



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

Escola Tècnica Superior d'Enginyeria Informàtica



Tema 1. Introducción

Percepción (PER)

Curso 2019/2020

Departamento de Sistemas Informáticos y Computación

Índice

- 1 Percepción ▷ 3
- 2 Teoría de la decisión estadística ▷ 7
- 3 Teoría de la decisión estadística en interacción ▷ 13
- 4 Aprendizaje en sistemas interactivos ▷ 17
- 5 Evaluación en sistemas interactivos ▷ 21

Índice

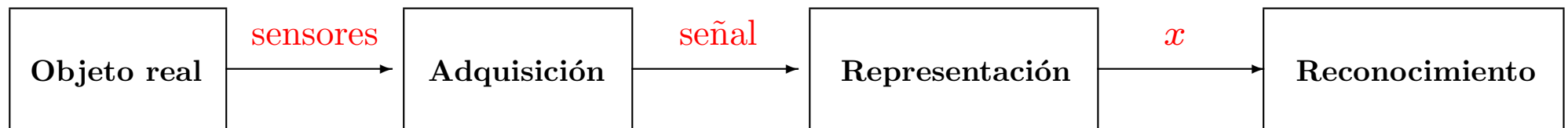
- 1 *Percepción* ▷ 3
- 2 Teoría de la decisión estadística ▷ 7
- 3 Teoría de la decisión estadística en interacción ▷ 13
- 4 Aprendizaje en sistemas interactivos ▷ 17
- 5 Evaluación en sistemas interactivos ▷ 21

Percepción

- *Percepción* como proceso cognitivo para captar información
- La percepción va estrechamente ligada al *Reconocimiento*
- Esta asignatura: *Percepción automática y Reconocimiento automático*
- Percepción automática como procesamiento informático de la información procedente de sensores
- La información es representada para su tratamiento informático posterior

Adquisición, Representación y Reconocimiento


- Percepción engloba adquisición, representación y reconocimiento



- Sensor: cámara, micrófono, escáner, . . .
- Señal: fichero de imagen, audio, vídeo, texto, . . .
- Representación (x): vector de características, cadena de símbolos, grafo, . . .

Reconocimiento

- Reconocimiento como *clasificación* $c(x) = c$ con $c \in \{1, \dots, c, \dots, C\}$
- Reconocimiento como *secuencia de símbolos* (frase), i.e., reconocimiento de texto manuscrito:

 $\xrightarrow{\text{reconocimiento}}$ **may be consumed**

- En esta asignatura veremos reconocimiento como clasificación
- Clasificación desde la teoría de la decisión estadística

Índice

- 1 Percepción ▷ 3
- 2 *Teoría de la decisión estadística* ▷ 7
- 3 Teoría de la decisión estadística en interacción ▷ 13
- 4 Aprendizaje en sistemas interactivos ▷ 17
- 5 Evaluación en sistemas interactivos ▷ 21

Teoría de la decisión estadística

- La representación del objeto \mathbf{x} a clasificar es una *variable aleatoria*
- Clasificador de Bayes (clasificador de mínimo error):

$$\hat{c}(\mathbf{x}) = \underset{c=1\dots C}{\operatorname{argmax}} P(c \mid \mathbf{x}) = \underset{c=1\dots C}{\operatorname{argmax}} P(c) p(\mathbf{x} \mid c)$$

donde

$P(c)$ es la probabilidad a priori de la clase c

$p(\mathbf{x} \mid c)$ es la distribución de probabilidad condicionada a la clase c

- $p(\mathbf{x} \mid c)$ es una distribución de probabilidad sobre valores de x en la clase c
 - Ej: Modelo de Markov oculto (SIN) u otra distribución (Gaussiana, . . .)
 - Se estima a partir de datos de la clase c (entrenamiento)

Ejercicio: clasificador de Bayes

Se dispone de 1000 correos electrónicos etiquetados como *Spam* (100) y *NoSpam* (900). Asimismo, la palabra *Bingo* aparece en 40 correos de los etiquetados como Spam (S) y en 18 de los etiquetados como NoSpam (N).

Dado el siguiente correo:

"Get \$1000 Free - Try the New Slot Machines at Bingo Palace"

¿Cómo lo etiquetaríamos para minimizar la probabilidad de error?

Ejercicio: clasificador de Bayes

Sea $x = 1$ si la palabra *Bingo* está en el correo. El clasificador de Bayes sería:

$$\hat{c}(x) = \operatorname{argmax}_{c \in \{S, N\}} P(c \mid x = 1) = \operatorname{argmax}_{c \in \{S, N\}} P(c) p(x = 1 \mid c)$$

Estimamos empíricamente las distribuciones de probabilidad involucradas:

$$\begin{aligned} P(c = S) &= \frac{100}{1000} & P(c = N) &= \frac{900}{1000} \\ p(x = 1 \mid c = S) &= \frac{40}{100} & p(x = 1 \mid c = N) &= \frac{18}{900} \end{aligned}$$

Calculamos para cada clase:

$$\begin{aligned} P(c = S) p(x = 1 \mid c = S) &= \frac{100}{1000} \cdot \frac{40}{100} = 0.04 \\ P(c = N) p(x = 1 \mid c = N) &= \frac{900}{1000} \cdot \frac{18}{900} = 0.018 \end{aligned}$$

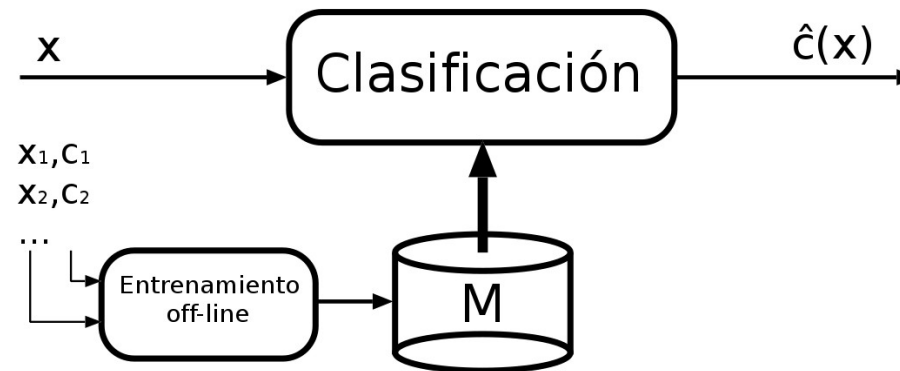
Por lo tanto, $\hat{c}(x) = S$

Algunas consideraciones

- La variable aleatoria x tendría que dar cuenta de la aparición o ausencia no sólo de la palabra *Bingo*, sino de todas las palabras del vocabulario
- De esta forma, x podría ser un vector donde cada componente del mismo estuviera asociado a una palabra distinta (Tema 2)
- Este tipo de representación conlleva una alta dimensionalidad que puede ser interesante reducir (Temas 3 y 6)
- Estos objetos pueden compararse por distancias (Tema 4)
- Existe probabilidades condicionales $p(x | c)$ bien conocidas más allá de la aparición o ausencia de un objeto (palabra) en particular (Tema 5)
- Los objetos de cada clase pueden no ser linealmente separables y podría convenir que lo fueran (Tema 6)
- Podemos usar más de un clasificador y combinarlos (Tema 7)

Clasificación con entrenamiento off-line

- Muestras de entrenamiento etiquetadas $\{(x_1, c_1), \dots, (x_n, c_n)\}$
- Estimación de modelo M ($P(c)$ y $p(\mathbf{x} | c)$) a partir de entrenamiento
- Clasificación de x de acuerdo a modelo M



- En esta asignatura solo estudiaremos el entrenamiento off-line

Índice

- 1 Percepción ▷ 3
- 2 Teoría de la decisión estadística ▷ 7
- 3 *Teoría de la decisión estadística en interacción* ▷ 13
- 4 Aprendizaje en sistemas interactivos ▷ 17
- 5 Evaluación en sistemas interactivos ▷ 21

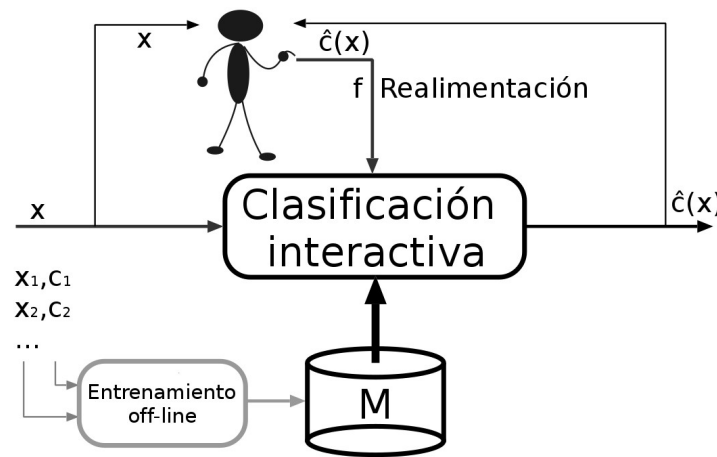
Teoría de la decisión estadística en interacción

Motivación:

- Sistema de reconocimiento (clasificación) puede cometer errores
 - Clasificación de textos, traducción automática, reconocimiento del habla
- Es necesaria la supervisión (interacción) de un usuario (humano)
- Existe supervisión humana en
 - Desarrollo: anotación de los datos empleados en estimar los modelos
 - Producción: corrección de posibles errores
 - Realimentación: considerar las correcciones para mejorar el sistema
- Es necesario adaptar la teoría de la decisión en presencia de interacción

Clasificación con realimentación sin reentrenamiento

- El usuario realiza una corrección f sobre la clasificación del sistema $\hat{c}(x)$

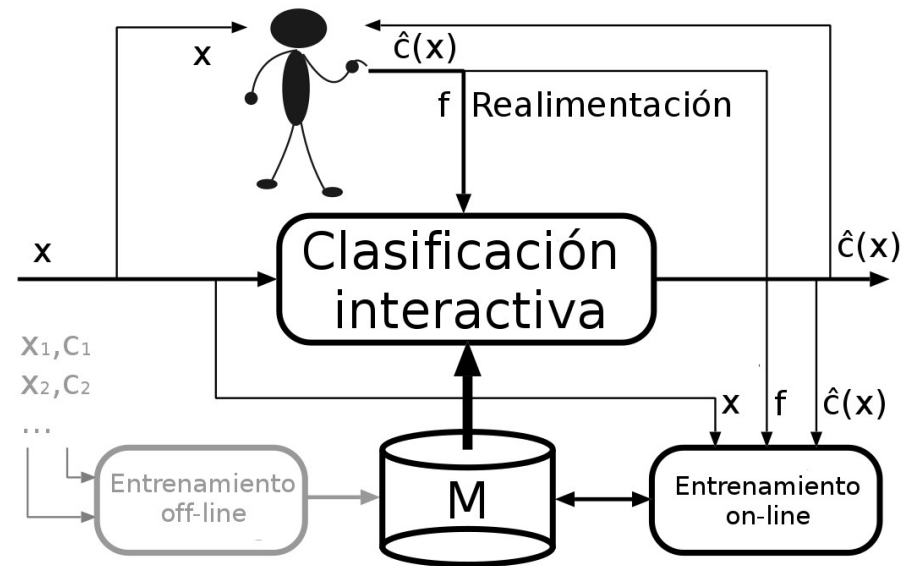


$$\hat{c}(x) = \operatorname{argmax}_{c=1 \dots C} P(c \mid x, f)$$

- El modelo M no se modifica, pero el sistema emplea la realimentación f
- Una posible realimentación es solicitar una clasificación alternativa de x

Clasificación con realimentación con reentrenamiento

- La realimentación proporciona nuevas muestras de entrenamiento etiquetadas
- Es posible reentrenar el modelo M para incorporar las nuevas muestras



- ¿Cómo contribuyen las nuevas muestras al modelo existente M en un sistema interactivo?

Índice

- 1 Percepción ▷ 3
- 2 Teoría de la decisión estadística ▷ 7
- 3 Teoría de la decisión estadística en interacción ▷ 13
- 4 *Aprendizaje en sistemas interactivos* ▷ 17
- 5 Evaluación en sistemas interactivos ▷ 21

Aprendizaje en sistemas interactivos

Algunas estrategias de entrenamiento en sistemas interactivos son:

- On-line learning
- Active learning

On-line learning

- Muestras de entrenamiento iniciales $T = \{(x_1, c_1), \dots, (x_n, c_n)\}$
- La realimentación del usuario genera un nuevo conjunto de entrenamiento $T' = \{(x_{n+1}, c_{n+1}), \dots, (x_m, c_m)\}$
- El nuevo modelo M' se obtiene de la combinación de T y T' , empleando T' para actualizar sus parámetros (“probabilidades”)
- Se debe ponderar la contribución entre T y T' a M' mediante interpolación

Active learning

- Se dispone de:
 - Conjunto reducido de muestras etiquetadas $T = \{(x_1, c_1), \dots, (x_n, c_n)\}$
 - Conjunto amplio de muestras no etiquetadas $U = \{(x_{n+1}, ?), \dots, (x_m, ?)\}$
- El etiquetado de muestras no etiquetadas es costoso
- El sistema elige $T' \subset U$ del menor tamaño posible para etiquetar por el usuario
- El error del modelo entrenado con $T \cup T'$ debe ser menor que con T
- Algunos criterios de selección de muestras a etiquetar:
 - *Uncertainty sampling*: muestras cuyo etiquetado es más incierto
 - *Expected model change*: muestras que mayor cambio causan en el modelo

Índice

- 1 Percepción ▷ 3
- 2 Teoría de la decisión estadística ▷ 7
- 3 Teoría de la decisión estadística en interacción ▷ 13
- 4 Aprendizaje en sistemas interactivos ▷ 17
- 5 *Evaluación en sistemas interactivos* ▷ 21

Evaluación en sistemas interactivos

- La evaluación convencional basada en tasa de error no es apropiada
- La evaluación se basa en *esfuerzo* o *productividad* del usuario:
 - Esfuerzo: errores corregidos (interacciones requeridas)
 - Productividad: en términos de tiempo requerido para realizar una tarea
- La **evaluación automática** de un sistema interactivo se basa en **esfuerzo**
 - Modelo de usuario en base a un conjunto de interacciones (operaciones)
 - Es una aproximación a la función objetivo a maximizar (productividad)
 - Es posible evaluar y comparar sistemas interactivos inmediatamente
- La **evaluación manual** de un sistema interactivo se basa en **productividad**
 - Corresponde con la percepción del usuario de la utilidad del sistema
 - Es costosa: reclutar usuarios, organización, control de experimentos, etc.
 - Número de sistemas a evaluar limitado, resultados no inmediatos



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

Escola Tècnica Superior d'Enginyeria Informàtica



Tema 2. Representación de objetos

Percepción (PER)

Curso 2019/2020

Departamento de Sistemas Informáticos y Computación

Índice

- 1 Introducción ▷ 3
- 2 Representación de imágenes ▷ 7
- 3 Representación de voz ▷ 37
- 4 Representación de texto ▷ 51

Índice

- 1 *Introducción* ▷ 3
- 2 Representación de imágenes ▷ 7
- 3 Representación de voz ▷ 37
- 4 Representación de texto ▷ 51

Extracción de características

- Captura y representa aquella información *discriminativa* del objeto a clasificar:
 - La similitud entre las representaciones de dos objetos debe estar en relación directa con la similitud con la que se suelen *percibir* dichos objetos
 - Objetos de la misma clase deben tener representaciones similares, mientras que objetos de clases distintas deben tener representaciones diferentes
- Representaciones invariantes a transformaciones/distorsiones usuales
 - Escalado, rotación, translación, oclusión
 - Variaciones de un mismo objeto deben tener representaciones similares
- Representación vectorial o simbólica del objeto
- Representación definida dentro de un *espacio* de representación E

Conceptos básicos

Clases: $\mathbb{C} = \{1, \dots, C\}$ si no se dice lo contrario

- Cada objeto o se manifiesta en un *Espacio Primario* o Universo U
- Suponemos que cada objeto $o \in U$ pertenece a una única *clase* $\varsigma(o)$
- \mathbb{C} denota el conjunto de posibles *identificadores* o *etiquetas de clase*

Espacio de representación: Generalmente $E = \mathbb{R}^D$ ó $E = \Sigma^*$

- Sea $\mathbf{x} = f(o)$ el resultado de la extracción de características de $o \in U$
- E incluye las representaciones del objeto o : $\{\mathbf{x} : \mathbf{x} = f(o), o \in U\} \subset E$
- f no es inyectiva: dos objetos pueden tener la misma representación

Conceptos básicos

Clasificador: $G : E \rightarrow \mathbb{C}$

- G se aprende con *muestras etiquetadas* $(\mathbf{x}_1, c_1), \dots, (\mathbf{x}_n, c_n) \in E \times \mathbb{C}$
- Recordad que $G = \{g_1(\mathbf{x}), g_2(\mathbf{x}), \dots, g_C(\mathbf{x})\}$ y $G(\mathbf{x}) \equiv \operatorname{argmax}_c g_c(\mathbf{x})$
- El objetivo es maximizar el acierto del clasificador

$$\sum_{o \in U} \delta(G(f(o)), \varsigma(o)) \quad \delta(i, j) = \begin{cases} 1 & \text{si } i = j \\ 0 & \text{si } i \neq j \end{cases}$$

Índice

- 1 Introducción ▷ 3
- 2 *Representación de imágenes* ▷ 7
- 3 Representación de voz ▷ 37
- 4 Representación de texto ▷ 51

Representación de imágenes

- Imagen: soporte de uno de los medios más importantes del sistema perceptivo
- Aplicaciones:
 - Reconocimiento óptico de caracteres (OCR)
 - Reconocimiento de huellas dactilares
 - Reconocimiento facial
 - Reconocimiento de elementos en imágenes
 - Robótica
 - ...
- Reto: gran cantidad de información a manejar en algunas ocasiones
- El procesado de imágenes tiene interés *per se*, con independencia de su uso para el reconocimiento

Adquisición

- **Imagen:** Función $f(x, y)$ es el *brillo* en cada punto de *coordenadas* x, y
- **Imagen Digital:** $f(x, y)$ discretizada en su *dominio* (coordenadas) y *rango* (brillo)

Muestreo:

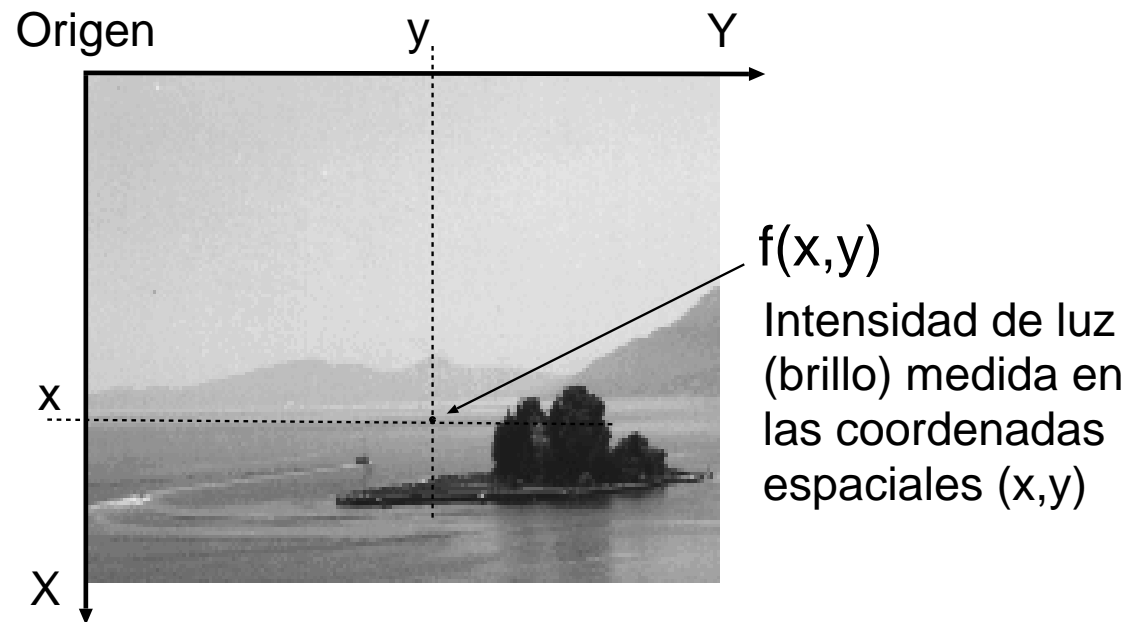
Discretización del *dominio*

Resolución: píxeles/pulgada

Cuantificación:

Discretización del *rango*

Niveles o colores: bits/píxel



Adquisición: escáner

Scanner plano



Resolución óptica: 600 ppp

Modos de exploración:

1 bit (blanco y negro)

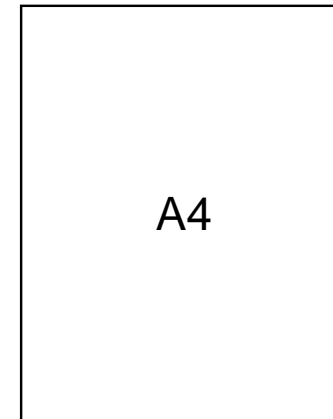
4 bits (16 niveles de gris)

8 bits (256 niveles de gris)

24 bits (16,7 millones colores)

8,27 p (21 cm)

11,69 p (29,7 cm)



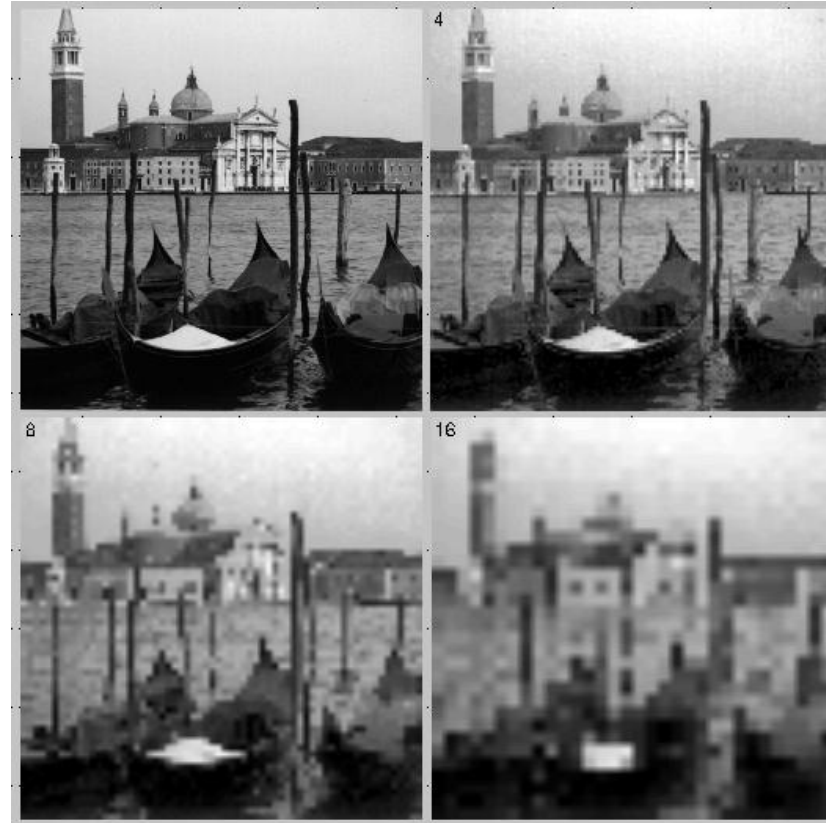
A 100 ppp y 8 bits:

*827 pixels/linea x 1169 lineas x 1 byte/pixel
= 1 Mbyte*

Límites de resolución

Imagen original

Muestreada 128 ppp



Muestreada 64 ppp

Muestreada 32 ppp

Frecuencia espacial = $\frac{T_r}{T_d}$ T_r : tamaño referencia, T_d : tamaño detalle

