Visión por Computadora I

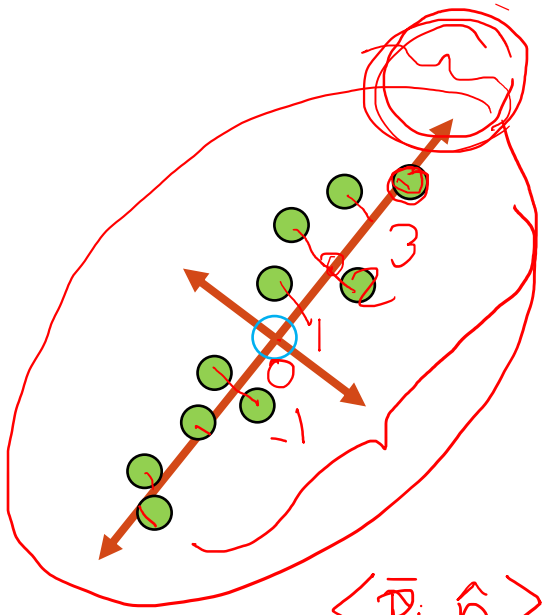
Ing. Andrés F. Brumovsky

(abrumov@fi.uba.ar)

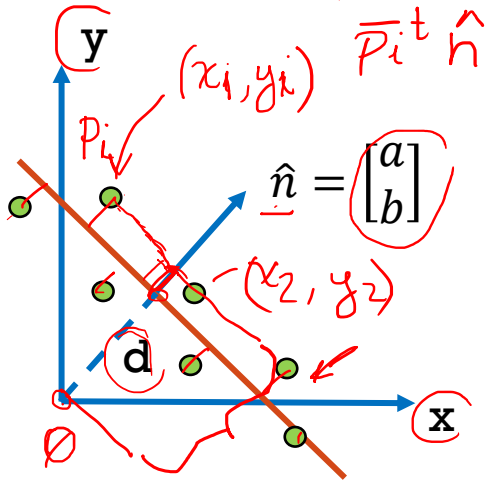
Laboratorio de Sistemas Embebidos -FIUBA

FACULTAD
DE INGENIERIA
Universidad de Buenos Aires

PCA (PRINCIPAL COMPONENTS ANALYSIS)



- Los modelos generativos funcionan mejor en espacios con pocas dimensiones (estimar las funciones de densidad en espacios de muchas dimensiones requieren una gran cantidad de datos)
- PCA es una técnica de reducción de dimensiones
  1. El primer componente principal es la dirección de máxima varianza (alrededor de la media).
  2. Los siguientes componentes principales son ortogonales a los previos y describen la máxima variancia residual



- El error de ajuste a una línea lo podemos escribir como

$$E(a,b,d) = \sum_i (ax_i + by_i - d)^2$$

$$\frac{\partial E}{\partial d} = 0 \rightarrow -2\sum_i (ax_i + by_i - d) = 0$$

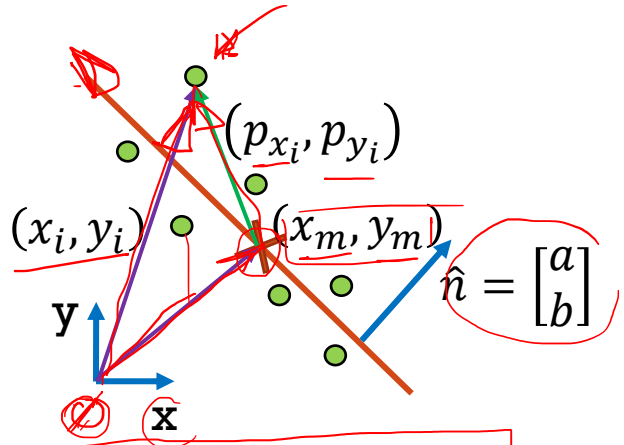
$$d = ax_m + by_m$$

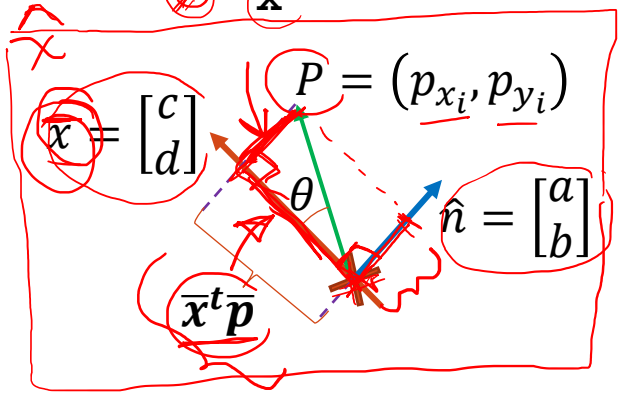
- Substituyendo en la primer ecuación

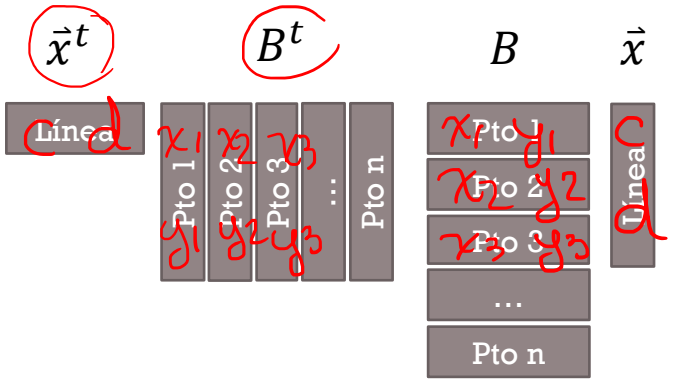
$$E = \sum_i [a(x_i - x_m) + b(y_i - y_m)]^2 = \|\boldsymbol{B}\hat{n}\|_2^2 \rightarrow \boldsymbol{B} = \begin{bmatrix} x_1 - x_m & y_1 - y_m \\ \cdots & \cdots \\ x_n - x_m & y_n - y_m \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix}$$

Donde minimizar $\|\boldsymbol{B}\hat{n}\|^2$ devuelve los ejes de menor inercia

PCA – INTERPRETACIÓN ALGEBRAICA I







- Por suma de vectores podemos escribir

$$(x_m, y_m) + (p_{x_i}, p_{y_i}) = (x_i, y_i)$$
$$(p_{x_i}, p_{y_i}) = (x_i, y_i) - (x_m, y_m)$$

- Minimizar la proyección sobre la normal $\hat{n}^t\bar{p}$ es equivalente a maximizar la proyección sobre la dirección unitaria en dirección de la línea $\bar{x}^t\bar{p}$
- De la diapositiva anterior sabemos que queremos minimizar la suma de todas estas distancias (para cada punto) al cuadrado ¿Cómo podemos escribir esto de manera algebraica?
- Podemos escribirlo matricialmente de la siguiente manera:

$$E = \bar{x}^t \boldsymbol{B^t B} \bar{x} \text{ sujeto a } \bar{x}^t\bar{x} = 1$$

