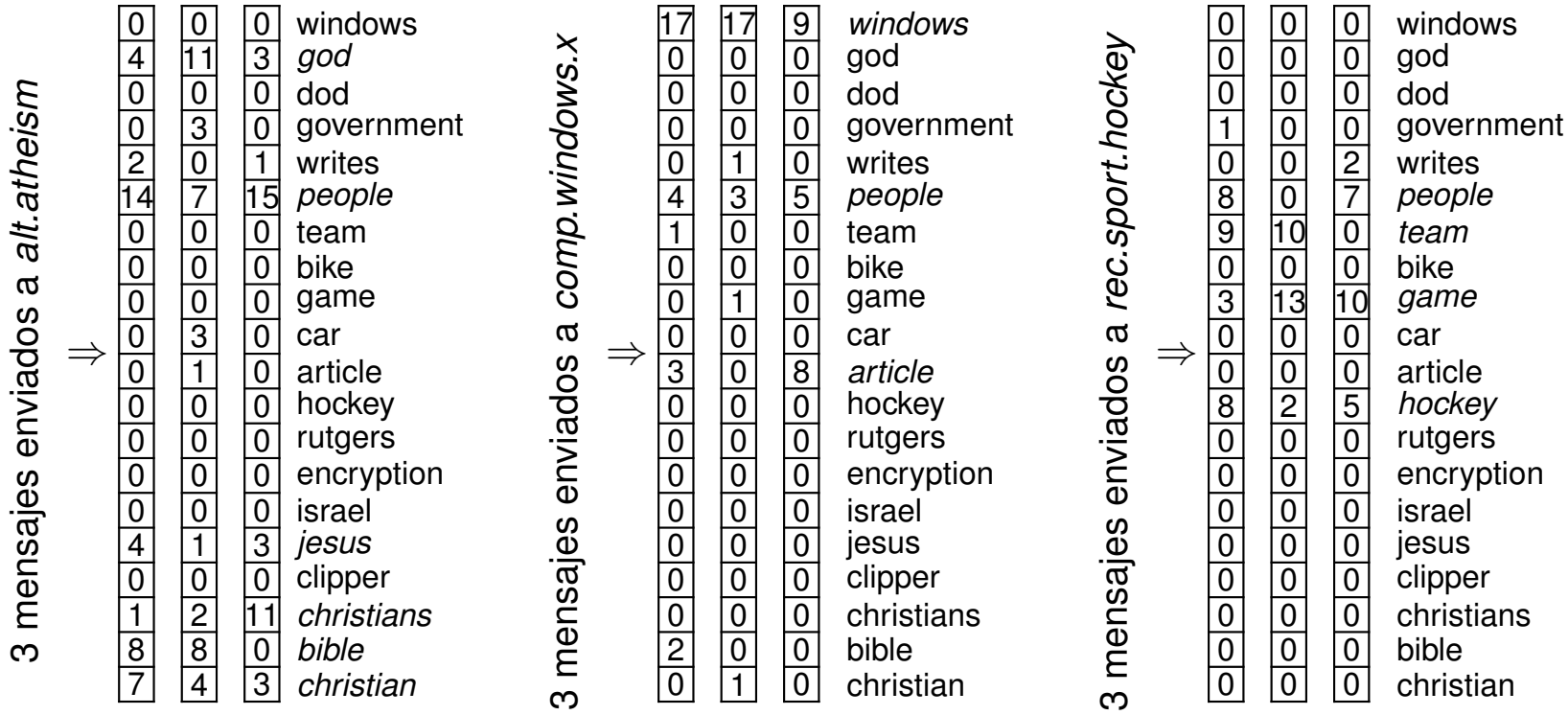


Idea: distribuciones de Bernoulli para modelar la condicional $p(x|c)$

Multinomial: Motivación

Algunas tareas de RF representan objetos como *vectores de cuentas*

Ejemplo: Texto representado como *bag-of-words*

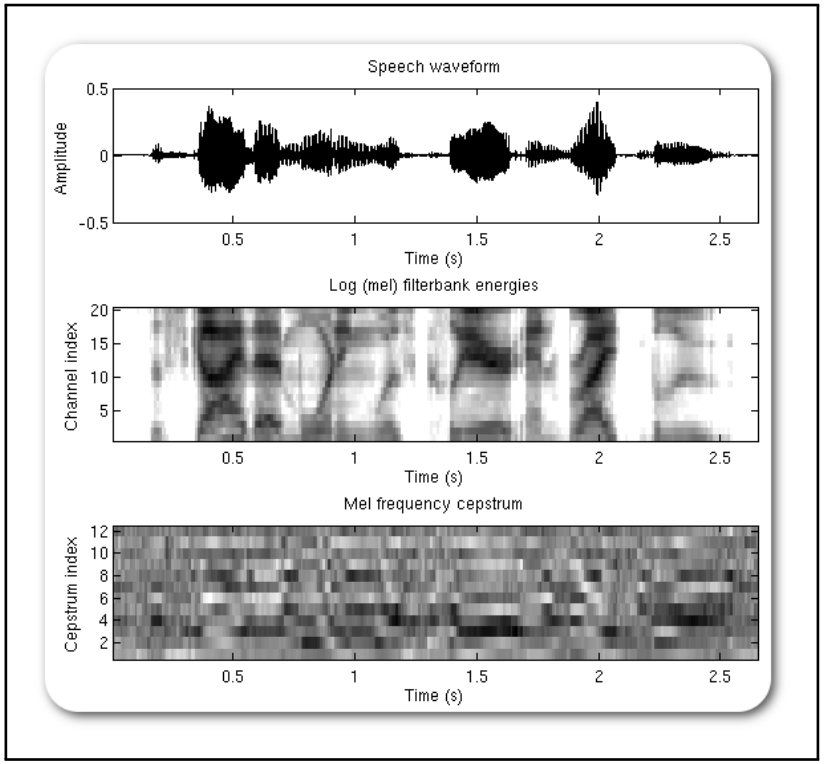


Idea: usar la *distribución multinomial* para modelizar la condicional $p(\boldsymbol{x}|c)$

Gaussiana: Motivación

Algunas tareas representan objetos por *vectores de características reales* ($\mathbb{R}^D$)

Ejemplo: Señal acústica mediante vectores de coeficientes cepstrales



$\Rightarrow$

18637.949219 520.131897 -421.241852 1373.556641 53.300270 1169.587769 -212.973862 . . . 266.111328 1038.843018 -41.131569 284.150818 -51.623795 27.055664 177.560684

15014.219727 -261.046661 -803.480652 1420.215332 -190.737427 1422.750366 -405.564545 . . . -120.428261 -108.468079 29.809811 69.221519 94.645294 20.136948 -27.475578

. . .

10495.829102 -551.998047 -471.013458 815.385010 -325.771301 891.692322 -281.717865 . . . -267.436554 -107.579346 -85.941284 -184.720337 -33.269119 95.332092 -81.534462

Idea: usar la *distribución gaussiana* para modelizar la condicional $p(\boldsymbol{x}|c)$

Introducción

Para cada distribución de probabilidad veremos:

- Su definición formal
- El clasificador asociado
- Su estimación por máxima verosimilitud (MV)
- Las técnicas de suavizado asociadas

Índice

Definición: Bernoulli unidimensional

Sea $p \in [0, 1]$ y $q = 1 - p$.

