2020

Sistemas inteligentes

Practica 2

Francisco Joaquín Murcia Gómez
48734281H
Universidad de Alicante

Indice

Introducción

Esta práctica consiste en el reconocimiento de tipos de imágenes, hay 10 tipos (avión, coches, pájaros, gatos, ciervos…), para ello usamos el algoritmo "AdaBoost" que es un algoritmo de Machine Learning, este algoritmo aprende a diferenciar los tipos de imágenes.
Para aprender a diferenciarlas usamos una base de datos de 2000 imágenes donde un 80% se usa para entrenar y el restante para pruebas, esas imágenes se pasan por adaboost y obtenemos las semillas para testear (clasificadores fuertes), estos están compuestos por clasificadores débiles que son los que se encargan de analizar los pixeles de las imágenes. Jugando con el numero de clasificadores que tarda cada clasificador fuerte y jugando con el numero de pruebas que realiza para elegir los mejores clasificadores débiles que formaran el fuerte podemos generar los clasificadores óptimos para acertar un número elevado de imágenes posibles.

Programa

Clase ClasificadorDebil

Esta clase se almacena y se calcula todo lo necesario para trabajar y generar clasificadores débiles, en esta clase se almacenan el pixel umbral, la dirección del clasificador y la confianza, y dos datos auxiliares como el error del clasificador y un vector donde guardamos a que clase pertenece

```
private int pixel;
private int umbral;
private int direccion;
private int[] perteneceClase;//guardamos si pertenece o no a la clase
private double error;
double confianza; //probabilidad de acierto
```

Para construir un clasificador débil, se le asigna un píxel umbral dirección y la cantidad de imágenes que ese clasificador ha de analizar, la confianza, el error se calculan aparte

```
public ClasificadorDebil(int Pixel, int Umbral, int Direccion, int TAM) {
    this.pixel = Pixel;
    this.umbral = Umbral;
    this.direccion = Direccion;
    this.perteneceClase=new int[TAM];
    this.error=0;
}
```

Para calcular la confianza se utilizo un método que aplica la siguiente formula

$$confianza = \frac{1}{2}\log\left(\frac{1-error}{error}\right)$$

A su vez, para calcular el error comprobamos cuantas veces no pertenece a dicha clase y le sumamos el peso de esa imagen

```
public void obtenerErrorClasif(ClasificadorDebil clasificador,ArrayList <Imagen> X, int[] Y, double[] D ){
    double err=0;
    for(int i=0; i<X.size();i++){
        if(Y[i]!=clasificador.perteneceClase[i]){//fallo
           err += D[i];
        }
    }
    clasificador.error=err;
}
```

Para saber si pertenece o no a una clase, comprobamos que el pixel de la imagen entre umbral pertenece a la clase, de lo contrario no

```
public void aplicarClasifDebil(ClasificadorDebil clasificador,ArrayList <Imagen> X){

    for(int i=0; i<X.size();i++){
        if(clasificador.direccion==1){
            if(X.get(i).getImageData()[clasificador.pixel]>clasificador.umbral){
               clasificador.perteneceClase[i]=1;
            }else{
               clasificador.perteneceClase[i]=-1;
            }
        }else{
            if(X.get(i).getImageData()[clasificador.pixel]>clasificador.umbral){
                clasificador.perteneceClase[i]=-1;
            }else{
                clasificador.perteneceClase[i]=1;
            }
        }

    }
}
```

Los umbrales junto con los pixeles y las direcciones se generan al azar, de tal modo que el píxel (p) genera valores entre 0-3072 (32x32x3) el umbral (u) entre 0-255 y la dirección (d) valores de 1 y -1

```
public ClasificadorDebil generarClasifAzar(int TAM){

    Random r = new Random();
    int p = r.nextInt(3072);
    int u = r.nextInt(255);
    int d = r.nextInt(2)+1;//1-2
    if(d==2)d=-1;//si es 2 pasa a -1
    return new ClasificadorDebil(p, u, d,TAM);

}
```

Para aplicar el clasificador débil a una sola imagen usamos una adaptación de la anterior funcion pero pasando una sola imagen

```
private int clasDebil(ClasificadorDebil clasificador, Imagen x){//optieene el clasifi
    if(clasificador.getDireccion()==1){
            if(x.getImageData()[clasificador.getPixel()]>clasificador.getUmbral()){
               return 1;
            }else{
               return-1;
            }
        }else{
            if(x.getImageData()[clasificador.getPixel()]>clasificador.getUmbral()){
                return-1;
            }else{
                return 1;
            }
        }
}
```

A parte de estas funciones están los gueters, seters de los datos junto a un constructor de copia y uno por defecto y un toString para la impresión por pantalla y la escritura del fichero