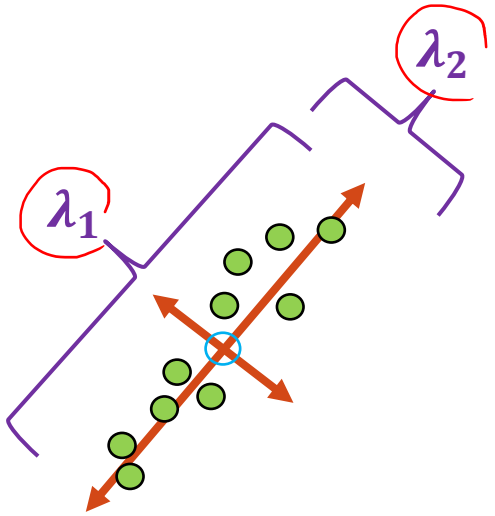
- Esto es lo mismo que escribir

$$\sum_i (\bar{x}^t\bar{p}_i)^2$$

- Veamos el caso para solo dos puntos (aunque puede haber m)

$$\begin{aligned}
&[c \quad d]\begin{bmatrix} p_{x_1} & p_{x_2} \\ p_{y_1} & p_{y_2} \end{bmatrix}\begin{bmatrix} p_{x_1} & p_{y_1} \\ p_{x_2} & p_{y_2} \end{bmatrix}\begin{bmatrix} c \\ d \end{bmatrix} = [c \quad d]\begin{bmatrix} p_{x_1}^2 + p_{x_2}^2 & p_{x_1}p_{y_1} + p_{x_2}p_{y_2} \\ p_{x_1}p_{y_1} + p_{x_2}p_{y_2} & p_{y_1}^2 + p_{y_2}^2 \end{bmatrix}\begin{bmatrix} c \\ d \end{bmatrix} \\
&= c\left(c(p_{x_1}^2 + p_{x_2}^2) + d(p_{x_1}p_{y_1} + p_{x_2}p_{y_2})\right) + d\left(c(p_{x_1}p_{y_1} + p_{x_2}p_{y_2}) + d(p_{y_1}^2 + p_{y_2}^2)\right) \\
&= c^2p_{x_1}^2 + c^2p_{x_2}^2 + cdp_{x_1}p_{y_1} + cdp_{x_2}p_{y_2} + cdp_{x_1}p_{y_1} + cdp_{x_2}p_{y_2} + d^2p_{y_1}^2 + d^2p_{y_2}^2 \\
&= (cp_{x_1} + dp_{y_1})^2 + (cp_{x_2} + dp_{y_2})^2 \\
&= \sum_i (\bar{x}^t\bar{p}_i)^2
\end{aligned}$$

PCA - INTERPRETACIÓN ALGEBRAICA II

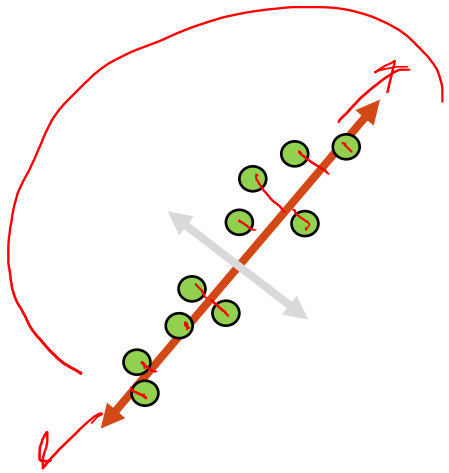


- Dado que lo que queremos hacer es encontrar el máximo de la ecuación anterior pero sujeto a la restricción $x^t x = 1$ (dado que la dirección normal es un vector unitaria) utilizamos el método de multiplicadores de Lagrange.

$$\boldsymbol{M} = \boldsymbol{B^t B} \rightarrow E = \bar{x}^t \boldsymbol{M} \bar{x}, \quad x^t x = 1$$
$$E' = \bar{x}^t \boldsymbol{M} \bar{x} - \lambda(\bar{x}^t \bar{x} - 1) \quad \text{Lagrange.}$$
$$\frac{\partial E'}{\partial \bar{x}} = 2\boldsymbol{M}\bar{x} - 2\lambda\bar{x} = 0 \rightarrow \boldsymbol{M}\bar{x} = \lambda\bar{x}$$

- Es decir, x es un autovector de $\boldsymbol{M} = \boldsymbol{B^t B}$
- De la diapositiva anterior también podemos obtener una interpretación algebraica distinta:

$$B^t B = \begin{bmatrix} \sum_{i=1}^{n} x_i^2 & \sum_{i=1}^{n} x_i y_i \\ \sum_{i=1}^{n} x_i y_i & \sum_{i=1}^{n} y_i^2 \end{bmatrix} \rightarrow \textit{matrix de covarianza alrededor del origen}$$



- De donde se deduce que los componentes principales son las direcciones ortogonales de la matriz de covarianza de un conjunto de puntos (los ejes mayores de la elipsoide en el espacio). Es decir, los autovectores de esta matriz de covarianza.
- A su vez los autovalores de esta matriz de covarianza indican la magnitud de las variaciones alrededor de estas direcciones
  - El autovalor λ_M más grande captura la mayor variación alrededor de los vectores de características $\boldsymbol{x}$
  - El autovalor λ_m más pequeño define las direcciones de menor variación
- Para el grupo de características de la derecha podemos describirlo por su posición a lo largo de la dirección de mayor variación.

EIGENFACES

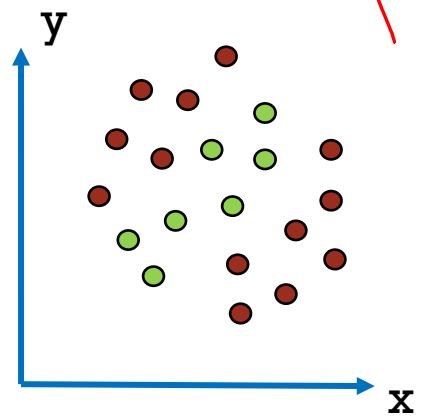
- Supongamos todo el universo de imágenes de 100 x 100 píxeles
  - Si tomamos como característica el valor de intensidad de cada píxel → $dim(\boldsymbol{x}) = 10000$
  - *¿Cuántas de estas imágenes son solo caras?*
- Queremos modelar el subespacio de imágenes de caras
  - Es decir construir un subespacio lineal de menor dimensión que mejor explique la variación en un conjunto de imágenes de caras.
- Supongamos M puntos $(\boldsymbol{x_1}, .., \boldsymbol{x_M})$ en un espacio $\mathcal{R}^d$ con d grande (por ejemplo 10000 o más)
  - Queremos las direcciones $\bar{u}$ en $\mathcal{R}^d$ que capturen la mayor variación de los x_i, sus coeficientes (proyecciones) serán:

$$u(x_i) = \bar{u}^t(\bar{x}_i - \bar{x}_m)$$

- La varianza de los datos proyectados sobre la dirección $\bar{u}$ se puede escribir entonces como:

$$var(\bar{u}) = \frac{1}{M}\sum_{i=1}^{M} \bar{u}^t(\bar{x}_i - \bar{x}_m)\left(\bar{u}^t(\bar{x}_i - \bar{x}_m)\right)^t$$
$$= \bar{u}^t\left[\frac{1}{M}\sum_{i=1}^{M}(\bar{x}_i - \bar{x}_m)(\bar{x}_i - \bar{x}_m)^t\right]\bar{u}$$
$$= \bar{u}^t \boldsymbol{\Sigma} \bar{u}$$

- $\boldsymbol{\Sigma}$ es la matriz de covarianza y la dirección de u (vector unitario) que maximiza la varianza es el autovector asociado con el autovalor más grande de $\boldsymbol{\Sigma}$.