Sea x una variable aleatoria binaria que sigue una distribución de Bernoulli de parámetro p ($x \sim Be(p)$)

La f.d. de x es:

$$p(x) = \begin{cases} p & \text{si } x = 1 \\ q & \text{si } x = 0 \end{cases} \quad = \quad p\,x + q\,(1 - x) \quad = \quad p^x\,q^{1-x}$$

Nota: $\quad 0^0 = 1 \quad$ y $\quad 0 \log 0 = 0$

Definición: Bernoulli multidimensional

Sean $x_1 \sim Be(p_1), \ldots, x_D \sim Be(p_D)$ independientes

En ese caso, $\boldsymbol{x} = (x_1, \ldots, x_D)^t$ sigue una Bernoulli D-dimensional de parámetro $\boldsymbol{p} = (p_1, \ldots, p_D)^t$

La f.d. de $\boldsymbol{x}$ es:

$$p(\boldsymbol{x}) = \prod_{d=1}^{D} p(x_d) = \prod_{d=1}^{D} p_d\, x_d + q_d\, (1 - x_d) = \prod_{d=1}^{D} p_d^{x_d}\, q_d^{(1-x_d)}$$

Clasificador Bernoulli

Clasificador Bernoulli: clasificador de Bayes en el caso particular en que la f.d. condicional $p(\boldsymbol{x}|c)$ es una Bernoulli:

$$p(\boldsymbol{x} \,|\, c) \sim Be_D(\boldsymbol{p}_c), \quad c = 1, \ldots, C.$$

Por tanto:

$$\begin{aligned} c^*(\boldsymbol{x}) &= \underset{c=1,\ldots,C}{\text{argmax}} \; \log P(c) + \log p(\boldsymbol{x} \mid c) \\ &= \underset{c=1,\ldots,C}{\text{argmax}} \; \log P(c) + \log \prod_{d=1}^{D} p_{cd}^{x_d} (1 - p_{cd})^{(1-x_d)} \\ &= \underset{c=1,\ldots,C}{\text{argmax}} \; \log P(c) + \sum_{d=1}^{D} x_d \log p_{cd} + (1 - x_d) \log(1 - p_{cd}) \end{aligned}$$

Clasificador Bernoulli

Agrupando términos dependientes e independientes de x_d:

$$c^*(\boldsymbol{x}) = \underset{c=1,\ldots,C}{\text{argmax}} \left(\sum_{d=1}^{D} x_d (\log p_{cd} - \log(1 - p_{cd})) \right) + \left(\log\, P(c) + \sum_{d=1}^{D} \log(1 - p_{cd}) \right)$$

Reescribimos la expresión anterior como:

$$c^*(\boldsymbol{x}) = \underset{c=1,\ldots,C}{\text{argmax}} \sum_{d=1}^{D} w_{cd}\, x_d + w_{c0}$$

donde

$$w_{cd} = \log p_{cd} - \log(1 - p_{cd}) \qquad w_{c0} = \log\, P(c) + \sum_{d=1}^{D} \log(1 - p_{cd})$$

Clasificador Bernoulli

Por tanto, es un ***clasificador lineal*** sobre $\boldsymbol{x}$:

$$c^*(\boldsymbol{x}) = \underset{c=1,\ldots,C}{\text{argmax}}\ g_c(\boldsymbol{x}) = \underset{c=1,\ldots,C}{\text{argmax}} \sum_{d=1}^{D} w_{cd}\, x_d + w_{c0}$$

Reescribiendo la expresión anterior como un producto escalar de dos vectores:

$$c^*(\boldsymbol{x}) = \underset{c=1,\ldots,C}{\text{argmax}}\ \boldsymbol{w}_c^t\, \boldsymbol{x} + w_{c0}$$

donde

$$\boldsymbol{w}_c = \log \boldsymbol{p}_c - \log(\mathbf{1} - \boldsymbol{p}_c)$$

Entrenamiento por máxima verosimilitud

Sean un conjunto de entrenamiento de N muestras independientes e idénticamente distribuidas (i.i.d.) extraídas aleatoriamente de C distribuciones Bernoulli:

$$\{(\boldsymbol{x}_n, c_n)\}_{n=1}^{N} \quad \text{i.i.d.} \quad p(\boldsymbol{x}, c) = P(c)\, p(\boldsymbol{x}|c), \quad p(\boldsymbol{x}|c) \sim Be_D(\boldsymbol{p}_c)$$

Conjunto de parámetros a estimar $\boldsymbol{\Theta}$:

- Probabilidades *a priori*: $P(1) \ldots, P(C)$
- Parámetros de las Bernoulli para cada clase c: $\boldsymbol{p}_c$, $c = 1, \ldots, C$

Por ***criterio de máxima verosimilitud*** (MV), se estima $\boldsymbol{\Theta}$ como:

$$\hat{P}(c) = \frac{N_c}{N} \qquad c = 1, \ldots, C$$

$$\hat{\boldsymbol{p}}_c = \frac{1}{N_c} \sum_{n\,:\,c_n = c} \boldsymbol{x}_n \qquad c = 1, \ldots, C$$

Suavizado de la distribución Bernoulli

Problema: muchos criterios de entrenamiento (incluído MV) pueden generar clasificadores sobreentrenados

Soluciones:

- Cambiar el criterio de aprendizaje
- ***Suavizar*** los parámetros estimados

Opciones de suavizado en Bernoulli:

- Truncamiento simple
- Muestra ficticia

Suavizado de la distribución Bernoulli

Truncamiento simple

Dado ϵ, $0 \leq \epsilon \leq 0.5$, redefinir $\hat{p}_{cd}$ como:

$$\tilde{p}_{cd} = \begin{cases} \epsilon & \text{si } \hat{p}_{cd} < \epsilon \\ 1 - \epsilon & \text{si } \hat{p}_{cd} > 1 - \epsilon \\ \hat{p}_{cd} & \text{en otro caso} \end{cases}$$

Muestra ficticia

Añadir al conjunto de aprendizaje $(\mathbf{0}, c)$ y $(\mathbf{1}, c)$, $c = 1, \ldots, C$.

Equivale a redefinir la estimación de $\hat{\boldsymbol{p}}_c$ como:

$$\tilde{\boldsymbol{p}}_c = \frac{1}{N_c + 2} \left(\mathbf{1} + \sum_{n \,:\, c_n = c} \boldsymbol{x}_n \right)$$

Índice

Definición: distribución multinomial

Sea una población $\mathcal{Y} = \{y_1, \ldots, y_l\}$ con $y_i \in \{1, \ldots, D\}$

Sean las proporciones p_d de los tipos de elemento $\{1, \ldots, D\}$ dadas por:

$$\boldsymbol{p} = (p_1, \ldots, p_D)^t \in [0,1]^D \qquad \text{con} \qquad \sum_{d=1}^{D} p_d = 1$$

Sea una secuencia de N elementos formada por extracción aleatoria con reemplazo desde $\mathcal{Y}$

$$w_1^N = w_1\, w_2 \cdots w_N$$

Número de secuencias distintas de longitud N:

$$\mathrm{VR}_{D,N} = D^N$$

Definición: distribución multinomial

Asumiendo independencia entre elementos:

$$p(w_1^N) = p_{w_1} p_{w_2} \cdots p_{w_N}$$

No depende del orden de los elementos, sino de su número de ocurrencias:

- x_d: el número de ocurrencias del elemento d en w_1^N
- $\boldsymbol{x} = (x_1, \ldots, x_D)^t$: vector de ocurrencias (número de ocurrencias de cada elemento en w_1^N)

$$p(w_1^N) = p_1^{x_1} \cdots p_D^{x_D} = \prod_{d=1}^{D} p_d^{x_d}$$

El número de secuencias diferentes con el mismo vector de ocurrencias es un *coeficiente multinomial:*

$$\binom{N}{\boldsymbol{x}} = \binom{N}{x_1, \ldots, x_D} = \frac{N!}{x_1! \cdots x_D!}$$

Definición: distribución multinomial

Distribución multinomial: se define sobre el espacio de vectores de ocurrencias

La probabilidad de $\boldsymbol{x}$ es la suma de probabilidades de todas las secuencias con vector de ocurrencias $\boldsymbol{x}$:

$$p(\boldsymbol{x}) = \binom{N}{\boldsymbol{x}} \prod_{d=1}^{D} p_d^{x_d}$$

$p(\boldsymbol{x})$ es una f.d. multinomial:

- D-dimensional
- Longitud $N = \sum_{d=1}^{D} x_d$
- Prototipo $\boldsymbol{p}$

De ahora en adelante, usaremos $x_+ = N$.

