Dudas prácticas: Algoritmo perceptrón

1. *Duda: ¿De dónde salen las fórmulas para calcular el intervalo de confianza al 95% para el error estimado (transparencia 10)?*

```
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);
```

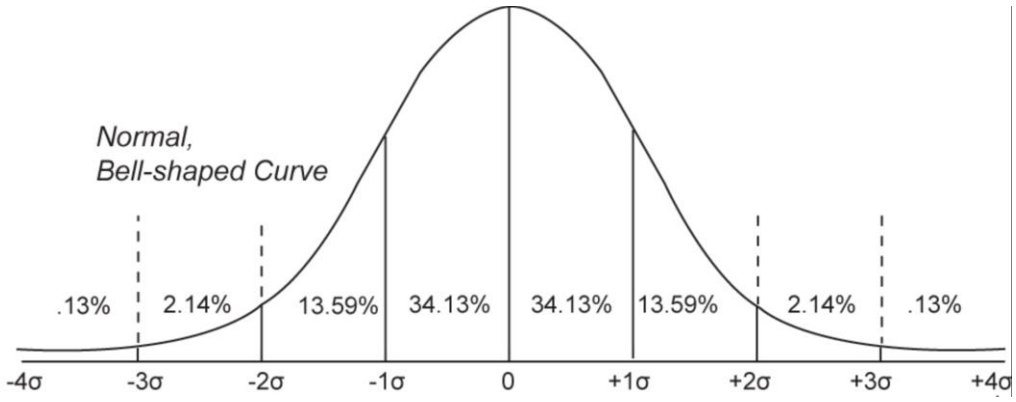
Las fórmulas salen de la distribución muestral de la proporción. La proporción es el número de elementos de un cierto tipo presentes en un conjunto dividido entre el número total de elementos (en este caso los errores de clasificación **nerr** dividido por el número total de elementos probados en el test **M**).

La proporción de elementos se ha denotado por **m**, y la proporción de elementos que no son errores con **1-m**.

La proporción m en muestras grandes (n>30) extraídas al azar tiende a seguir una distribución de probabilidad normal, caracterizada por:

a. Su media coincide con la proporción de elementos (por eso se ha denotado como **m=nerr/M**).
b. Su varianza se calcula con **m*(1-m)/M** y la desviación típica se calcula con **s=sqrt(m*(1-m)/M)**

Así pues, la distribución muestral de la proporción es **N(m, m(1-m)/n)** y sabemos que en una distribución normal el **95%** de las observaciones están en el intervalo dado por **m ± 1.96 * s**.



El motivo por el que damos un intervalo de confianza es porque el error no lo sabemos con total exactitud, ya que **nerr** depende de los elementos elegidos al azar del total (30% de los 1000 en el ejemplo de OCR).

2. *Duda: Sintaxis al definir funciones en octave*

```
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.
```

1- Son los parámetros de salida, pueden ser de tipos distintos.
2- Parámetros de entrada, se van leyendo de izquierda a derecha, pero no tienen que estar todos, los **if** con el parámetro **nargin** (3) controlan cuantos parámetros pasó el usuario al llamar a la función y si no los pasó los inicializa a valores por defecto.

3. *Duda: ¿Que és cada variable en la tabla?*

```
#      a        b    E    k Ete Ete (%)     Ite (%)
#------- -------- --- --- --- ------- ----------
```

Se obtiene de:

```
...
for a=as
  for b=bs
  [w,E,k]=perceptron(data(1:NTr,:),b,a); rl=zeros(M,1);
  ...
```

Y el rango de error de test de:

```
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)
```

Ete

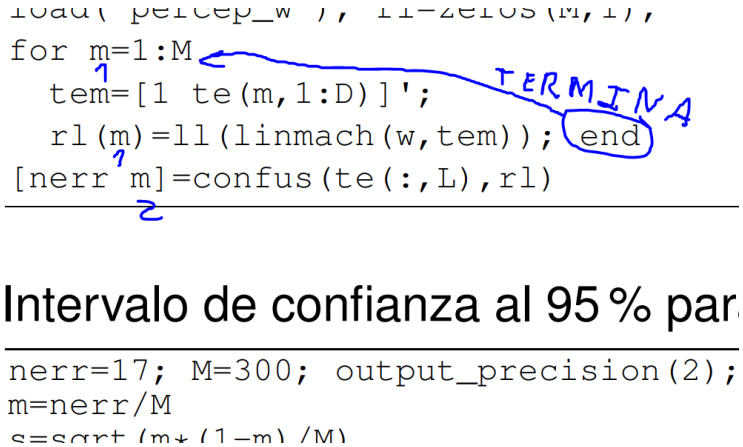
Intervalo de confianza al 95 % par

```
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);
```

Ete %

Ite %

4. *Duda: En la transparencia 10, ¿Qué es la variable m?*



La variable m se utiliza para almacenar tres valores diferentes, lo que puede llevar a confusión si no entendemos el código:

1- En los sitios marcado con 1 es un entero entre 1 y M, esta variable m es destruida cuando termina el bucle después del **end**.

