



UNIVERSITAT
POLITÈCNICA
DE VALÈNCIA

Algoritmo Perceptrón: aplicación a tareas de clasificación

DSIC

Departamento de Sistemas
Informáticos y Computación

Objetivos formativos

- Implementar clasificadores lineales
- Programar el algoritmo Perceptrón
- Aplicar el algoritmo Perceptrón a tareas de clasificación

Índice

1	Funciones discriminantes lineales	3
2	Algoritmo Perceptrón	5
3	Aplicación a tareas de clasificación: OCR	7
3.1	Entrenamiento	8
3.2	Estimación del error	10
3.3	Efecto de α	11
3.4	Efecto de b	12
3.5	Entrenamiento del clasificador final	13
4	Ejercicio: aplicación a otras tareas	14

1. Funciones discriminantes lineales

Todo clasificador puede representarse como:

$$c(x) = \arg \max_c g_c(x)$$

donde cada clase c utiliza una **función discriminante** $g_c(x)$ que mide el grado de pertenencia de un objeto x a la clase c

Las funciones discriminantes más utilizadas son **lineales** (con x):

$$g_c(\mathbf{x}) = \mathbf{w}_c^t \mathbf{x} + w_{c0} \quad \text{donde} \quad \mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_D \end{pmatrix} \quad \text{y} \quad \mathbf{w}_c = \begin{pmatrix} w_{c1} \\ \vdots \\ w_{cD} \end{pmatrix}$$

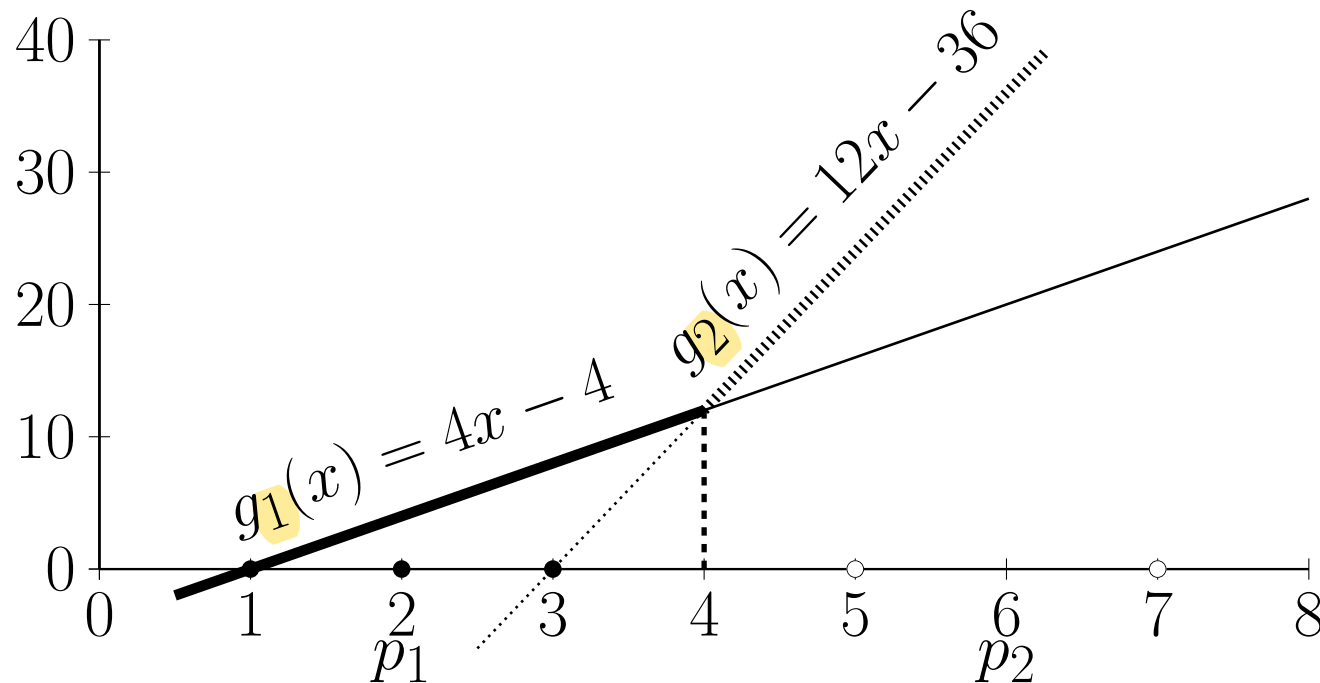
Con notación **homogénea**:

$$g_c(\mathbf{x}) = \mathbf{w}_c^t \mathbf{x} \quad \text{donde} \quad \mathbf{x} = \begin{pmatrix} 1 \\ \mathbf{x} \end{pmatrix} \quad \text{y} \quad \mathbf{w}_c = \begin{pmatrix} w_{c0} \\ \mathbf{w}_c \end{pmatrix}$$

linmach.m

```
function cstar=linmach(w,x)
    C=columns(w); cstar=1; max=-inf;
    for c=1:C
        g=w(:,c)'\*x;
        if (g>max) max=g; cstar=c; endif; end
endfunction
```

```
w=[-4 -36; 4 12];
for x=1:8;
    printf("c(%d)=%d\n",x,linmach(w,[1 x]')) ; end
```



c(1)	=1
c(2)	=1
c(3)	=1
c(4)	=1
c(5)	=2
c(6)	=2
c(7)	=2
c(8)	=2

2. Algoritmo Perceptrón

determinan el comportamiento del algoritmo

Entrada: $\{(\mathbf{x}_n, c_n)\}_{n=1}^N$, $\{\mathbf{w}_c\}_{c=1}^C$, $\alpha \in \mathbb{R}^{>0}$ y $b \in \mathbb{R}$

Salida: $\{\mathbf{w}_c\}^* = \arg \min_{\{\mathbf{w}_c\}} \sum_n \left[\max_{c \neq c_n} \mathbf{w}_c^t \mathbf{x}_n + b \stackrel{[P]}{>} \mathbf{w}_{c_n}^t \mathbf{x}_n \right]$

Método:

$$[P] = \begin{cases} 1 & \text{si } P = \text{verdadero} \\ 0 & \text{si } P = \text{falso} \end{cases}$$

repetir

para todo dato \mathbf{x}_n

$err = \text{falso}$

para toda clase c distinta de c_n

si $\mathbf{w}_c^t \mathbf{x}_n + b > \mathbf{w}_{c_n}^t \mathbf{x}_n$: $\mathbf{w}_c = \mathbf{w}_c - \alpha \cdot \mathbf{x}_n$; $err = \text{verdadero}$

si err : $\mathbf{w}_{c_n} = \mathbf{w}_{c_n} + \alpha \cdot \mathbf{x}_n$

hasta que no queden muestras mal clasificadas
(o se llegue a un máximo de iteraciones prefijado)

perceptron.m

```
function [w,E,k]=perceptron(data,b,a,K,iw)
[N,L]=size(data); D=L-1;
labs=unique(data(:,L)); C=numel(labs);
if (nargin<5) w=zeros(D+1,C); else w=iw; end
if (nargin<4) K=200; end;
if (nargin<3) a=1.0; end;
if (nargin<2) b=0.1; end;
for k=1:K
    E=0;
    for n=1:N
        xn=[1 data(n,1:D)]';
        cn=find(labs==data(n,L));
        er=0; g=w(:,cn)'\*xn;
        for c=1:C; if (c!=cn && w(:,c)'\*xn+b>g)
            w(:,c)=w(:,c)-a*xn; er=1; end; end
        if (er)
            w(:,cn)=w(:,cn)+a*xn; E=E+1; end; end
        if (E==0) break; end; end
endfunction
```

```
data=[0 0 1;1 1 2];
[w,E,k]=perceptron(data);
disp(w); printf("E=%d k=%d\n",E,k);
```

```
1    -1
-1    1
-1    1
E=0 k=3
```