<https://www.ucalgary.ca/pip369/mod4/spatial/frequency1>

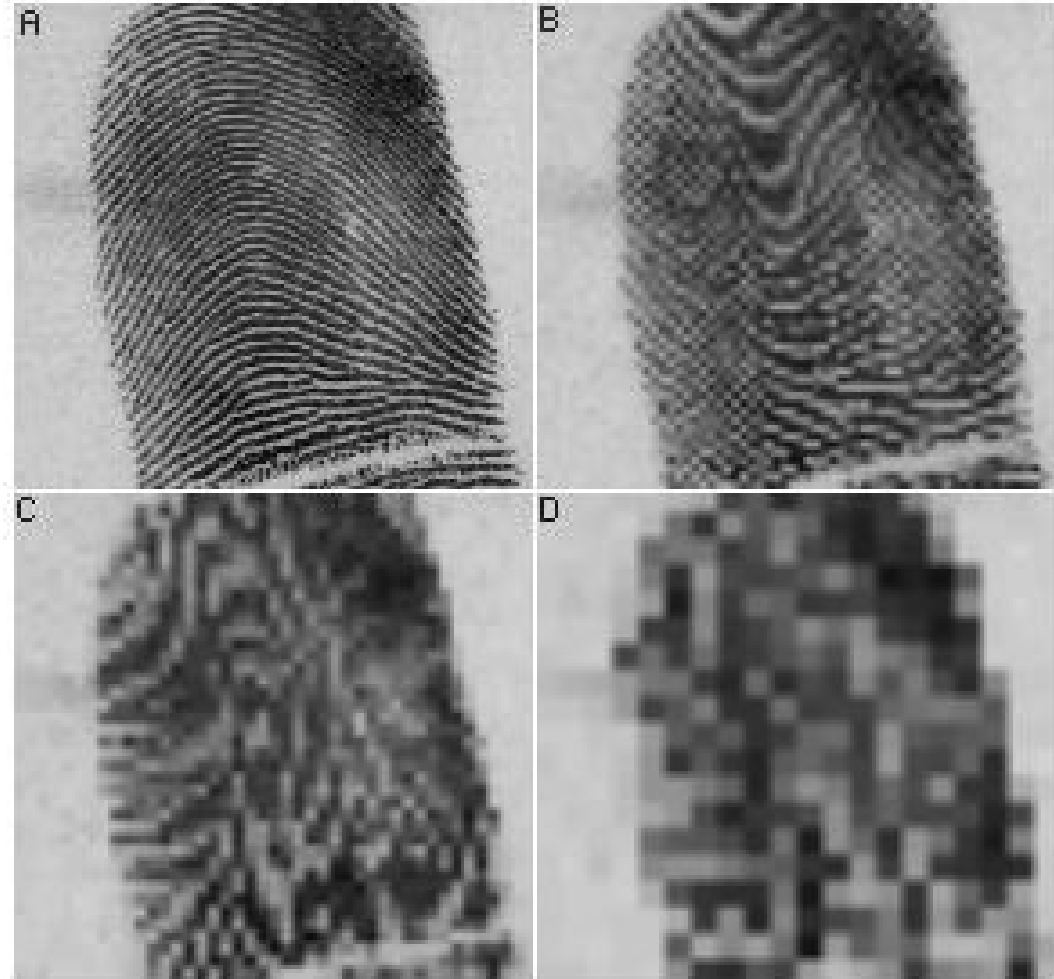
Aliasing y teorema del muestreo

- Imagen Original (1x1 pulgadas)
- Frecuencia espacial de las periodicidades (“detalles”) de interés más finas: $P \approx 50ppp$
- Imágenes Digitales: A:140ppp, B:70ppp, C:44ppp, D:22ppp

- **Frecuencia de Nyquist**
(Teorema del Muestreo):

Si P es la frecuencia espacial de la periodicidad más fina en una imagen y F es la resolución de muestreo, la imagen original sólo podrá ser fielmente reproducida si:

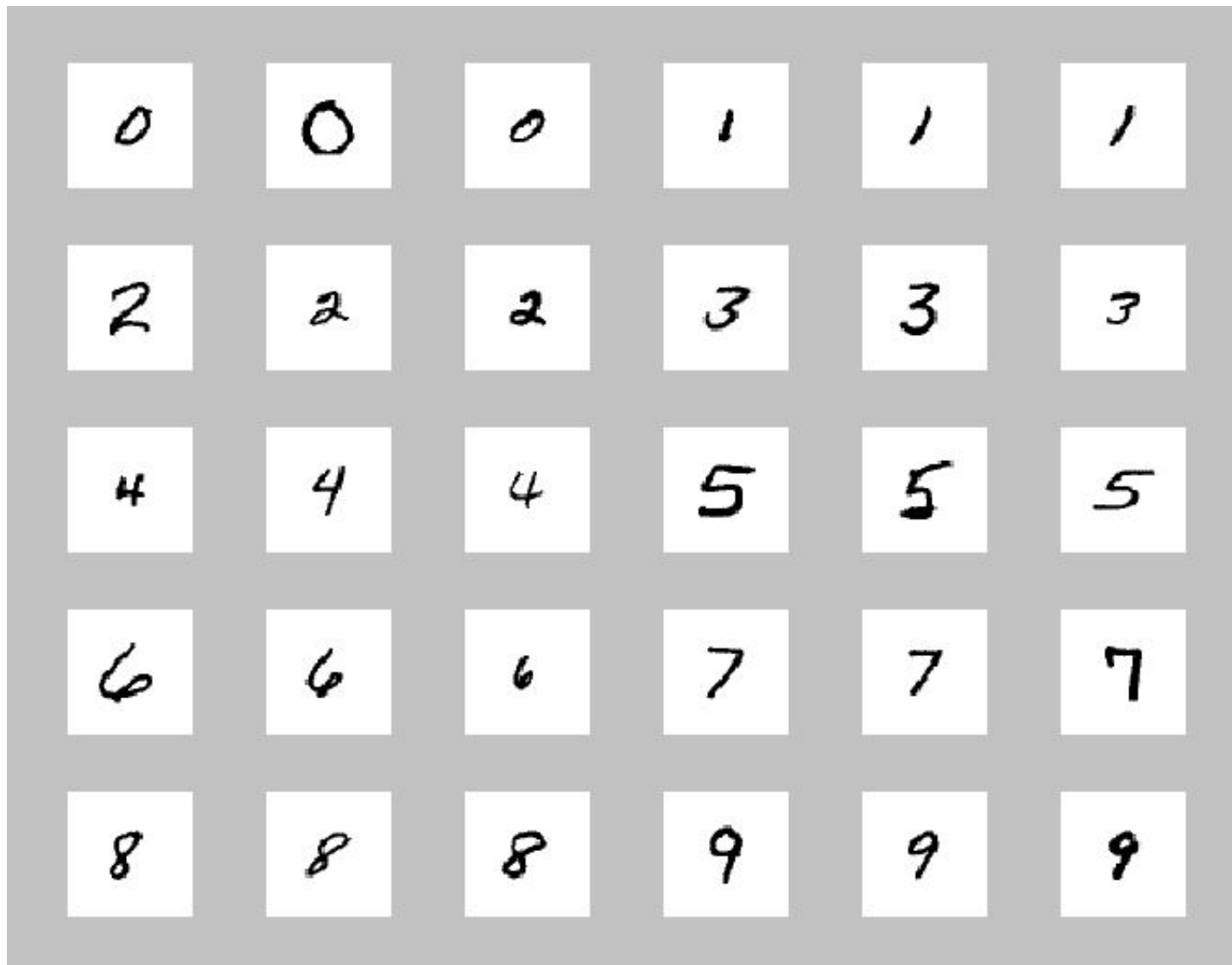
$$F > 2P$$



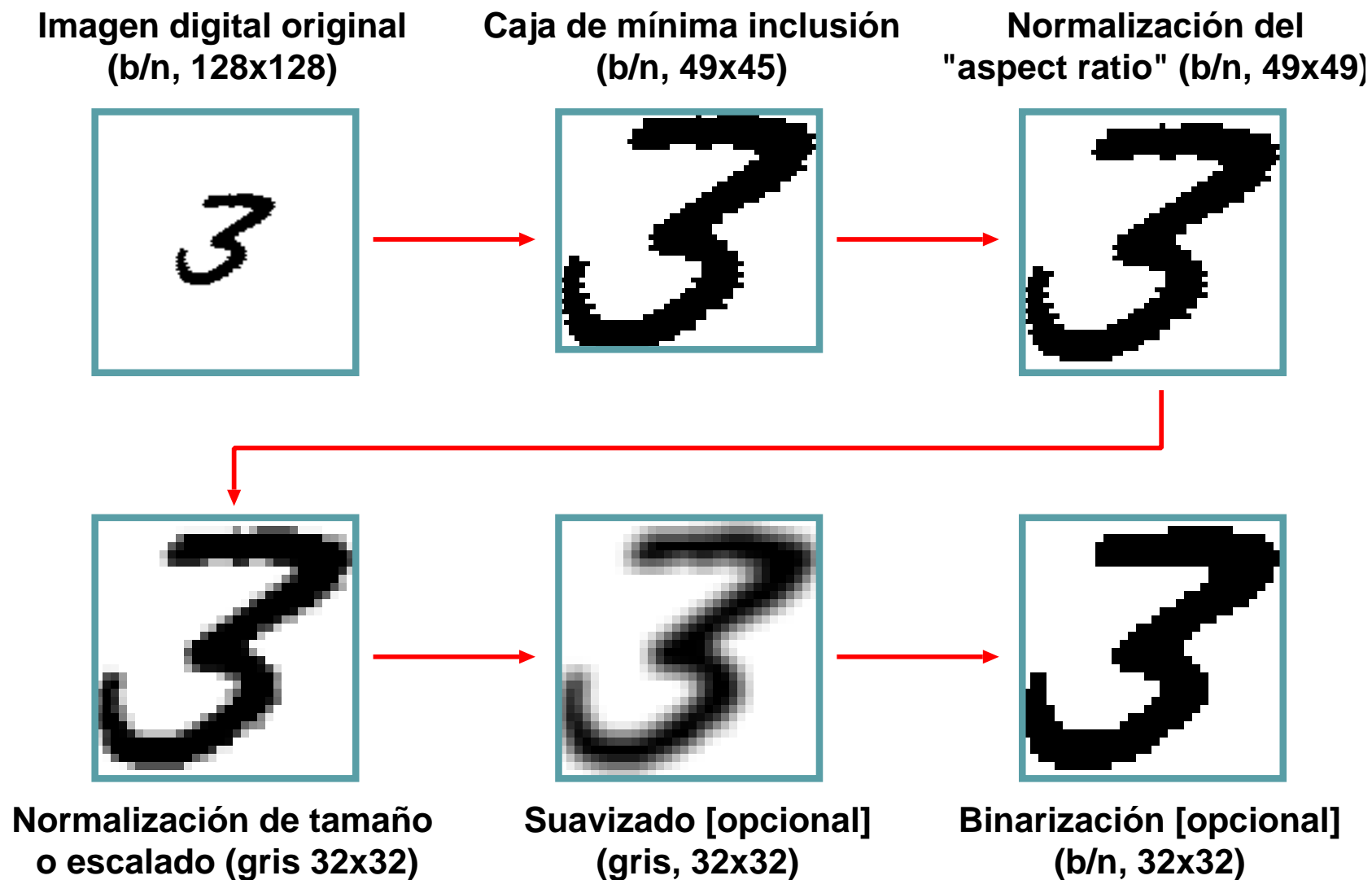
Reconocimiento de caracteres manuscritos (OCR)

- El objetivo es reconocer *texto manuscrito*
- Reconocimiento de caracteres *aislados* versus *texto continuo*
- Sistemas “On-Line” y “Off-Line”
- La tecnología actualmente disponible permite alcanzar prestaciones cercanas a las humanas en reconocimiento de caracteres aislados
- Sistemas comerciales ya disponibles con buenas prestaciones para caracteres *impresos* aislados y con prestaciones aceptables para caracteres manuscritos

Reconocimiento de caracteres manuscritos (OCR)



Preproceso de imágenes en OCR



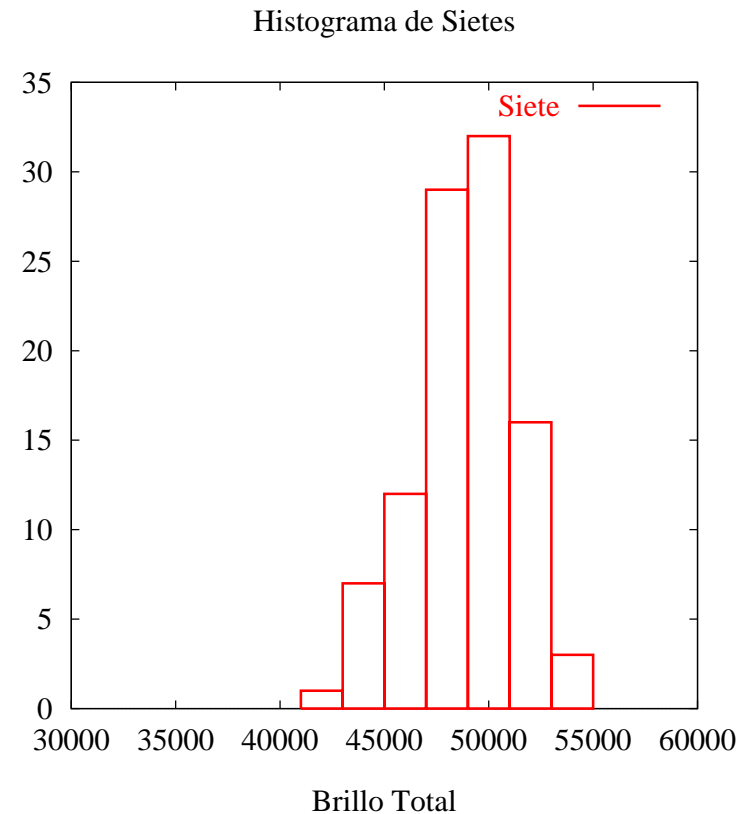
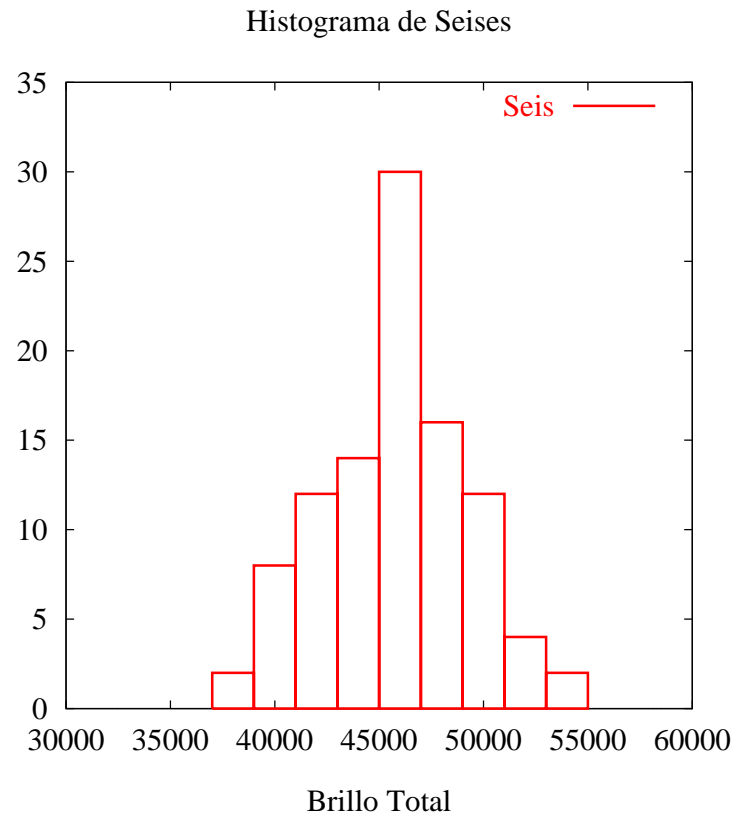
Extracción de características en OCR

Ejemplos de dígitos manuscritos “6” y “7” normalizados

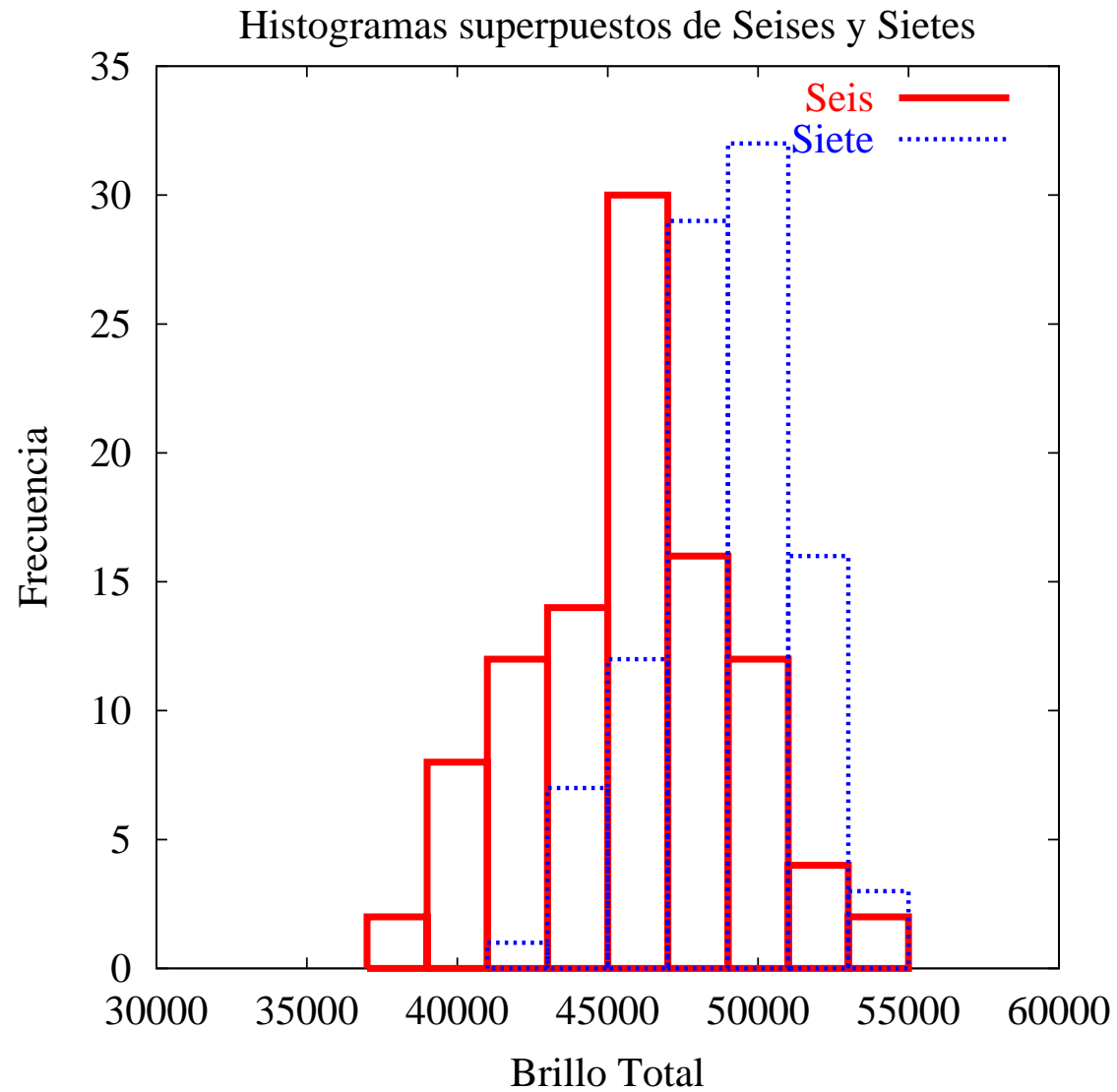


Extracción de características en OCR

Representación en 1 dimensión (histogramas de brillos)

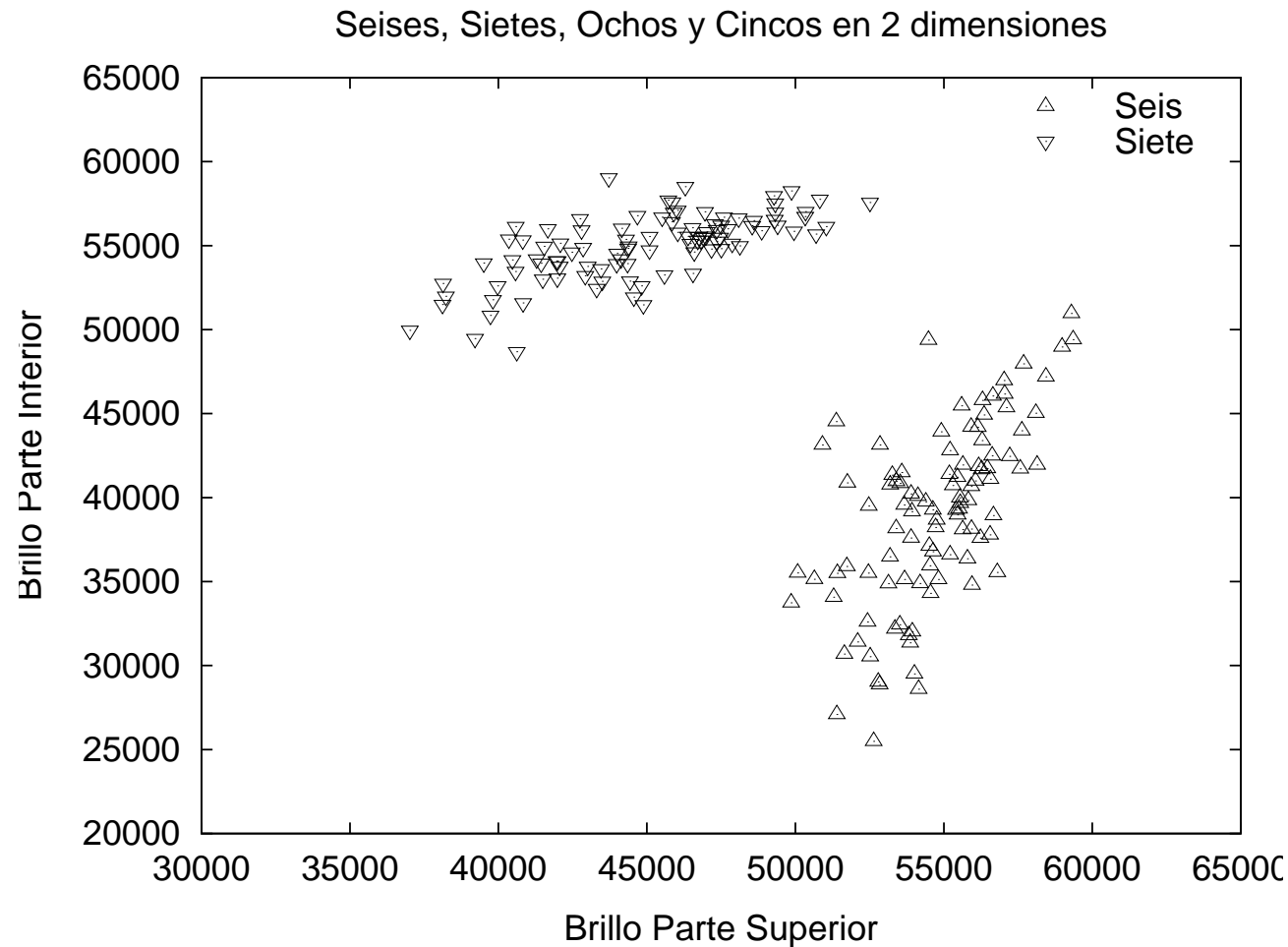


Extracción de características en OCR



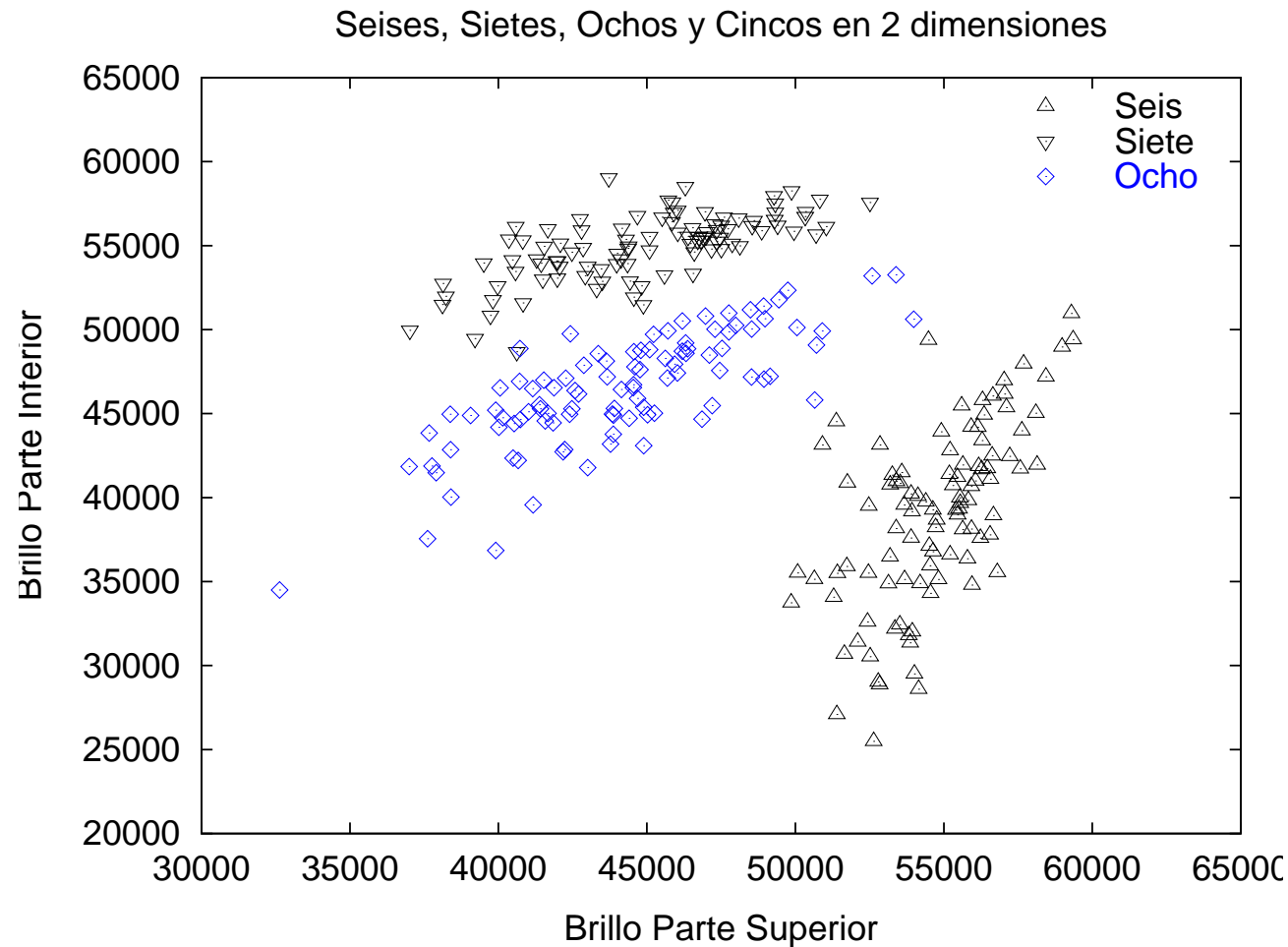
Extracción de características en OCR

Representación en 2 dimensiones (brillo superior frente a inferior)



