

# Detección de rostros con filtros de Haar.

Jorge Luis Vargas Barrera  
CIMAT. Unidad Monterrey.

June 10, 2021

# Contenido

Introducción.

Filtros de Haar.

Integral Image.

AdaBoost.

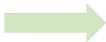
Clasificador en cascada.

# Introducción.

1. Dada una imagen queremos determinar si hay rostros presentes y en donde.
2. Aplicaciones en: Vigilancia, vídeo conferencia, identificación biométrica.



detección



reconocimiento



Paul



Michael



Viola



John

# Introducción

- \* Un detector de rostros tiene que identificar si una imagen de tamaño arbitrario contiene un rostro humano o no, y donde se encuentra.
- \* Un marco natural para considerar este problema es el de la clasificación binaria, en donde se construye un clasificador para reducir el error de una clasificación incorrecta.

# Filtros de Haar.



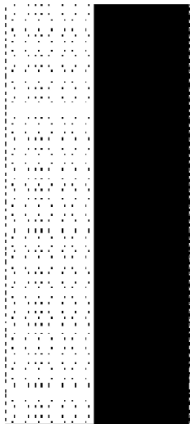
$$\sum_{1 \leq i \leq N} \sum_{1 \leq j \leq N} I(i,j) 1_{P(i,j) \text{ es blanco}} - \sum_{1 \leq i \leq N} \sum_{1 \leq j \leq N} I(i,j) 1_{P(i,j) \text{ es negro}}$$

# Filtros de Haar.



# Filtros de Haar

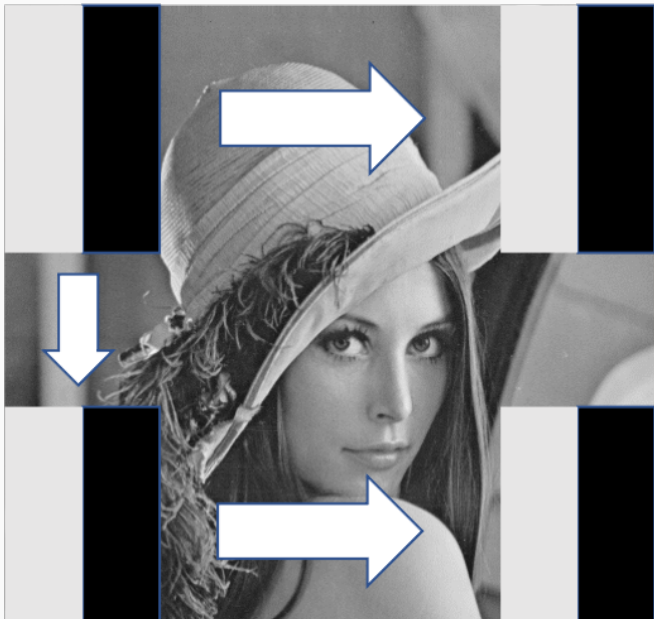
En este caso  
nuestra característica  
es:  $(0.1 + 0.1 + 0.2 + 0.2 + 0.2 + 0.1 + 0.2 + 0.2 + 0.2 + 0.3 + 0.2 + 0.1 + 0.2 + 0.3 + 0.2 + 0.2) - (0.5 + 0.4 + 0.5 + 0.6 + 0.4 + 0.7 + 0.5 + 0.4 + 0.4 + 0.5 + 0.6 + 0.8 + 0.5 + 0.7 + 0.6 + 0.6)$   
 $W - B = 3 - 8.7$   
 $= -5.7$



0.1	0.2	0.5	0.4
0.1	0.3	0.4	0.5
0.2	0.2	0.5	0.6
0.2	0.1	0.6	0.8
0.2	0.2	0.4	0.5
0.1	0.3	0.7	0.7
0.2	0.2	0.5	0.6
0.2	0.2	0.4	0.6



# Filtros de Haar.



# Problemas de con filtros de Haar.


1. Una ventana de  $24 \times 24$  tiene 16000+ características.
2. El costo computacional de calcular las características se vuelve grande.