3. Aplicación a tareas de clasificación: OCR

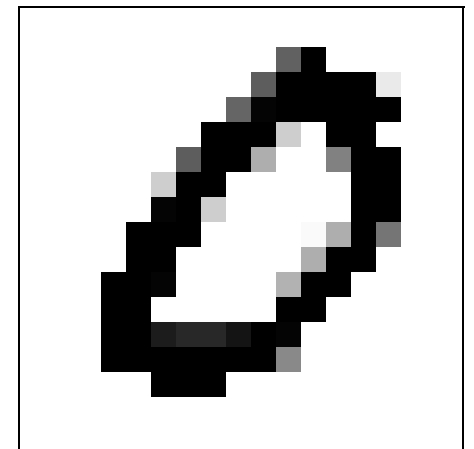
El corpus `OCR_14x14` es una matriz `data` de 1000 filas (muestras) y 197 columnas (196 características y etiqueta de clase):

```
head -n 7 OCR_14x14
```

```
# name: data
# type: matrix
# rows: 1000
# columns: 197
0 0 0 0 0 0 0 0 0.62 1 0 0 0 0 0 0 0 0 0 0.63 1 1 1 1 ... 0 0
0 0 0 0 0 0 0 0.38 1 1 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 ... 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0.91 1 0.99 ... 0 0
```

Cada muestra corresponde a una imagen de dígito manuscrito normalizada a 14x14 grises y leída en el orden de lectura usual:

```
load("OCR_14x14");
[N,L]=size(data); D=L-1;
I=reshape(data(1,1:D),14,14)';
imshow(1-I);
rand("seed",23); data=data(randperm(N),:);
for n=1:1000
    I=reshape(data(n,1:196),14,14)';
    imshow(1-I); pause(0.5);
end
```



3.1. Entrenamiento

```
load("OCR_14x14"); [N,L]=size(data); D=L-1;
ll=unique(data(:,L)); C=numel(ll);
rand("seed",23); data=data(randperm(N),:);
[w,E,k]=perceptron(data(1:round(.7*N),:));
save_precision(4); save("percep_w","w");
output_precision(2); w
```

```
w =
-39.00 -30.00 -31.00 -35.00 -34.00 -27.00 -33.00 -30.00 -46.00 -31.00
  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00  0.00
 -1.00  0.00 -2.00  0.00 -1.00  2.00  0.00  0.00 -2.00  0.00
 -1.38 -1.69 -2.53 -1.84 -1.53  0.69 -0.69  4.15 -3.22 -1.69
 -1.69 -1.77  0.54 -3.46 -1.46 -3.00 -2.00  5.15 -3.46 -3.00
 -3.54 -7.48 -1.15 -3.00  0.25 -6.71 -5.08  1.77 -1.85 -8.41
...
```

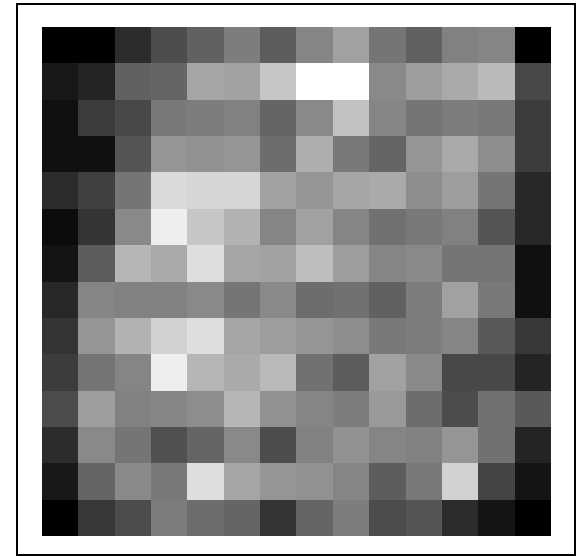
La pseudo-probabilidad de que \mathbf{x} (con $x_0 = 1$) sea del dígito c es $g_c(\mathbf{x}) = \mathbf{w}_c^t \mathbf{x}$, donde \mathbf{w}_c viene dado por la columna $c + 1$ de \mathbf{w} :

```
load("OCR_14x14"); load("percep_w"); [N,L]=size(data); D=L-1;
for n=1:N; xn=[1 data(n,1:D)]';
    for c=0:9 printf("g_%d(x_%d)=%.0f ",c,n,w(:,c+1)'*xn); end
    printf("\n"); end
```

```
g_0(x_1)=-665 g_1(x_1)=-840 g_2(x_1)=-813 g_3(x_1)=-798 ...
g_0(x_2)=-518 g_1(x_2)=-659 g_2(x_2)=-592 g_3(x_2)=-604 ...
g_0(x_3)=-635 g_1(x_3)=-802 g_2(x_3)=-826 g_3(x_3)=-740 ...
...
```