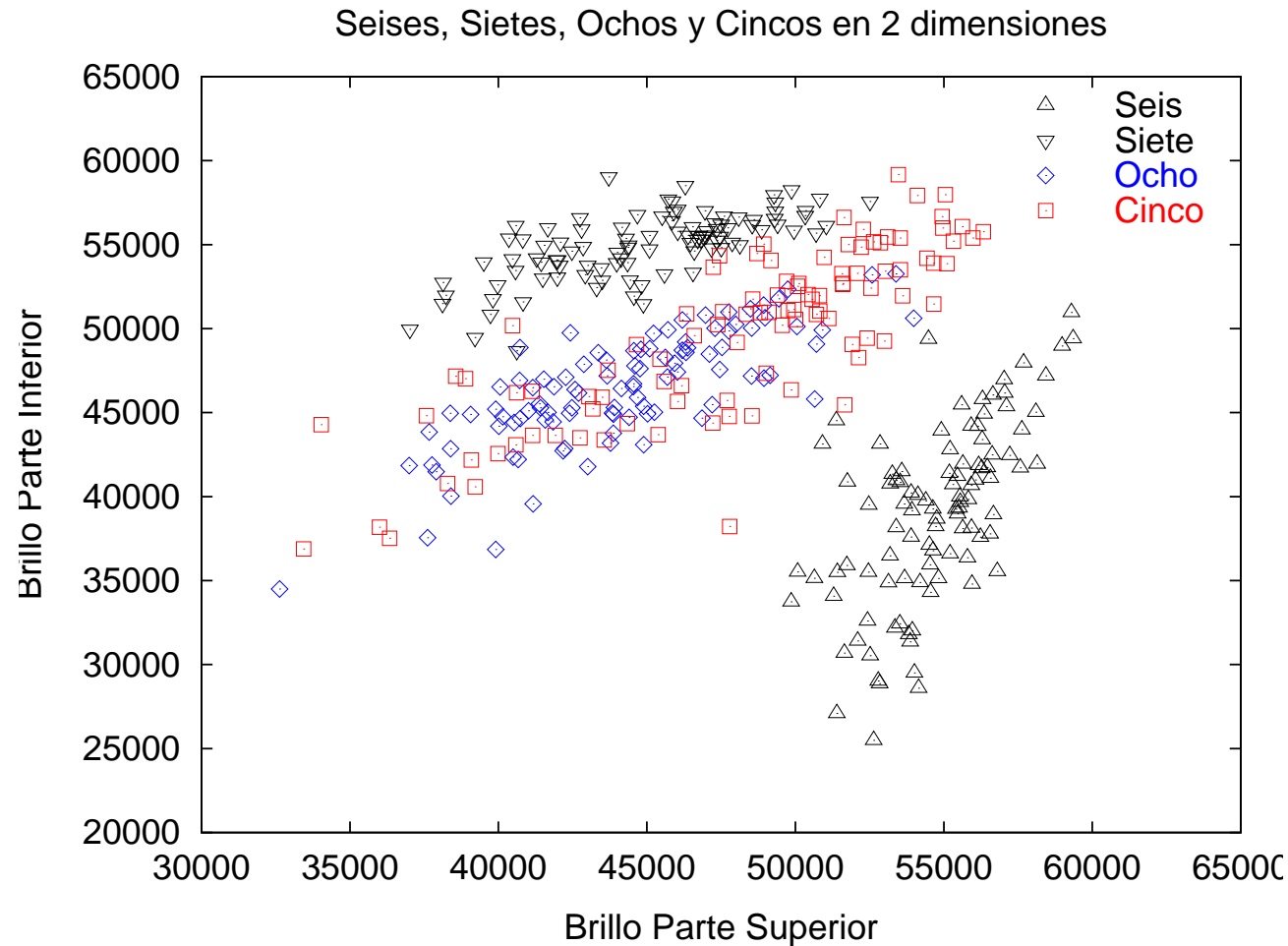
Extracción de características en OCR

Representación en 2 dimensiones (brillo superior frente a inferior)



Extracción de características en OCR

Representación en 2 dimensiones (brillo superior frente a inferior)

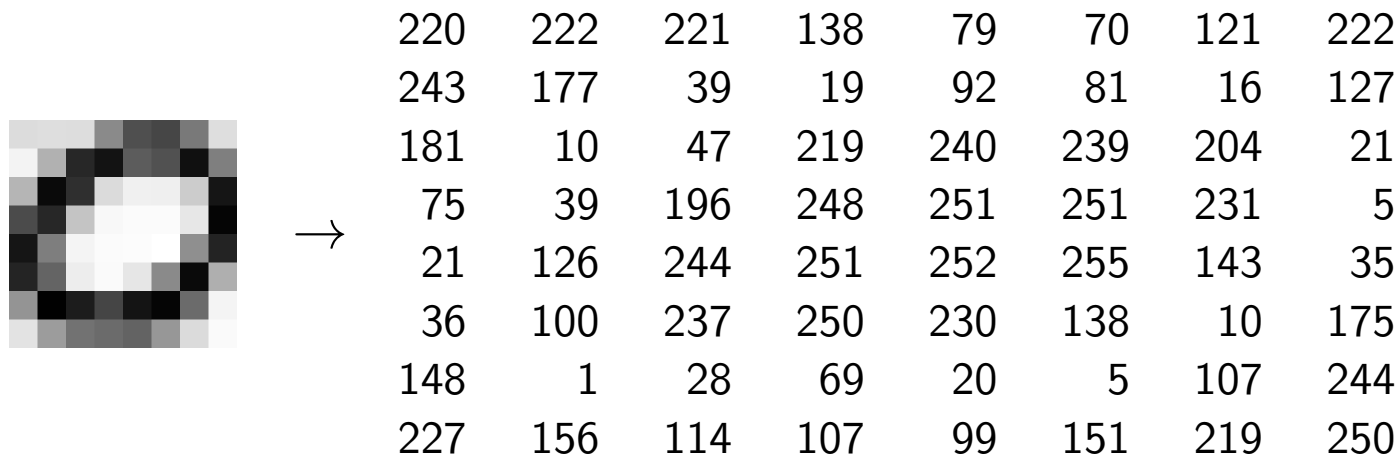


Solapamiento de clases. Posible solución: aumentar la dimensionalidad

Extracción de características en OCR

Una técnica de extracción de características en OCR es la representación **directa**:

- Preproceso
- Normalización a un tamaño vertical y horizontal fijo $I \times J$.
- Generación de imagen en escala grises (formato PGM).
- El nivel de gris de cada píxel es una componente de la representación vectorial

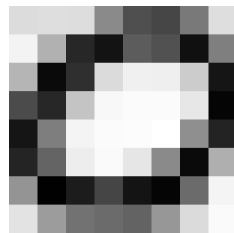


Vector de características de 8×8 componentes: (220, 222, 221, 138, . . . , 219, 250)

Extracción de características en OCR

Otra posibilidad es la representación **por histograma**:

- Tantas componentes como niveles de gris (usualmente, 256)
- La componente i es el número de píxeles con nivel de gris i en la imagen



220	222	221	138	79	70	121	222
243	177	39	19	92	81	16	127
181	10	47	219	240	239	204	21
75	39	196	248	251	251	231	5
21	126	244	251	252	255	143	35
36	100	237	250	230	138	10	175
148	1	28	69	20	5	107	244
227	156	114	107	99	151	219	250

Nivel	0	1	2	3	4	5	6	...	254	255
Número de píxeles	0	1	0	0	0	2	0	...	0	1

Vector de características de 256 componentes: $(0, 1, 0, 0, 0, 2, 0, \dots, 0, 1)$

Extracción de características en OCR

Cada representación ocupa un tamaño distinto en memoria

Dados n píxeles y l niveles de gris

- Representación directa: $n \cdot \left\lceil \frac{\log_2 l}{8} \right\rceil$ bytes

$$(220, 222, 221, 138, \dots, 219, 250) \rightarrow 64 \cdot 1 = 64 \text{ bytes}$$

- Representación por histograma: $l \cdot \left\lceil \frac{\log_2(n+1)}{8} \right\rceil$ bytes

$$(0, 1, 0, 0, 0, 2, 0, \dots, 0, 1) \rightarrow 256 \cdot 1 = 256 \text{ bytes}$$

Extracción de características: métodos locales

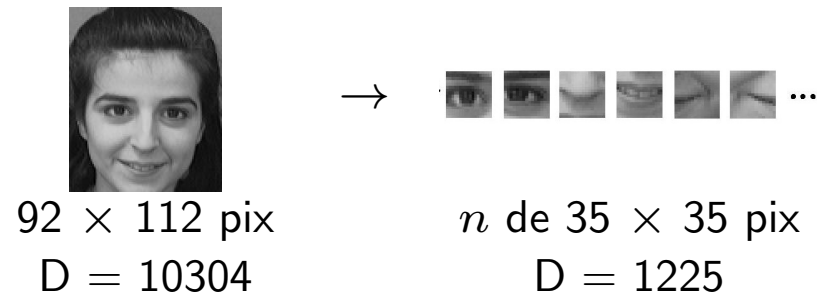
- Globalmente dos imágenes del mismo objeto pueden diferir



- Localmente existen partes (*patches*) semejantes
- Demo en PoliformaT

Extracción de características: métodos locales

- Cada imagen es representada por varias partes de la misma
- Se escogen ventanas de la imagen que sean informativas (discriminativas); por ejemplo, ventanas con alta varianza en niveles de grises

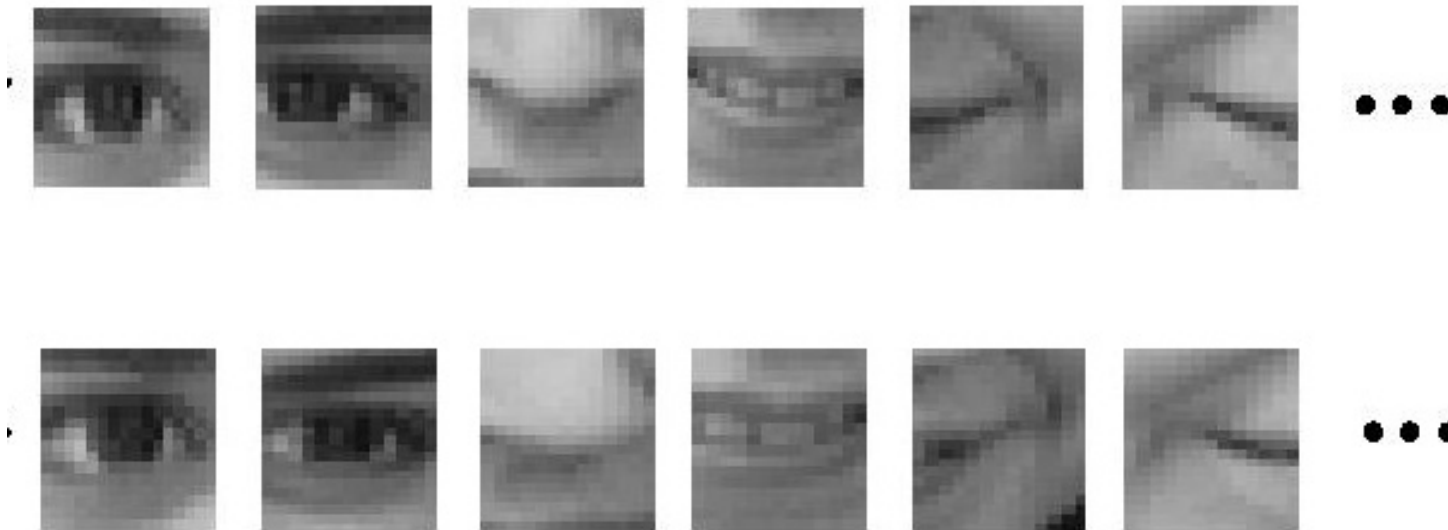


- Un objeto se representa por un conjunto de CL (vectores)
- Se obtienen representaciones de una dimensionalidad menor
- No se almacena la posición de la característica local dentro de la imagen
- Las representaciones locales son invariantes a la traslación



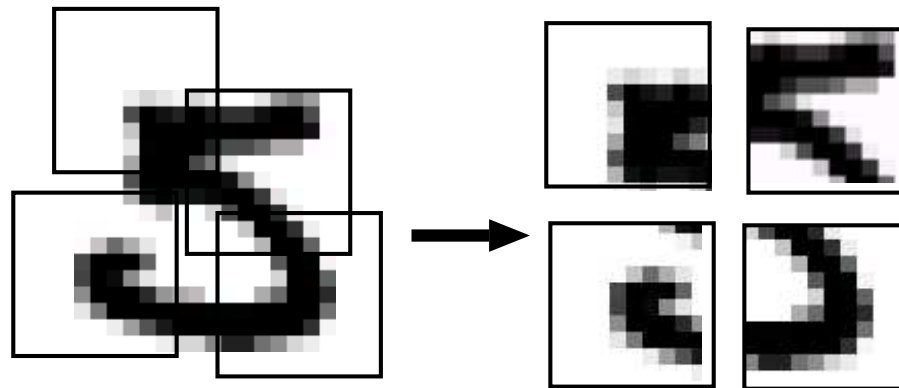
Reconocimiento facial con métodos locales

- Las CL son útiles cuando el objeto está formado por estructuras más sencillas
- Cara = cejas + ojos + nariz + boca + barbilla + contorno
- Representación global: mucha diferencia entre ambas imágenes
- Representación local: sólo varía la posición relativa de las partes
- Apariencia local: las CL extraídas son muy similares:



OCR con métodos locales

- Imagen original 20×20 píxeles (vector 400-dimensional)



- Representación local: 4 CL de 11×11 píxeles (4 vectores 121-dimensionales)
- Se aprovecha la invarianza a la traslación de las partes discriminantes

Puntos de interés

- Puntos de interés: píxeles de los cuales extraer CL
- Criterio para definir puntos de interés:
 - Deben de poder ser formalmente (matemáticamente) definidos
 - La información *local* alrededor debe de ser discriminativa
 - En algunas aplicaciones, robustas a ciertas transformaciones: iluminación, perspectiva, distorsión, . . .
- Basados en información e invarianza
 - Detectores de contorno
 - Detectores de esquinas
- Basados en exploración espacial de la imagen
 - Extracción a partir de una rejilla
 - Extracción aleatoria

Puntos de interés: detectores de contorno

- Evitan obtener CL de zonas homogéneas de la imagen
- Se detectan los contornos y se umbralizan
- Puntos de interés con alta respuesta al detector de contorno (alta varianza)



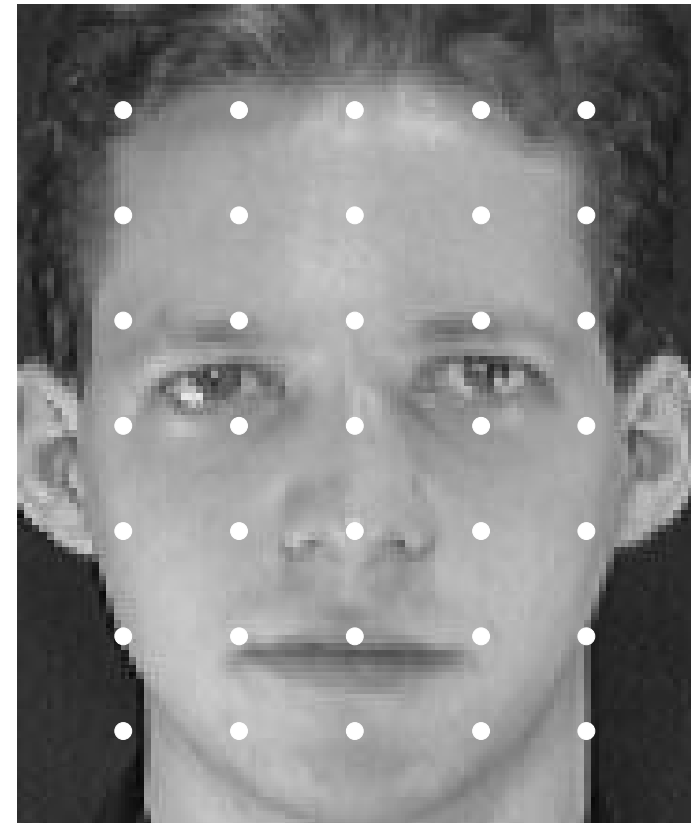
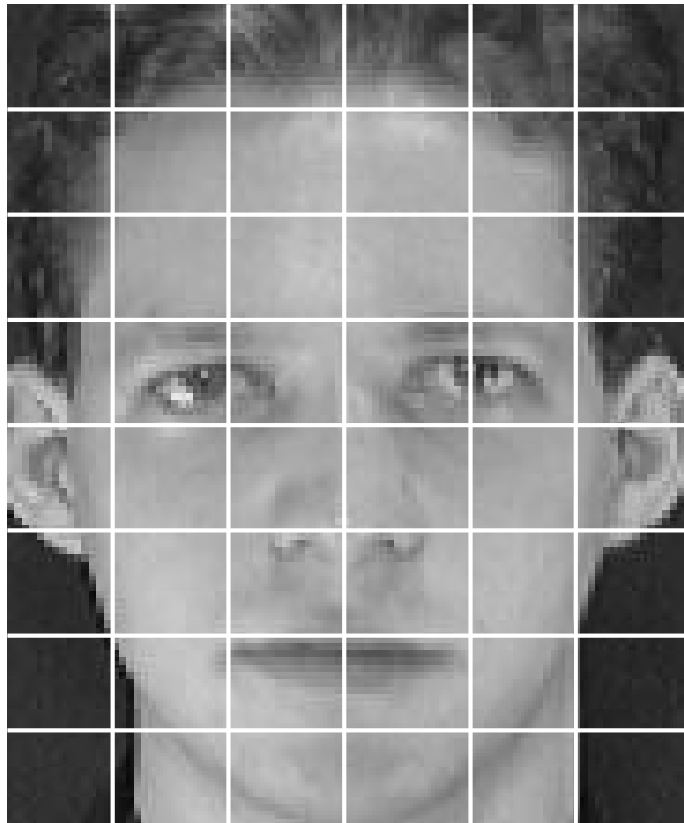
Puntos de interés: detectores de esquinas

- Más restrictivos que el detector de contorno
- Puntos de interés con alta respuesta a un detector de esquinas
- Se usan habitualmente en tareas de detección de objetos



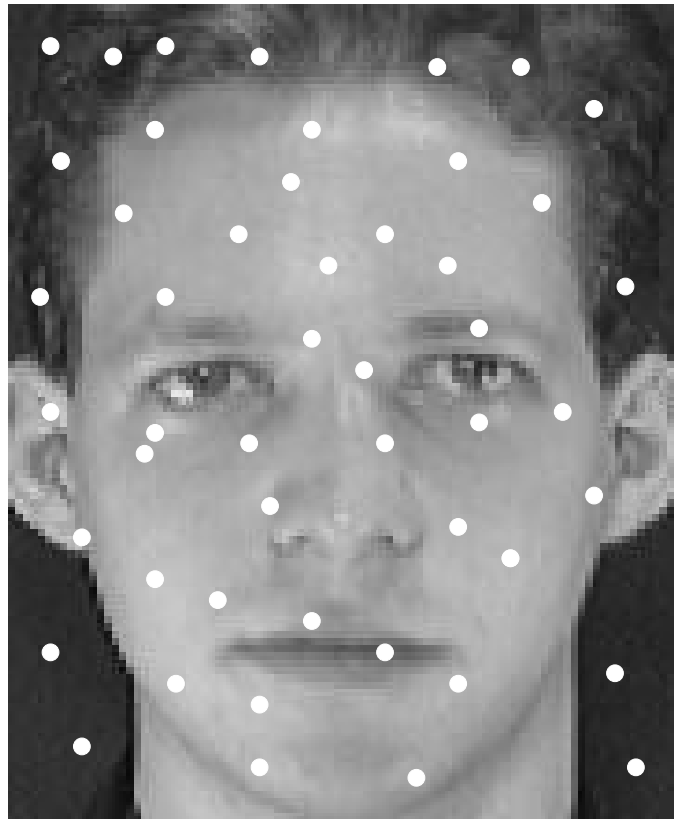
Puntos de interés: extracción por rejilla

- Puntos de interés: puntos de unión de una rejilla
- Puntos espaciados en un número de píxeles fijo, tanto horizontal como vertical (puede ser distinto en cada eje)



Puntos de interés: extracción aleatoria

- Puntos de interés: subconjunto aleatorio de los píxeles de la imagen
- Suele ser un buen complemento a la extracción por rejilla



Extracción de características: métodos locales

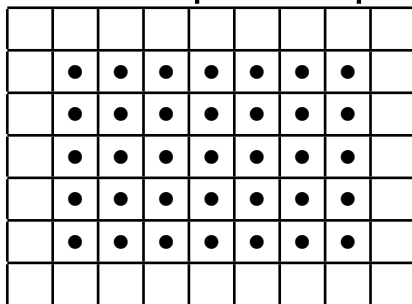
La representación por CL ocupa una memoria dependiente de:

- Número de puntos de interés/ventanas tomado (n)
- Tamaño (en bytes) de la representación de cada ventana (s)

n : dependiente de extracción y tamaño de ventana \rightarrow Tamaño final:
 s : como en extracción global $n \cdot s$ bytes

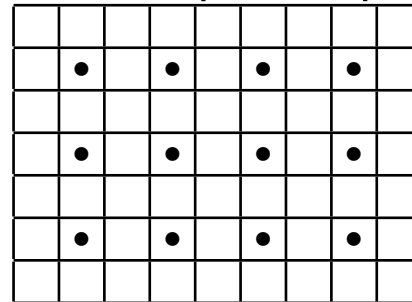
Ejemplo: extracciones por rejilla (ventana, desplazamiento horizontal y vertical)

3×3 , 1 píx., 1 píx.



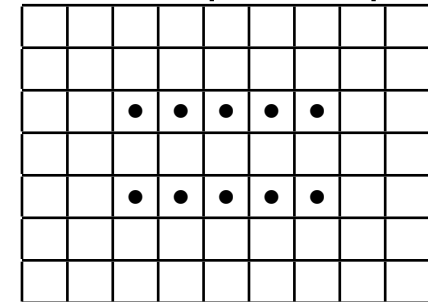
$$n = (7 - 3 + 1) \cdot (9 - 3 + 1) = 35$$

3×3 , 2 píx., 2 píx.



$$n = \left\lceil \frac{(7-3+1)}{2} \right\rceil \cdot \left\lceil \frac{(9-3+1)}{2} \right\rceil = 12$$

5×5 , 1 píx., 2 píx.