Clasificador multinomial

Clasificador multinomial: clasificador de Bayes donde la f.d. condicional $p(\boldsymbol{x}|c)$ es una multinomial

$$p(\boldsymbol{x} \mid c) \sim Mult_D(x_+, \boldsymbol{p}_c), \quad c = 1, \ldots, C.$$

Por tanto:

$$c^*(\boldsymbol{x}) = \underset{c=1,\ldots,C}{\text{argmax}} \ \log P(c) + \log p(x \mid c)$$

$$= \underset{c=1,\ldots,C}{\text{argmax}} \ \log P(c) + \log \frac{x_+!}{x_1! \cdots x_D!} \prod_{d=1}^{D} p_{cd}^{x_d}$$

$$= \underset{c=1,\ldots,C}{\text{argmax}} \ \log P(c) + \log \frac{x_+!}{x_1! \cdots x_D!} + \sum_{d=1}^{D} x_d \log p_{cd}$$

Clasificador multinomial

Eliminando el término independiente de c:

$$c^*(\boldsymbol{x}) = \underset{c=1,\ldots,C}{\text{argmax}} \ \log \ P(c) + \sum_{d=1}^{D} x_d \log p_{cd}$$

Expresando el sumatorio en forma de producto escalar:

$$c^*(\boldsymbol{x}) = \underset{c=1,\ldots,C}{\text{argmax}} \ (\log \boldsymbol{p}_c)^t \, \boldsymbol{x} + \log \ P(c)$$

En forma de clasificador lineal:

$$c^*(\boldsymbol{x}) = \underset{c=1,\ldots,C}{\text{argmax}} \ g_c(\boldsymbol{x}) = \underset{c=1,\ldots,C}{\text{argmax}} \ \boldsymbol{w}_c^t \, \boldsymbol{x} + w_{c0}$$

Con:

$$\boldsymbol{w}_c = \log \boldsymbol{p}_c \qquad w_{c0} = \log P(c)$$

Entrenamiento por máxima verosimilitud

Sean N muestras de entrenamiento aleatoriamente extraídas de C distribuciones multinomiales independientes:

$$\{(\boldsymbol{x}_n, c_n)\}_{n=1}^{N} \quad \text{i.i.d.} \quad p(\boldsymbol{x}, c) = P(c)\, p(\boldsymbol{x}|c), \quad p(\boldsymbol{x}|c) \sim Mult_D(x_+, \boldsymbol{p}_c)$$

Conjunto de parámetros a estimar $\boldsymbol{\Theta}$:

- Probabilidades *a priori*: $P(1) \ldots, P(C)$
- Prototipos de las multinomiales para cada clase c: $\boldsymbol{p}_c$, $c = 1, \ldots, C$

Por ***criterio de máxima verosimilitud*** (MV), se estima $\boldsymbol{\Theta}$ como:

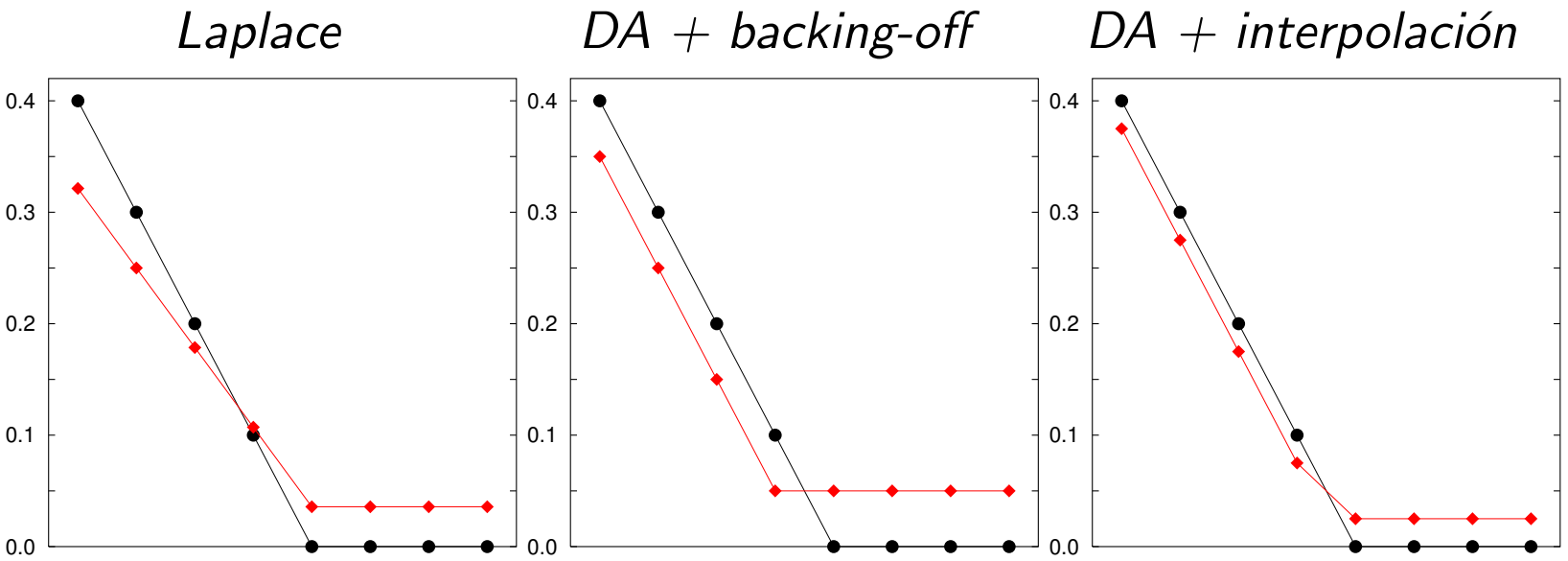
$$\hat{P}(c) = \frac{N_c}{N} \qquad \hat{\boldsymbol{p}}_c = \frac{1}{\sum\limits_{n\,:\,c_n=c} \sum\limits_{d=1}^{D} x_{nd}} \sum_{n\,:\,c_n=c} \boldsymbol{x}_n \qquad c = 1, \ldots, C$$

Suavizado de la distribución multinomial

Laplace: suma una constante $\epsilon > 0$ a cada parámetro y renormaliza

Descuento Absoluto (DA):

1. Descuenta una constante $\epsilon > 0$ (pequeña) a cada parámetro mayor que cero
2. Distribuir la probabilidad descontada según una *distribución generalizada*:
   - Entre todos los parámetros nulos *(backing-off)*
   - Entre todos los parámetros *(interpolación)*



Índice

Definición: distribución gaussiana unidimensional

Sea x una variable aleatoria unidimensional

Gaussiana unidimensional estandarizada

$x \sim \mathcal{N}(0, 1)$ presenta una distribución de probabilidad

$$p(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2} x^2\right)$$

Gaussiana unidimensional general

$x \sim \mathcal{N}(\mu, \sigma)$, con media $\mu \in \mathbb{R}$ y varianza $\sigma^2 \in \mathbb{R}^+$, presenta una distribución de probabilidad

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right)$$

Definición: distribución gaussiana multidimensional

Sea $\boldsymbol{x} = (x_1, \ldots, x_D)^t$ una variable aleatoria D-dimensional

Gaussiana estandarizada

$\boldsymbol{x} \sim \mathcal{N}_D(\mathbf{0}, I_D)$, donde $x_1, \ldots, x_D \sim \mathcal{N}(0,1)$ independientes, presenta una distribución de probabilidad:

$$p(\boldsymbol{x}) = (2\pi)^{-\frac{D}{2}} \exp\left(-\frac{1}{2}\boldsymbol{x}^t\boldsymbol{x}\right)$$

Definición: distribución gaussiana multidimensional

Gaussiana general

Sean:

- $\boldsymbol{z} \sim \mathcal{N}_D(\mathbf{0}, I_D)$
- $\boldsymbol{\mu} \in \mathbb{R}^D$
- $A \in \mathbb{R}^{D \times D} : |A| \neq 0$
- $\Sigma = AA^t$ (simétrica y definida positiva) con:
  - $A = W\Delta^{\frac{1}{2}}$
  - W vectores propios de Σ
  - Δ valores propios de Σ
- $\boldsymbol{x} = A\boldsymbol{z} + \boldsymbol{\mu}$

$\boldsymbol{x} \sim \mathcal{N}_D(\boldsymbol{\mu}, \Sigma)$, con media $\boldsymbol{\mu}$ y matriz de covarianzas Σ, presenta una distribución de probabilidad:

$$p(\boldsymbol{x}) = (2\pi)^{-\frac{D}{2}} |\Sigma|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^t \Sigma^{-1} (\boldsymbol{x} - \boldsymbol{\mu})\right)$$

Clasificador gaussiano

Clasificador gaussiano: clasificador de Bayes donde la f.d. condicional $p(\boldsymbol{x}|c)$ es una gaussiana

$$p(\boldsymbol{x} \,|\, c) \sim \mathcal{N}_D(\boldsymbol{\mu}_c, \Sigma_c), \quad c = 1, \ldots, C$$