# Imagen Integral.

1. Permite obtener de forma rápida y eficiente el cálculo de la suma de píxeles en una parte rectangular de la imagen.
2. Se puede calcular en tiempo lineal respecto a la dimensión de la imagen.

$$II(i,j) := \begin{cases} \sum_{1 \leq s \leq i} \sum_{1 \leq t \leq j} I(s,t) & , \quad 1 \leq i \leq N \text{ y } 1 \leq j \leq N \\ 0 & , \quad \text{de otra forma} \end{cases}$$

# Imagen Integral



1	3	7	5
12	4	8	2
0	14	16	9
5	11	6	10

1	4	11	16
13	20	35	42
13	34	65	81
18	50	87	113

# Imagen Integral

Set  $II(1,1)=I(1,1);$

**for**  $i=1$  to  $N$  **do**

**for**  $j=1$  to  $M$  **do**

$II(i,j)=I(i,j)+II(i,j-1)+II(i-1,j)-II(i-1,j-1);$

        II es definido como cero cuando  $(i,j)$   
        están fuera de I;

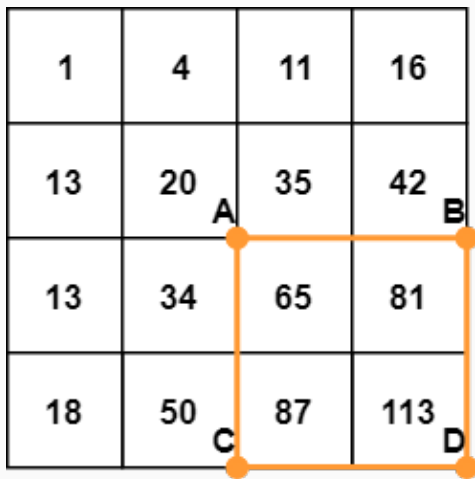
**end**

**end**

**Algorithm 1:** Image Integral Recursiva

# Calculo en rectángulos.

1	4	11	16
13	20	35	42
13	34	65	81
18	50	87	113



A 4x4 grid of numbers. An orange rectangle is drawn around the intersection of the second and third columns and the second and third rows. The vertices of the rectangle are labeled A (top-left), B (top-right), C (bottom-left), and D (bottom-right). The numbers inside the rectangle are 20, 35, 34, 65, 50, 87, and 81.

**ABCD = ?**

# Calculo en rectángulos.

1	4	11	16
13	20	35	42
13	34	65	81
18	50	87	113

D = 113

1	4	11	16
13	20	35	42
13	34	65	81
18	50	87	113

C = 50

1	4	11	16
13	20	35	42
13	34	65	81
18	50	87	113

B = 42

1	4	11	16
13	20	35	42
13	34	65	81
18	50	87	113

A = 20

$$\boxed{113} - \boxed{50} - \boxed{42} + \boxed{20} = 41$$

# Podemos aplicar lo que sabemos.

- \* Estamos en terreno conocido.
- \* Dado un conjunto de entrenamiento de imágenes positivas y negativas, podríamos usar cualquiera de los métodos de clasificación que se vieron en el curso.



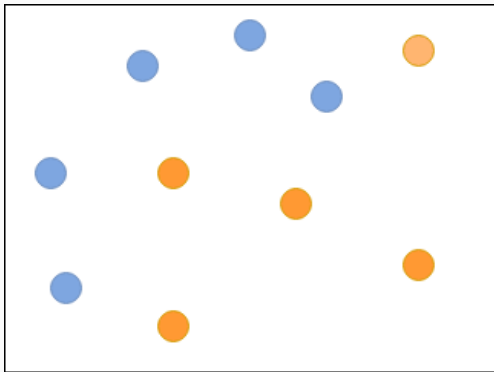
# Más Problemas

Pero... Calcular todas las características de Haar resulta computacionalmente caro aún cuando se tiene un método eficiente para calcularlas.

# AdaBoost

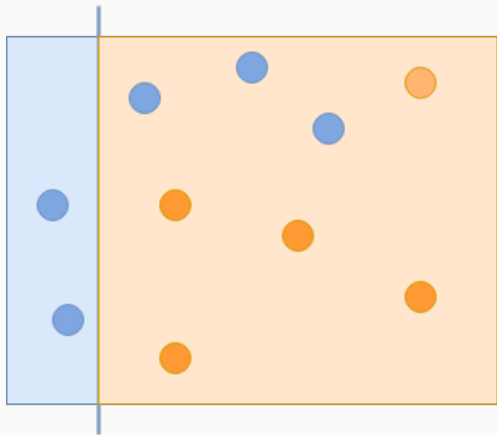
\* Usaremos AdaBoost para seleccionar un conjunto pequeño de características y entrenar el clasificador.