Clase ClasificadorFuerte

Esta clase almacena todo lo necesario para almacenar y trabajar un clasificador fuerte, como ya comentamos, un clasificador fuerte es un conjunto de clasificadores fuertes por este motivo la clase está compuesta por una lista de clasificadores débiles y dos datos auxiliares como los aciertos que ha tenido el clasificador fuerte y la predicción que ha tenido el clasificador a una foto

```
private ArrayList<ClasificadorDebil>Debiles;
private int [] predicciones;
private double aciertos;
```

Para construir el clasificador simplemente se asignamos un conjunto de clasificadores débiles

```
public ClasificadorFuerte(ArrayList<ClasificadorDebil> Debiles) {
    this.Debiles = Debiles;
}
```

Para aplicar los distintos clasificadores débiles y saber si reconocen una imagen o no tenemos esta función que aplica el clasificador débil a cada imagen, dependiendo de la confianza de este y las veces que acierte podremos decir que el clasificador sirve o no

```
public void aplicarClasFuerte(ArrayList<Imagen> X){

    predicciones=new int[X.size()];
    for(int i=0;i<X.size();i++){//recorremos las imagenes
        double h=0;
        for (int j = 0; j <Debiles.size() ; j++) {
            ClasificadorDebil debil=Debiles.get(j);
             h+=debil.getConfianza()*clasDebil(debil, X.get(i));
        }
        if(h<0){
         predicciones[i]= -1;
        }else if(h>0){
         predicciones[i]= 1;
        }

    }
}
```

Finalmente, para obtener el error es tan sencillo como ver cuantos fallos hay comparando nuestra predicción con el resultado

```
public void obtenerAciertosClasif(int[] Y, ArrayList<Imagen>X){
    double match=0;
    for(int i=0; i<X.size();i++){
        if(Y[i]==this.predicciones[i]){//aciertos
            match++;
        }
    }
    aciertos=match;
}
```

A parte de estas funciones tenemos los geters seters y toString pertinentes para la impresión por pantalla y la escritura del fichero

Adaboost

En primer lugar, tenemos el número de clasificadores débiles que tendrá un fuerte (T) y el numero de pruebas que realizará adaboost para generar los mejores clasificadores clasificadores (A), después se asigna valores por defecto al tamaño repartiendo valores entre 0 y 1 equitativamente a todas las imágenes

```
int T=50;
int A=500;
ClasificadorDebil c;
double D[]=new double[X.size()];//array de pesos
for(int i=0; i<X.size(); i++){//iteramos por cada clasificdor que hay en cada imagen
    D[i]=1.0/X.size();
}
ArrayList<ClasificadorDebil>bestClas=new ArrayList<>();
```

Luego generamos los T clasificadores para ese conjunto de imágenes, para cada clasificador generamos A clasificadores para escoger el mejor, para ello generamos un clasificador, lo aplicamos y obtenemos su error (con las funciones de la clase ClasificadorDebil), con estos clasificadores generados escogemos el que mejor error tenga

```
for(int i=0;i<A;i++){//generamos pruebas
    c=new ClasificadorDebil().generarClasifAzar(X.size());
    c.aplicarClasifDebil(c, X);
    c.obtenerErrorClasif(c, X, Y, D);
    if(i==0){
        err=c.getError();
        cDebil=c;
    }else{
        if(c.getError()<err){//cogemos el error mas pequeño
            err=c.getError();
            cDebil=c;
        }
    }

}
```

El proceso se repite A veces.

Después, calculamos la confianza de el clasificador elegido y actualizamos el pedo

```
cDebil.setConfianza(err);//calclamos y colocamos la confianza en el clasificador debil

D=updatePesos(X.size(),cDebil,D,Y);//actualizamos los pesos de la imagen
bestClas.add(cDebil);
```

Para actualizar los pesos se aplica la siguiente formula:

$$D_{t+1} = \frac{D_t(i) \cdot e^{-\alpha_t \cdot y_i h_t(x_i)}}{Z_t}$$
$$Z_t = \sum_i D_t(i)$$

Para cada imagen se genera un numerador(peso) y un denominador (Z) al generar las dos componentes por cada imagen se realiza la fracción y se coloca en la ubicación correspondiente en el vector de pesos

```
private static double[] updatePesos(int tam,ClasificadorDebil cDebil,double[] D,int[] Y){//actualiza los pesos
    double peso[]=new double[tam];
    double Z=0;
    for(int i=0;i<tam;i++){//calculamos numerador y denoinador
            peso[i]=D[i]*Math.pow(Math.E, -cDebil.getConfianza()*Y[i]*cDebil.getPerteneceClase()[i]);
            Z+=peso[i];
        }
    for(int j=0;j<tam;j++){//actualizamos los pesos
      D[j]=peso[j]/Z;
    }
    return D;
}
```