Por tanto:

$$\begin{aligned}
c^*(\boldsymbol{x}) &= \underset{c=1,\ldots,C}{\text{argmax}} \; \log P(c) + \log p(\boldsymbol{x} \mid c) \\
&= \underset{c=1,\ldots,C}{\text{argmax}} \; \log P(c) - \frac{D}{2}\log 2\pi - \frac{1}{2}\log |\Sigma_c| - \frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_c)^t\Sigma_c^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_c) \\
&= \underset{c=1,\ldots,C}{\text{argmax}} \; \log P(c) - \frac{1}{2}\log |\Sigma_c| - \frac{1}{2}\boldsymbol{x}^t\Sigma_c^{-1}\boldsymbol{x} + \boldsymbol{\mu}_c^t\Sigma_c^{-1}\boldsymbol{x} - \frac{1}{2}\boldsymbol{\mu}_c^t\Sigma_c^{-1}\boldsymbol{\mu}_c \\
&= \underset{c=1,\ldots,C}{\text{argmax}} \; -\frac{1}{2}\boldsymbol{x}^t\Sigma_c^{-1}\boldsymbol{x} + \boldsymbol{\mu}_c^t\Sigma_c^{-1}\boldsymbol{x} + \left(\log P(c) - \frac{1}{2}\log |\Sigma_c| - \frac{1}{2}\boldsymbol{\mu}_c^t\Sigma_c^{-1}\boldsymbol{\mu}_c\right)
\end{aligned}$$

Clasificador gaussiano

$$c^*(\boldsymbol{x}) = \underset{c=1,\ldots,C}{\text{argmax}} \; -\frac{1}{2}\boldsymbol{x}^t\Sigma_c^{-1}\boldsymbol{x} + \boldsymbol{\mu}_c^t\Sigma_c^{-1}\boldsymbol{x} + \left(\log P(c) - \frac{1}{2}\log\,|\Sigma_c| - \frac{1}{2}\boldsymbol{\mu}_c^t\Sigma_c^{-1}\boldsymbol{\mu}_c\right)$$

Clasificador ***cuadrático*** con $\boldsymbol{x}$:

$$c^*(\boldsymbol{x}) = \underset{c=1,\ldots,C}{\text{argmax}} \; g_c(\boldsymbol{x}) = \underset{c=1,\ldots,C}{\text{argmax}}\, \boldsymbol{x}^t\, W_c\, \boldsymbol{x} + \boldsymbol{w}_c^t\, \boldsymbol{x} + w_{c0}$$

Con:

$$W_c = -\frac{1}{2}\Sigma_c^{-1} \qquad \boldsymbol{w}_c = \Sigma_c^{-1}\boldsymbol{\mu}_c$$

$$w_{c0} = \log P(c) - \frac{1}{2}\log\,|\Sigma_c| - \frac{1}{2}\boldsymbol{\mu}_c^t\Sigma_c^{-1}\boldsymbol{\mu}_c$$

Clasificador gaussiano

Caso particular: ***matriz de covarianzas común***, $\Sigma_c = \Sigma$

En ese caso, tanto $-\frac{1}{2}\boldsymbol{x}^t\Sigma^{-1}\boldsymbol{x}$ como $-\frac{1}{2}\log|\Sigma|$ son independientes de c

$$c^*(\boldsymbol{x}) = \underset{c=1,\ldots,C}{\text{argmax}}\ \boldsymbol{\mu}_c^t\Sigma^{-1}\boldsymbol{x} + \left(\log P(c) - \frac{1}{2}\boldsymbol{\mu}_c^t\Sigma^{-1}\boldsymbol{\mu}_c\right)$$

El clasificador gaussiano es ***lineal***:

$$c^*(\boldsymbol{x}) = \underset{c=1,\ldots,C}{\text{argmax}}\ g_c(\boldsymbol{x}) = \underset{c=1,\ldots,C}{\text{argmax}}\ \boldsymbol{w}_c^t\,\boldsymbol{x} + w_{c0}$$

Con:

$$\boldsymbol{w}_c = \Sigma^{-1}\boldsymbol{\mu}_c \qquad w_{c0} = \log P(c) - \frac{1}{2}\boldsymbol{\mu}_c^t\Sigma^{-1}\boldsymbol{\mu}_c$$

Entrenamiento por máxima verosimilitud

Sean N muestras de entrenamiento aleatoriamente extraídas de C distribuciones gaussianas independientes

$$\{(\boldsymbol{x}_n, c_n)\}_{n=1}^{N} \quad \text{i.i.d.} \quad p(\boldsymbol{x}, c) = P(c)\, p(\boldsymbol{x}|c), \quad p(\boldsymbol{x}|c) \sim \mathcal{N}_D(\boldsymbol{\mu}_c, \Sigma_c)$$

Conjunto de parámetros a estimar Θ:

- Probabilidades *a priori*: $P(1), \ldots, P(C)$
- Medias para cada clase: $\mu_1, \ldots, \mu_C$
- Matrices de covarianza para cada clase: $\Sigma_1, \ldots, \Sigma_C$

Por ***criterio de máxima verosimilitud*** (MV), se estima $\boldsymbol{\Theta}$ como:

$$\hat{P}(c) = \frac{N_c}{N} \qquad (1)$$

$$\hat{\boldsymbol{\mu}}_c = \frac{1}{N_c} \sum_{n:c_n=c} \boldsymbol{x}_n \qquad (2)$$

$$\hat{\Sigma}_c = \frac{1}{N_c} \sum_{n\,:\,c_n=c} (\boldsymbol{x}_n - \hat{\boldsymbol{\mu}}_c)(\boldsymbol{x}_n - \hat{\boldsymbol{\mu}}_c)^t = \left(\frac{1}{N_c} \sum_{n\,:\,c_n=c} \boldsymbol{x}_n \boldsymbol{x}_n^t \right) - \hat{\boldsymbol{\mu}}_c \hat{\boldsymbol{\mu}}_c^t \qquad (3)$$

Entrenamiento por máxima verosimilitud

En el caso de Σ común para todas las clases ($\Sigma_c = \Sigma$), el conjunto de parámetros a estimar Θ es:

- Probabilidades *a priori*: $P(1), \ldots, P(C)$
- Medias para cada clase: $\mu_1, \ldots, \mu_C$
- Matriz de covarianza común: Σ

Por ***criterio de máxima verosimilitud***, la estimación de $\boldsymbol{\Theta}$ se calcula como en el caso general (Ecuaciones (1) y (2) para $\hat{P}(c)$ y $\hat{\boldsymbol{\mu}}_c$ respectivamente) y:

$$\hat{\Sigma} = \sum_c \hat{P}(c)\,\hat{\Sigma}_c = \frac{1}{N}\sum_n \boldsymbol{x}_n \boldsymbol{x}_n^t - \sum_c \hat{P}(c)\,\hat{\boldsymbol{\mu}}_c \hat{\boldsymbol{\mu}}_c^t \tag{4}$$

Con $\hat{\Sigma}_c$ calculada según Ecuación (3)

Suavizado

Umbralizado de covarianza [Duda01, pág. 113]

Covarianzas con magnitud de la correlación no cercana a uno valen cero:

$$\tilde{\sigma}^2_{cdd'} = \begin{cases} \hat{\sigma}^2_{cdd'} & \text{si } |\hat{\rho}_{cdd'}| > 1 - \epsilon \\ 0 & \text{otro caso} \end{cases} \qquad \forall c, d, d' = 1, \ldots, D;\ d \neq d'$$

Donde:

- ϵ es una constante pequeña no negativa ($\epsilon = 0 \rightarrow \Sigma$ diagonal)
- Coeficiente de correlación: $\hat{\rho}_{cdd'} = \frac{\hat{\sigma}^2_{cdd'}}{\hat{\sigma}_{cdd}\,\hat{\sigma}_{cd'd'}}$

Flat smoothing

Combinación lineal de cada $\hat{\Sigma}_c$ y $\tilde{\Sigma}$ (matriz de covarianza global suavizada):

$$\tilde{\Sigma}_c = \alpha\,\hat{\Sigma}_c + (1 - \alpha)\,\tilde{\Sigma} \qquad \forall c \quad \alpha \in [0, 1]$$