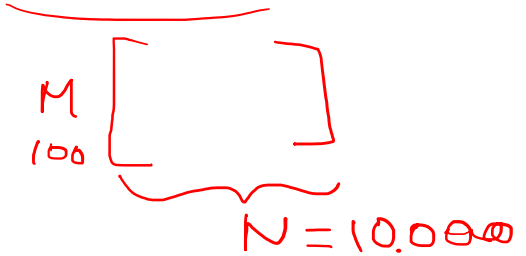


EIGENFACES

- Entonces, la dirección que contiene la máxima covarianza de los datos es el autovector correspondiente al autovalor más grande de la matriz de covarianza de los datos.
- Aún más, las k direcciones ortogonales (autovectores) correspondientes a los k autovalores más grandes contienen las direcciones de mayor variación de los datos.
- Entonces podríamos pasar de 10,000 dimensiones (como en el ejemplo) a 40, 30 o inclusive 10 direcciones ortogonales (dimensiones) de mayor variación.
- Ahora, supongamos que tenemos justamente un conjunto de M puntos con N dimensiones.

$$\boldsymbol{B}_{M\times N} \rightarrow (\boldsymbol{B^t B})_{N\times N}$$

$$Si\ N = 10{,}000 \Rightarrow (\boldsymbol{B^t B})\ tiene\ 100{,}000{,}000\ datos!$$



- **Truco:**

$$\boldsymbol{B^t B}\,\bar{v}_i = \lambda \bar{v}_i$$
$$\boldsymbol{B}\,\boldsymbol{B^t B}\,\bar{v}_i = \lambda \boldsymbol{B}\bar{v}_i$$
$$(\boldsymbol{B}\,\boldsymbol{B^t})\boldsymbol{B}\,\bar{v}_i = \lambda \boldsymbol{B}\bar{v}_i$$

$(BB^t)_{M\times M}$

- Finalmente, si calculamos los autovectores de $\boldsymbol{BB^t}$ (de dimensión MxM) y los premultiplicamos por la matriz $\boldsymbol{B}$ obtendremos los autovectores de la matriz $\boldsymbol{B^t B}$
- M entonces será la cantidad de caras posibles y entre todo el universo de caras resultará $M \ll N$
- ¿Ahora, cuántos autovectores tiene esta matriz? → M-1 (los datos tienen restada la media)

EIGENFACES

- Mathew A. Turk y Alex P. Pentland – 1991

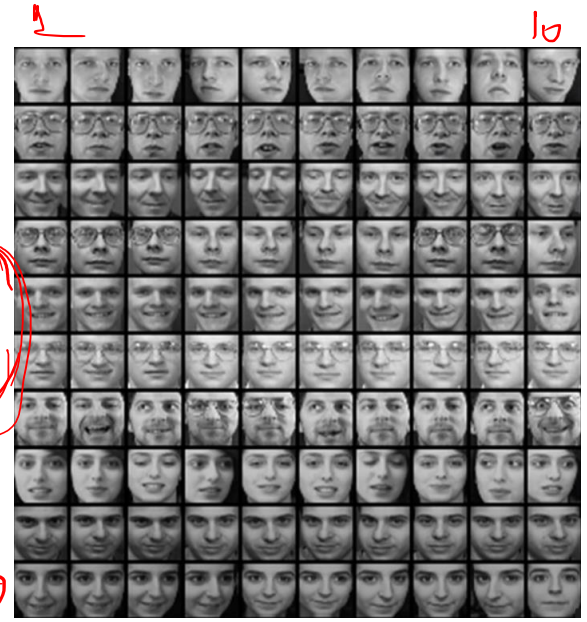
1. Asumen que la mayoría de las caras se encuentran en un subespacio de dimensión baja definido por las primeras k direcciones de máxima varianza ($k \ll d$)
2. Usan PCA para determinar los autovectores (o "eigenfaces") $u_1, u_2, \dots, u_k$ que representan el subespacio.
3. Representan todas las imágenes de rostros en el conjunto de datos (dataset) como combinaciones lineales de eigenfaces

- Imágenes de entrenamiento $\bar{x}_1, \dots, \bar{x}_M$
- Las eigenfaces son en realidad vectores (arreglados con matriz y mostrados como imagen) de 10.000 posiciones (en este ejemplo)
- Si quisiéramos hacer el producto interno (proyección) de una cara nueva $\bar{x}$ en este espacio de eigenfaces solo tenemos que superponer la imagen de la cara sobre cualquiera de estas eigenfaces y multiplicar punto a punto.

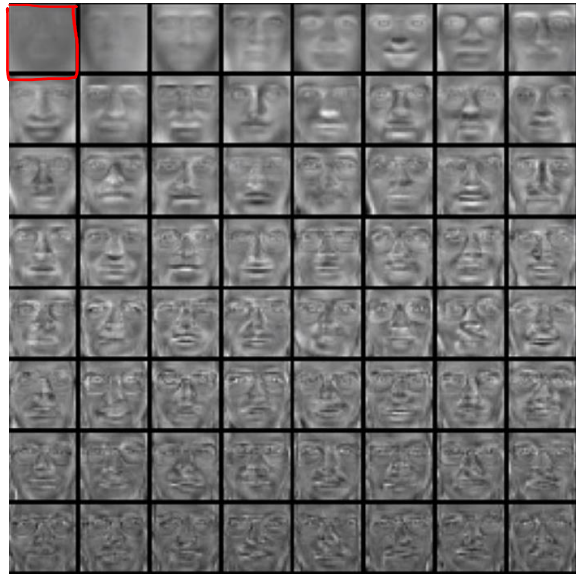
$$\bar{x} \to [\bar{u}_1^t(\bar{x} - \bar{u}_0), \dots, \bar{u}_k^t(\bar{x} - \bar{u}_0)]$$
$$= [w_1, \dots, w_k]$$

- Donde estos pesos w_i son la representación de la cara, de 10.000 valores a k

$\hat{x}$ $\bar{u}_0$ $w_1\bar{u}_0 + w_2\bar{u}_2 + w_3\bar{u}_3 + w_4\bar{u}_4 + \cdots$



Cara promedio ($\bar{u}_0$)



Eigenfaces: $u_1, \dots, u_k$

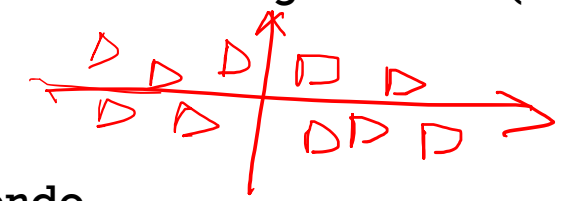
EIGENFACES

$\bar{u}_1, \ldots, \bar{u}_k$

$\bar{u}_0 + 3\sigma_k \bar{u}_k$

$\bar{u}_0 - 3\sigma_k \bar{u}_k$

- ¿Cómo lo usamos para reconocimiento?. Dada una nueva imagen $\bar{x}$
  1. Proyectamos en el subespacio: $[w_1, \ldots, w_k] = [\bar{u}_1^t(\bar{x} - \bar{u}_0), \ldots, \bar{u}_k^t(\bar{x} - \bar{u}_0)]$
  2. Opcionalmente podemos verificar el error de reconstrucción $\bar{x} - \hat{x}$ para determinar si es o no una imagen real
  3. La clasificamos como la cara de entrenamiento más cercana en el espacio k-dimensional
  4. Podemos pensar en este procedimiento como en un modelo generativo (se modelan las poblaciones)



- Contras
  1. No es robusto frente a desalineaciones
  2. No es robusto frente a fuertes variaciones del fondo
  3. A veces la dirección de máxima varianza no es la óptima para la clasificación