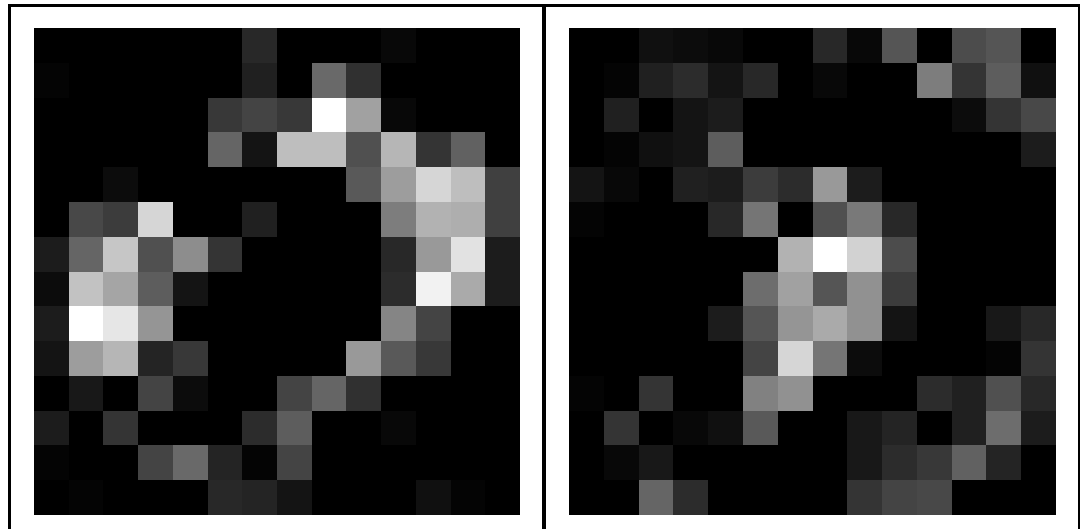
Los pesos $\{w_{cd}\}$ de mayor variabilidad en c tienen mayor efecto discriminativo que los pesos que varían poco. Derecha: $\sigma(\{w_{1d}, \dots, w_{Cd}\})$ para cada $d > 0$.

```
load("percep_w"); sw=std(w(2:197,:),1,2);  
I=reshape(sw,14,14)'; imshow(I,[,]);
```



Las pesos de una clase c comparativamente mayores que los del resto de clases (mayores que la media) indican qué características (no neg.; p.e. grises) son “pro- c ”; los menores son “anti- c ”.

```
load("percep_w");  
mw=mean(w(2:197,:),2);  
for c=0:9  
    wc=w(2:197,c+1);  
    pw=max(0,wc-mw);  
    I=reshape(pw,14,14)';  
    imshow(I,[,]); pause(3);  
    nw=-min(0,wc-mw);  
    I=reshape(nw,14,14)';  
    imshow(I,[,]); pause(3);  
end
```



3.2. Estimación del error

Estimación del error de clasificación mediante las **muestras no empleadas en entrenamiento** (*muestras de test*):

```
load("OCR_14x14");  
[N,L]=size(data); D=L-1;  
ll=unique(data(:,L));  
C=numel(ll); rand("seed",23);  
data=data(randperm(N),:);  
M=N-round(.7*N); te=data(N-M+1:N,:);  
load("percep_w"); rl=zeros(M,1);  
for m=1:M  
    tem=[1 te(m,1:D)]';  
    rl(m)=ll(linmach(w,tem)); end  
[nerr m]=confus(te(:,L),rl)
```

```
nerr = 17  
m =  
37  0  0  0  0  0  0  0  0  0  
0 29  0  0  0  0  0  0  1  0  
0  1 32  0  0  0  0  0  0  0  
1  0  1 26  0  2  0  0  0  0  
0  0  0  0 27  0  0  1  0  0  
0  0  0  2  0 26  0  0  0  0  
0  0  0  0  0  0 28  0  0  0  
0  0  0  0  0  0  0 27  0  2  
1  2  0  0  0  1  0  0 24  2  
0  0  0  0  0  0  0  0  0 27
```