Donde: $\tilde{\Sigma} = \beta\hat{\Sigma} + (1 - \beta)I$, $\beta \in [0, 1]$

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escola Tècnica Superior d'Enginyeria **Informàtica**

etsinf

Tema 6. Representación basada en Kernels y LDA

Percepción (PER)

Curso 2019/2020

Departamento de Sistemas Informáticos y Computación

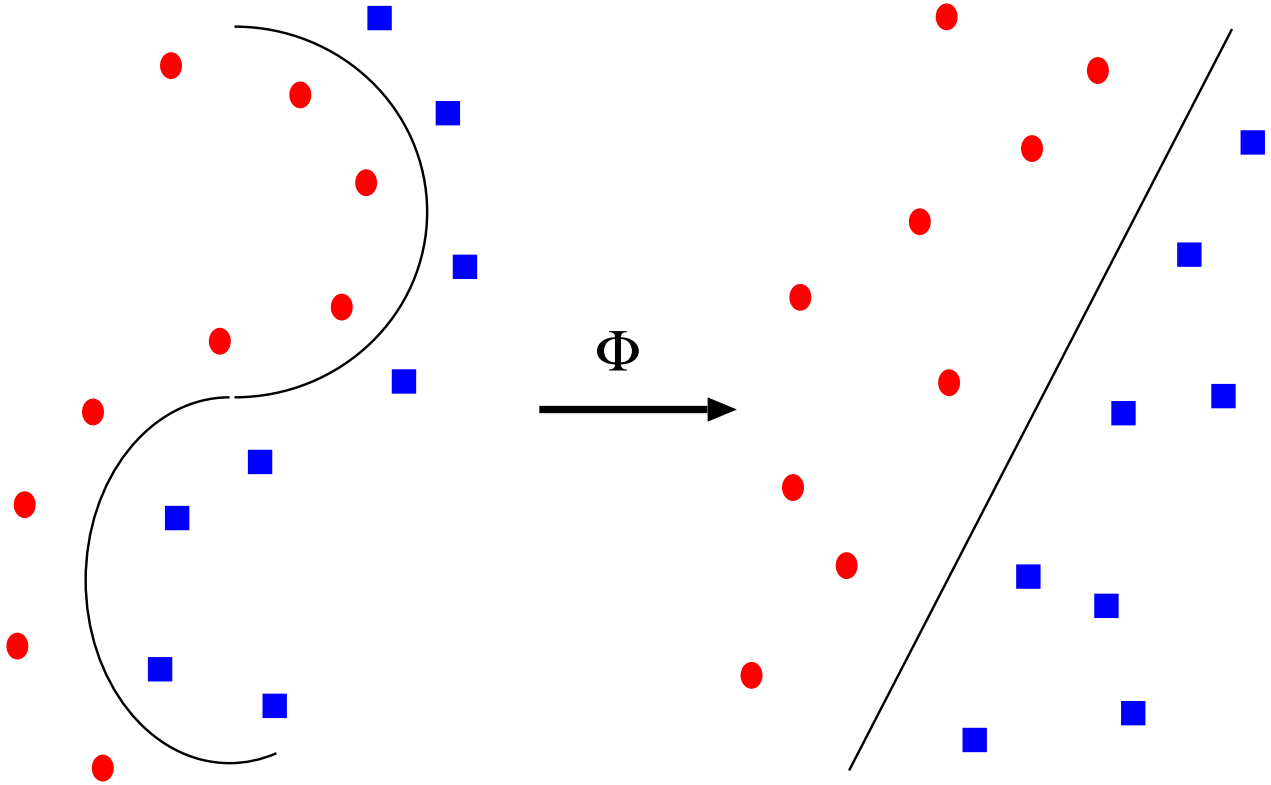
Índice

Índice

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

Clasificación binaria y kernels

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

- Conjunto de entrenamiento

$$X = \{(\mathbf{x}_1, c_1), (\mathbf{x}_2, c_2), \cdots, (\mathbf{x}_n, c_n)\} \quad \text{con} \quad c_i \in \{-1, +1\}$$

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

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

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

Aprendizaje - Perceptron de 2 clases

- Entrada: $X = \{(\mathbf{x}_1, c_1), (\mathbf{x}_2, c_2), \cdots, (\mathbf{x}_n, c_n)\}$ 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 $(i = 1; i \leq n; i{+}{+})$
        $g(\mathbf{x}_i) = \mathbf{w}^t \cdot \mathbf{x}_i + w_0$
        if $c_i \cdot g(\mathbf{x}_i) \leq 0$ **then** // Si hay un error de clasificación
            $\mathbf{w} = \mathbf{w} + \alpha\, c_i\, \mathbf{x}_i$; $w_0 = w_0 + \alpha\, c_i$
        else
            $m = m + 1$
while $(m < n)$

Aprendizaje - Perceptron de 2 clases

- Entrada: $X = \{(\mathbf{x}_1, c_1), (\mathbf{x}_2, c_2), \cdots, (\mathbf{x}_n, c_n)\}$,
  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 $(i = 1; \; i \leq n; \; i{+}{+})$

    if $c_i \cdot g(\mathbf{x}_i) \leq 0$ **then** // Si hay un error de clasificación
      $g(\mathbf{x}) = g(\mathbf{x}) + \alpha_i \, c_i \, (\mathbf{x}_i^t \cdot \mathbf{x}) + \alpha_i \, c_i$
    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:

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i \, c_i \, \mathbf{x}_i \qquad w_0 = \sum_{i=1}^{n} \alpha_i \, c_i$$

- $g(\mathbf{x})$ 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:

$$g(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i \, c_i \, (\mathbf{x}_i^t \cdot \mathbf{x}) + \alpha_i \, c_i$$

- Generalizar producto escalar para resolver tareas no linealmente separables
- Cambio por una función ***kernel*** $K(\mathbf{x}_i, \mathbf{x})$: 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:

$$g(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i \, c_i \, K(\mathbf{x}_i, \mathbf{x}) + \alpha_i \, c_i = \sum_{i=1}^{n} \alpha_i \, c_i \, (\Phi(\mathbf{x}_i^t) \cdot \Phi(\mathbf{x})) + \alpha_i \, c_i$$

Clasificación binaria y kernels

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

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

- Usualmente la representación es vectorial, entonces:

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

- 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

Aprendizaje - Kernel Perceptron

- El algoritmo Kernel Perceptron aprende la siguiente función:

$$g(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i \, c_i \, K(\mathbf{x}_i, \mathbf{x}) + \alpha_i \, c_i$$

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

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

con

$$\mathbf{w} = \sum_{i=1}^{n} \alpha_i \, c_i \, \Phi(\mathbf{x}_i) \qquad w_0 = \sum_{i=1}^{n} \alpha_i \, c_i$$

- Los únicos parámetros a aprender son los α_i
- 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: $X = \{(\mathbf{x}_1, c_1), (\mathbf{x}_2, c_2), \cdots, (\mathbf{x}_n, c_n)\}$
- Salida: $g(\mathbf{x})$
- Algoritmo:

$g(\mathbf{x}) = 0$;
do
    $m = 0$; // número de muestras bien clasificadas
    for $(i = 1;\ i \leq n;\ i{+}{+})$
        if $c_i \cdot g(\mathbf{x}_i) \leq 0$ **then** // Si hay un error de clasificación
            $g(\mathbf{x}) = g(\mathbf{x}) + c_i\, K(\mathbf{x}_i, \mathbf{x}) + c_i$
        else
            $m = m + 1$
while $(m < n)$

Aprendizaje - Kernel Perceptron

Desde el punto de vista de los parámetros α:

- Entrada: $X = \{(\mathbf{x}_1, c_1), (\mathbf{x}_2, c_2), \cdots, (\mathbf{x}_n, c_n)\}$
- Salida: $\alpha \in \mathbb{R}^n$
- Algoritmo:

$\alpha = \mathbf{0}$;
do
    $m = 0$; // número de muestras bien clasificadas
    for $(i = 1; i \leq n; i{+}{+})$
        if $c_i \cdot g(\mathbf{x}_i) \leq 0$ **then** // Si hay un error de clasificación
            $\alpha_i = \alpha_i + 1$
        else
            $m = m + 1$
while $(m < n)$

Índice

Tipos de Kernel

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

- Kernel polinomial:

$$K(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^t \cdot \mathbf{y} + c)^d$$