CLASIFICACIÓN DISCRIMINATIVA

- **"No buscamos modelar una clase sino aprender a discriminar entre ellas"**
- Distintos métodos de clasificación discriminativa
  - **Nearest Neighbor** – Shakhnarovich, Viola, Darrel (2003) – Berg, Berg, Malik (2005)
  - **SVMs (Support Vector Machine)** – Guton, Vapnik, Heisele, Serre, Poggio (2001)
  - **Boosting** – Viola, Jones (2001) – Torralba et. al. (2004) – Opelt et. al. (2006)
  - **Random Forest** – Breiman (1984) – Shotton, et al CVPR 2008
  - **Redes neuronales** – LeCun, Bottou, Bengio, Haffner (1998) – Rowley, Baluja, Kanade (1998)
- Asumimos
  - Un número fijo conocido de clases (no aparecerán clases nuevas)
  - Amplio número de datos de entrenamiento de cada clase
  - Igual costo de cometer errores de clasificación (entre clases)
  - No sabemos "a priori" las características que definen una clase
- Dividimos el procedimiento en dos partes
  - **Entrenamiento**
    - Conseguir las imágenes modelo: Casos positivos + casos negativos
    - Definir las características a evaluar: LBP, HoG, etc.
    - Enseñar/entrenar un clasificador
  - **Testeo**
    - Generar candidatos (en una nueva imagen)
      - Ventana deslizante, diferentes escalas
    - Puntuar los candidatos

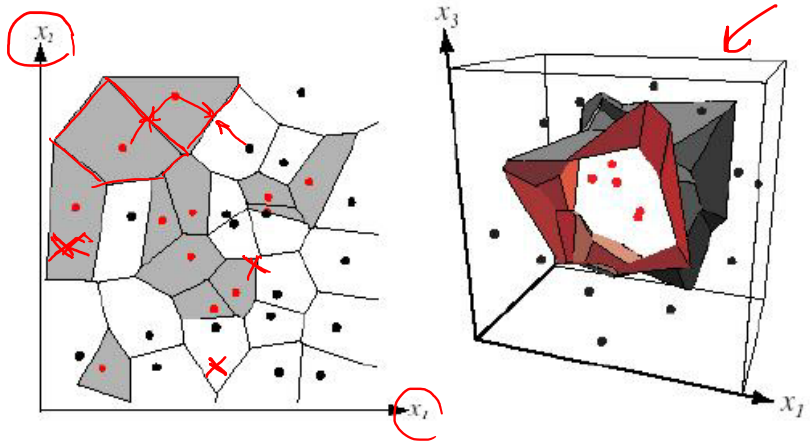
NEAREST NEIGHBOR

- La idea es asignar a un nuevo caso la etiqueta del punto de entrenamiento más cercano
- La partición del espacio de características es a través de celdas de Voronoi
  - Las fronteras se definen como:

$$R_k = \{x \in X | d(x, P_k) \leq d(x, P_j) \; \forall \, j \neq k\}$$

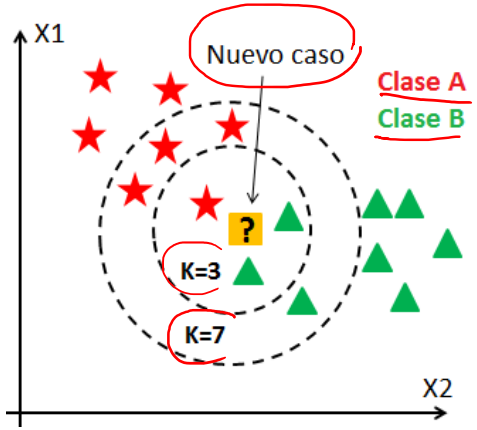
  - La forma de las celdas pueden depender de la definición de distancia considerada
- Es fácil de implementar
- Es muy demandante en cuanto a recursos de memoria y procesamiento en la fase de testeo
  - Se deben recordar todos los datos de entrenamiento
  - Al comprobar un nuevo caso se debe evaluar la distancia contra todos los datos de entrenamiento (o utilizar un KD tree)
- k-NN: Robustece el criterio de decisión
  - Para un nuevo punto encuentra los k puntos más cercanos
  - Realiza una votación entre esos k puntos para clasificar
  - Este método de consenso funciona muy bien, aunque sigue siendo intensivo en el uso de datos.

Pattern Classification
Peter Hart, Richard Duda, David Stork



Negros: Casos positivos

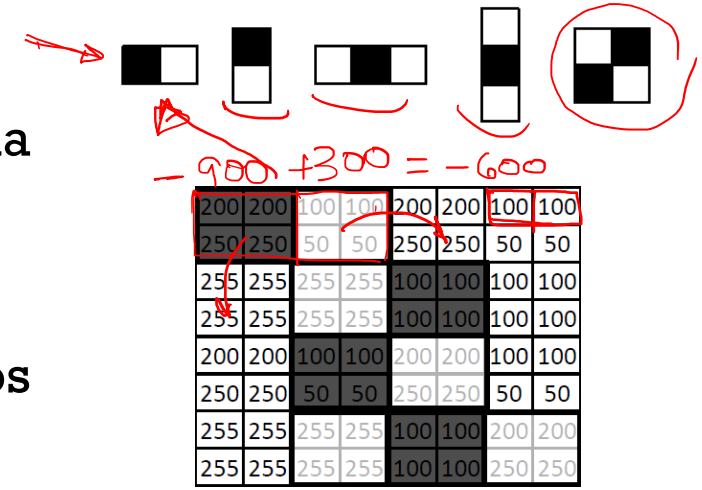
Rojos: Casos negativos

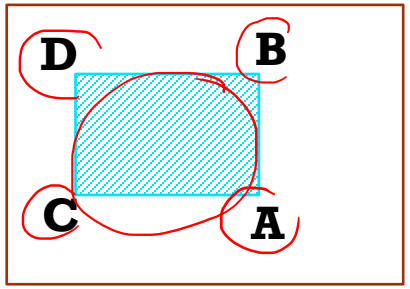


FILTROS DE HAAR

- Buscan detectar patrones de brillo como características de la imagen (diferencias de intensidad entre zonas adyacentes)
- Se pueden definir a múltiples escalas y posiciones
- El resultado del filtro es la diferencia en la suma de los valores de los píxeles entre zonas claras y oscuras
- Se pueden computar muy rápidamente utilizando imágenes integrales → solo tres operaciones de suma para encontrar el área de cualquier tamaño de rectángulo
- Considerando todos los posibles parámetros (posición y escala y tipo) de estos filtros se pueden armar en imágenes de 24x24 más de 180.000 características

Filtros de Haar Básicos





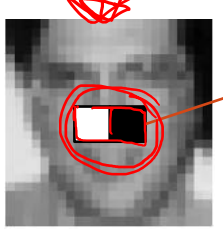
Suma= A-B-C+D

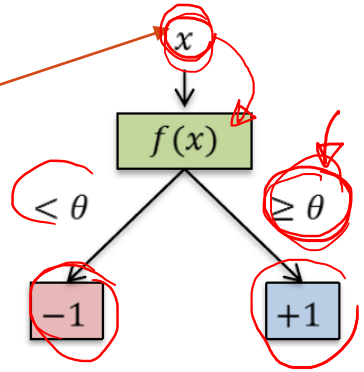
BOOSTING (ADABOOST)