Este proceso lo repetimos hasta generar los T clasificadores débiles

Finalmente devolvemos el clasificador fuerte

Programa principal

En el main de programa tenemos dos partes, una de entrenamiento y otra de prueba

La parte de entrenamiento su función es entrenar los clasificadores para obtener los clasificadores fuertes óptimos, este modo dedica un porcentaje de imágenes para generar dichos clasificadores y el restante para pruebas, también guarda estos clasificadores en un fichero para la posterior lectura de ese por parte de la ejecución modo test

La parte de test lee el fichero con los clasificadores fuertes guardados por la parte de test y lo aplica a una sola imagen para decir su tipo (avión coche…)

Entrenamiento

Al principio inicializo variable útiles ara la ejecución como los fallos del test el array de las imágenes (X) el array de las carpetas (Y) y genero los conjuntos de imágenes tanto de test como de entrenamiento estas están dentro de un bucle que me recorre las 10 clases con las que estamos trabajando, de tal modo que me ejecutara 10 adaboost

```
int fallos=0;
int fallosTest=0;
int totalImagenes=0;
int totalImagenesTest=0;
double porcentajePromedio=0.0;
double porcentajePromedioTest=0.0;
try {
    FileWriter fichero = new FileWriter(FILE);
    for (int numCarpeta = 0; numCarpeta < 10; numCarpeta++) {//iteramos por ca
        ArrayList<Imagen> X=new ArrayList<>();
        File carpeta = new File("./cifar10_2000/" + numCarpeta);//obtenemos el
        File[] lista= carpeta.listFiles();
        int imagenes=lista.length*80/100;// el 80% de las imagenes
        int Y[]=new int[imagenes*2];
        ArrayList<Imagen> Xtest=new ArrayList<>();
        int Ytest[]=new int[imagenes*2];
        conjuntosImagenes(X, Y, imagenes,numCarpeta);//optenemos los conjuntos
        conjuntosImagenesTest(Xtest,Ytest,imagenes,numCarpeta,lista.length);
```

Para el conjunto de test y de entrenamiento uso dos funciones parecidas, las cuales me recorren la carpeta de la clase en la que este y me lo añaden al array de imágenes y al array de las clases que dependiendo si estoy en la clase elegida valdrá 1 o si no valdrá -1. Para test es identico solo que me escoge en mi caso el 20% de imágenes no elegidas por entrenamiento

```
public static void conjuntosImagenes(ArrayList<Imagen>X, int[] Y,int imagenes,int carpeta){//
    CIFAR10Loader m1=new CIFAR10Loader();
    m1.loadDBFromPath("./cifar10_2000");
    int tam=0;
     for(int i=0; i < 10; i++) {
        ArrayList d1imgs = m1.getImageDatabaseForDigit(i);
        if(i==carpeta){// si estoy en la carpeta de la clase indicada, optengo sus imagenes
            for (int j = 0; j < imagenes; j++) {//carpeta elegida
                Imagen img = (Imagen) d1imgs.get(j);
                X.add(img);
                Y[tam]=1;
                tam++;
            }
        }else{//resto de crpetas
            for (int j = 0; j < imagenes/9; j++) {
                Imagen img = (Imagen) d1imgs.get(j);
                X.add(img);
                Y[tam]=-1;
                tam++;
            }
        }
    }
}
```

Generados los conjuntos, aplicamos adaboost para ese conjunto, en mi caso mis mejores datos son 120 clasificadores generados 100 veces , una vez optenido el clasificador fuerte obtenemos sus aciertos

```
/*
*ENTRENAMIENTO
*/
System.out.println("********** PRUEBA N°"+(numCarpeta+1)+ " **********");
/**
 * llamada a adaboost mejores valores 120/100
 */
ClasificadorFuerte clasFuerte=Adaboost(X,Y,Xtest,Ytest,120,100);//llamada a adaboost
clasFuerte.aplicarClasFuerte(X);//aplicacion del clasificador
clasFuerte.obtenerAciertosClasif(Y,X);
```