- Ejemplo $d = 2$

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

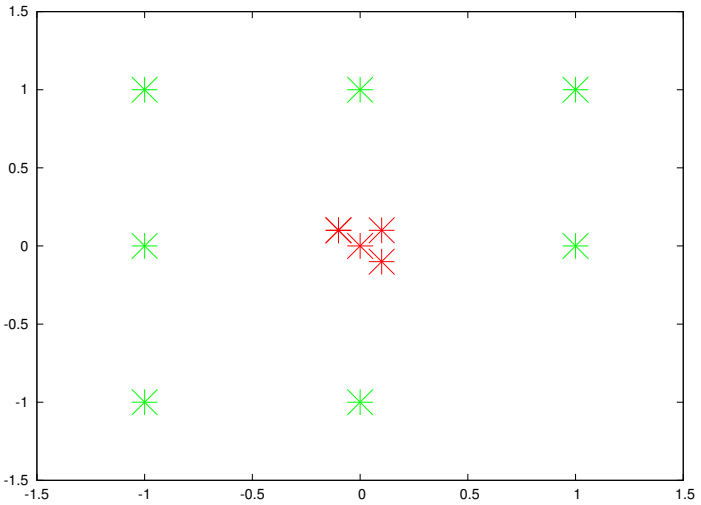
$$= x_1^2\, y_1^2 + x_2^2\, y_2^2 + 2\, x_1\, y_1\, x_2\, y_2 + 2\, x_1\, y_1\, c + 2\, x_2\, y_2\, c + c^2$$

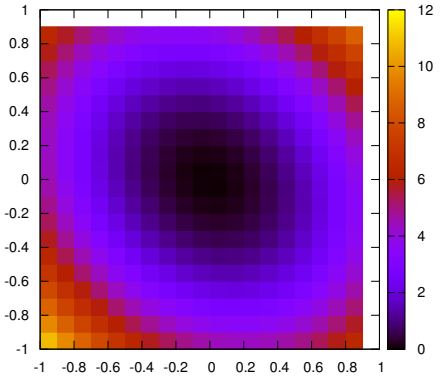
$$= \begin{bmatrix} x_1^2 & x_2^2 & \sqrt{2}\, x_1\, x_2 & \sqrt{2}\, x_1 & \sqrt{2}\, x_2 & c \end{bmatrix} \begin{bmatrix} y_1^2 \\ y_2^2 \\ \sqrt{2}\, y_1\, y_2 \\ \sqrt{2}\, y_1 \\ \sqrt{2}\, y_2 \\ c \end{bmatrix}$$

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

Kernel polinomial

- Conjunto de entrenamiento (ver figura izquierda)
- Representación de $g(\mathbf{x})$ con $\mathbf{x} \in [-1, 1]$ y $\alpha_i = 1, \forall i$ (ver figura derecha)

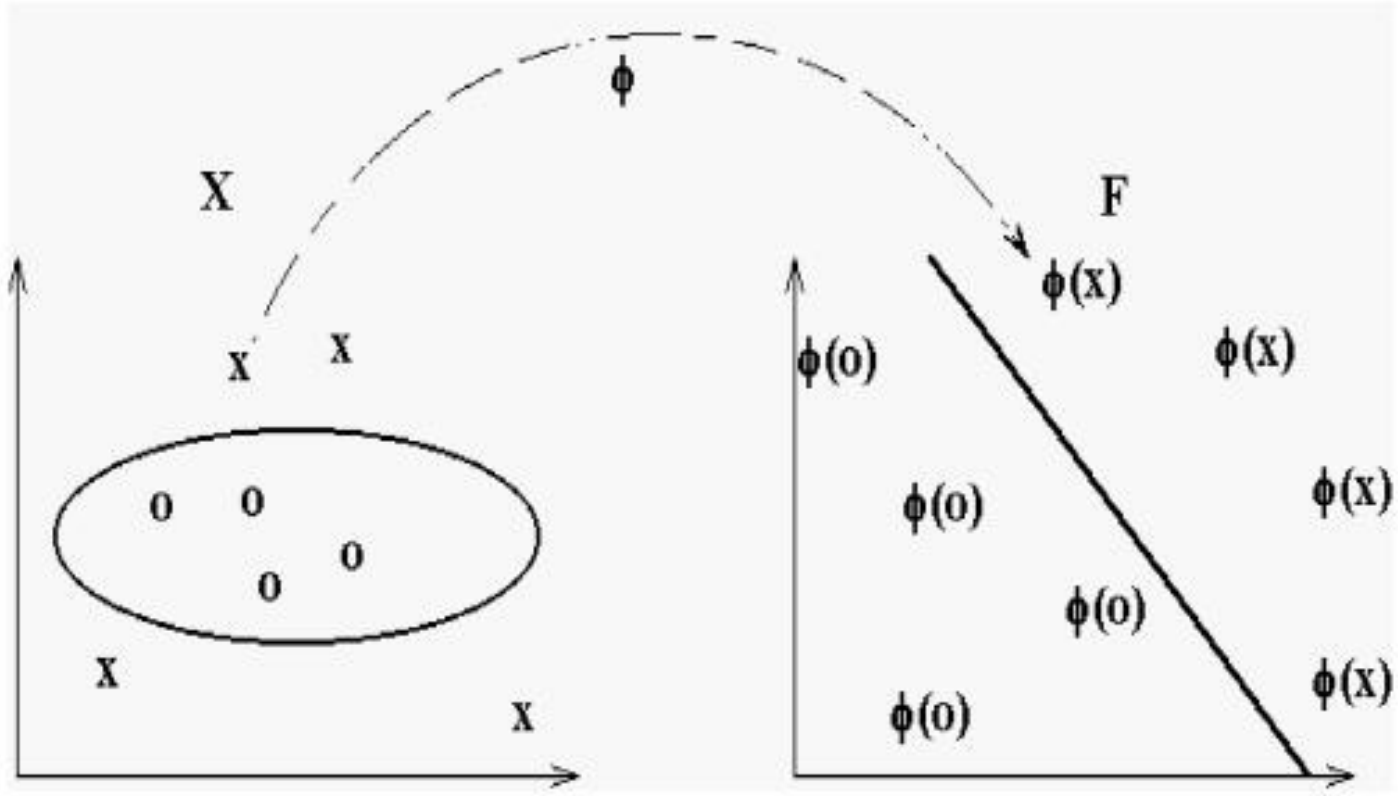




con $g(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i \, c_i \, K(\mathbf{x}_i, \mathbf{x}) + \alpha_i \, c_i$ siendo $K(\mathbf{x}_i, \mathbf{x})$ un kernel polinómico

Kernel polinomial

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



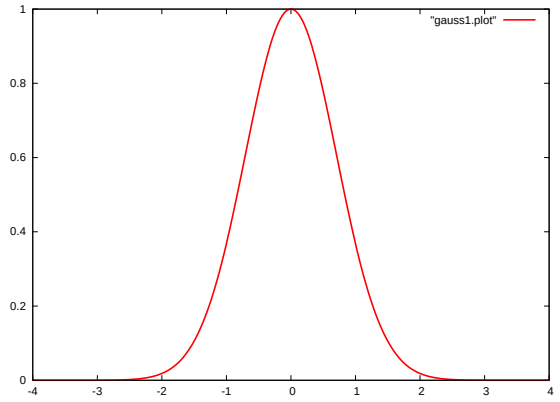
http://upload.wikimedia.org/wikipedia/commons/b/b1/Svm_8_polinomial.JPG

Kernel gaussiano

- Kernel Gaussiano:

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

- 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

Kernels generalizados

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

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

- 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 $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ 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:

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

- 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:

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

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

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:
  - $\overline{\mathbf{x}}' = W^t\overline{\mathbf{x}}$ (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: $\mathbb{C} = \{1, 2, \ldots, C\}$, $c_i \in \mathbb{C}$
- Media total: $\overline{\mathbf{x}} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i$
- Número de muestras de la clase c: n_c
- Media de clase c: $\overline{\mathbf{x}}_c = \frac{1}{n_c} \sum_{i:c_i=c}^{n} \mathbf{x}_i$
- Matriz de covarianzas de clase c: $\Sigma_c = \frac{1}{n_c} \sum_{i:c_i=c}^{n} (\mathbf{x}_i - \overline{\mathbf{x}}_c)(\mathbf{x}_i - \overline{\mathbf{x}}_c)^t$