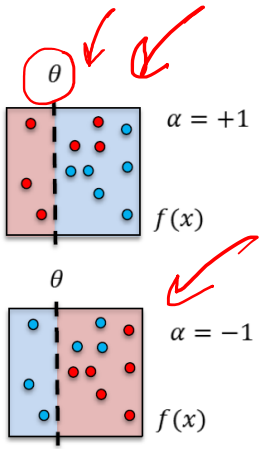
- "Rapid Object Detection using a Boosted Cascade of Simple Features" – Paul Viola, Michael Jones, 2001
- Es un método iterativo que se basa en dos pasos
  1. Encontrar el clasificador "débil" que obtiene el menor error de entrenamiento
  2. Elevar los pesos de las muestras de entrenamiento mal clasificadas por ese clasificador
- En el caso de características de Haar cada clasificador débil se basa en el resultado de la aplicación de una característica de Haar sobre la imagen.

$$h(x) = \begin{cases} -\alpha & f(x) < \theta \\ \alpha & f(x) \geq \theta \end{cases}$$

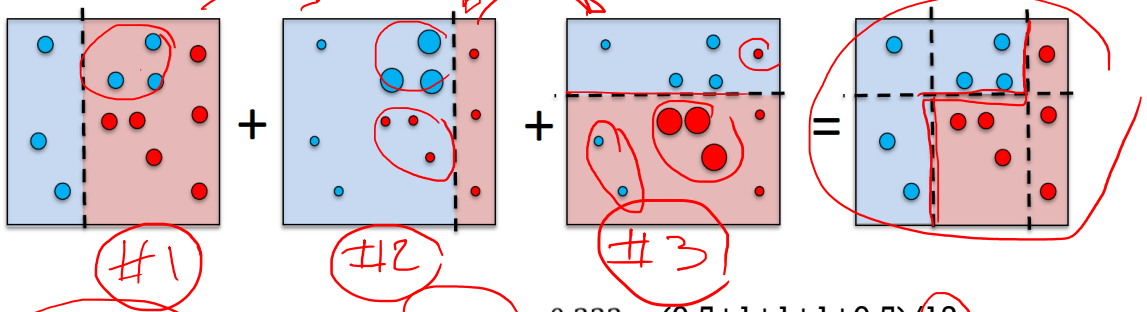
$$\alpha = \{-1, +1\}$$



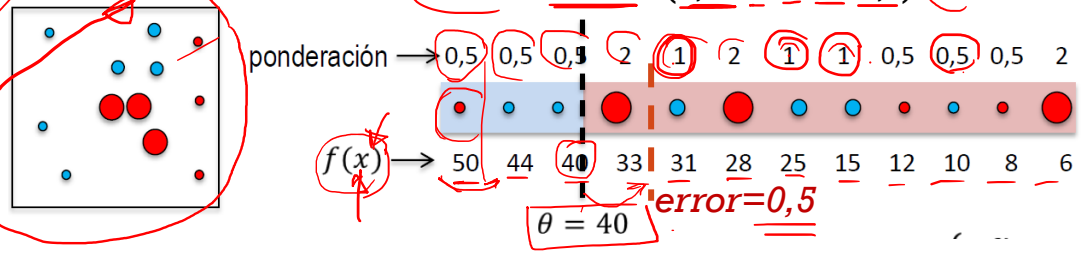


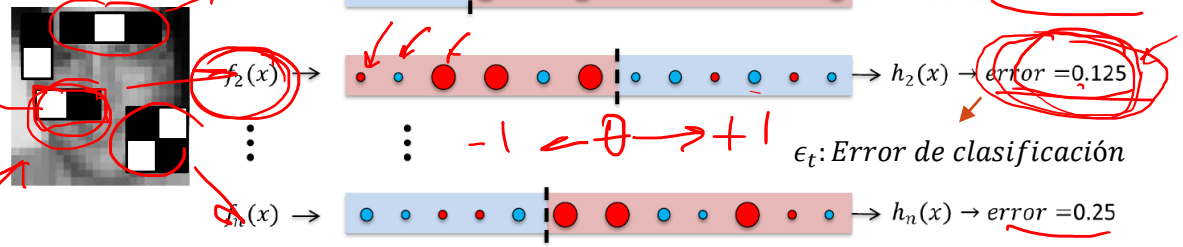


Imágenes: Universitat Autònoma de barcelona



$error = 0.333 = (0{,}5+1+1+1+0{,}5)/12$





Actualización de pesos

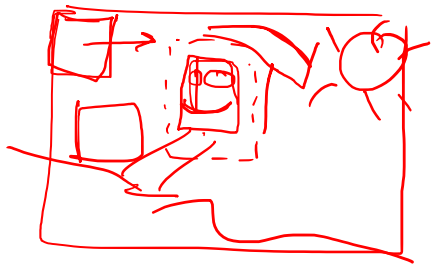
$$w_i(t+1) \begin{cases} \frac{1}{\epsilon_t}\frac{w_i(t)}{2} & h(x_i) \neq y_i (casos\ mal\ clasificados) \\ \frac{1}{1-\epsilon_t}\frac{w_i(t)}{2} & h(x_i) = y_i\ (casos\ bien\ clasificados) \end{cases}$$