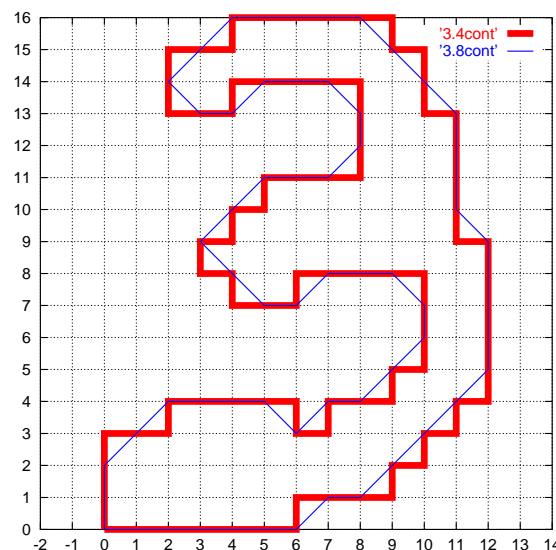
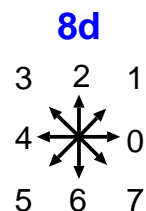
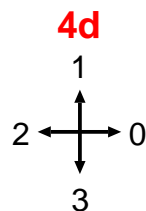
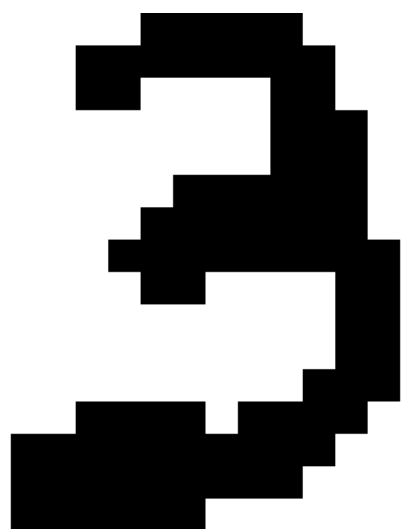


$$n = \left\lceil \frac{(7-5+1)}{2} \right\rceil \cdot (9 - 5 + 1) = 10$$

Representación estructural: códigos de contorno

Aproximación estructural/sintáctica al RF: objetos compuestos por la concatenación de *primitivas* (subobjetos elementales), representación por *modelos sintácticos* (gramáticas)

Extracción de primitivas basada en *códigos de contorno de 4 y 8 direcciones*

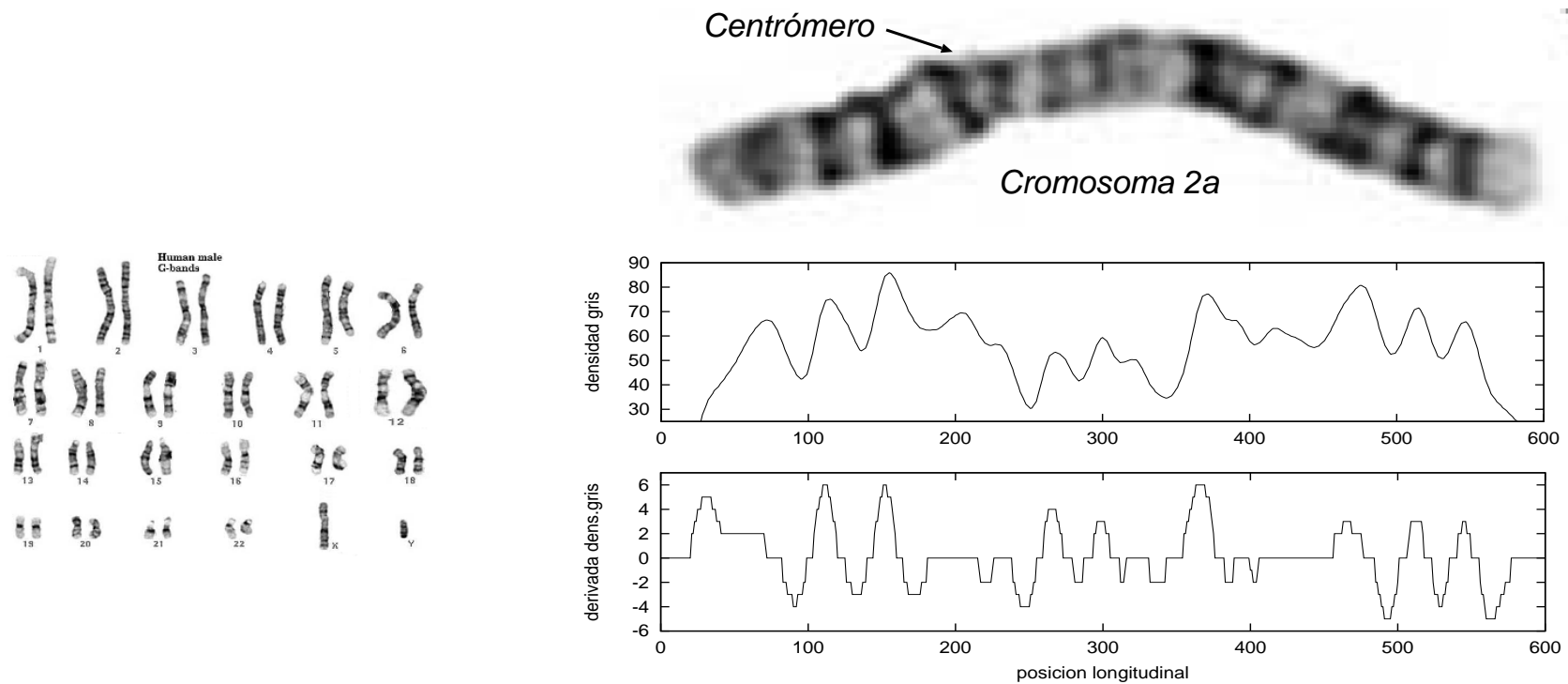


4d: 00000303303333033333232322232222221110010000301001011122223221210101000111222232211001

8d: 0000777666766665555454444442211000710112344543311001234454311

Representación estructural: otras representaciones

Representación estructural de un cromosoma



"====CDFDCBBBBBBBA==bcd==DGFB=bccb== ==cffc=CCC==cdb==BCB==dfdcb===="

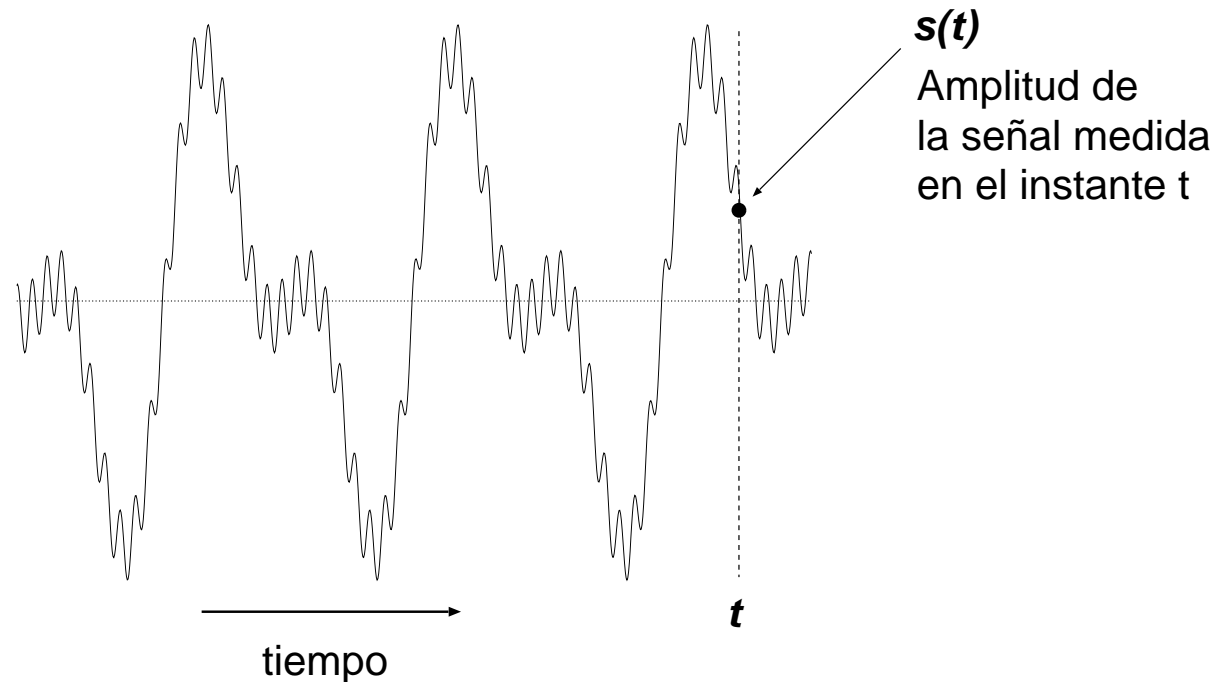
Cadena de Primitivas

Índice

- 1 Introducción ▷ 3
- 2 Representación de imágenes ▷ 7
- 3 *Representación de voz* ▷ 37
- 4 Representación de texto ▷ 51

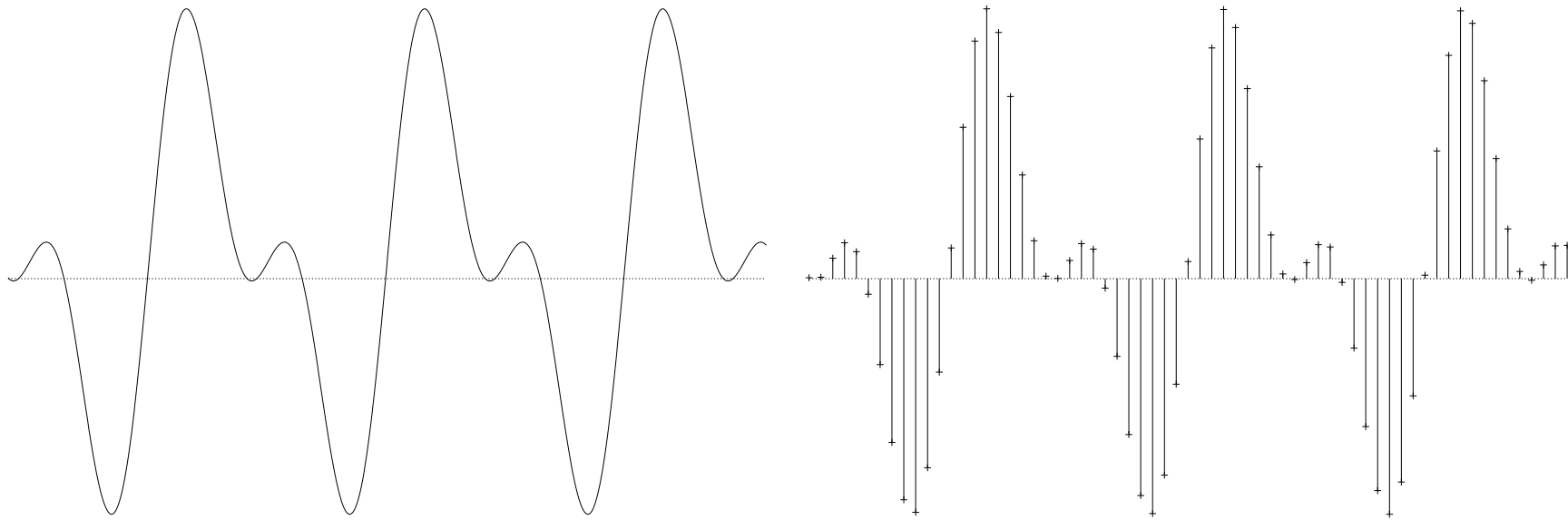
Adquisición y preprocesado

- Señal acústica: función temporal de variaciones de *amplitud* de la presión del aire $s(t)$
- Digitalización: discretización de $s(t)$ a nivel:
 - Temporal (dominio): *muestreo*
 - Amplitud (rango): *cuantificación*



Adquisición y preprocesado

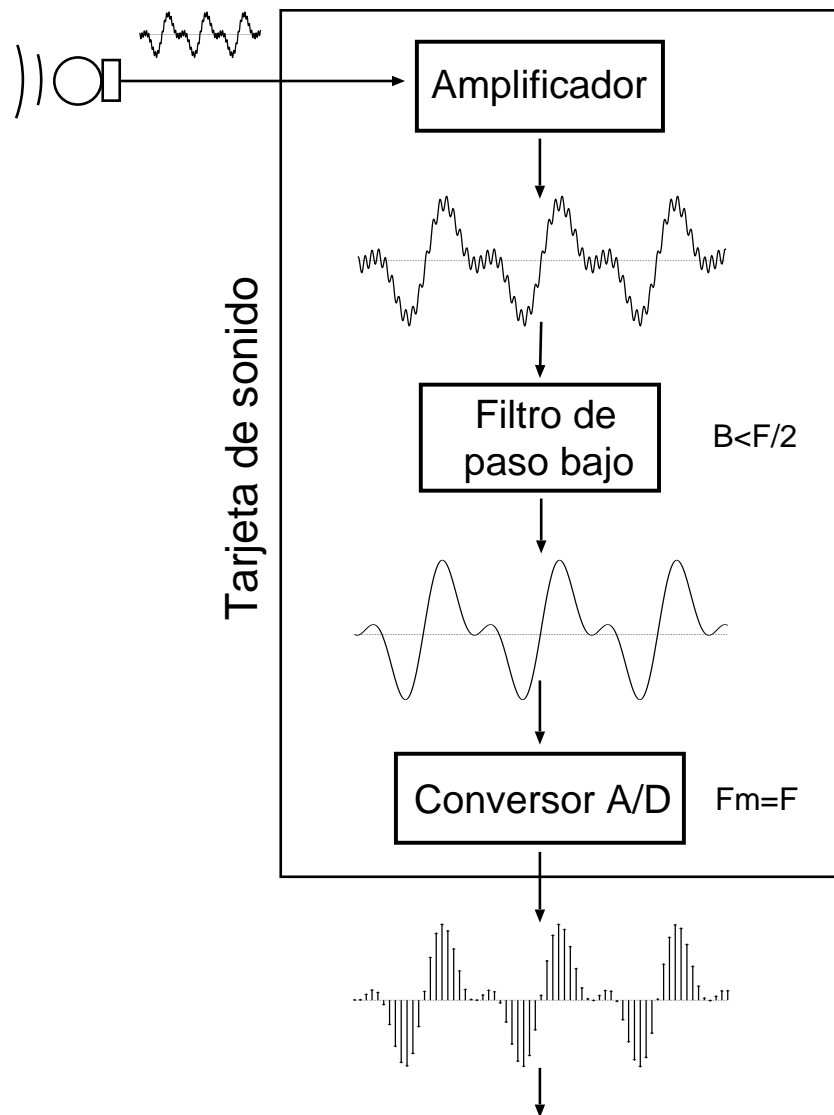
Ejemplo de **muestreo**:



Teorema del muestreo: para reconstruir una señal de ancho de banda B , la frecuencia de muestreo F_m debe cumplir $F_m > 2 \cdot B$

Cuantificación: discretizar valores en el rango de la amplitud a una cierta representación digital (número de bits)

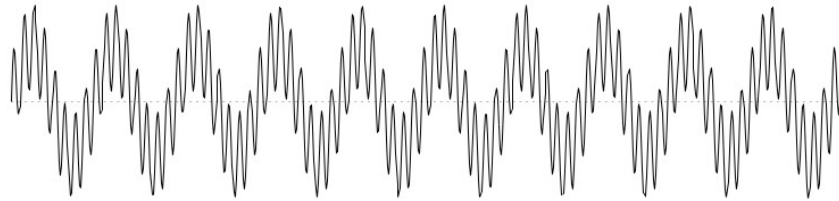
Adquisición y preprocesado



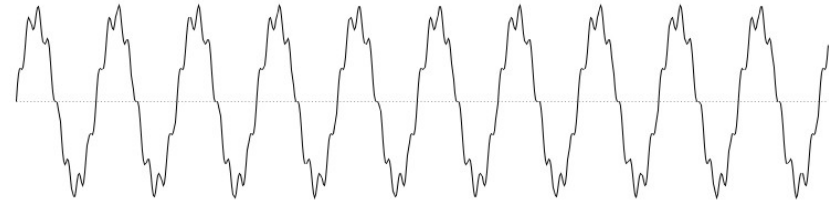
	Voz telefónica $B=3.6\text{kHz}$	Voz Calidad $B=8\text{KHz}$	Audio CD (HI-FI) $B=20\text{kHz}$
Muestreo (kHz)	8	16	44.1
Cuantificación (bits)	8 (12)	16 (12)	2x16 (20)
Flujo de datos (Mbytes/hora)	28.1	112.5	620.2
Segundos en 1 Mbyte	128	32	5.8

Adquisición y preproceso

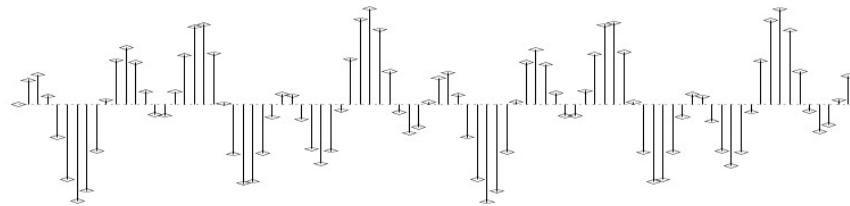
Violación del teorema de muestreo



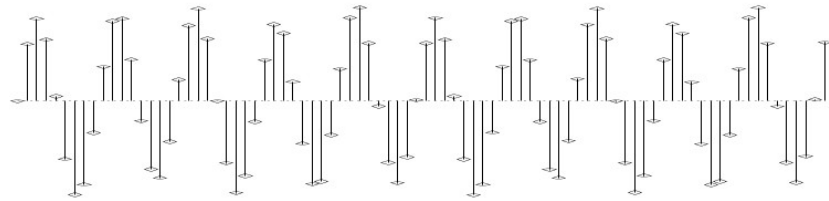
Señal original (sin filtrar): $F_0=600\text{hz}$; $F_1=4800\text{hz}$



Señal original (filtrada): $F_0=600\text{hz}$; $F_1=4800\text{hz}$



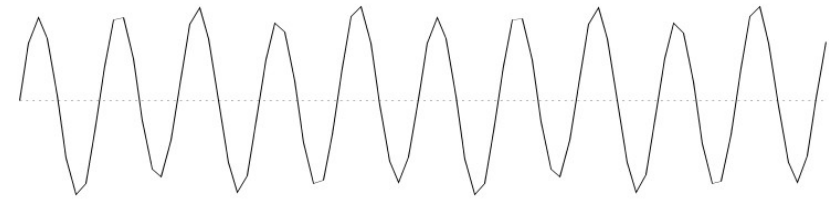
Señal muestreada: $F_m = 5000 \text{ hz} < 2 \cdot 4800 \text{ hz}$



Señal muestreada: $F_m = 5000 \text{ hz} > 2 \cdot 600 \text{ hz}$



Señal muestreada reconstruida

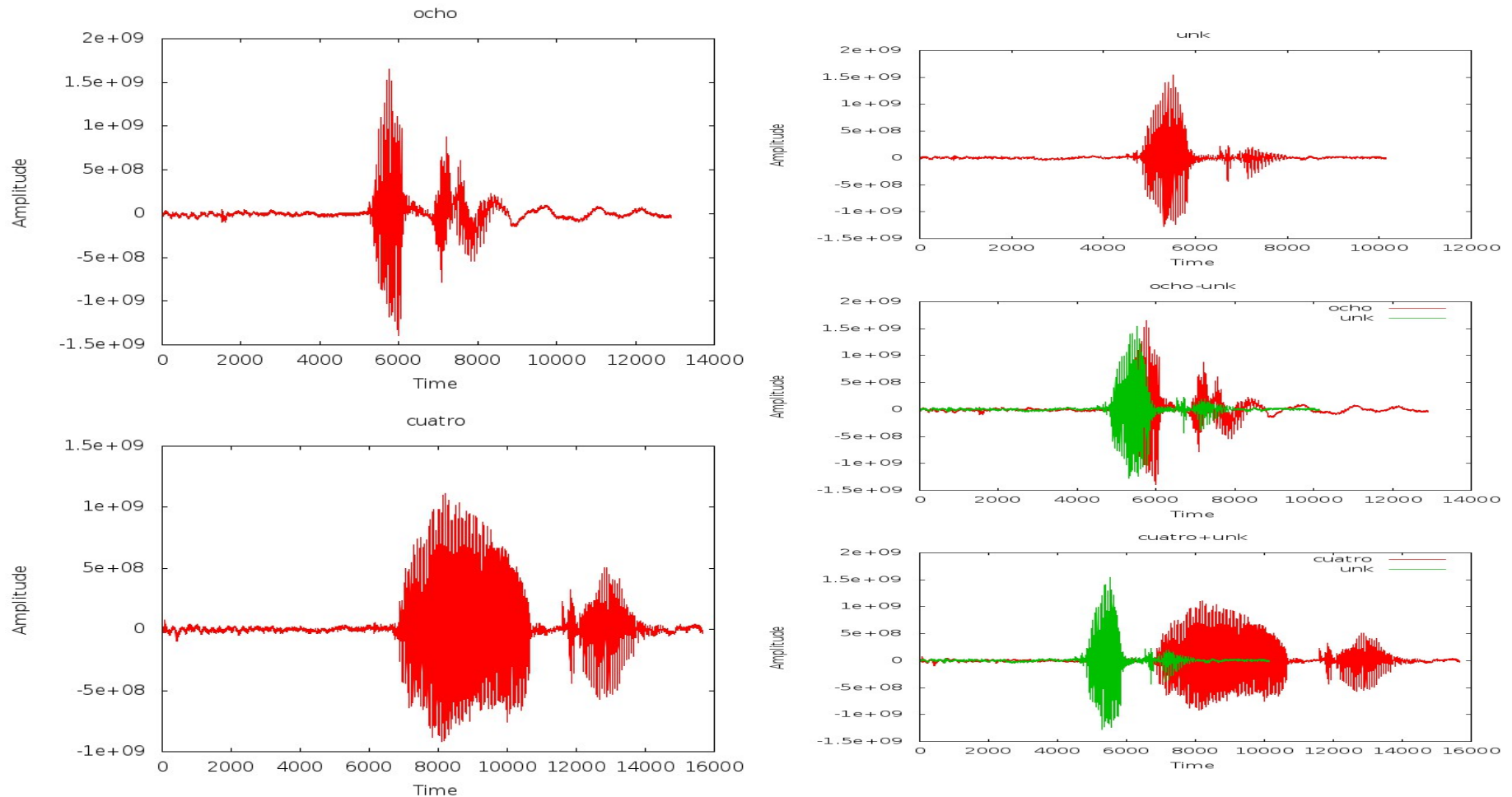


Señal muestreada reconstruida

Audios en PoliformaT

Adquisición y preproceso

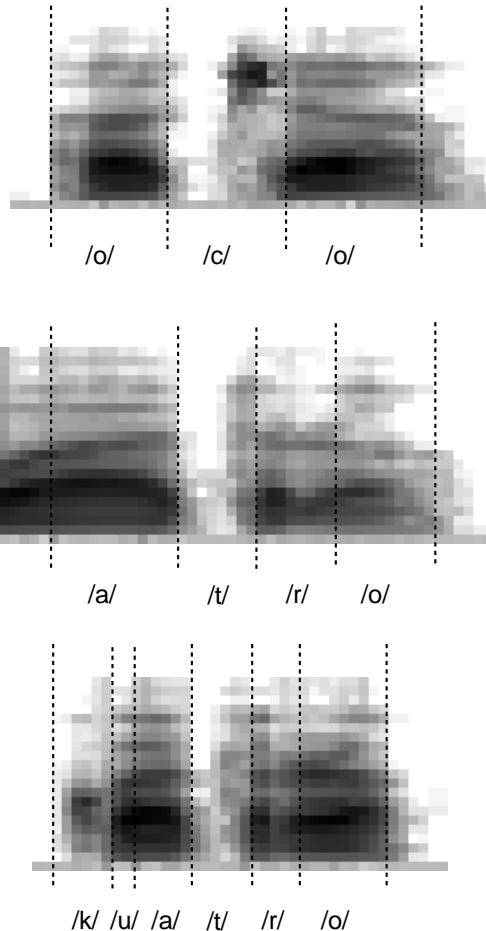
La representación temporal no es lo bastante discriminativa



Alternativa: **representación frecuencial** (*espectrograma*)

Adquisición y preproceso

Representación
en el dominio
de la frecuencia

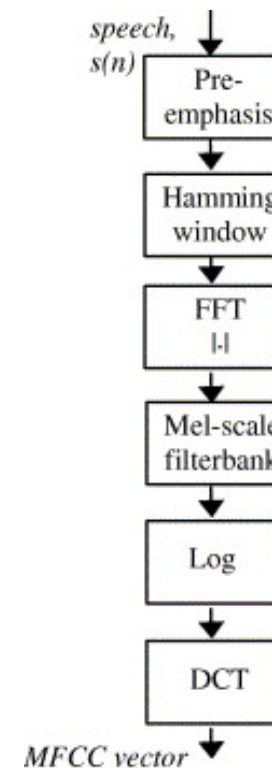


- Mayor capacidad discriminativa
- Distintas regiones identifican distintos sonidos
- Problema: longitud de las regiones no uniforme (variabilidad temporal no lineal)
 - Vocálicos: elásticos
 - Consonánticos: duración más regular

Extracción de características

Proceso habitual para obtención de coeficientes ceptrales (**estándar ETSI**):