LDA propone encontrar una matriz de proyección W que separe las medias $\overline{\mathbf{x}}_c$ 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]:

$$S_b = \sum_{c=1}^{C} n_c \, (\overline{\mathbf{x}}_c - \overline{\mathbf{x}})(\overline{\mathbf{x}}_c - \overline{\mathbf{x}})^t$$

  - 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

[1] between-class
[2] within-class

LDA

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

$$S'_b = W^t S_b W \qquad\qquad S'_w = W^t S_w W$$

- 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
- Para calcular la varianza total de una matriz se puede utilizar el operador traza Tr, que es la suma de los 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]

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

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

Problema de optimización

- Por simplicidad, optimizaremos la proyección a una única dimensión

$$\widehat{\mathbf{w}} = \underset{\mathbf{w}}{\text{argmax}} \frac{\mathbf{w}^t S_b \mathbf{w}}{\mathbf{w}^t S_w \mathbf{w}}$$

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

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

- Expresado con el multiplicador de Lagrange correspondiente

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

- 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

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

que se puede expresar mediante un sumatorio de multiplicadores de Lagrange

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

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

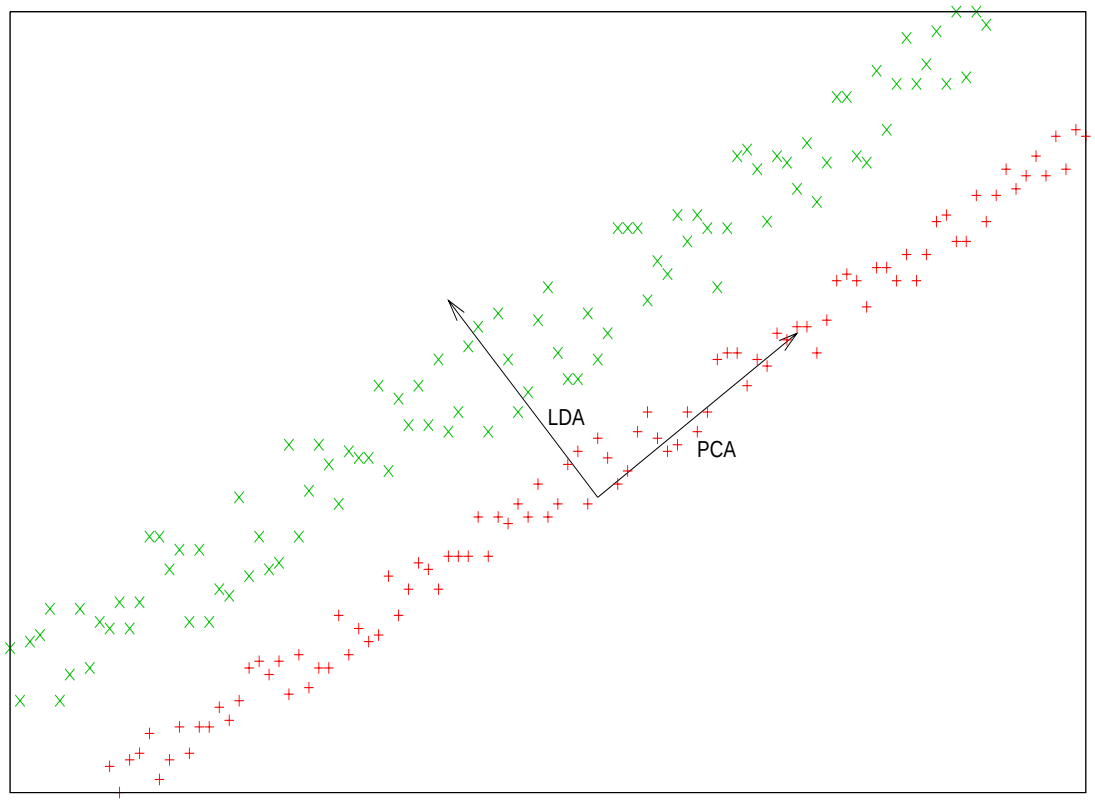
$$W_{D\times k} = \begin{pmatrix} \mathbf{w}_1 & \mathbf{w}_2 & \dots & \mathbf{w}_k \end{pmatrix}, \quad \lambda_1 > \lambda_2 > \cdots > \lambda_k$$

- 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,$\mathcal{X} = \{(\mathbf{x}_1, c_1), (\mathbf{x}_2, c_2), \cdots, (\mathbf{x}_n, c_n)\}$
- Salida: W
- Algoritmo:
  1. Calcular la media de los datos: $\overline{\mathbf{x}} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i$
  2. Calcular la media por clase: $\overline{\mathbf{x}}_c = \frac{1}{n_c} \sum_{i:c_i=c}^{n} \mathbf{x}_i$
  3. Calcular la matriz de covarianzas de clase: $\Sigma_c = \frac{1}{n_c} \sum_{i:c_i=c}^{n} (\mathbf{x}_i - \overline{\mathbf{x}}_c)\,(\mathbf{x}_i - \overline{\mathbf{x}}_c)^t$
  4. Calcular $S_w = \sum_{c=1}^{C} \Sigma_c$
  5. Calcular $S_b = \sum_{c=1}^{C} n_c\,(\overline{\mathbf{x}}_c - \overline{\mathbf{x}})(\overline{\mathbf{x}}_c - \overline{\mathbf{x}})^t$
  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:

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

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

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escola Tècnica Superior d'Enginyeria **Informàtica**

etsinf

Tema 7. Combinación de clasificadores

Percepción (PER)

Curso 2019/2020

Departamento de Sistemas Informáticos y Computación

Índice

Índice

Introducción

- Las fuentes de error de un clasificador son:
  - ***Bias*** (sesgo): asunciones erróneas, error en la selección del tipo de clasificador. Relacionado con la capacidad de ajuste del clasificador elegido a los datos.
  - ***Variance*** (varianza): dependencia de los datos de entrenamiento. Relacionado con la bondad del aprendizaje del clasificador en función de la cantidad de datos disponibles.
  - ***Noise*** (ruido): ruido inherente en los datos
- Compromiso entre *bias* y *variance* para el diseño de un buen clasificador
- Caracterización de *bias* y *variance* de los distintos clasificadores

Caracterización del error

Clasificador G como regresor (aprendido en entrenamiento): $G(x) : E \to \mathbb{R}$

Valor verdadero y: $y = F(x) + \epsilon$

- $F(x)$: función verdadera
- ϵ: ruido inherente de los datos

Representación del error como el ***valor esperado del error cuadrático***:

$$\mathbb{E}[(y - G(x))^2]$$

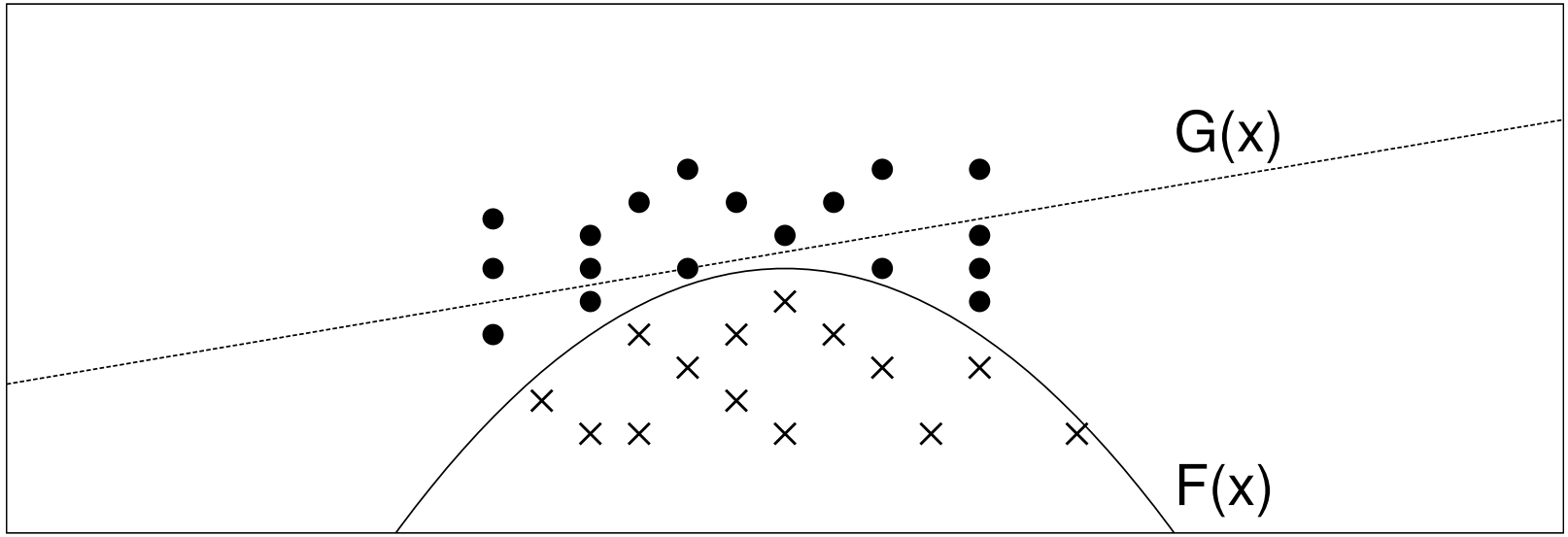
Definiendo $\overline{G(x)} = \mathbb{E}[G(x)]$, finalmente se tiene:

$$\mathbb{E}\left[(y - G(x))^2\right] = \underbrace{\mathbb{E}\left[\left(G(x) - \overline{G(x)}\right)^2\right]}_{\text{Variance}} + \underbrace{\left(\overline{G(x)} - F(x)\right)^2}_{\text{Bias}} + \underbrace{\mathbb{E}\left[(y - F(x))^2\right]}_{\text{Noise}}$$

Detalles de los calculos en documento en PoliformaT

Caracterización del error

- ***Variance***: variación de $G(x)$ según datos de entrenamiento
- ***Bias***: error del clasificador promedio, capacidad de adaptarse al entrenamiento
- ***Noise***: ruido presente en los datos



Tipos de clasificadores

- Clasificadores con *bias* alto y *variance* bajo: (p.ej., clasificador lineal)
  - Poco flexibles
  - Pocos parámetros
  - Bajo requerimiento de datos de entrenamiento
  - Clasificadores débiles (*weak learners*): apenas mejores que el aleatorio
- Clasificadores con *bias* bajo y *variance* alto: (p.ej., k-NN)
  - Muy flexibles (aprenden cualquier frontera de decisión)
  - Muchos parámetros
  - Alto requerimiento de datos de entrenamiento
  - Clasificadores fuertes (*strong learners*): *arbitrariamente* precisos
- ***Ensemble learning***: combinación de clasificadores
  - ***Bagging***:
    combinación de clasificadores fuertes modificando el entrenamiento
  - ***Boosting***:
    construcción de clasificadores fuertes a partir de clasificadores débiles

Índice

Bagging

Bagging: **B**ootstrap **Ag**regat**ing**

Clasificadores G_i a partir de variación de los datos de entrenamiento X

- Obtener X_i por *bootstrapping* desde X
- *Bootstrapping*: muestreo aleatorio con reemplazamiento
- Entrenar G_i con X_i

Combinación de clasificadores G_i por suma no ponderada

Bagging

Algoritmo Bagging:

- Entrenamiento:

  For $i = 1 \cdots M$

  Obtener X_i a partir de X

  Entrenar G_i con X_i

  End

- Clasificación:

$$G(x) = \frac{1}{M} \sum_{i=1}^{M} G_i(x)$$

Bagging se emplea en clasificadores binarios, con $\hat{c}(x) = \text{sgn}(G(x))$

Propiedades de Bagging

- ***Variance***:

$$\mathbb{E}\left[\left(G(x) - \overline{G(x)}\right)^2\right] \qquad G(x) = \frac{1}{M}\sum_{i=1}^{M} G_i(x), \textit{ variance} \text{ se reduce}$$

- ***Bias***:

$$\left(\overline{G(x)} - F(x)\right)^2 \qquad \overline{G(x)} \text{ no cambia, y } \textit{bias} \text{ no cambia}$$

- El error del clasificador generado mediante Bagging se reduce

- Bagging adecuado para combinar clasificadores fuertes (flexibles, *bias* bajo)

Índice

Boosting

- Combinación de clasificadores débiles ponderando los datos de entrenamiento
- Se dispone de un conjunto de L clasificadores débiles: $\mathcal{G} = \{G_1, \ldots, G_L\}$
- Se asumen clasificadores débiles binarios: $G_l(x) \in \{-1, 1\}$
- Conjunto de entrenamiento: $\mathcal{X} = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ con $y_n \in \{-1, 1\}$
- En cada iteración, toma $C_i \in \mathcal{G}$ de menor error sobre $\mathcal{X}$ ponderado por $w^{(i)}$
- $G(x)$ es la combinación lineal de los seleccionados hasta iteración m:

$$G(x) = G^{(m)}(x) = \sum_{i=1}^{m} \alpha_i \, C_i(x) \;\; \text{donde} \;\; C_i \in \mathcal{G}$$

Boosting

En la iteración m seleccionamos un clasificador C_m junto con su peso α_m

$$G^{(m)}(x) = G^{(m-1)}(x) + \alpha_m \, C_m(x)$$

El criterio de error E a minimizar es la pérdida exponencial en cada dato

$$E = \sum_{i=1}^{N} \exp(-y_i \, G^{(m)}(x_i)) = \sum_{i=1}^{N} \exp(-y_i \, G^{(m-1)}(x_i) - y_i \, \alpha_m \, C_m(x_i))$$

Definiendo el peso de x_i para la iteración m ($w_i^{(m)}$) como su pérdida exponencial:

$$w_i^{(m)} = \exp(-y_i \, G^{(m-1)}(x_i)) \;\; \longrightarrow \;\; E = \sum_{i=1}^{N} w_i^{(m)} \, \exp(-y_i \, \alpha_m \, C_m(x_i))$$

Se buscan C_m y α_m que minimicen E

Boosting

- Minimización respecto a C_m
  - $E \approx \sum_{y_i \neq C_m(x_i)} w_i^{(m)}$
  - Por tanto, ***se selecciona el clasificador $C_m \in \mathcal{G}$ que minimice el error de clasificación sobre los datos ponderados***

- Minimización respecto a α_m: por derivación e igualación a cero
  - Error en iteración m:

$$\epsilon_m = \frac{\sum_{y_i \neq C_m(x_i)} w_i^{(m)}}{\sum_{i=1}^{N} w_i^{(m)}}$$

  - Valor de α_m:

$$\alpha_m = \frac{1}{2} \ln \left(\frac{1 - \epsilon_m}{\epsilon_m} \right)$$

Detalles de los cálculos en documento en PoliformaT

Algoritmo AdaBoost

Entrada:

- Conjunto de entrenamiento $\mathcal{X} = \{(x_1, y_1) \ldots (x_N, y_N)\}$
- Conjunto clasificadores débiles (binarios) $\mathcal{G} = \{G_1, \ldots, G_L\}$

Proceso:

1. $w_i^{(1)} = \frac{1}{N} \quad i = 1, \ldots, N$

2. Para $m = 1 \ldots M$

2.1. $C_m = \text{argmin}_{g \in \mathcal{G}} \sum_{y_i \neq g(x_i)} w_i^{(m)}$

2.2. $\epsilon_m = \text{mín}_{g \in \mathcal{G}} \sum_{y_i \neq g(x_i)} w_i^{(m)}$

2.3. Si $\epsilon_m > 0.5 \quad$ fin

2.3. $\alpha_m = \frac{1}{2} \ln \left(\frac{1-\epsilon_m}{\epsilon_m} \right)$

2.4. $w_i^{(m+1)} = \frac{w_i^{(m)} \exp(-y_i \, \alpha_m \, C_m(x_i))}{\sum_{i'=1}^{N} w_{i'}^{(m)} \exp(-y_{i'} \, \alpha_m \, C_m(x_{i'}))}$

Salida: $G(x) = \sum_{m=1}^{M} \alpha_m \, C_m(x)$

Propiedades de AdaBoost

Boosting:

- Aprovecha el bajo *variance* de los clasificadores (débiles) combinados
- Reduce el *bias*
- Es más sensible a datos ruidosos
- En comparación con Bagging, puede comportarse peor según los datos