Clasificador fuerte

$$H(x) = signo\left(\sum_{i=1}^{T} \alpha_i h_i(x)\right)$$

$$\alpha_t = \frac{1}{2}\log\frac{1-\epsilon_t}{\epsilon_t}$$

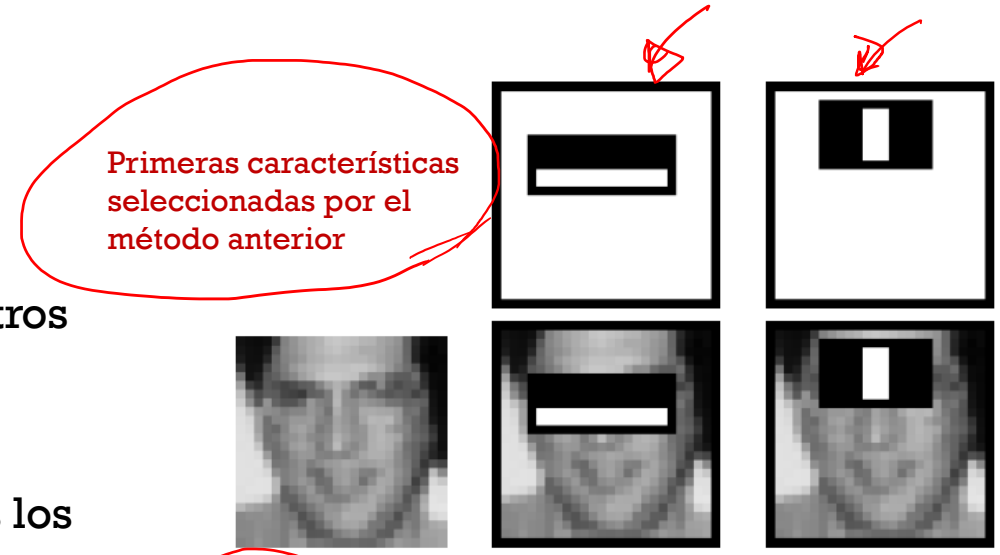
ADABOOST

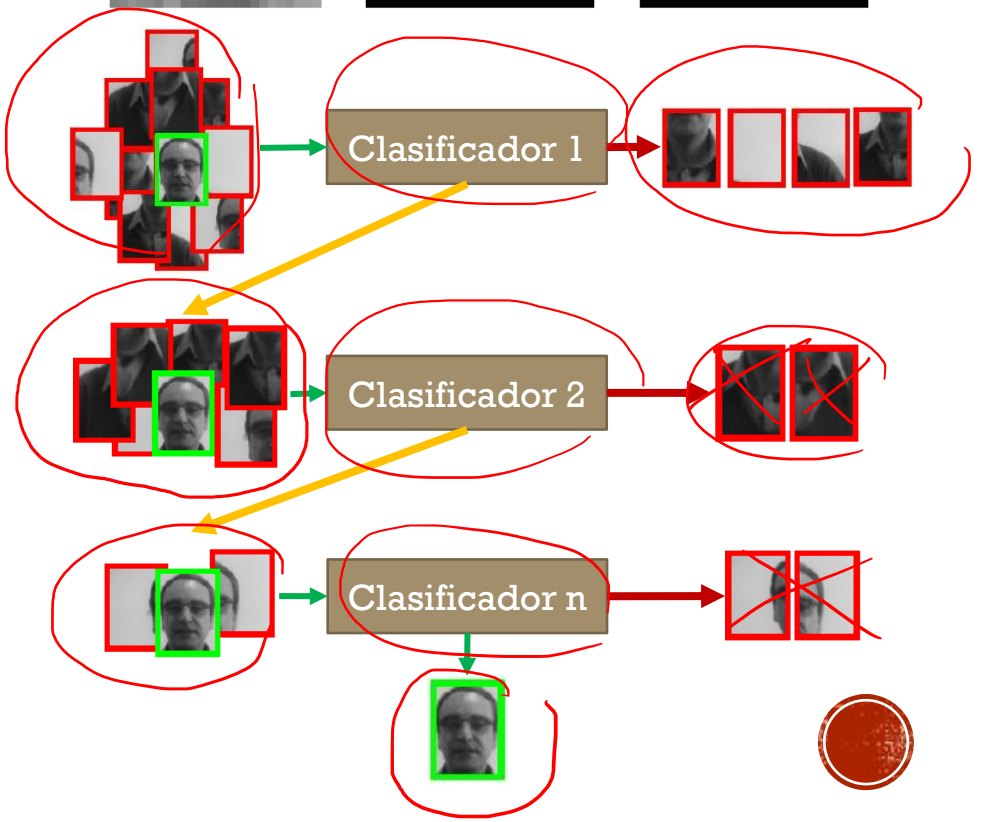


- Detector en cascada
  1. La mayoría de las ventanas en una búsqueda de rostros (u objetos) no contiene rostros!
  2. Detectar más rápido si NO hay rostros que si hay
  3. Combinación secuencial de clasificadores
  4. Una imagen es solo reconocida como rostro si todos los clasificadores la aceptan

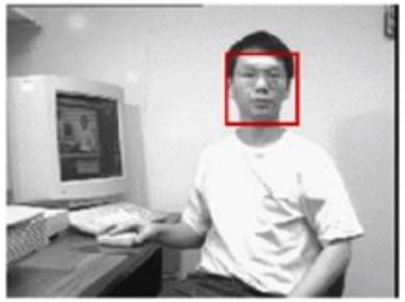
- **Ventajas**
  1. Integra la clasificación con la selección de características
  2. Flexibilidad en la selección de los clasificadores débiles (esquema de boosting)
  3. La complejidad es lineal en el número de ejemplos de entrenamiento
  4. La etapa de testeo es realmente rápida
  5. Es fácil de implementar

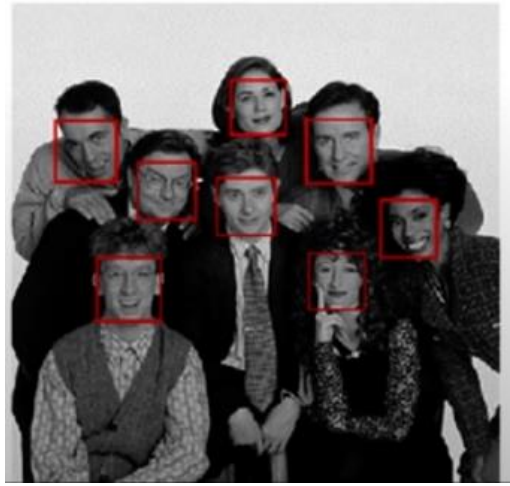
- **Desventajas**
  1. Necesita muchos datos de entrenamiento
  2. Principalmente en casos con muchas clases no anda tan bien como otros métodos (por ejemplo SVM)

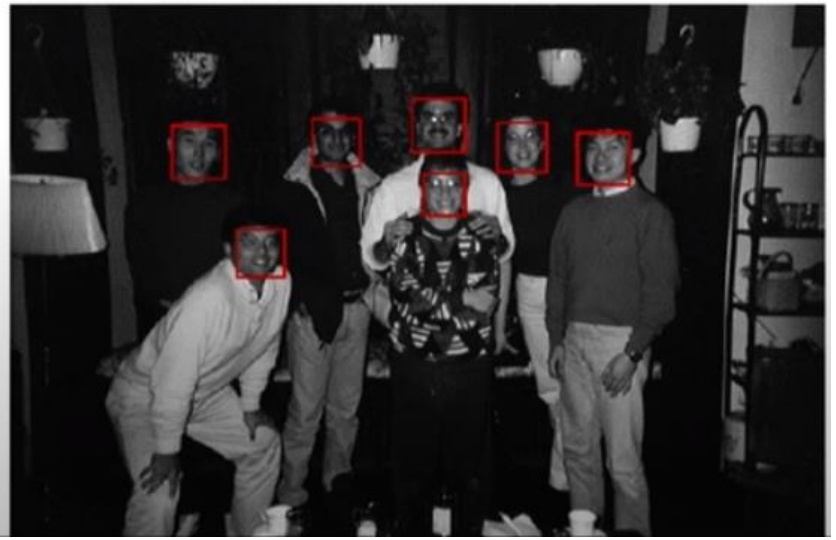




ADABOOST







SVM (SUPPORT VECTOR MACHINES)

- Clasificadores lineales
  - Podemos escribir la recta en coordenadas homogéneas como

$$px + qy + b = 0$$

$$\bar{w} = \begin{bmatrix} p \\ q \end{bmatrix} \bar{x} = \begin{bmatrix} x \\ y \end{bmatrix} \Rightarrow \bar{w}^t\bar{x} + b = 0$$

  - La distancia de punto (x_0, y_0) a esta recta será (en este caso $\bar{w}$ puede no ser unitario)

$$D = \frac{|px_0 + qy_0 + b|}{\sqrt{p^2 + q^2}} = \frac{|\bar{w}^t\bar{x} + b|}{\|\bar{w}\|}$$

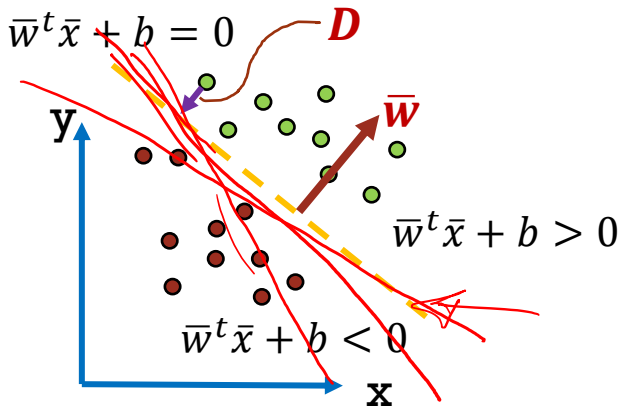
  - ¿Ahora…cuál es la mejor recta que separa las poblaciones?
    - Línea de separación óptima (en el caso 2D)
    - Las líneas punteadas son la distancia más corta a puntos de las poblaciones
    - La idea es maximizar el margen entre los casos positivos y negativos
    - Puntos especiales → puntos o "vectores" de soporte (modelo discriminativo)
    - Si cambian las muestras del vector de soporte cambia la solución del SVM
    - Si cambia cualquier otra muestra no cambia la solución del SVM
  - La condición de clasificación podemos escribirla como