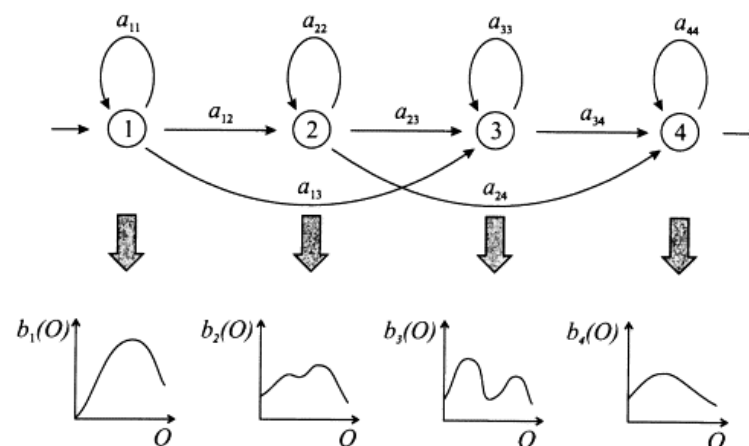
- Preénfasis
 - Paso alto, equilibrado frecuencial
- Ventana de Hamming
 - Obtención de subseñales (marcos, *frames*)
- Transformada rápida de Fourier (FFT)
 - Paso a dominio frecuencial
- Banco de filtros en la escala de Mel
 - Filtro basado en percepción humana
- Logaritmo
 - Sensibilidad no lineal
- Transformada discreta del coseno (DCT)
 - Tracto vocal, decorrelado



<http://ars.els-cdn.com/content/image/1-s2.0-S0167639305002359-gr2.jpg>

Representación continua: coeficientes cepstrales

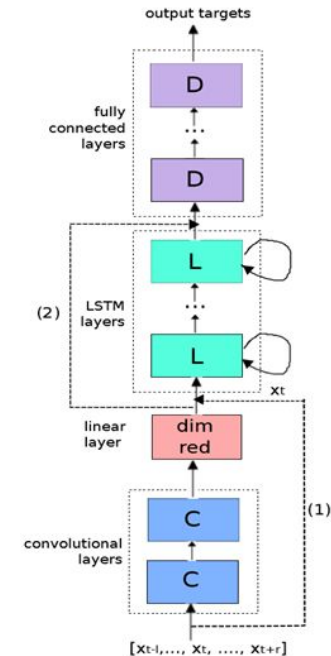
- La representación continua permite el uso de múltiples modelos estadísticos
- Representación directa del vector de características (MFCC)
- Modelo tradicional: modelo oculto de Markov (HMM) (1970s-2010s)
 - MFCC se usan para estimar los parámetros del HMM
 - Distribución Gaussiana para probabilidad de emisión en los estados
 - Posible reemplazar Gaussiana por una red neuronal profunda (DNN)



<http://ars.els-cdn.com/content/image/1-s2.0-S0262885699000554-gr1.gif>

Representación continua: coeficientes cepstrales

- Modelos actuales: aprendizaje profundo (*deep learning*)
- Basados en redes neuronales de gran número de capas y densidad
- Diversidad de aproximaciones y modelos
 - Modelos generativos: basados en redes recurrentes (RNN)
 - Modelos discriminativos (*end-to-end*): basados en clasificación temporal conexionista (CTC)



<http://slideplayer.com/slide/8844792/26/images/34/Innovation:+Ensemble+Deep+Learning.jpg>

Representación discreta: cuantificación vectorial

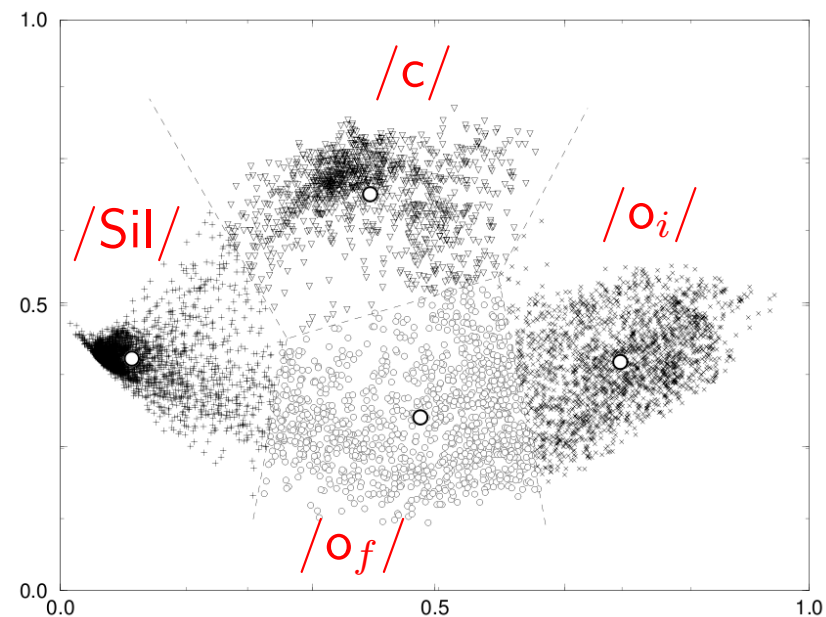
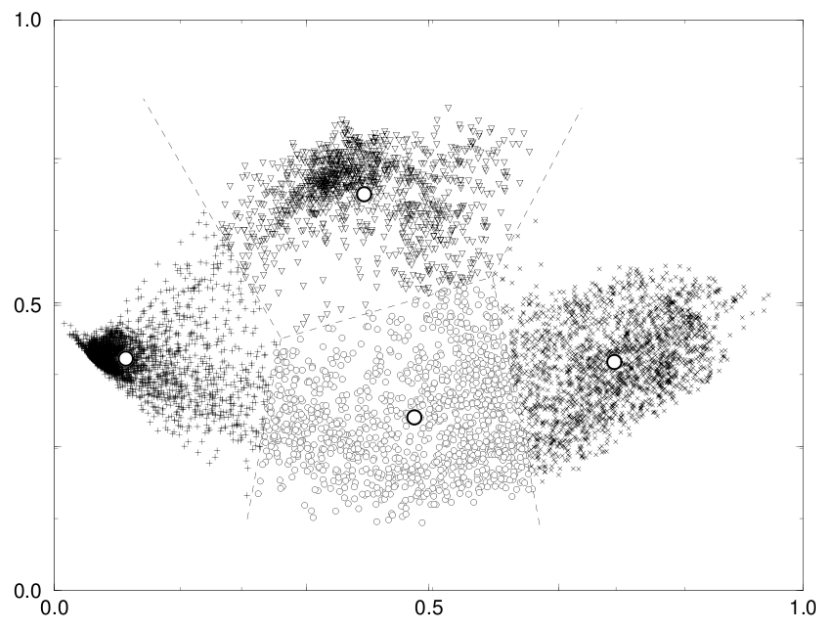
- Para tareas sencillas (palabras aisladas) puede aplicarse esta alternativa
- Idea general: asociar un símbolo a cada tipo de vector de características
- De secuencia de vectores de características a cadena de símbolos
- Proceso en dos pasos:
 1. Obtención de tipos de vectores (*codebook*)
 2. Cuantificación vectorial (asignación de vectores a símbolos)

Representación discreta: cuantificación vectorial

Proceso de creación del *codebook*

- Se toma vectores de características de entrenamiento
- Se elige un número k de tipos de vectores (k etiquetas)
- Se realiza un particionado en k *clusters* (p.ej., con k -medias)
- Se obtiene la media de cada *cluster* como prototipo o *codeword*
- Se asocia una etiqueta a cada *codeword*

Ejemplo: vectores de características de “ocho” proyectados en dos dimensiones



Representación discreta: cuantificación vectorial

Proceso de cuantificación vectorial

- Se toma el *codebook* obtenido del paso previo
- Se toma los vectores de características a decodificar
- Se asigna a cada vector la etiqueta del *codeword* más cercano (en distancia euclídea)

Espectrograma
(21 canales, 100hz)



52x21

Cadena de
etiquetas acústicas

<bbffbfefeehhhhhhhhhhhhhhhhhhhhhhggddddddddddddddff>

52 símbolos
($|\Sigma| = 16$)

Representaciones alternativas

- Segmentación de traza
- Coeficientes lineales predictivos (LPC)
- Predicción lineal perceptiva (PLP)
- Reconocimiento en tándem

Índice

- 1 Introducción ▷ 3
- 2 Representación de imágenes ▷ 7
- 3 Representación de voz ▷ 37
- 4 *Representación de texto* ▷ 51

Extracción de texto

- Selección de la unidad de documento
 - Varios ficheros pueden constituir un único documento
 - Un fichero grande puede requerir dividirse en varios documentos
- Problema de determinar la granularidad de la unidad de documento
- Texto almacenado en ficheros como secuencia de bytes
- Conversión a secuencia de caracteres con cierto formato
 - Ejemplo: extracción desde un fichero Word, PDF, comprimido, XML, etc.
- Necesidad de determinar el formato de fichero y su extractor asociado
- La secuencia de bytes puede contener información útil

Extracción de texto

Fases de extracción de características:

- Tokenización
 - Separación palabras, signos puntuación, etc.
- Normalización
 - A minúsculas, sin tildes, etc.
- Eliminación de *stop words*
 - Palabras con alta frecuencia
- *Stemming*
 - Eliminación de terminaciones (algoritmo de Porter)
- Lematización
 - Análisis morfológico, eliminación de desinencias

Más información: [Introduction to Information Retrieval](#) Cap. 2

Extracción de texto: ejemplo

- Extraído de la tarea *20 Newsgroups*¹
- Consiste en clasificar los mensaje enviados en 20 grupos de noticias
- Ejemplo de mensaje enviado al grupo de noticias *alt.atheism*

From: b711zbr@utarlg.uta.edu (JUNYAN WANG)

Subject: Bible contradictions

I would like a list of Bible contadictions from those of you who dispite being free from Christianity are well versed in the Bible.

- La codificación es ASCII y la unidad de documento es el mensaje
- Elimina mensajes duplicados y algunos campos de la cabecera del mensaje

¹Disponible en <http://qwone.com/~jason/20Newsgroups>

Extracción de texto: ejemplo

Original

From: b711zbr@utarlg.uta.edu (JUNYAN WANG)

Subject: Bible contradictions

I would like a list of Bible contadictions from those of you who dispite being free from Christianity are well versed in the Bible.

Tokenización

From : b711zbr @ utarlg.uta.edu (JUNYAN WANG)

Subject : Bible contradictions

I would like a list of Bible contadictions from those of you who dispite being free from Christianity are well versed in the Bible .

Normalización

from : b711zbr @ utarlg.uta.edu (junyan wang)

subject : bible contradictions

i would like a list of bible contadictions from those of you who dispite being free from christianity are well versed in the bible .

Eliminación

stop words

++++ : b711zbr @ utarlg.uta.edu (junyan wang)

subject : bible contradictions

+ +++++ like + list ++ bible contadictions +++++ +++++ ++ +++ +++ dispite being free +++++ christianity +++ well versed ++ +++ bible .

Stemming

++++ : b711zbr @ utarlg.uta.edu (junyan wang)

subject : bibl- contradict----

+ +++++ like + list ++ bibl- contadict---- +++++ +++++ ++ +++ +++ dispit- be--- free +++++ christian--- +++ well vers-- ++ +++ bibl- .

Lematización

++++ : b711zbr @ utarlg.uta.edu (junyan wang)

subject : bibl- contradict----

+ +++++ like + list ++ bibl- contadict---- +++++ +++++ ++ +++ +++ dispit- be--- free +++++ christ@@@--- +++ well vers-- ++ +++ bibl- .

Representación *bag-of-words*

- Determinar el vocabulario V (tokens diferentes) de la colección de D documentos
- Cada documento d es representado mediante un vector x cuya dimensionalidad es igual al tamaño de vocabulario $|V|$
- Cada dimensión t del vector está asociada a un token del vocabulario
- Representaciones:
 - Binaria: los valores del vector indican la presencia (1) o no (0) de un determinado token en el documento que está representando

$$x_{dt} \in \{0, 1\}$$

- Entera (***term-frequency***): los valores del vector indican el número de ocurrencias de dicho token en el documento

$$x_{dt} \in \mathbb{N}$$

Representación *bag-of-words*

3 mensajes enviados a <i>alt.atheism</i>	⇒	0	0	0	windows
		4	11	3	god
		0	0	0	dod
		0	3	0	government
		2	0	1	writes
		14	7	15	people
		0	0	0	team
		0	0	0	bike
		0	0	0	game
		0	3	0	car
		0	1	0	article
		0	0	0	hockey
		0	0	0	rutgers
		0	0	0	encryption
		0	0	0	israel
		4	1	3	jesus
		0	0	0	clipper
		1	2	11	christians
		8	8	0	bible
		7	4	3	christian
3 mensajes enviados a <i>comp.windows.x</i>	⇒	17	17	9	windows
		0	0	0	god
		0	0	0	dod
		0	0	0	government
		0	1	0	writes
		4	3	5	people
		1	0	0	team
		0	0	0	bike
		0	1	0	game
		0	0	0	car
		3	0	8	article
		0	0	0	hockey
		0	0	0	rutgers
		0	0	0	encryption
		0	0	0	israel
		0	0	0	jesus
		0	0	0	clipper
		0	0	0	christians
		2	0	0	bible
		0	1	0	christian
3 mensajes enviados a <i>rec.sport.hockey</i>	⇒	0	0	0	windows
		0	0	0	god
		0	0	0	dod
		1	0	0	government
		0	0	2	writes
		8	0	7	people
		9	10	0	team
		0	0	0	bike
		3	13	10	game
		0	0	0	car
		0	0	0	article
		8	2	5	hockey
		0	0	0	rutgers
		0	0	0	encryption
		0	0	0	israel
		0	0	0	jesus
		0	0	0	clipper
		0	0	0	christians
		0	0	0	bible
		0	0	0	christian

Representación *bag-of-words*

Tendremos un conjunto de documentos $\{T_1, T_2, \dots, T_D\}$

El tamaño de la representación de un documento T_d en memoria depende de:

- El tamaño de vocabulario: $|V|$
- La longitud del documento d : l_d (máximo de ocurrencias de una palabra)

Para un documento T_d :

$$|V| \cdot \left\lceil \frac{\log_2(l_d + 1)}{8} \right\rceil \text{ bytes}$$

Para la colección $\{T_1, T_2, \dots, T_D\}$, con $l_{max} = \max_{d=1, \dots, D} l_d$

$$D \cdot |V| \cdot \left\lceil \frac{\log_2(l_{max} + 1)}{8} \right\rceil \text{ bytes}$$

Representaciones en el área de extracción de información

- Las representaciones *bag-of-words* sólo tienen en cuenta un documento
- La capacidad discriminativa puede depender de la colección de documentos
- Posible solución: definir el peso de un token como el producto de una función local (documento) por una global (colección)

$$w_{dt} = L(d, t) \cdot G(t)$$

- Representaciones *bag-of-words*:
 - Funciones locales:
 - Conteo: $L(d, t) = x_{dt}$
 - Logaritmo: $L(d, t) = \log(x_{dt} + 1)$
 - Función global unitaria: $G(t) = 1$

Representaciones en el área de extracción de información

Representaciones globales:

- Funciones locales: las mismas que para *bag-of-words*
- Funciones globales más utilizadas:

$$\text{Normal} \quad G(t) = \left(\sum_d x_{dt}^2 \right)^{-\frac{1}{2}}$$

$$\text{GfIdf} \quad G(t) = \frac{\sum_d x_{dt}}{\sum_{d:x_{dt}>0} 1}$$

$$\text{Idf} \quad G(t) = \log \frac{D}{\sum_{d:x_{dt}>0} 1}$$

- *Idf* es el más usado al atenuar tokens con presencia en muchos documentos

Representación: secuencia de tokens (n-gramas)

- La representación basada en *bag-of-words* pierde información de contexto

- Ejemplo:

Mary is quicker than John
John is quicker than Mary

- Número de ocurrencias de secuencias de tokens de longitud n (n -gramas)
- Captura relación entre tokens consecutivos
- Puede presentar problemas de dimensionalidad y dispersión
- Ha demostrado obtener mejor rendimiento que la aproximación *bag-of-words*

Representación: secuencia de tokens (n-gramas)

3 mensajes enviados a *alt.atheism*

0	0	0	x windows
2	3	0	in god
0	0	0	dod security
0	1	0	government of
1	0	0	writes about
7	2	4	other people
0	0	0	team sponsored
0	0	0	your bike
0	0	0	best game
0	1	0	a car uses
0	0	0	article contains
0	0	0	hockey playoffs
0	0	0	rutgers university
0	0	0	inexpensive encryption
0	0	0	in israel
2	0	1	jesus appeared
0	0	0	clipper project
1	1	3	respectable christians
4	2	0	christian bible
3	1	1	christian morality

3 mensajes enviados a *alt.atheism*

0	0	0	for x windows
1	2	0	believe in god
0	0	0	meets dod security
0	1	0	government of the
1	0	0	writes about radical
3	1	2	other people have
0	0	0	team sponsored by
0	0	0	ride your bike
0	0	0	the best game
0	1	0	a car uses
0	0	0	this article contains
0	0	0	hockey playoffs have
0	0	0	rutgers university newark
0	0	0	inexpensive encryption devices
0	0	0	voting in israel
1	0	0	suddenly jesus appeared
0	0	0	delta clipper project
0	0	1	many respectable christians
2	2	0	the christian bible
2	0	0	that christian morality



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

Escola Tècnica Superior d'Enginyeria Informàtica



Tema 3. Reducción de dimensionalidad

Percepción (PER)

Curso 2019/2020

Departamento de Sistemas Informáticos y Computación

Índice

- 1 Introducción ▷ 3
- 2 Proyecciones lineales ▷ 9
- 3 Vectores propios ▷ 17
- 4 Principal Component Analysis (PCA) ▷ 25
- 5 Aplicación de PCA: EigenFaces ▷ 39

Índice

- 1 *Introducción* ▷ 3
- 2 Proyecciones lineales ▷ 9
- 3 Vectores propios ▷ 17
- 4 Principal Component Analysis (PCA) ▷ 25
- 5 Aplicación de PCA: EigenFaces ▷ 39

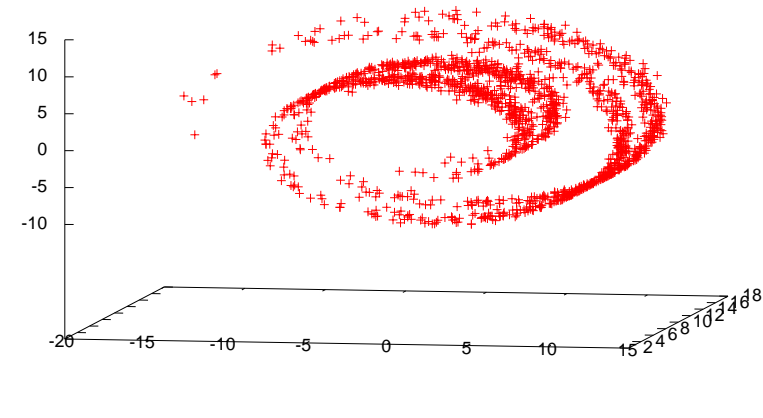
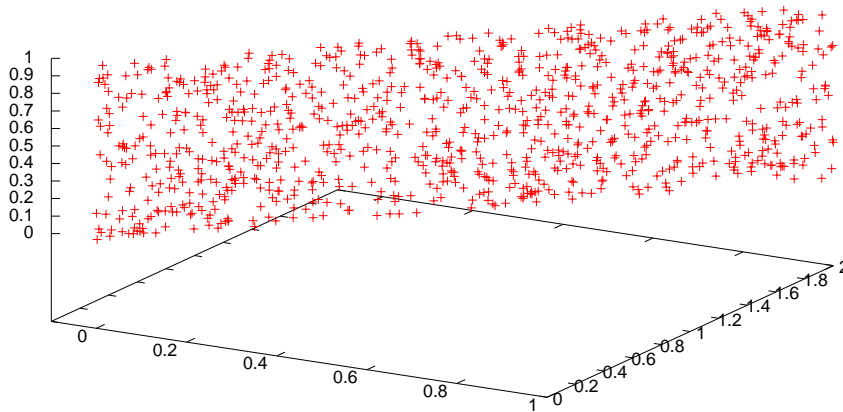
Introducción

- La representación vectorial de un objeto \mathbf{x} puede ser de muy alta dimensionalidad $\mathbf{x} \in \mathbb{R}^D$ siendo $D > 1000$
- En un espacio de alta dimensionalidad las técnicas de aprendizaje se ven afectadas por lo que se conoce como la *maldición de la dimensionalidad*¹
- Razones para aplicar técnicas de reducción de dimensionalidad:
 - La dimensionalidad intrínseca puede ser menor que la de la representación
 - Las componentes de cada vector pueden presentar fuertes correlaciones
 - Reducción de parámetros a estimar en el clasificador mejorando su estimación
 - Por eficiencia de tiempo de cómputo y ocupación en memoria
 - Eliminación del ruido durante la adquisición de datos

¹En inglés, *curse of dimensionality*.

Dimensionalidad intrínseca de los datos

El espacio de representación puede ser \mathbb{R}^D pero los objetos tienen una representación intrínseca $\mathbb{R}^{D'}$ con $D' < D$



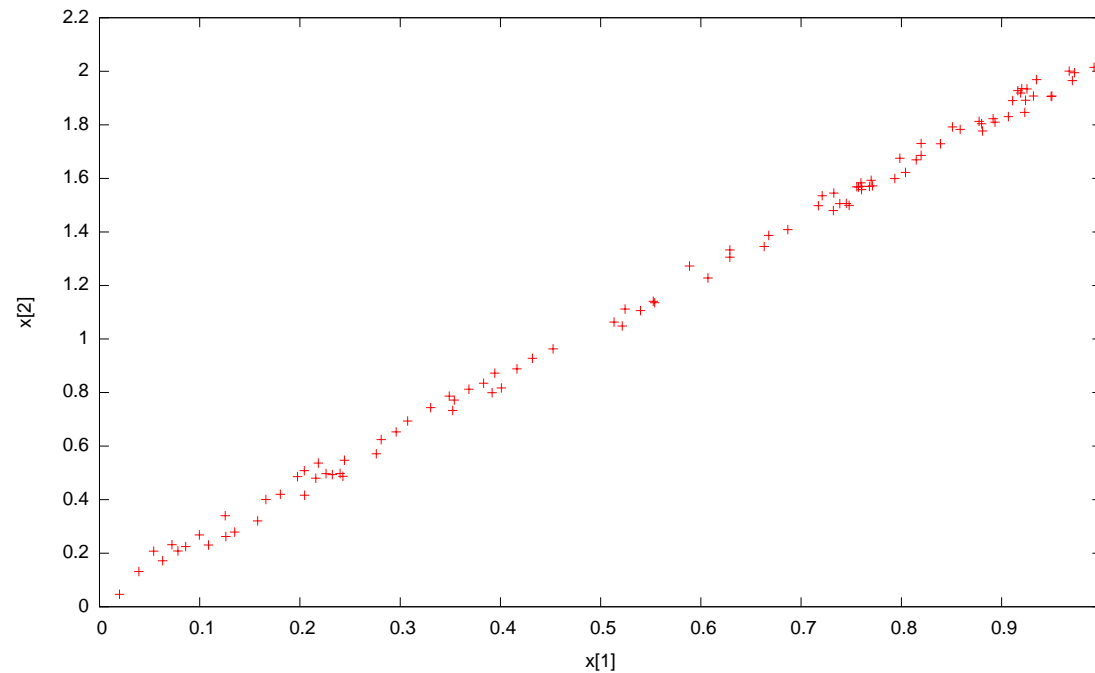
En ambos ejemplos $D = 3$ y $D' = 2$

Correlaciones entre componentes

Algunas dimensiones están correladas y no aportan información extra

Ejemplo $\mathbf{x} \in \mathbb{R}^2$:

- $x_2 = 2x_1 \rightarrow$ Dimensionalidad intrínseca $D' = 1$
- $x_2 = 2x_1 + \epsilon$ con ϵ independiente de $x_1 \rightarrow D' = 2$
pero x_2 no aporta mucha más información que x_1



Estimación de los parámetros del clasificador

- Sea un clasificador por funciones discriminantes lineales (FDLs)

- $G \equiv \{g_1, g_2, \dots, g_C\}$, $g_c(\mathbf{x}) = \mathbf{w}_c \cdot \mathbf{x} + b$

- Si $\mathbf{x} \in \mathbb{R}^D$, hay $C \times (D + 1)$ parámetros a estimar

- Ejemplo:

Clasificar mediante FDLs las caras de 100 individuos. De cada individuo se tienen 4 imágenes. Cada imagen consta de 75×75 píxeles. Considerando una representación geométrica global tenemos que:

$$100 \times (5625 + 1) = 562600 \text{ parámetros a estimar con sólo 400 muestras}$$

- Problema:

Un espacio de 75×75 dimensiones se puede considerar *vacío* con sólo 400 ejemplos. Es más, para que dejara de considerarse *vacío* haría falta un número de muestras que nunca vamos a tener disponibles

Reducción de dimensionalidad

- Proceso para reducir la dimensionalidad del espacio de representación de los datos
- Siendo $E = \mathbb{R}^D$ el espacio de representación de los datos, la reducción de dimensionalidad se puede ver como una función:

$$f(\mathbf{x}) : \mathbb{R}^D \rightarrow \mathbb{R}^k, \quad k \ll D$$

- Así, dada una representación $\mathbf{x} \in \mathbb{R}^D$, tendremos $\mathbf{x}' = f(\mathbf{x})$, $\mathbf{x}' \in \mathbb{R}^k$