2- En los sitios marcados como 2 m es una matriz donde se compara la clase real a la que pertenece cada observación con la clase que devuelve linmach usando la w entrenada con perceptrón. Las observaciones proceden del 30% separado para test.

   En la diagonal podemos ver 37 ceros que han sido correctamente clasificados como ceros, 29 unos clasificados como unos, etc. En el resto de la matriz podemos ver los errores, según los índices de fila y columna sabemos la clase a la que pertenecían y en la que fueron clasificados por error de w mal entrenado en linmach.

```
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
```

3- En 3 se vuelve a machacar el valor, m es la media de la distribución muestral de la proporción de errores, es decir la media de errores de test (duda 1).

5. *Duda: En la transparencia 10, ¿Qué hace el código de randperm?*

```
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");
```

A la hora de estimar el error de clasificación debemos emplear un conjunto de observaciones que no haya sido utilizado para entrenar el clasificador. Por ello vamos a entrenar el Perceptron con un 70% de los datos y reservamos el 30% restante (test) para estimar el error.

En el caso de OCR_14x14 los datos están agrupados por clase, por lo que si cogemos los 700 primeros dígitos de los 1000 disponibles vamos a dejarnos los 8 y 9, que están en las 300 filas al final de data.

Randperm reordena aleatoriamente las filas de data, así al coger el 70% inicial hay números de todas las clases y obtenemos una muestra aleatoria. Primero inicializamos una semilla para generar números aleatorios:

```
rand("seed",23);
```

Después permutamos las filas:

```
data=data(randperm(N),:);
```

Randperm devuelve vector de permutaciones del tamaño indicado, podemos usar dichos vectores como índices de fila para permutar:

```
>> a = [10,20,30,40,50]
a =

   10   20   30   40   50

>> randperm(5)
ans =

   3   1   5   4   2

>> randperm(5)
ans =

   4   5   1   2   3

>> a([3,1,2,5,4])
ans =

   30   10   20   50   40

>> a(randperm(5))
ans =

   50   10   20   30   40
```

6. *Duda: Este printf me da error de formato y como redondeo valores:*

printf(" a b E k Ete Ete(%) Ite(%)\n");

El símbolo % se usa para formatear las variables que se quieren imprimir en el printf, al ser un carácter reservado da error. Para escaparlo hay que escribirlo dos veces seguidas.

printf(" a b E k Ete Ete(%%) Ite(%%)\n");

Para redondear valores o limitar los decimales pueden usarse estas expresiones

%8.1f – Reserva espacio para 8 dígitos en la parte entera y 1 en la parte decimal
%.1f –Reserva espacio para un dígito en la parte decimal
%3d - Un entero reservando espacio para 3 dígitos

Podéis consultar la referencia del lenguaje para una explicación más exhaustiva.

7. *Duda: ¿Cómo se selecciona el mejor valor de los parámetros a y b?*

Para elegir el mejor valor y, siguiendo las indicaciones de la práctica, vemos que el parámetro a no tiene una gran influencia en el algoritmo, pero la escala de a influirá en la escala de b. Podemos dejar el parámetro a en un valor fijo (0.1 o 1) e ir variando b, primero usando una escala exponencial (0.1 1 10 100 1000 10000) y cuando sepamos más o menos por donde está la mejor b usaremos una escala lineal (100 200 300 400).

El mejor valor viene dado por el porcentaje de error de test (Ete %), ya que tener en cuenta E sería una medida optimista del error (es el error minimizado al entrenar). Ete % sin embargo es el error al clasificar datos que no se usaron durante el entrenamiento.

Aunque E salga algo peor que Ete % hay que tener en cuenta que estamos limitando el número de iteraciones del perceptrón, puede que entrenemos con datos que no se puedan separar fácilmente o que el algoritmo haya parado antes de tiempo.

Le damos prioridad a Ete %: Apuntamos la combinación de a y b que minimiza Ete %.

8. *Comentario profesor: Intervalo de confianza al 95 % para el error estimado*

Este intervalo se obtiene restando y sumando a la media (duda 3). Hay que comprobar que el valor inferior no sea menor que cero (no es posible una proporción de errores negativa) o mayor que el 100%.

```
#       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]
```

9. *Duda: Errores incomprensibles relacionados con los saltos de línea en Windows y linux siendo diferentes*

Al pasar el fichero desde Windows a linux o copiar en Windows y pegar en una máquina virtual Linux o una terminal con un entorno unix (ya sea el subsistema de Linux para Windows, el cygwin o cualquier otra) estamos pegando texto con la codificación de saltos de línea de Windows.

El octave compilado para Windows no entiende esta codificación de fichero. Podemos usar el programa de unix “dos2unix” para arreglar esto:

```
$ apt-get install dos2unix (comando de instalación según la distribución)
$ dos2unix experiment.m
```

Va a convertir todos los caracteres extraños al formato de Linux.

https://askubuntu.com/questions/803162/how-to-change-windows-line-ending-to-unix-version