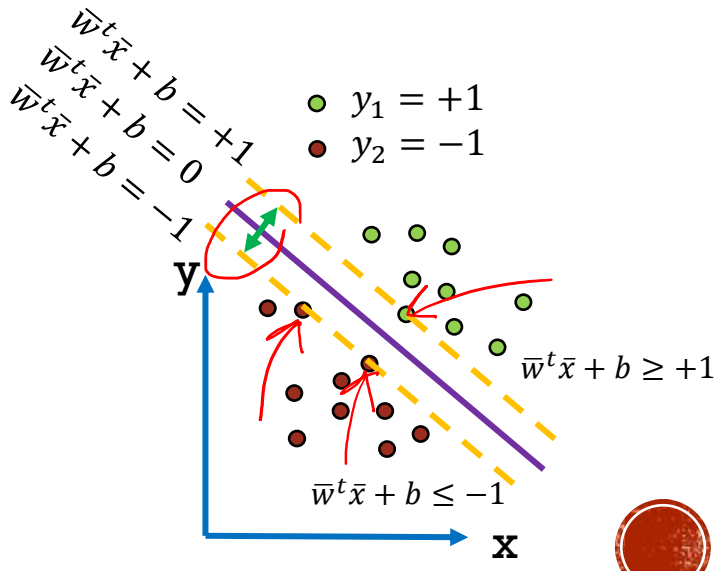
$$y_i(\bar{w}^t\bar{x} + b) \geq 1$$

  - ¿Cómo medimos el margen?

$$D_{(+1,-1)} = \frac{|\bar{w}^t\bar{x} + b|}{\|\bar{w}\|} = \frac{|\pm 1|}{\|\bar{w}\|}$$

$$M = \mathrm{D}_{+1} + D_{-1} = \frac{2}{\|\bar{w}\|}$$





SVM (SUPPORT VECTOR MACHINES)

- Entonces la idea es maximizar el margen…que sería lo mismo que minimizar

$$\frac{1}{2}\bar{w}^t\bar{w}\,, restringido\ a{:}\ y_i\,(\bar{w}^t\bar{x}+b)\geq 1$$

- Y este es un problema de optimización cuadrática estándar!. Ya sabemos plantear este problema a través del Lagrangiano.

$$\mathcal{L}(\bar{w},b,\lambda_i)=\frac{1}{2}\bar{w}^t\bar{w}-\sum_i\lambda_i[y_i(\bar{w}^t\bar{x_i}+b)-1]$$

- La minimización del Lagrangiano (SVM Primal) respecto de $\bar{w}$ y b implica

$$\frac{\partial\mathcal{L}}{\partial\bar{w}}=\bar{w}-\sum_i\lambda_i\,y_i\bar{x_i}=0\Rightarrow\bar{w}=\sum_i\lambda_i\,y_i\bar{x_i}$$

$$\frac{\partial\mathcal{L}}{\partial b}=-\sum_i\lambda_i\,y_i=0\qquad\Rightarrow\sum_i\lambda_i\,y_i=0\,,\lambda_i\geq 0$$

- Reemplazando estas soluciones nuevamente en el Lagrangiano y operando podemos obtener una ecuación para la resolución de optimización (maximización) del SVM Dual

$$\Theta(\lambda)=-\frac{1}{2}\sum_i\sum_j\lambda_i\,\lambda_j\,y_iy_j\bar{x_i}^t\bar{x_j}+\sum_i\lambda_i$$

- A partir de la solución para $\bar{w}$ y sabiendo que $b=y_i-\bar{w}^t\bar{x_i}$ para cada vector de soporte podemos plantear una función de clasificación

$$f(\bar{x})=signo(\bar{w}^t\bar{x}+b)$$

$$f(\bar{x})=signo\left(\sum_i\lambda_i\,y_i\bar{x}^t\bar{x_i}+b\right)$$

Nueva muestra
vectores soporte

SVM (SUPPORT VECTOR MACHINES)

- ¿Qué pasa si no estamos en un caso linealmente separable?
  - Mapeamos los datos a una dimensión superior

$$\Phi: x \to \varphi(x)$$

  - Entonces el producto escalar se transforma en

$$K(\bar{x}_i, \bar{x}_j) = \varphi(\bar{x}_i)^T \varphi(\bar{x}_j)$$

  - Este K es el "kernel". Una función kernel es una función de similaridad que corresponde a un producto interno en un espacio expandido de características
  - Ejemplo: $\bar{x} = (x_1, x_2), K(\bar{x}_i, \bar{x}_j) = (1 + \bar{x}_i^{\mathrm{t}} \bar{x}_j)^2$

$$K(\bar{x}_i, \bar{x}_j) = 1 + x_{i1}^2 x_{j1}^2 + 2x_{i1}x_{j1}x_{i2}x_{j2} + x_{i2}^2 x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2}$$

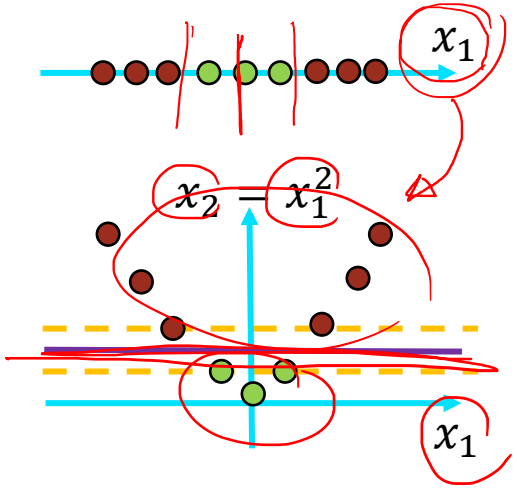
$$= \begin{bmatrix} 1 & x_{i1}^2 & \sqrt{2}x_{i1}x_{i2} & x_{i2}^2 & \sqrt{2}x_{i1} & \sqrt{2}x_{i2} \end{bmatrix}^t \begin{bmatrix} 1 & x_{j1}^2 & \sqrt{2}x_{j1}x_{j2} & x_{j2}^2 & \sqrt{2}x_{j1} & \sqrt{2}x_{j2} \end{bmatrix}$$

$$= \varphi(\bar{x}_i)^T \varphi(\bar{x}_j)$$

$$\varphi(\bar{x}) = \begin{bmatrix} 1 & x_1^2 & \sqrt{2}x_1x_2 & x_2^2 & \sqrt{2}x_1 & \sqrt{2}x_2 \end{bmatrix} \to cumple$$

  - Entonces en lugar de calcular explícitamente la transformación $\varphi(\bar{x})$, definimos la función kernel K y la incluimos en la función de clasificación para tener **"una decisión de borde no lineal en el espacio original de características"**.

$$f(\bar{x}) = signo\left(\sum_i \lambda_i y_i \bar{x}^t \bar{x}_i + b\right) \to signo\left(\sum_i \lambda_i y_i K(\bar{x}, \bar{x}_i) + b\right)$$



Ejemplos de Kernel

Polinomial
$K(\bar{x}, \bar{z}) = \langle \bar{x}, \bar{z} \rangle^d$

Func. Base radial (Gaussian)
$K(\bar{x}, \bar{z}) = e^{-\|\bar{x} - \bar{z}\|^2 / 2\sigma}$