Índice

- 1 Introducción ▷ 3
- 2 *Proyecciones lineales* ▷ 9
- 3 Vectores propios ▷ 17
- 4 Principal Component Analysis (PCA) ▷ 25
- 5 Aplicación de PCA: EigenFaces ▷ 39

Proyecciones lineales

- Sólo consideraremos proyecciones lineales
- Son el resultado de multiplicar \mathbf{x} por una *matriz de proyección* W

$$\begin{array}{ccccc} \mathbf{x}' & = & W^t & \cdot & \mathbf{x} \\ k \times 1 & & k \times D & & D \times 1 \end{array}$$

donde $W \in \mathbb{R}^{D \times k}$ y $\mathbf{x} \in \mathbb{R}^D$, $k < D$

- La proyección resulta en un vector $\mathbf{x}' \in \mathbb{R}^k$
- Ejemplo para $D = 3$ y $k = 2$

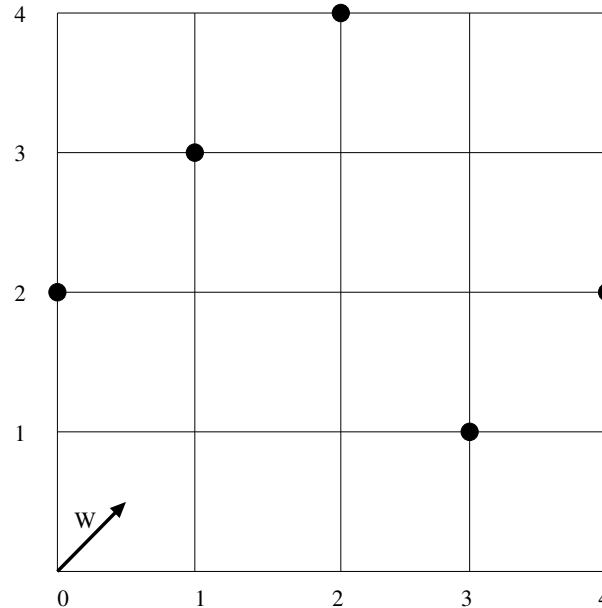
$$\begin{array}{ccccc} \begin{pmatrix} 2 \\ 0 \end{pmatrix} & = & \begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix} & \cdot & \begin{pmatrix} 2 \\ 0 \\ 1 \end{pmatrix} \\ \mathbf{x}' & = & W^t & \cdot & \mathbf{x} \end{array}$$

Proyecciones lineales

- Sean los siguientes puntos en un espacio \mathbb{R}^2 :

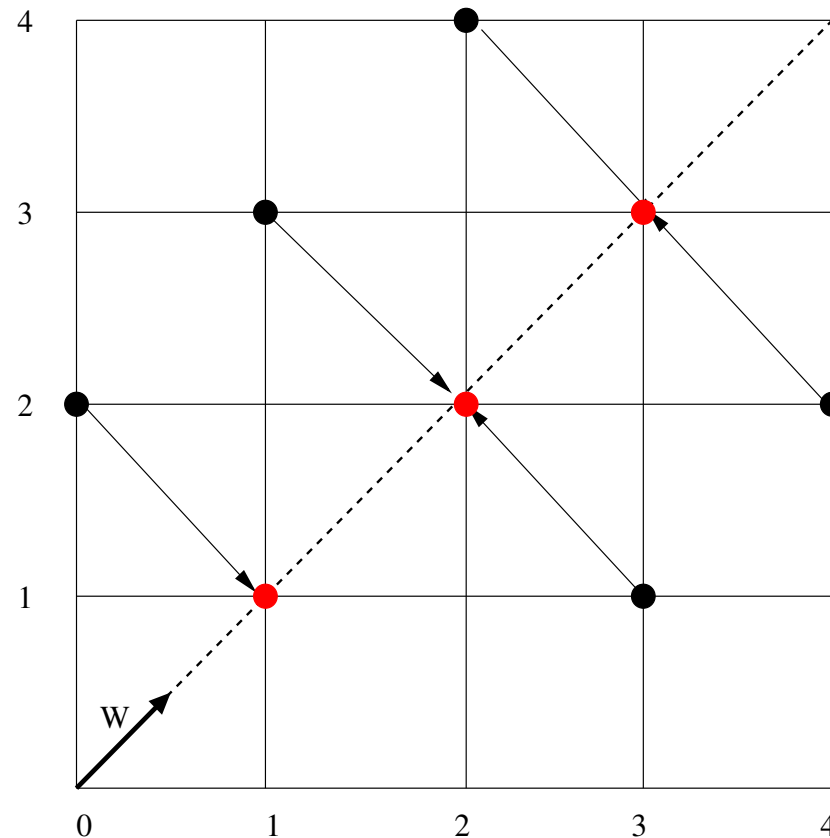
$$\mathbf{x}_1 = \begin{pmatrix} 0 \\ 2 \end{pmatrix} \quad \mathbf{x}_2 = \begin{pmatrix} 1 \\ 3 \end{pmatrix} \quad \mathbf{x}_3 = \begin{pmatrix} 2 \\ 4 \end{pmatrix} \quad \mathbf{x}_4 = \begin{pmatrix} 3 \\ 1 \end{pmatrix} \quad \mathbf{x}_5 = \begin{pmatrix} 4 \\ 2 \end{pmatrix}$$

- Calculemos la proyección con $W = \begin{pmatrix} \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \end{pmatrix}$



Proyecciones lineales

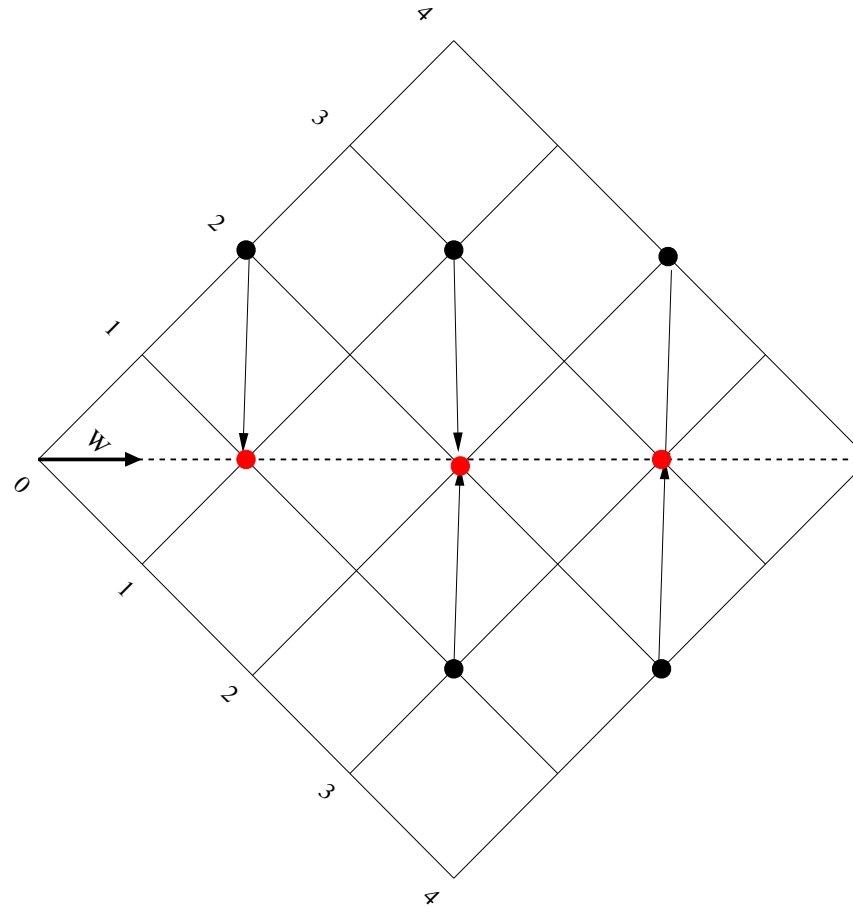
¿Qué está pasando con x_2 y x_4 ?



Proyección ortogonal a W

Proyecciones lineales

Desde el punto de vista de W , esta reducción de dimensionalidad es una proyección sobre la recta definida por dicha matriz $W = \begin{pmatrix} \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \end{pmatrix}$



Proyecciones lineales: interpretación geométrica

- Una proyección es un producto escalar de los vectores columna \mathbf{w} de W por \mathbf{x}

$$\mathbf{w}^t \cdot \mathbf{x} = |\mathbf{w}| \cdot |\mathbf{x}| \cdot \cos \alpha$$

donde α es el ángulo definido entre \mathbf{w} y \mathbf{x}

- Calculemos ahora la proyección con $W = \begin{pmatrix} \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} \end{pmatrix}$ de:

$$\mathbf{x}_1 = \begin{pmatrix} 2 \\ 0 \end{pmatrix} \quad \mathbf{x}_2 = \begin{pmatrix} 2 \\ 2 \end{pmatrix} \quad \mathbf{x}_3 = \begin{pmatrix} 0 \\ 2 \end{pmatrix} \quad \mathbf{x}_4 = \begin{pmatrix} 0 \\ -2 \end{pmatrix} \quad \mathbf{x}_5 = \begin{pmatrix} -2 \\ -2 \end{pmatrix}$$

- ¿Cómo podríamos interpretar el signo y magnitud del resultado?

Proyecciones lineales: interpretación geométrica

- En toda proyección lineal la dimensión i -ésima de \mathbf{x}' se calcula como:

$$\mathbf{x}'_i = \mathbf{w}_i^t \cdot \mathbf{x}$$

donde \mathbf{w}_i es la columna i -ésima de W

- En la práctica, \mathbf{x} se multiplica por cada uno de los vectores columna de W
- El producto escalar $\mathbf{w}_i^t \cdot \mathbf{x}$ normalizado es la similitud coseno entre \mathbf{w}_i y \mathbf{x} :

$$\cos \alpha = \frac{\mathbf{w}_i^t \cdot \mathbf{x}}{|\mathbf{w}_i| \cdot |\mathbf{x}|}$$

Proyecciones lineales y reducción de dimensionalidad

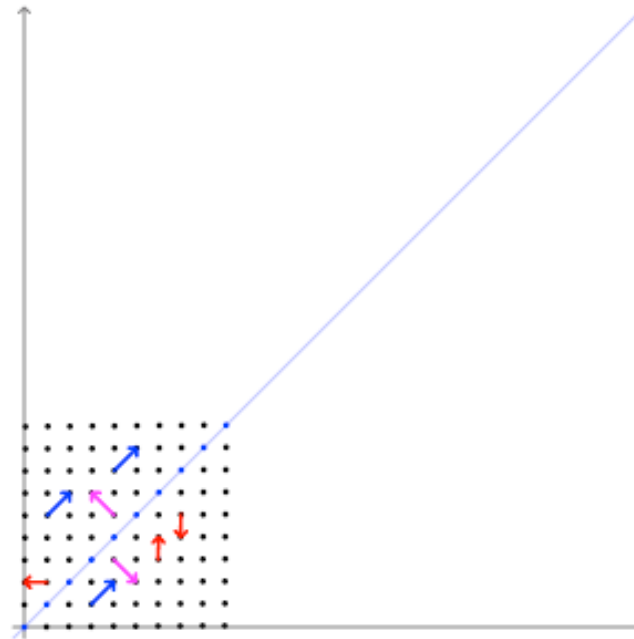
- En las reducciones de dimensionalidad lineales se busca una W adecuada
- La nueva representación en $E' \equiv \mathbb{R}^k$ debería cumplir las propiedades de:
 - *Continuidad*: cosas cercanas permanecen cercanas
 - *Discriminativa*: datos de distintas clases están separados
 - *Invarianza*: a transformaciones usuales en el espacio original
- Una proyección lineal con estas propiedades es en general difícil de hallar
- Propuesta:
 - Búsqueda de la W adecuada como un problema de optimización
 - Selección de criterio de optimización según una cierta propiedad deseable
- Este problema de optimización requiere el uso de **vectores y valores propios** (*eigenvectors* y *eigenvalues*, respectivamente)

Índice

- 1 Introducción ▷ 3
- 2 Proyecciones lineales ▷ 9
- 3 *Vectores propios* ▷ 17
- 4 Principal Component Analysis (PCA) ▷ 25
- 5 Aplicación de PCA: EigenFaces ▷ 39

Vectores propios de una matriz

Ejemplo de transformación lineal ($\mathbf{x}' = W^t \mathbf{x}$ donde $\mathbf{x}', \mathbf{x} \in \mathbb{R}^2$):



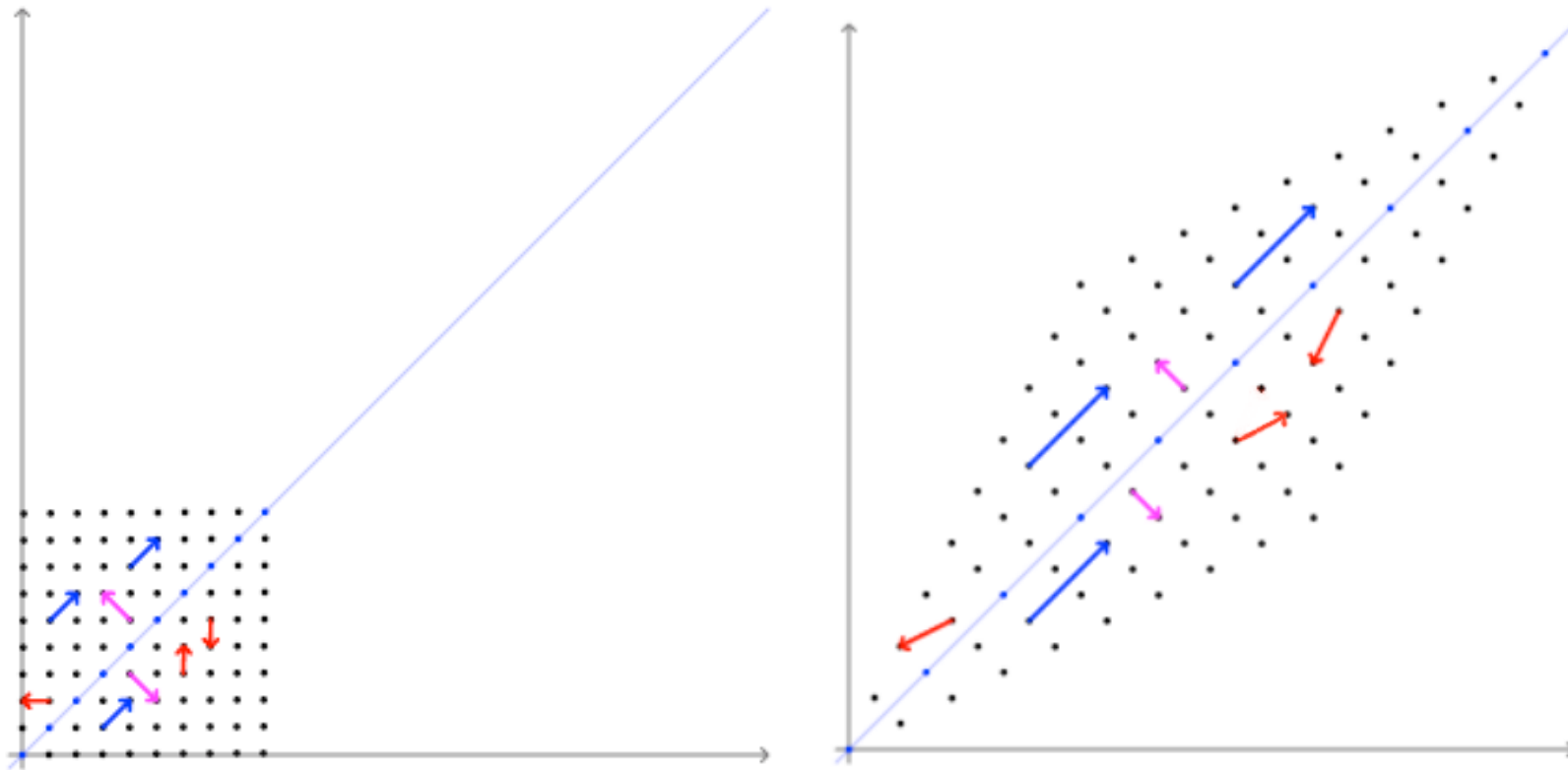
incluyendo vectores

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

Below the vectors, there are colored arrows: a blue arrow pointing up-right under \mathbf{x}_1 , a red arrow pointing down under \mathbf{x}_2 , a red arrow pointing up under \mathbf{x}_3 , a pink arrow pointing down-right under \mathbf{x}_4 , a red arrow pointing left under \mathbf{x}_5 , and a pink arrow pointing up-left under \mathbf{x}_6 .

Vectores propios de una matriz

Si aplicamos la transformación lineal con $W = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}$ tenemos:



¿Qué vectores no son modificados en dirección, aunque puede que en magnitud?

Vectores propios de una matriz

- Observamos que los vectores

$$\begin{pmatrix} 1 \\ 1 \end{pmatrix} \quad \begin{pmatrix} -1 \\ 1 \end{pmatrix} \quad \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

son simplemente escalados pero no rotados

- Dichos vectores son **vectores propios** (*eigenvectors*) de la matriz W
- Se dice que \mathbf{x} es un vector propio de W si:

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

con $\lambda \in \mathbb{R}$

- A este escalar se le conoce como **valor propio** (*eigenvalue*)

Vectores propios de una matriz. Problema

- Muchas técnicas usan el cálculo de vectores y valores propios de una matriz
- Los valores propios se interpretan según qué represente esa matriz
- Encontrar los vectores propios supone resolver el sistema:

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

o lo que es lo mismo

$$W^t \mathbf{x} - \lambda \mathbf{x} = 0 \quad \longrightarrow \quad (W^t - \lambda I) \mathbf{x} = 0$$

- Por simplicidad en la presentación del cálculo de vectores y valores propios utilizaremos la notación W sin transponer

$$W \mathbf{x} - \lambda \mathbf{x} = 0 \quad \longrightarrow \quad (W - \lambda I) \mathbf{x} = 0$$

Vectores propios de una matriz. Solución

- Según un teorema fundamental del álgebra lineal:

$$(W - \lambda I)\mathbf{x} = 0 \quad \Leftrightarrow \quad \det(W - \lambda I) = 0$$

- Pasos de resolución:
 1. Calcular valores propios λ (los que hacen que $\det(W - \lambda I) = 0$)
 2. Utilizar λ para resolver el sistema lineal $(W - \lambda I)\mathbf{x} = 0$
- Por lo tanto cada valor propio tiene asociado un vector propio:

$$(W - \lambda_1 I)\mathbf{x}_1 = 0$$

...

$$(W - \lambda_n I)\mathbf{x}_n = 0$$

- El valor 0 denota un vector D -dimensional de ceros
- $n = D$ si la matriz $W \in \mathbb{R}^{D \times D}$ es de rango completo

Vectores propios de una matriz. Ejemplo

Sea $W = \begin{pmatrix} 1 & 2 \\ 2 & 1 \end{pmatrix}$ vamos a calcular sus valores y vectores propios

$$\det(A - \lambda I) = 0 \rightarrow \det \left(\begin{pmatrix} 1-\lambda & 2 \\ 2 & 1-\lambda \end{pmatrix} \right) = 0 \rightarrow (1-\lambda)^2 - 4 = 0 \rightarrow \boxed{\lambda = \{-1, 3\}}$$

Para $\lambda_1 = -1$, encontrar \mathbf{x}_1 que cumpla $(W - \lambda_1 I)\mathbf{x}_1 = 0$

$$\begin{pmatrix} 2 & 2 \\ 2 & 2 \end{pmatrix} \begin{pmatrix} x_{11} \\ x_{12} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \rightarrow \begin{matrix} (2x_{11} + 2x_{12}) = 0 \\ (2x_{11} + 2x_{12}) = 0 \end{matrix} \rightarrow \mathbf{x}_1 = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

Para $\lambda_2 = 3$, encontrar \mathbf{x}_2 que cumpla $(W - \lambda_2 I)\mathbf{x}_2 = 0$

$$\begin{pmatrix} -2 & 2 \\ 2 & -2 \end{pmatrix} \begin{pmatrix} x_{21} \\ x_{22} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \rightarrow \begin{matrix} (-2x_{21} + 2x_{22}) = 0 \\ (2x_{21} - 2x_{22}) = 0 \end{matrix} \rightarrow \mathbf{x}_2 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

En realidad \mathbf{x}_1 y \mathbf{x}_2 definen una *base*, ya que existen infinitos vectores propios.

Consideraciones prácticas

En general, no existe solución cerrada para polinomios de orden > 4 (Teorema de Abel-Ruffini)

Existen algoritmos numéricos iterativos para encontrar soluciones:

- QR
- Householder
- Lanczos

Índice

- 1 Introducción ▷ 3
- 2 Proyecciones lineales ▷ 9
- 3 Vectores propios ▷ 17
- 4 *Principal Component Analysis (PCA)* ▷ 25
- 5 Aplicación de PCA: EigenFaces ▷ 39

PCA: Principal Component Analysis

- PCA es probablemente la técnica de reducción de dimensionalidad más conocida
- Es una técnica de reducción de dimensionalidad *no supervisada* (la capacidad discriminativa entre clases no se tiene en cuenta)
- Es muy empleada por preservar la mayor parte de la *varianza* de los datos, que se asume que incluye la capacidad de discriminar entre clases
- Se suele emplear como un preproceso previo a otras técnicas de reducción de dimensionalidad supervisadas

Problema de optimización

- Objetivo PCA: encontrar una matriz de proyección W que minimice el error de reconstrucción
- Sea un conjunto de datos $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ con $\mathbf{x}_i \in \mathbb{R}^D$ y $\bar{\mathbf{x}} = \frac{1}{n} \sum_i \mathbf{x}_i$
- Dado un vector $\mathbf{x}_i \in \mathbb{R}^D$, el proceso de reconstrucción es:
 - Se resta $\bar{\mathbf{x}}$ a \mathbf{x}_i y se proyecta el resultado al espacio reducido \mathbb{R}^k

$$\mathbf{x}'_i = W^t(\mathbf{x}_i - \bar{\mathbf{x}}) \quad W \in \mathbb{R}^{D \times k} \quad \text{y} \quad \mathbf{x}'_i \in \mathbb{R}^k$$

- Se proyecta \mathbf{x}'_i al espacio original \mathbb{R}^D y se suma $\bar{\mathbf{x}}$

$$\hat{\mathbf{x}}_i = \bar{\mathbf{x}} + V^t \mathbf{x}'_i \quad V \in \mathbb{R}^{k \times D} \quad \text{y} \quad \hat{\mathbf{x}}_i \in \mathbb{R}^D$$

- El error de reconstrucción del conjunto de datos al proyectar a \mathbb{R}^k es:

$$\text{error}_k = \sum_{i=1}^n \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|^2 = \sum_{i=1}^n (\mathbf{x}_i - \hat{\mathbf{x}}_i)^t (\mathbf{x}_i - \hat{\mathbf{x}}_i)$$

Problema de optimización

- W está formada por k vectores *columna* $\mathbf{w}_j \in \mathbb{R}^{D \times 1}$
- W define una base ortonormal donde $W^t W = I$:

$$\begin{aligned}\mathbf{w}_j^t \mathbf{w}_j &= 1 \\ \mathbf{w}_j^t \mathbf{w}_i &= 0 \quad \text{si } i \neq j\end{aligned}$$

- Con estas condiciones $V = W^t$ minimiza el error de reconstrucción, es decir:

$$\hat{\mathbf{x}}_i = \bar{\mathbf{x}} + W W^t (\mathbf{x}_i - \bar{\mathbf{x}}) \quad \text{con } W \in \mathbb{R}^{D \times k}$$

- Objetivo: encontrar matriz de proyección W que minimice error de reconstrucción

$$\widehat{W} = \underset{W}{\operatorname{argmin}} \sum_{i=1}^n (\mathbf{x}_i - \hat{\mathbf{x}}_i)^t (\mathbf{x}_i - \hat{\mathbf{x}}_i)$$

Problema de optimización

- Operando convenientemente, equivale a:

$$\begin{aligned}\widehat{W} &= \operatorname{argmin}_W \sum_{i=1}^n (\mathbf{x}_i - \hat{\mathbf{x}}_i)^t (\mathbf{x}_i - \hat{\mathbf{x}}_i) = \operatorname{argmin}_W \sum_{j=k+1}^D \mathbf{w}_j^t \Sigma_{\mathcal{X}} \mathbf{w}_j \\ &= \operatorname{argmax}_W \sum_{j=1}^k \mathbf{w}_j^t \Sigma_{\mathcal{X}} \mathbf{w}_j\end{aligned}$$

donde $\Sigma_{\mathcal{X}}$ es la matriz de covarianza de los datos.