# AdaBoost



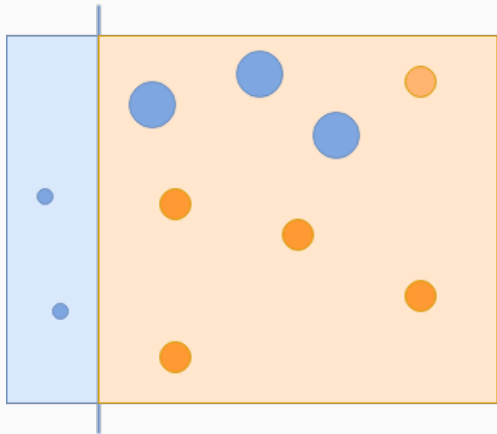
**Figure:** Queremos clasificar los círculos azules y naranjas usando clasificadores débiles.

# AdaBoost



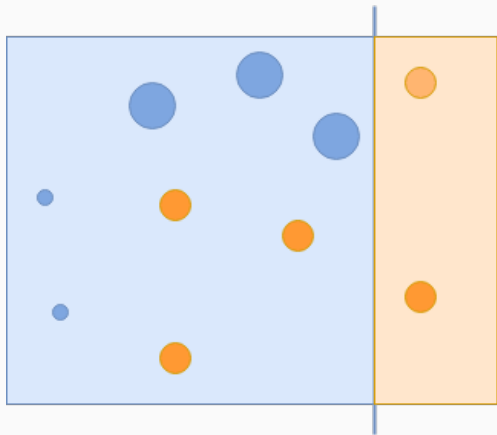
**Figure:** El primer clasificador captura algunos de los círculos azules. En la siguiente iteración se le dará más peso a los ejemplos donde fallo.

# AdaBoost



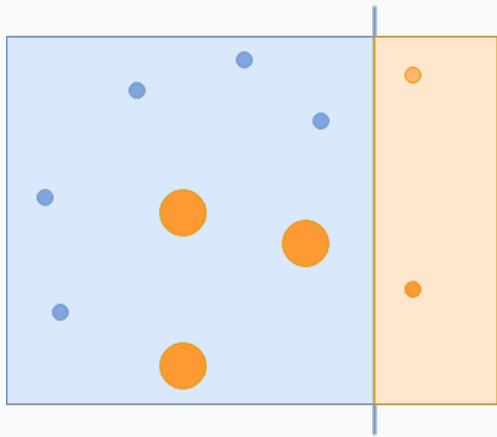
**Figure:** Se le da más importancia a los mal clasificados.

# AdaBoost



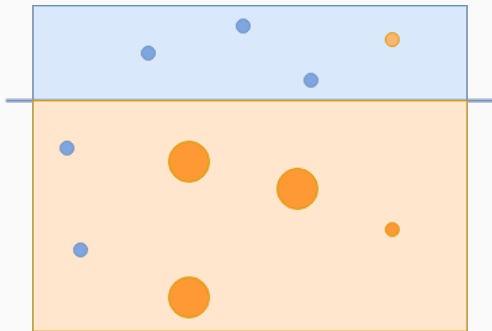
**Figure:** EL segundo clasificador logra capturar todos los círculos azules, pero incorrectamente captura naranjas.

# AdaBoost



**Figure:** Se le da mayor importancia a círculos mal clasificados.

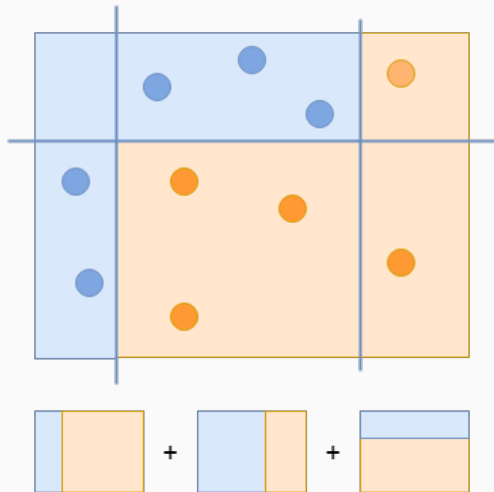
# AdaBoost



**Figure:** El tercer clasificador captura lo círculos naranjas faltantes.



# AdaBoost



**Figure:** Para formar el clasificador fuerte combinamos los clasificadores débiles

# AdaBoost

Dadas ejemplos de imágenes  $(x_1, y_1), \dots, (x_n, y_n)$  donde  $y_i = 0, 1$  para ejemplos negativos y positivos respectivamente;