Sigmoide
$K(\bar{x}, \bar{z}) = \tanh(k\langle \bar{x}, \bar{z} \rangle - \delta)$

Multi-cuadrático inverso
$K(\bar{x}, \bar{z}) = \left(\|\bar{x} - \bar{z}\|^{1/2} 2\sigma + c^2\right)^{-1}$

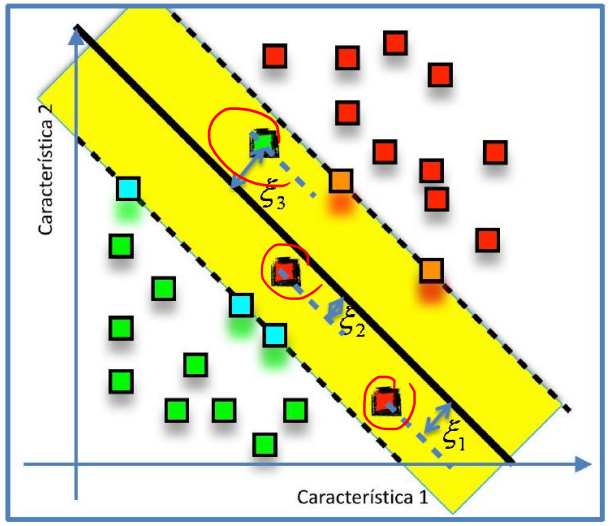
Kernel de intersección
$K(\bar{x}, \bar{z}) = \sum_{i=1}^{n} \min(x_i, z_i)$

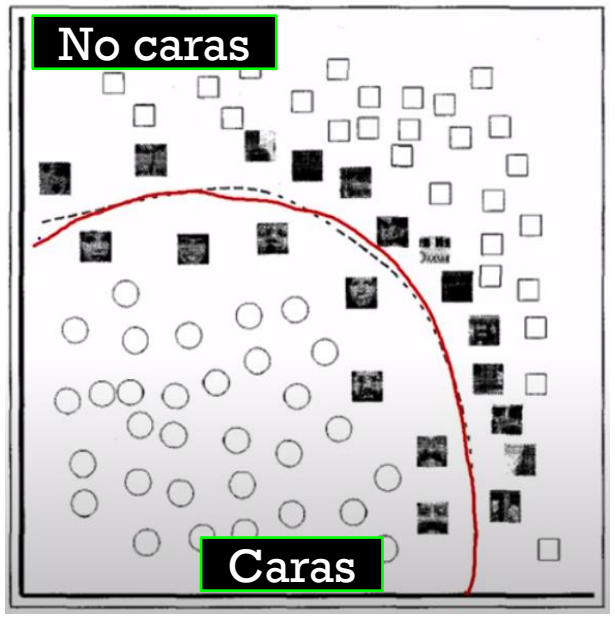
SVM (SUPPORT VECTOR MACHINES)

- ¿Qué pasa si aún así no son linealmente separables?
  - Se puede permitir un cierto "margen suave" que permita pequeños errores de clasificación en la frontera
  - Se introduce una variable de holgura $\xi_i \geq 0$ y pedimos
    1. Minimizar $\Phi(\bar{w}) = \frac{1}{2}\bar{w}^t\bar{w} + C\sum_i \xi_i$ sujeto a: $y_i\,(\bar{w}^t\varphi(\bar{x_i}) + b) \geq 1 - \xi_i$
    2. Maximizar $\Theta(\lambda) = -\frac{1}{2}\sum_i\sum_j \lambda_i\,\lambda_j\,y_i y_j K(\bar{x},\bar{x_i}) + \sum_i \lambda_i$ sujeto a:

$$\begin{cases} 0 \leq \lambda_i \leq C \quad b = y_i(1-\xi_i) - \sum_j y_j\lambda_i K(\bar{x_j},\bar{x_i}) \\ \sum_i \lambda_i\,y_i = 0 \quad f(\bar{x}) = signo\left(\sum_i \lambda_i\,y_i K(\bar{x},\bar{x_i}) + b\right) \end{cases}$$

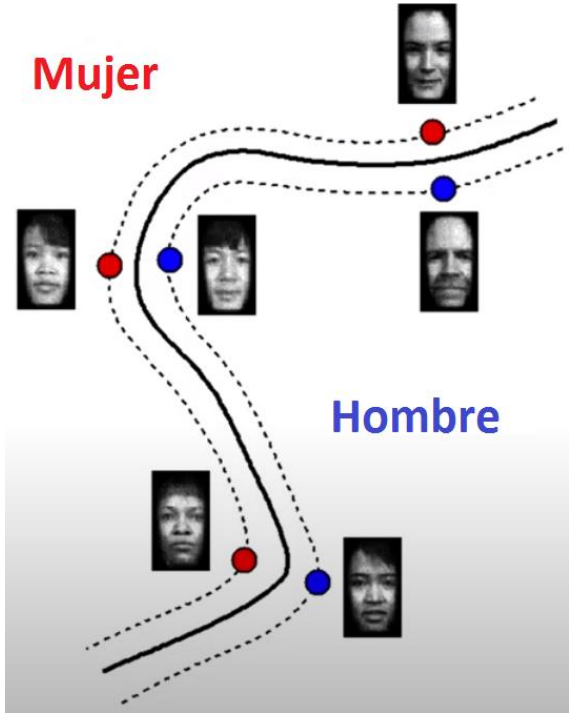
- Entonces, ¿cuál es el procedimiento?
  - En etapa de entrenamiento
    1. Definir la representación (espacio de características)
    2. Elegir la función de kernel
    3. Calcular los pares del kernel sobre las muestras etiquetadas
    4. Usar esa "matriz kernel" para resolver para los vectores de soporte y los pesos.
  - En etapa de testeo
    1. Calcular los valores del kernel entre la nueva muestra y los vectores de soporte
    2. Aplicar los pesos
    3. Verificar la salida (signo)





SVM (SUPPORT VECTOR MACHINES)

- ¿Y si tenemos muchas clases distintas?→ Multi-Class SVMs
  1. Uno Vs. Todos
     - **Entrenamiento:** Aprender un SVM para cada clase Vs. el resto
     - **Testeo:** Aplicar cada SVM a la muestra ejemplo y asignarla a la clase del SVM que devuelva el valor de decisión más alto
  2. Uno Vs. Uno
     - **Entrenamiento:** Aprender un SVM para cada par de clases
     - **Testeo:** Cada SVM "vota" por la clase a ser asignada a la muestra ejemplo



- **Ventajas**
  - Hay muchas librerías públicas disponibles
    - http://www.kernel-machines.org/software
    - https://www.csie.ntu.edu.tw/~cjlin/libsvm/
    - OpenCV
  - La estructura basada en kernels es muy potente y flexible
  - Suele devolver un pequeño número de vectores soporte (es rápido en testeo)
  - Funciona muy bien en la práctica (inclusive mejor que boosting)
- **Desventajas**
  - No hay manera (nativa) de entrenar un SVM multi-clase (se deben combinar)
  - Puede ser demandante elegir el kernel correcto para un problema
  - Difícil de programar (se suele usar a través de librerías). Se pierde intuición sobre los resultados
  - Tiempo de cómputo y memoria
    - Durante el entrenamiento se debe calcular la matriz de kernel para cada par de muestras
    - El aprendizaje puede tomar mucho tiempo para problemas a gran escala

TP6

- Utilizando la webcam
  1. Implementar un algoritmo utilizando filtros de Haar en cascada que:
     a) Detecte Rostros
     b) Dentro de los rostros detecte ojos
     c) Dentro de los rostros detecte sonrisas