- Por simplicidad, consideramos un único vector \mathbf{w} que minimiza el error de reconstrucción cuando proyectamos a una única dimensión

$$\hat{\mathbf{w}} = \operatorname{argmax}_{\mathbf{w} \in \mathbb{R}^D} \mathbf{w}^t \Sigma_{\mathcal{X}} \mathbf{w} \quad \text{sujeto a que} \quad \mathbf{w}^t \mathbf{w} = 1$$

Problema de optimización

- Maximización con restricciones, resolución por *multiplicadores de Lagrange*

$$\hat{\mathbf{w}} = \operatorname{argmax}_{\mathbf{w} \in \mathbb{R}^D} \max_{\lambda \in \mathbb{R}} \mathbf{w}^t \Sigma_{\mathcal{X}} \mathbf{w} + \lambda (1 - \mathbf{w}^t \mathbf{w})$$

- Tras derivar respecto a \mathbf{w} y λ , e igualar a cero, se tiene que

$$\Sigma_{\mathcal{X}} \mathbf{w} = \lambda \mathbf{w}$$

- Es decir, \mathbf{w} es un vector propio de la matriz de covarianza de los datos $\Sigma_{\mathcal{X}}$ y λ su valor propio asociado.

Detalles de los cálculos en documento en PoliformaT

Problema de optimización

- ¿Qué vectores propios se deben usar en la proyección para minimizar el error de reconstrucción?
- Para minimizar el error de reconstrucción al proyectar a una dimensión:

$$\hat{\mathbf{w}} = \operatorname{argmax}_{\mathbf{w} \in \mathbb{R}^D} \max_{\lambda \in \mathbb{R}} \mathbf{w}^t \Sigma_{\mathcal{X}} \mathbf{w} + \lambda (1 - \mathbf{w}^t \mathbf{w})$$

- Como $\Sigma_{\mathcal{X}} \mathbf{w} = \lambda \mathbf{w}$ y $\mathbf{w}^t \mathbf{w} = 1$

$$\hat{\mathbf{w}} = \operatorname{argmax}_{\mathbf{w} \in \mathbb{R}^D} \max_{\lambda \in \mathbb{R}} \mathbf{w}^t \lambda \mathbf{w} + \lambda (1 - \mathbf{w}^t \mathbf{w}) = \operatorname{argmax}_{\mathbf{w} \in \mathbb{R}^D} \max_{\lambda \in \mathbb{R}} \lambda$$

- Por tanto, se minimiza el error de reconstrucción al proyectar a una única dimensión cuando se toma el vector propio \mathbf{w} asociado al mayor valor propio

Problema de optimización

- Para obtener el segundo vector propio que minimiza el error de reconstrucción al proyectar a dos dimensiones, lo expresaríamos como:

$$\hat{\mathbf{w}}_2 = \operatorname{argmax}_{\mathbf{w}_2 \in \mathbb{R}^D} \mathbf{w}_2^t \Sigma_{\mathcal{X}} \mathbf{w}_2 \quad \text{sujeto a que} \quad \mathbf{w}_2^t \mathbf{w}_2 = 1 \quad \text{y} \quad \mathbf{w}_1^t \mathbf{w}_2 = 0$$

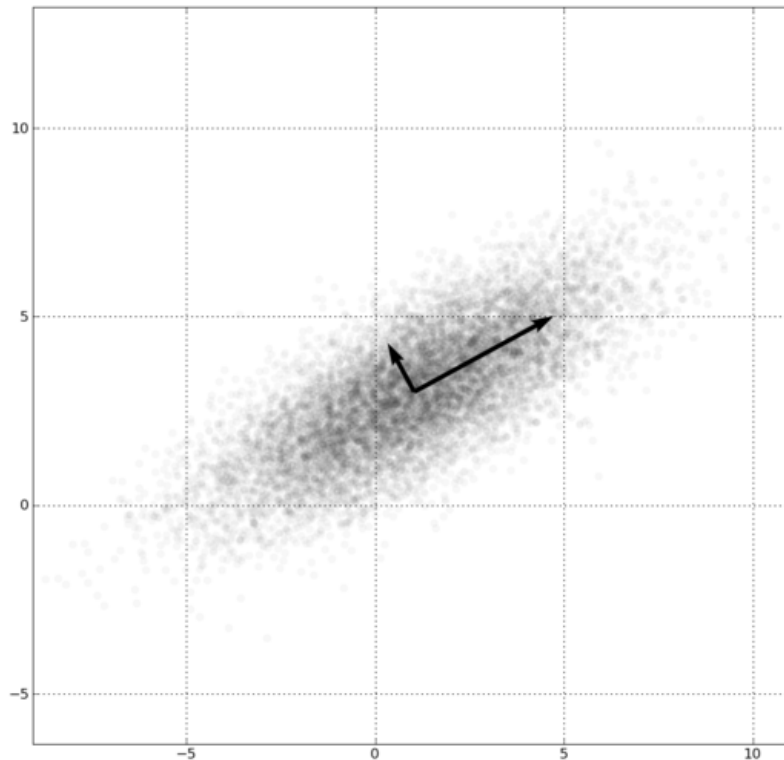
es decir, el segundo vector de proyección debe ser unitario y ortogonal a \mathbf{w}_1

- Proceso iterativo de cálculo de vectores de proyección que se resume en:
 - Cada vector propio \mathbf{w}_j tienen un valor propio asociado λ_j
 - \widehat{W} son los vectores propios de $\Sigma_{\mathcal{X}}$ de mayor a menor valor propio
 - $\Sigma_{\mathcal{X}}$ puede tener hasta D vectores propios (si es de rango completo)
- El error de reconstrucción cuando se proyecta a k dimensiones es:

$$\text{error}_k = \sum_{j=k+1}^D \mathbf{w}_j^t \Sigma_{\mathcal{X}} \mathbf{w}_j = \sum_{j=k+1}^D \mathbf{w}_j^t \lambda_j \mathbf{w}_j = \sum_{j=k+1}^D \lambda_j$$

Interpretación gráfica de PCA

La proyección PCA se hace en la dirección de los vectores propios de mayor a menor valor propio, es decir, en las direcciones del espacio de mayor a menor varianza



- El primer vector propio w_1 indica la dirección del espacio donde hay mayor varianza (λ_1) en los datos
- El segundo vector propio w_2 indica una dirección del espacio ortogonal a w_1 donde reside la mayor cantidad de varianza restante (no capturada en λ_1)

Problema de optimización. Resumen

- Para *minimizar el error de reconstrucción* a k dimensiones se proyecta con $W = (\mathbf{w}_1 \dots \mathbf{w}_k)$ que son los k vectores propios de $\Sigma_{\mathcal{X}}$ con *mayor valor propio asociado*
- Dado $\mathbf{y} \in \mathbb{R}^D$ para proyectarlo a k dimensiones

$$\begin{array}{ccccc} \mathbf{y}' & = & W^t & \cdot & (\mathbf{y} - \bar{\mathbf{x}}) \\ k \times 1 & & k \times D & & D \times 1 \end{array}$$

siendo $\bar{\mathbf{x}} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i$

Algoritmo PCA

- Entrada: $n, D, k, \mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$
- Salida: $W, \bar{\mathbf{x}}$
- Algoritmo:
 1. Calcular la media de los datos: $\bar{\mathbf{x}} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i$
 2. Restar a todos los datos la media: $\mathbf{x}_i \leftarrow \mathbf{x}_i - \bar{\mathbf{x}}$
 3. Calcular la matriz de covarianza: $\Sigma_{\mathcal{X}} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i (\mathbf{x}_i)^t$
 4. Encontrar todos los vectores propios de $\Sigma_{\mathcal{X}}$
 5. Ordenarlos descendientemente según los valores propios asociados
 6. Definir W como la matriz con los k primeros vectores propios
- Importante: para aplicar la proyección lineal a cualquier otro dato $\mathbf{y} \in \mathbb{R}^D$ hay que restarle previamente la media estimada en el paso 1

Diagonalización y varianza

- En general PCA es el proceso que diagonaliza la matriz de covarianzas:

$$\Sigma_{\mathcal{X}} W = W \Lambda \quad \rightarrow \quad \Sigma_{\mathcal{X}} = W \Lambda W^t \rightarrow \quad \Lambda = W^t \Sigma_{\mathcal{X}} W$$

donde:

- $\Sigma_{\mathcal{X}} \in \mathbb{R}^{D \times D}$ es la matriz de covarianza de los datos
 - $W \in \mathbb{R}^{D \times D}$ es la matriz con todos los vectores propios
 - $\Lambda \in \mathbb{R}^{D \times D}$ es una matriz diagonal con los todos valores propios
- Λ es la matriz de covarianzas de los datos proyectados por $W \in \mathbb{R}^{D \times D}$
 - Los datos proyectados están decorrelados (covarianzas nulas), y la varianza de estos datos en la dimensión j es $\Lambda_{jj} = \lambda_j$
 - La varianza total acumulada de los datos proyectados a k dimensiones es:

$$\sum_{j=1}^k \Lambda_{jj} = \sum_{j=1}^k \lambda_j$$

Consideraciones prácticas

- Sean:
 - $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$, datos de entrenamiento
 - $\bar{\mathbf{x}} = \frac{1}{n} \sum_{i=1}^n \mathbf{x}_i$, media de dichos datos
 - $A_{D \times n} = (\mathbf{x}_1 - \bar{\mathbf{x}}, \mathbf{x}_2 - \bar{\mathbf{x}}, \dots, \mathbf{x}_n - \bar{\mathbf{x}})$
- $\Sigma_{\mathcal{X}}$ puede expresarse como $\Sigma_{\mathcal{X}} = \frac{1}{n} A A^t$
- PCA consiste en obtener W y Λ que diagonalicen $\Sigma_{\mathcal{X}}$
- Problema: cuando $n \ll D$.
 - $\Sigma_{\mathcal{X}}$ no tendrá más de n vectores propios
 - Almacenar $\Sigma_{\mathcal{X}} \in \mathbb{R}^{D \times D}$ puede ser problemático con D grande
- Solución: diagonalizar $\Sigma'_{\mathcal{X}} = \frac{1}{D} A^t A$, con $\Sigma'_{\mathcal{X}} \in \mathbb{R}^{n \times n}$
- Pero hay que obtener obtener W y Λ que diagonalicen $\Sigma_{\mathcal{X}}$

Consideraciones prácticas

■ Método:

- Obtener W' y Λ' de la diagonalización de $\Sigma'_{\mathcal{X}}$
- Expresarlo en términos de la diagonalización de $\Sigma_{\mathcal{X}}$

$$\Sigma'_{\mathcal{X}} W' = \frac{1}{D} A^t A W' = W' \Lambda'$$

multiplicamos a ambas partes de la igualdad por $\frac{D}{n} A$

$$\boxed{\frac{1}{n} A A^t} A W' = \frac{D}{n} A W' \Lambda' \quad \longrightarrow \quad \Sigma \quad \boxed{A W'} = \boxed{A W'} \quad \boxed{\frac{D}{n} \Lambda'}$$

\downarrow
 W

\downarrow
 Λ

- Por tanto, $W = A W'$, $\Lambda = \frac{D}{n} \Lambda'$
- Los vectores propios obtenidos son ortogonales pero no ortonormales
- Se debe dividir cada vector por su módulo para obtener vectores ortonormales

Índice

- 1 Introducción ▷ 3
- 2 Proyecciones lineales ▷ 9
- 3 Vectores propios ▷ 17
- 4 Principal Component Analysis (PCA) ▷ 25
- 5 *Aplicación de PCA: EigenFaces* ▷ 39

Aplicación de PCA: EigenFaces

- Muestras $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ que representan imágenes de caras
- Representación global: $\mathbf{x}_i \in \mathbb{R}^D$, con D número de píxeles de la imagen
- Los vectores propios de la matriz de covarianza tienen esta apariencia:



²Imágenes de <http://www.chrisdecoro.com/eigenfaces/>

Aplicación de PCA: EigenFaces

- Toda imagen \mathbf{x}_i puede ser reconstruida como una suma ponderada de una selección de los k primeros vectores propios (imágenes) $\hat{\mathbf{x}}_i = \bar{\mathbf{x}} + WW^t(\mathbf{x}_i - \bar{\mathbf{x}})$
- A mayor valor de k mejor reconstrucción:



Resultado de la reconstrucción usando $k = 1 \dots 25$ vectores propios

Aplicación de PCA: EigenFaces

- En general, se puede reconstruir cualquier imagen del mismo tamaño que las imágenes empleadas en la obtención de los vectores propios
- La reconstrucción siempre tenderá a ser una cara:





UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

Escola Tècnica Superior d'Enginyeria Informàtica



Tema 4. Clasificación basada en distancias

Percepción (PER)

Curso 2019/2020

Departamento de Sistemas Informáticos y Computación

Índice

- 1 Introducción: espacio métrico y distancias ▷ 3
- 2 Vecino más cercano ▷ 8
- 3 k -vecinos más cercanos ▷ 14
- 4 Relación con la probabilidad a posteriori ▷ 18
- 5 Optimizaciones: aprendizaje de distancias, edición y condensado ▷ 23

Índice

- 1 *Introducción: espacio métrico y distancias* ▷ 3
- 2 Vecino más cercano ▷ 8
- 3 k -vecinos más cercanos ▷ 14
- 4 Relación con la probabilidad a posteriori ▷ 18
- 5 Optimizaciones: aprendizaje de distancias, edición y condensado ▷ 23

Introducción

- **Clasificadores basados en distancias**: basados en la similitud entre la muestras a clasificar, y las *muestras* etiquetadas o **prototipos** dados
- En clasificación un prototipo representa el objeto x y su clase c
- Los prototipos son almacenados para la clasificación de muestras sin etiquetar mediante el cálculo de distancias
- Formalmente:
 - Prototipos: conjunto de pares $X = \{(\mathbf{x}_1, c_1), \dots, (\mathbf{x}_n, c_n)\}$
 - Clasificación: dado y , clasificar según las distancias $d(y, \mathbf{x}_i)$, $1 \leq i \leq n$
- **Espacio de representación E** : debe ser *Métrico* o *Pseudométrico*
 - E puede ser un espacio no vectorial ($E \neq \mathbb{R}^D$)
- **Medida de disimilitud o distancia d** : función que dada la representación de dos objetos $(x, x') \in E \times E$ devuelve un valor de *disimilitud*, que debería correlarse con la disimilitud de dichos objetos en la realidad.

Espacios métricos y pseudométricos

Espacio métrico:

- Par (E, d)
- E : espacio de representación
- d : función *métrica* o ***distancia***, $d : E \times E \rightarrow \mathbb{R}$, que cumple $(\forall p, q, r \in E)$:
 - $d(p, q) \geq 0$; $d(p, q) = 0 \Leftrightarrow p = q$ *(Positiva o nula)*
 - $d(p, q) = d(q, p)$ *(Simétrica)*
 - $d(p, q) + d(q, r) \geq d(p, r)$ *(Desigualdad Triangular)*

Espacio pseudométrico:

- Versión “simplificada”
- No requiere *simetría* ni *desigualdad triangular*
- $d(\cdot, \cdot)$ se denomina ***medida de disimilitud***

Espacios métricos y pseudométricos

Los espacios métricos y pseudométricos son más “primitivos” que los *espacios vectoriales*

Todo espacio vectorial con *producto escalar* se convierte en métrico mediante la definición de la ***distancia euclídea***:

$$d(\mathbf{p}, \mathbf{q}) = \|\mathbf{p} - \mathbf{q}\| = \sqrt{(\mathbf{p} - \mathbf{q})^t(\mathbf{p} - \mathbf{q})} = \\ \sqrt{(p_1 - q_1)^2 + \cdots + (p_D - q_D)^2}$$

Distancias usuales

Representaciones vectoriales D -dimensionales ($E = \mathbb{R}^D$)

- Familia L_p : $d_p(\mathbf{a}, \mathbf{b}) = \left(\sum_{i=1}^D |a_i - b_i|^p \right)^{\frac{1}{p}}$

$L1$	$L2$ (Euclídea)	$L_\infty / L0$
$\sum_{i=1}^D (a_i - b_i) $	$\left(\sum_{i=1}^D (a_i - b_i)^2 \right)^{\frac{1}{2}}$	$\max_{1 \leq i \leq D} (a_i - b_i) $

- Distancia Euclídea Ponderada: $d(\mathbf{a}, \mathbf{b}) = \left(\sum_{i=1}^D w_i \cdot (a_i - b_i)^2 \right)^{\frac{1}{2}}$

Representaciones estructurales por cadenas de primitivas ($E \subseteq \Sigma^*$)

- Distancia (ponderada) de edición: $d(a, b) = \text{mínima talla (o peso) de una secuencia de operaciones de edición que transforma } a \text{ en } b$

Índice

- 1 Introducción: espacio métrico y distancias ▷ 3
- 2 *Vecino más cercano* ▷ 8
- 3 k -vecinos más cercanos ▷ 14
- 4 Relación con la probabilidad a posteriori ▷ 18
- 5 Optimizaciones: aprendizaje de distancias, edición y condensado ▷ 23

Clasificación por el vecino más cercano

Sea $d : E \times E \rightarrow \mathbb{R}$ una métrica y sea y un punto de E

Vecino más cercano (Nearest Neighbour, NN):

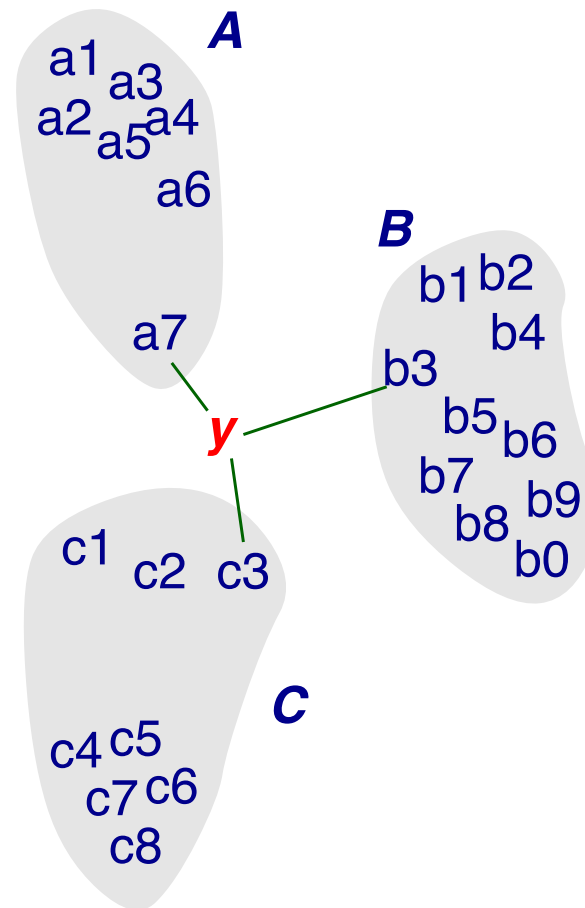
Sea X_c el conjunto de prototipos de la clase c :

$$y \in \hat{c} \Leftrightarrow \exists \mathbf{x} \in X_{\hat{c}} : d(y, \mathbf{x}) \leq d(y, \mathbf{x}') \quad \forall \mathbf{x}' \in X_c, 1 \leq c \leq C, c \neq \hat{c}$$

En caso de empate decidir, entre las empatadas, la clase con mayor número de prototipos o bien aleatoriamente

Clasificación por el vecino más cercano

Regla NN



$NN(y)=a7$

$\hat{c} = A$

Fronteras de decisión

Sea E espacio vectorial con *métrica euclídea*:

$$E = \mathbb{R}^D; \quad d(\mathbf{y}, \mathbf{x}) = \sqrt{(\mathbf{y} - \mathbf{x})^t (\mathbf{y} - \mathbf{x})}$$

La *frontera de decisión* entre las clases i y j son los puntos *equidistantes* al prototipo más cercano de cada clase:

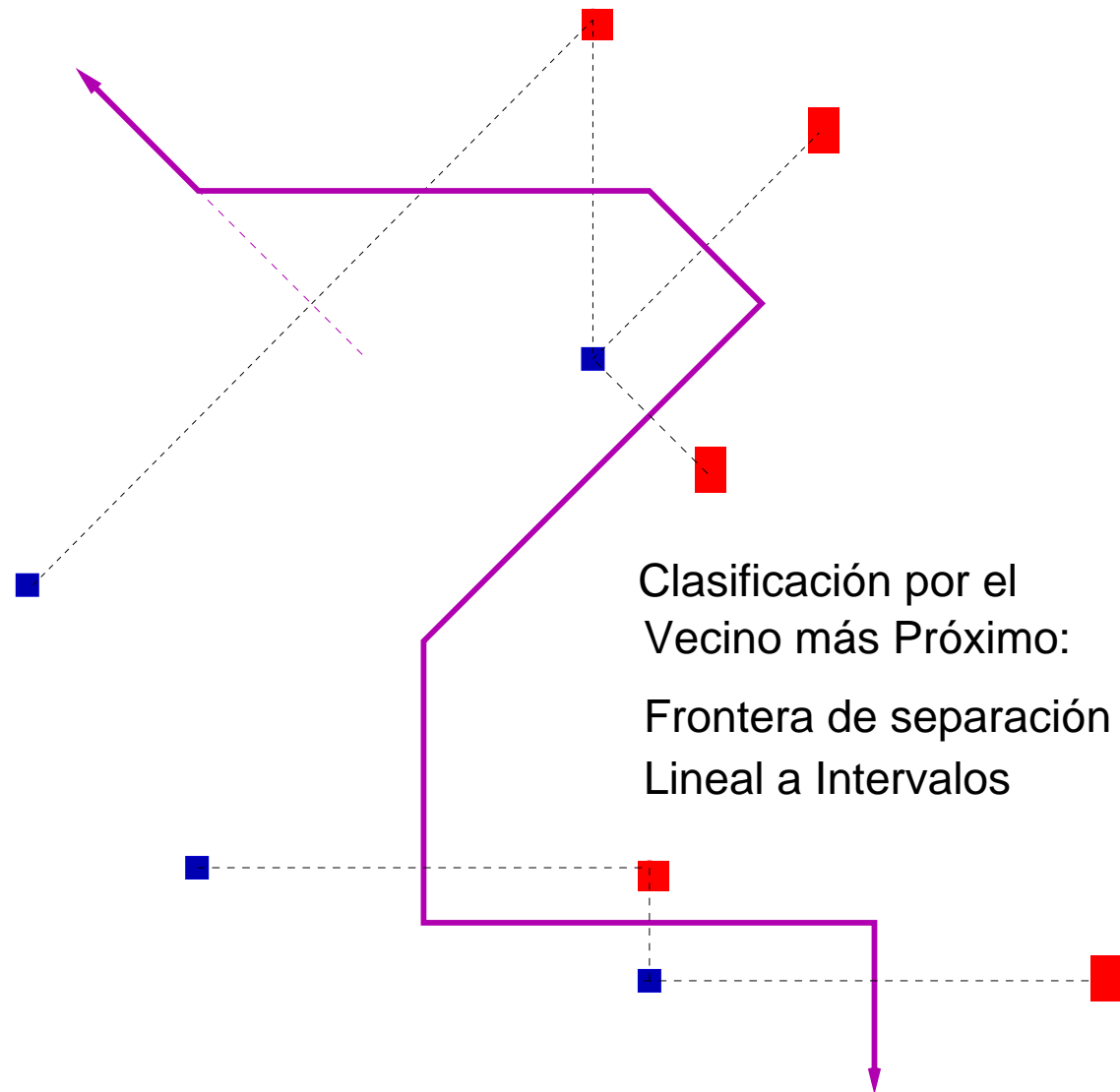
$$\min_{\mathbf{x} \in X_i} (d(\mathbf{y}, \mathbf{x})) = \min_{\mathbf{x} \in X_j} (d(\mathbf{y}, \mathbf{x})) \equiv \min_{\mathbf{x} \in X_i} (\|\mathbf{y} - \mathbf{x}\|^2) = \min_{\mathbf{x} \in X_j} (\|\mathbf{y} - \mathbf{x}\|^2) \equiv$$

$$\min_{\mathbf{x} \in X_i} (-2\mathbf{x}^t \mathbf{y} + \mathbf{x}^t \mathbf{x}) = \min_{\mathbf{x} \in X_j} (-2\mathbf{x}^t \mathbf{y} + \mathbf{x}^t \mathbf{x})$$