Inicializa los pesos  $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$  para  $y_i = 0, 1$  respectivamente, donde  $m$  y  $l$  son el número de negativos y positivos respectivamente;

**for**  $t = 1, \dots, T$  **do**

Normaliza los pesos,

$$w_{t,i} = \frac{w_{t,i}}{\sum_{j=1}^n w_{t,j}}$$

;

Para cada característica,  $j$ , entrena un clasificador  $h_j$  que se restringe a una sola característica. El error con respecto a  $w_t, \epsilon_t = \sum_i w_i |h_j(x_i) - y_i|$ ;

Escoge el clasificador  $t$  con el menor error  $\epsilon_t$ ;

Actualiza los pesos:

$$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$$

donde  $e_i = 0$  si el ejemplo  $x_i$  se clasifica correctamente,  $e_i = 1$  en otro caso, y  $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$ ;

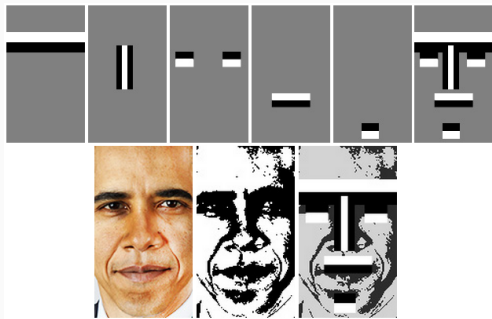
El clasificador final es:

$$h(x) = \begin{cases} 1 & \sum_{t=1}^T \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^T \alpha_t \\ 0 & \text{en otro caso} \end{cases}$$

donde  $\alpha_t = \log \frac{1}{\beta_t}$

**end**

# AdaBoost



**Figure:** Ejemplo de características importantes que se escogen con AdaBoost.

## Más Problemas.

- \* Muchas de las ventanas que se revisaran no contienen rostros. Pero aún así se procesan completamente. Es decir, se revisan miles de características.
- \* Queremos descartar rápidamente las ventanas que no contienen rostros y dedicar mas tiempo en zonas promisorias.

# Clasificador en cascada.

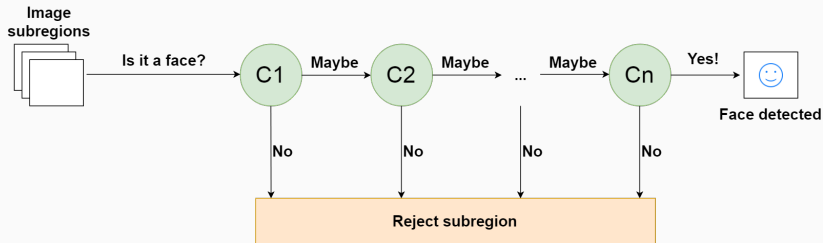
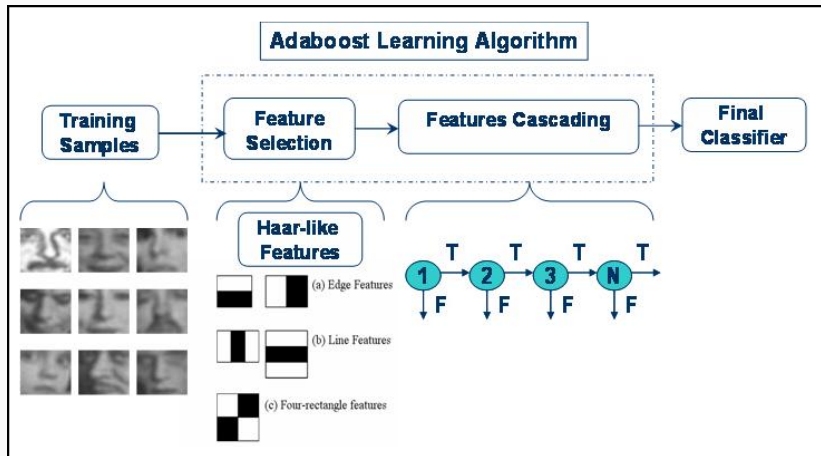


Figure: Clasificador en cascada.

# Resumen.



DEMO

# Referencias

- [1] Paul Viola y Michael Jones. Rapid object detection using a boosted cascade of simple features. 2001.
- [2] Yi-Qing Wang. "An Analysis of the Viola-Jones Face Detection Algorithm". En: Image Processing On Line 4 (2014). <https://doi.org/10.5201/ipol.2014.104>, págs. 128-148.