Intervalo de confianza al 95 % para el error estimado:

```
nerr=17; M=300; output_precision(2);  
m=nerr/M  
s=sqrt(m*(1-m)/M)  
r=1.96*s  
printf("I=[%.3f, %.3f]\n",m-r,m+r);
```

```
m = 0.057  
s = 0.013  
r = 0.026  
I=[0.031, 0.083]
```

3.3. Efecto de α

```
#!/usr/bin/octave -qf
load("OCR_14x14"); [N,L]=size(data); D=L-1;
ll=unique(data(:,L)); C=numel(ll);
rand("seed",23); data=data(randperm(N),:);
NTr=round(.7*N); M=N-NTr; te=data(NTr+1:N,:);
printf("#          a      E      k Ete\n");
printf("#----- --- --- ---\n");
for a=[.1 1 10 100 1000 10000 100000]
    [w,E,k]=perceptron(data(1:NTr,:),0.1,a); rl=zeros(M,1);
    for n=1:M rl(n)=ll(linmach(w,[1 te(n,1:D)]'))); end
    [nerr m]=confus(te(:,L),rl);
    printf("%8.1f %3d %3d %3d\n",a,E,k,nerr);
end
```

#	a	E	k	Ete
#-----	---	---	---	---
	0.1	0	16	20
	1.0	0	13	17
	10.0	0	8	15
	100.0	0	12	16
	1000.0	0	12	16
	10000.0	0	12	16
	100000.0	0	12	16

El parámetro α , $\alpha > 0$, **no** tiene gran efecto sobre el comportamiento de Perceptrón.

3.4. Efecto de b

```
#!/usr/bin/octave -qf
load("OCR_14x14"); [N,L]=size(data); D=L-1;
ll=unique(data(:,L)); C=numel(ll);
rand("seed",23); data=data(randperm(N),:);
NTr=round(.7*N); M=N-NTr; te=data(NTr+1:N,:);
printf("#          b      E      k Ete\n");
printf("#----- --- --- ---\n");
for b=[.1 1 10 100 1000 10000 100000]
    [w,E,k]=perceptron(data(1:NTr,:),b); rl=zeros(M,1);
    for n=1:M rl(n)=ll(linmach(w,[1 te(n,1:D)]'))); end
    [nerr m]=confus(te(:,L),rl);
    printf("%8.1f %3d %3d %3d\n",b,E,k,nerr);
end
```

#	b	E	k	Ete
#-----	---	---	---	---
	0.1	0	13	17
	1.0	0	16	20
	10.0	0	10	19
	100.0	0	22	16
	1000.0	0	125	13
	10000.0	165	200	10
	100000.0	544	200	29

El parámetro b **sí** tiene gran efecto.

Si las muestras son linealmente separables, escogeremos un b con el que Perceptrón converja ($E = 0$) y sea comparativamente elevado (p.e. $b = 1000$).

3.5. Entrenamiento del clasificador final

Entrenamos nuestro clasificador *final* con todas las muestras:

```
load("OCR_14x14");  
[w,E,k]=perceptron(data,1000); [E k]  
save_precision(4);  
save("OCR_14x14_w","w");    # nomfichero = nomcorpus_w
```