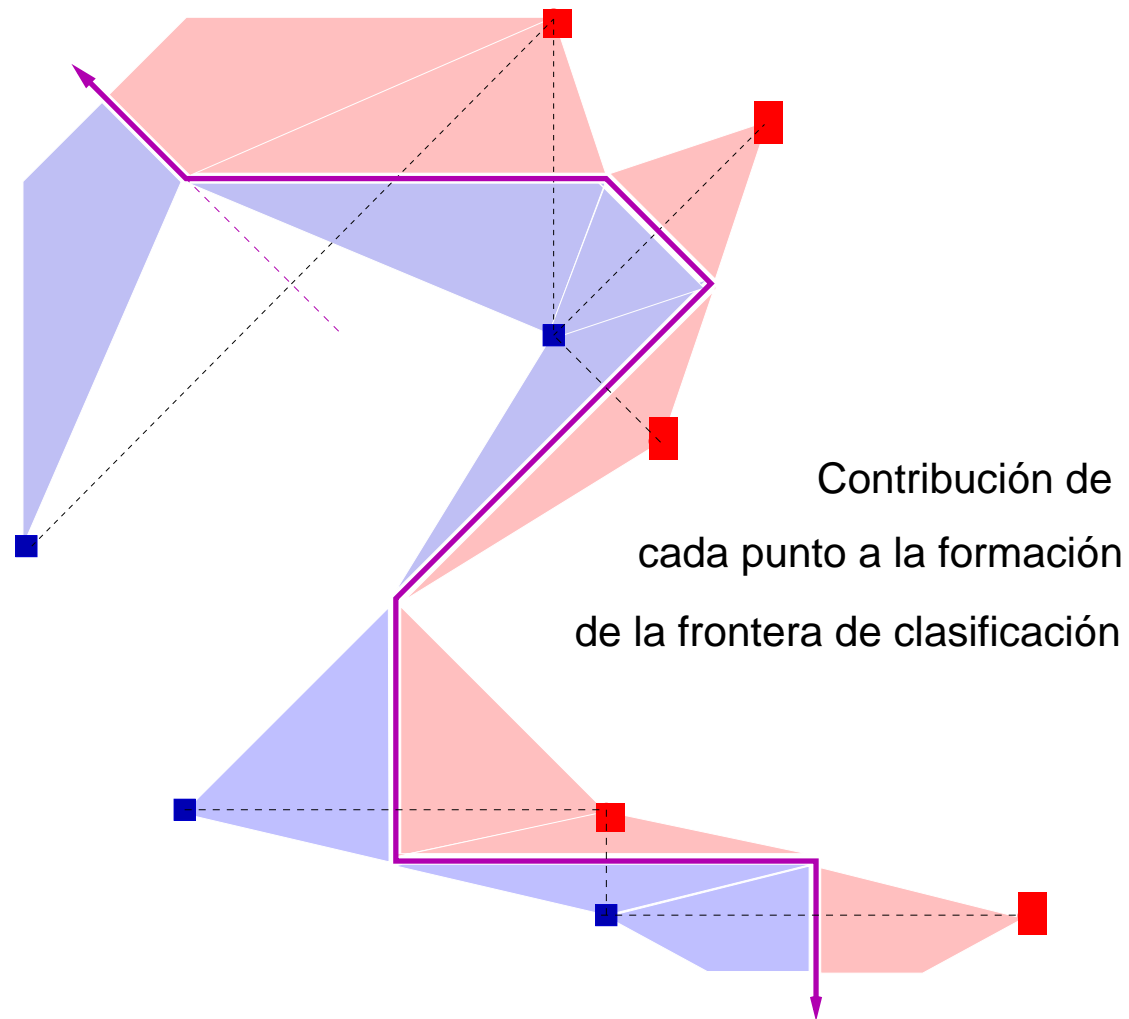
Separación lineal dependiente de los prototipos involucrados: *Funciones Discriminantes Lineales a Trozos (LT)*

Existen fronteras de decisión LT que no pueden obtenerse mediante ningún clasificador NN

Fronteras de decisión asociadas al clasificador NN



Fronteras de decisión asociadas al clasificador NN



Índice

- 1 Introducción: espacio métrico y distancias ▷ 3
- 2 Vecino más cercano ▷ 8
- 3 *k-vecinos más cercanos* ▷ 14
- 4 Relación con la probabilidad a posteriori ▷ 18
- 5 Optimizaciones: aprendizaje de distancias, edición y condensado ▷ 23

Clasificación por los k -vecinos más cercanos

Sea $d : E \times E \rightarrow \mathbb{R}$ una métrica y sea y un punto de E

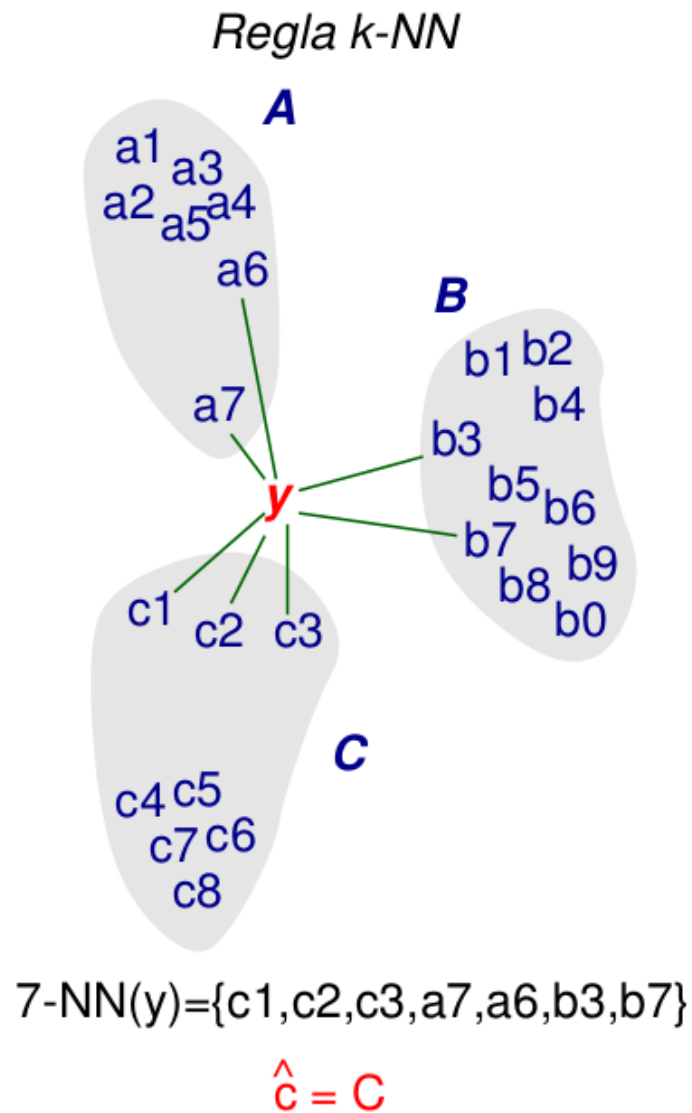
k -vecinos más cercanos (k -NN):

- Sea X_c el conjunto de prototipos representante de la clase c
- Sea X^k el conjunto de los $k \in \mathbb{N}^+$ prototipos más próximos a y

$$y \in \hat{c} \Leftrightarrow |X^k \cap X_{\hat{c}}| \geq |X^k \cap X_c| \quad 1 \leq c \leq C, c \neq \hat{c}$$

- En caso de empate, decidir entre las clases empatadas según 1-NN (1-NN equivale a NN)

Clasificación por los k -vecinos más cercanos



Fronteras de decisión

Sea E espacio vectorial con *métrica euclídea*:

$$E = \mathbb{R}^D; \quad d(\mathbf{y}, \mathbf{x}) = \sqrt{(\mathbf{y} - \mathbf{x})^t(\mathbf{y} - \mathbf{x})}$$

La *frontera de decisión* para *k-NN* vendrá dada por los puntos que empatan a número máximo de vecinos de dos clases distintas:

$$|X^k \cap X_i| = |X^k \cap X_j|$$

Separación dada por *Funciones Discriminantes Lineales a Trozos*

Toda frontera de decisión LT puede obtenerse mediante FDs *k-NN*

Índice

- 1 Introducción: espacio métrico y distancias ▷ 3
- 2 Vecino más cercano ▷ 8
- 3 k -vecinos más cercanos ▷ 14
- 4 *Relación con la probabilidad a posteriori* ▷ 18
- 5 Optimizaciones: aprendizaje de distancias, edición y condensado ▷ 23

k -NN y probabilidad a posteriori

Planteamiento habitual de estimación de $P(c \mid \mathbf{y})$: por fórmula de Bayes

- A partir de los datos de entrenamiento, estimar:
 - *Probabilidades a priori* de las clases $P(c)$, $1 \leq c \leq C$
 - *Probabilidad condicional* de cada clase $p(\mathbf{y}|c)$, $\mathbf{y} \in E$, $1 \leq c \leq C$
- Clasificar por máxima *probabilidad a posteriori* según la *regla de Bayes*:

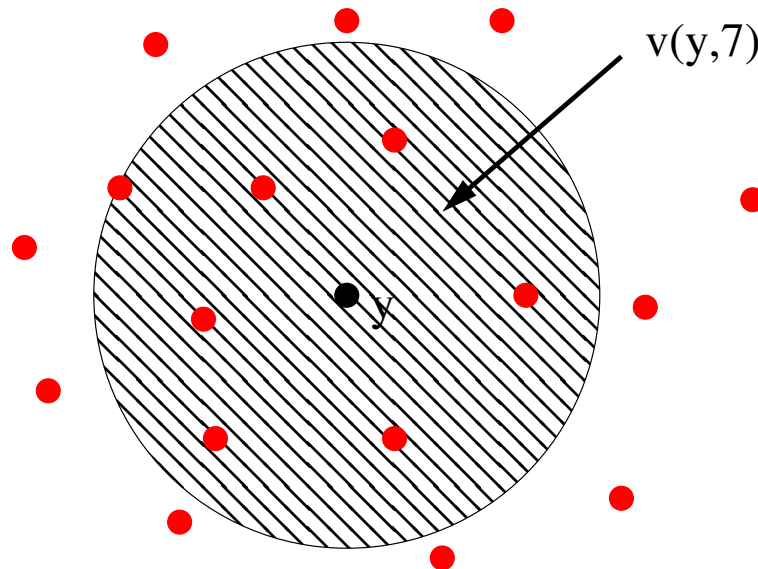
$$P(c \mid \mathbf{y}) = \frac{p(\mathbf{y} \mid c) P(c)}{\sum_{c'=1}^C p(\mathbf{y} \mid c') P(c')}$$

Posibilidad alternativa: *estimar directamente $P(c \mid \mathbf{y})$ a partir de los datos mediante k -NN*

k -NN y probabilidad a posteriori

Dado y un punto en el que queremos estimar $P(c \mid y)$, sean:

$v(y, k)$: volumen de la menor hiperesfera centrada en y que contiene a los k vecinos más próximos de y (de cualquier clase)



k_c : número de prototipos de la clase c de entre los k -NN, $\sum_{c=1}^C k_c = k$

N_c : número de prototipos de la clase c , $\sum_{c=1}^C N_c = N$ (todos los prototipos)

k -NN y probabilidad a posteriori

Estimadores:

- Probabilidades *a priori*: $\hat{P}(c) = \frac{N_c}{N}$, $1 \leq c \leq C$
- *Masa de probabilidad* de la clase c en la hiperesfera de volumen $v(\mathbf{y}, k)$ centrada en \mathbf{y} : k_c/N_c
- Con $v(\mathbf{y}, k)$ infinitesimal, la *condicional* es: $\hat{p}(\mathbf{y} \mid c) = \frac{k_c/N_c}{v(\mathbf{y}, k)}$
- *Probabilidad a posteriori* por regla de Bayes:

$$\hat{P}(c \mid \mathbf{y}) = \frac{\frac{k_c}{N_c} \frac{N_c}{N}}{\sum_{c'=1}^C \frac{k_{c'}}{N_{c'}} \frac{N_{c'}}{N}} = \frac{k_c}{k}$$

Regla de clasificación: $\hat{c} = \operatorname{argmax}_c \hat{P}(c \mid \mathbf{y}) = \operatorname{argmax}_c \frac{k_c}{k} = \operatorname{argmax}_c k_c$

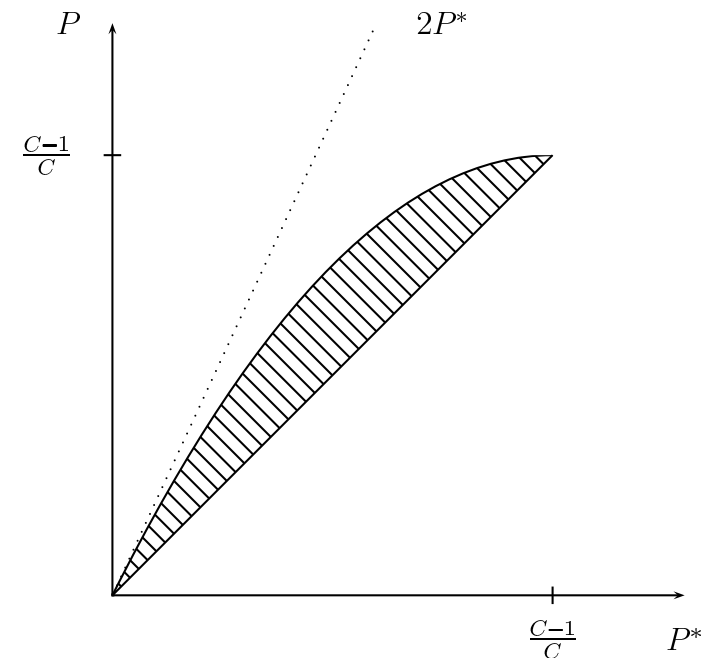
Es decir, clasificar \mathbf{y} en la clase ***a la que pertenezcan la mayoría de sus k vecinos más próximos***

Probabilidad de error de los clasificadores NN y k -NN

Sea P^* el error de Bayes

Cuando el número de prototipos es ilimitado ($N \rightarrow \infty$), el riesgo de **error del clasificador NN** puede acotarse por:

$$P^* \leq P \leq P^* \left(2 - \frac{C}{C-1} P^* \right) \leq 2P^*$$



El riesgo de **error del clasificador k -NN** tiende al error de Bayes si:

$$N \rightarrow \infty; \quad k \rightarrow \infty; \quad \frac{k}{N} \rightarrow 0$$

Las dos últimas condiciones se cumplen, por ejemplo, tomando $k = \sqrt{N}$

Índice

- 1 Introducción: espacio métrico y distancias ▷ 3
- 2 Vecino más cercano ▷ 8
- 3 k -vecinos más cercanos ▷ 14
- 4 Relación con la probabilidad a posteriori ▷ 18
- 5 *Optimizaciones: aprendizaje de distancias, edición y condensado* ▷ 23

Optimizaciones de k -NN

- En la clasificación por k -NN, el único parámetro *directo* a establecer es k
- Sin embargo, existen un par de *meta*-parámetros con gran influencia en la calidad de la clasificación:
 - ***Distancia empleada***: puede adecuarse a la distribución de prototipos de cada clase y eliminar efectos de escala en las distintas componentes
 - ***Conjunto de prototipos*** (puede no ser el disponible originalmente)
 - Puede *limpiarse*: fronteras de decisión más simples, mayor generalización
 - Puede *reducirse*: clasificador más compacto en memoria y más rápido en búsqueda

Aprendizaje de distancias

Limitación: aprendizaje de los pesos asociados a la distancia euclídea ponderada:

$$d(\mathbf{y}, \mathbf{p}) = \sqrt{\left(\sum_{i=1}^D w_i \cdot (y_i - p_i)^2 \right)}$$

- $\mathbf{y} \in \mathbb{R}^D$ es el objeto a clasificar
- $\mathbf{p} \in \mathbb{R}^D$ es un prototipo del conjunto de aprendizaje disponible
- $\mathbf{w} \in \mathbb{R}^D$ es el vector de pesos

Restricción: $w_i > 0, i = 1, \dots, D$ (para que la distancia sea una métrica)

Aprendizaje de distancias

Distancia euclídea normalizada o ***Mahalanobis-diagonal***:

$$d(\mathbf{y}, \mathbf{p}) = \sqrt{\left(\sum_{i=1}^D \frac{1}{\sigma_i^2} (y_i - p_i)^2 \right)}$$

- σ_i : desviación típica de la componente i -ésima de la representación vectorial de los datos
- $w_i = \frac{1}{\sigma_i^2}$: inversa de la varianza de la componente i -ésima

Equivale a pre-normalizar los datos dividiendo cada componente por la desviación típica y usar la distancia euclídea sobre los datos normalizados

Aprendizaje de distancias

Distancia *Mahalanobis-diagonal por clase*:

$$d(\mathbf{y}, \mathbf{p}) = \sqrt{\left(\sum_{i=1}^D \frac{1}{\sigma_{ic}^2} (y_i - p_i)^2 \right)}$$

- c : clase de \mathbf{p}
- σ_{ic}^2 : varianza de la componente i -ésima en clase c
- $w_i = \frac{1}{\sigma_{ic}^2}$: inversa de la varianza de la componente i -ésima considerando sólo prototipos de la clase c

Esta distancia *ya no es una métrica*: pesos diferentes según la clase de \mathbf{p}

Aprendizaje de distancias

Distancia *Mahalanobis-Local*:

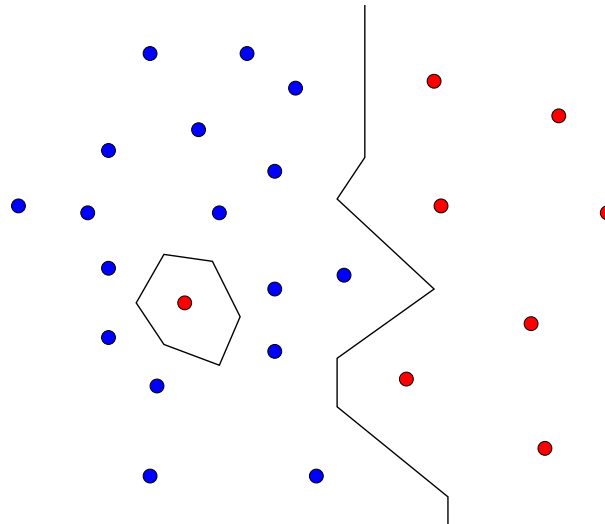
$$d(\mathbf{y}, \mathbf{p}) = \sqrt{\left(\sum_{i=1}^D \frac{1}{\sigma_{i\mathbf{p}}^2} (y_i - p_i)^2 \right)}$$

- $\sigma_{i\mathbf{p}}^2$: varianza de la componente i -ésima de los prototipos que son k -NN de \mathbf{p} de su misma clase
- $w_i = \frac{1}{\sigma_{i\mathbf{p}}^2}$: inversa de la varianza de la componente i -ésima calculada sobre los prototipos k -NN de \mathbf{p} de su misma clase \rightarrow estimación de la varianza local de la clase c

Esta distancia *tampoco es una métrica*: pesos diferentes dependiendo de \mathbf{p}

Edición de prototipos

- Objetivo: eliminar prototipos ruidosos
- Prototipo ruidoso: prototipo de una clase *aislado* dentro de la zona de prototipos de otra clase

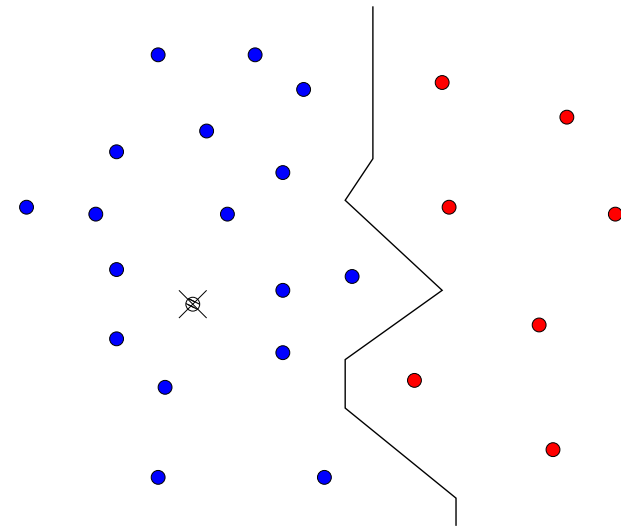


- El punto ruidoso genera *huecos* en las regiones de decisión
- Eliminar prototipos ruidosos da regiones de decisión simplemente conexas (sin *huecos*)

Edición de prototipos

Algoritmo de edición de Wilson

- Clasifica por k -NN (k parámetro) de cada prototipo frente al resto
- Elimina los prototipos cuya clasificación sea diferente de su propia clase
- Finalización: todos los prototipos se clasifican correctamente
- Coste computacional por recorrido para n prototipos: $O(n^2)$
 - Probar n prototipos
 - Para cada uno calcular vecinos más cercanos (n distancias)
- Técnicas para bajar el coste:
 - Almacenar las distancias ya ordenadas en una matriz en la primera iteración
 - Emplear técnicas de búsqueda rápida
- Usualmente consigue eliminar los prototipos ruidosos
- Crítica principal: resultado dependiente del orden de recorrido



Edición de prototipos

Algoritmo de Wilson:

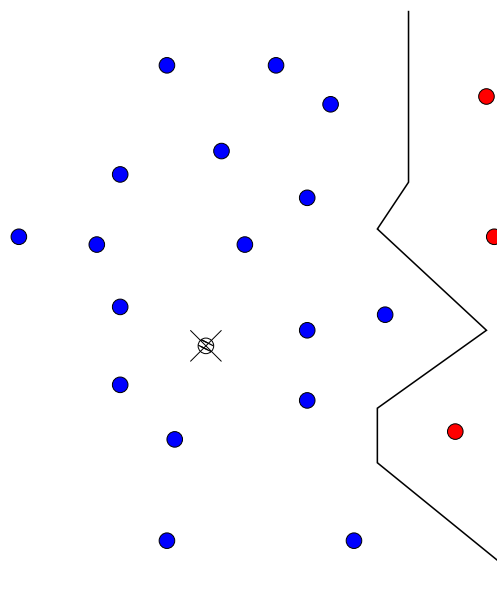
- Entrada: $X = \{(\mathbf{x}_1, c_1), \dots, (\mathbf{x}_N, c_N)\}$, k , d
- Salida: $X' = \{(\mathbf{x}_1, c_1), \dots, (\mathbf{x}_M, c_M)\}$ $M \leq N$
- Algoritmo:
 1. error=true; $X' = X$;
 2. while (error)
 3. error=false;
 4. for i=1:N
 5. $\hat{c} = knn(\mathbf{x}_i, X' - \mathbf{x}_i, d, k)$;
 6. if ($\hat{c} \neq c_i$) $X' = X' - \{\mathbf{x}_i\}$; error=true;
 7. endfor
 8. endwhile

Entrada: conjunto de prototipos original, valor de k , distancia d a emplear

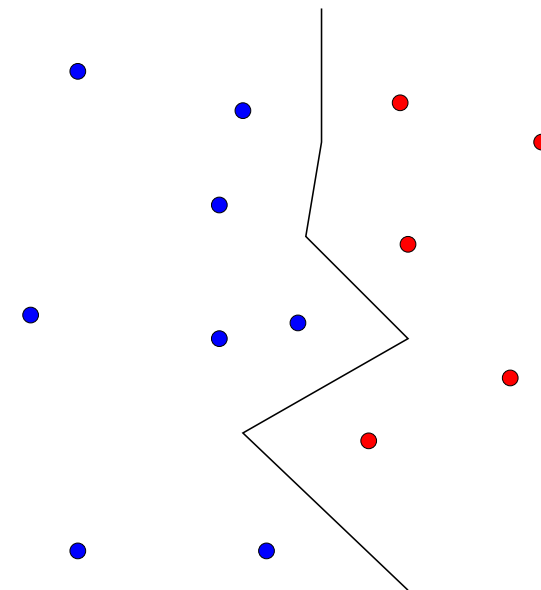
Salida: conjunto reducido o igual, $X' \subseteq X$

Condensado de prototipos

- Objetivo: reducir drásticamente el conjunto de prototipos sin modificar significativamente las fronteras de decisión
- Algunos algoritmos de condensado deben partir del conjunto de prototipos ya editado (sin ruido)



Editado



Condensado

Condensado de prototipos

Algoritmo CNN (Condensed Nearest Neighbor, Hart, 1968):

- Definir dos conjuntos de prototipos:
 - STORE (S): prototipos a retener
 - GARBAGE (G): prototipos a descartar
- Algoritmo en dos fases:
 1. Crear S y G desde los prototipos originales (conjunto X)
 2. Recorrer G hasta que quede vacío o no sufra modificaciones
- Esencialmente:
 - S mantiene los prototipos clasificados incorrectamente
 - G mantiene los prototipos clasificados correctamente
- Crítica principal: resultado dependiente del orden de recorrido

Condensado de prototipos

Algoritmo CNN

- Entrada: $X = \{(\mathbf{x}_1, c_1), \dots, (\mathbf{x}_N, c_N)\}$ editado, k, d
- Salida: S
- Algoritmo:

1. // Primera fase

- $S = G = \emptyset$
- $\{(\mathbf{x}_1, c_1)\} \rightarrow S$
- for $i=2:N$
- $\hat{c} = knn(\mathbf{x}_i, S, d, k);$
- if $(\hat{c} \neq c_i)$ $\{(\mathbf{x}_i, c_i)\} \rightarrow S$
- else $\{(\mathbf{x}_i, c_i)\} \rightarrow G$
- endfor

2. // Segunda fase

- error=true
- while $(G \neq \emptyset \ \&\& \text{error})$
- error=false;
- forall $(\mathbf{x}, c) \in G$
- $\hat{c} = knn(\mathbf{x}, S, d, k);$
- if $(\hat{c} \neq c)$
- $\{(\mathbf{x}, c)\} \rightarrow S;$
- $G = G - \{(\mathbf{x}, c)\};$
- error=true;
- endif
- endforall
- endwhile