Examinemos los pesos del clasificador final:

```
load("OCR_14x14_w")  
output_precision(2); w
```

```
w =  
-1847.00 -1622.00 -1686.00 -1847.00 -1736.00 -1527.00 -1643.00 -1657.00 -2207.00 -1853.00  
  0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00  
  0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00  
 -9.00   -14.33   -52.08   -22.16   -18.16    48.92    -4.08   -36.67   -49.08   -35.08  
-18.68   -74.45   -63.09   -52.68   -51.95    5.93   -22.55    74.31   -51.42   -48.13  
-35.28  -118.40    17.82   -78.14   -22.17   -76.07   -74.11   165.60   -67.45   -56.44  
-109.60 -189.10   -80.59   -73.37    21.95  -151.10   -91.60    66.40    61.42  -208.20  
-109.80 -246.70  -187.70  -130.10  -193.40  -319.20  -255.80  -185.80  -111.50  -88.27  
-336.50 -361.40  -458.70  -292.30  -415.70  -458.20  -325.30  -506.00  -292.60  -85.41  
-565.50 -346.70  -491.60  -592.10  -678.20  -442.80  -495.90  -789.50  -361.70  -310.00  
-520.70 -477.40  -410.40  -508.90  -668.20  -575.40  -548.10  -460.10  -437.00  -346.20  
-533.90 -472.60  -489.30  -522.00  -437.60  -495.90  -526.70  -504.80  -579.00  -534.40  
-284.10 -120.30  -285.90  -276.30  -139.90  -151.00  -236.40  -163.30  -282.40  -278.80  
-124.00    34.48  -179.10  -246.80   -67.34   148.80  -154.60  -108.30  -117.00   -68.68  
  0.00    -2.00    -4.00    -4.00    2.00    3.00    0.00    0.00    0.00    0.00  
  0.00    0.00    0.00   -24.80    0.00    0.00    0.00   24.80    0.00  -13.64  
 -1.52   -10.15   -1.53   -23.04   11.88   -10.65   -3.70   22.87   -1.46  -11.61  
-55.64   -79.97  -18.85  -146.60  -60.85  -31.97  -77.04   121.10  -112.10  -82.56  
...
```

4. Ejercicio: aplicación a otras tareas

Sean los siguientes 4 conjuntos de datos de sendas tareas:

1. **expressions**: 225 expresiones faciales representadas mediante vectores 4096-D y clasificadas en 5 clases (1=sorpresa, 2=felicidad, 3=tristeza, 4=angustia y 5=disgusto).
2. **gauss2D**: 4000 muestras sintéticas procedentes de dos clases equiprobables de forma Gaussiana bidimensional.
3. **gender**: 2836 expresiones faciales representadas mediante vectores 1280-D y clasificadas por género.
4. **videos**: 7985 vídeos de baloncesto o no-baloncesto representados mediante vectores 2000-D extraídos de histogramas de características locales.

Actividad

1. Elabora un script `experiment.m` en Octave para automatizar la aplicación del algoritmo Perceptron a otras tareas. Este script recibe como entrada los datos, y el rango de valores de α y b :

```
#!/usr/bin/octave -qf
if (nargin!=3)
    printf("Usage: ./experiment.m <data> <alphas> <bes>\n");
    exit(1);
end
arg_list=argv();
data=arg_list{1};
as=str2num(arg_list{2});
bs=str2num(arg_list{3});
load(data); [N,L]=size(data); D=L-1;
...
for a=as
    for b=bs
        [w,E,k]=perceptron(data(1:NTr,:),b,a); rl=zeros(M,1);
        ...
    end
end
```

Desde el intérprete de comandos ejecutaremos:

```
$ ./experiment.m OCR_14x14 "[.1 1 10 100 1000 10000]" "[0.1]"
```


Actividad

Una posible salida de resultados del script sería:

#	a	b	E	k	Ete	Ete (%)	Ite (%)
#	-----	-----	---	---	---	-----	-----
	0.1	0.1	0	16	20	6.7	[3.8, 9.5]
	1.0	0.1	0	13	17	5.7	[3.1, 8.3]
	10.0	0.1	0	8	15	5.0	[2.5, 7.5]
	100.0	0.1	0	12	16	5.3	[2.8, 7.9]
	1000.0	0.1	0	12	16	5.3	[2.8, 7.9]
	10000.0	0.1	0	12	16	5.3	[2.8, 7.9]

2. Obtén una tabla resumen de mejores resultados aproximadamente como la siguiente:

tarea	Ete (%)	Ite (%)
OCR_14x14	4.3	[2.0, 6.6]
expressions	3.0	[0.0, 7.1]
gauss2D	9.0	[7.4, 10.6]
gender	6.1	[4.5, 7.7]
videos	18.7	[17.1, 20.2]

Examen

- El examen de laboratorio consistirá en una modificación de tu script `experiment.m` para la realización de un experimento con un conjunto de datos ya conocido o nuevo.
- El día del examen deberás entregar:
 - Script `experiment.m` original
 - Script `experiment.m` modificado
 - Resultados obtenidos y comentarios sobre los mismos