Por ultimo tendríamos que obtener los fallos y que imprimir por pantalla el resultado de entrenamiento, por ultimo se escribe el clasificador en el fichero

```
totalImagenes+=X.size();
fallos+=(X.size()-clasFuerte.getAciertos());
porcentajePromedio+=resultadoPrueba(clasFuerte,(double)X.size());//optenemos el resultado
DecimalFormat df= new DecimalFormat("#.00");
.format(resultadoPrueba(clasFuerte, X.size())));
System.out.println("N° imagenes: "+X.size()+"\n"+
        "N° negativos: "+(X.size()-clasFuerte.getAciertos())+"\n"
           + "Porcentaje de acierto Promedio: "
               + ""+df.format(resultadoPrueba(clasFuerte,(double)X.size()))+"%\n");
fichero.write(clasFuerte.toString().replaceAll("\\,",","));// escritura en fichero
```

Para pasar las pruebas con el conjunto de testes igual, pero sin aplicar adaboost, se aplica directamente a el conjunto de test

```
System.out.println("********** Test N°"+(numCarpeta+1)+ " **********");
totalImagenesTest+=Xtest.size();
clasFuerte.aplicarClasFuerte(Xtest);//aplicacion del clasificador
clasFuerte.obtenerAciertosClasif(Ytest,Xtest);
```

Por último, tendríamos otra vez la parte estadística y de impresión que es igual que en entenamiento

Por ultimo imprimimos el resultado promedio de test y de entrenamiento

```
System.out.println("-------------------------------------\n\t   Promedios tota
 resultado(totalImagenes,fallos,porcentajePromedio/10,"Entrenamiento");
 resultado(totalImagenesTest,fallosTest,porcentajePromedioTest/10,"Test");
```

La impresión por pantalla quedaría así:

```
********** PRUEBA N°1 **********
N° imagenes: 314
N° negativos: 12.0
Porcentaje de acierto Promedio: 96,18%

********** Test N°1 **********
N° imagenes: 77
N° negativos: 14.0
Porcentaje de acierto Promedio: 81,82%

********** PRUEBA N°2 **********
N° imagenes: 296
N° negativos: 1.0
Porcentaje de acierto Promedio: 99,66%

********** Test N°2 **********
N° imagenes: 75
N° negativos: 13.0
Porcentaje de acierto Promedio: 82,67%

********** PRUEBA N°3 **********
N° imagenes: 324
N° negativos: 18.0
Porcentaje de acierto Promedio: 94,44%

********** Test N°3 **********
N° imagenes: 77
N° negativos: 18.0
Porcentaje de acierto Promedio: 76,62%
```

```
********** PRUEBA N°4 **********
N° imagenes: 309
N° negativos: 15.0
Porcentaje de acierto Promedio: 95,15%

********** Test N°4 **********
N° imagenes: 75
N° negativos: 19.0
Porcentaje de acierto Promedio: 74,67%

********** PRUEBA N°5 **********
N° imagenes: 342
N° negativos: 15.0
Porcentaje de acierto Promedio: 95,61%

********** Test N°5 **********
N° imagenes: 79
N° negativos: 18.0
Porcentaje de acierto Promedio: 77,22%

********** PRUEBA N°6 **********
N° imagenes: 290
N° negativos: 3.0
Porcentaje de acierto Promedio: 98,97%

********** Test N°6 **********
N° imagenes: 73
N° negativos: 12.0
Porcentaje de acierto Promedio: 83,56%
```

```
********** PRUEBA Nº7 **********
Nº imagenes: 327
Nº negativos: 4.0
Porcentaje de acierto Promedio: 98,78%

********** Test Nº7 **********
Nº imagenes: 78
Nº negativos: 11.0
Porcentaje de acierto Promedio: 85,90%

********** PRUEBA Nº8 **********
Nº imagenes: 312
Nº negativos: 9.0
Porcentaje de acierto Promedio: 97,12%

********** Test Nº8 **********
Nº imagenes: 76
Nº negativos: 16.0
Porcentaje de acierto Promedio: 78,95%

********** PRUEBA Nº9 **********
Nº imagenes: 324
Nº negativos: 4.0
Porcentaje de acierto Promedio: 98,77%

********** Test Nº9 **********
Nº imagenes: 77
Nº negativos: 12.0
Porcentaje de acierto Promedio: 84,42%
```

```
********** PRUEBA Nº10 **********
Nº imagenes: 324
Nº negativos: 1.0
Porcentaje de acierto Promedio: 99,69%

********** Test Nº10 **********
Nº imagenes: 77
Nº negativos: 17.0
Porcentaje de acierto Promedio: 77,92%

-------------------------------------
           Promedios totales
-------------------------------------

********** Entrenamiento **********
Nº imagenes: 3162
Nº negativos: 82
Porcentaje de acierto Promedio: 97,44%

********** Test **********
Nº imagenes: 764
Nº negativos: 150
Porcentaje de acierto Promedio: 80,37%
```

Test

Para la parte de test lo primero que hacemos es leer el fichero generado por entrenamiento, recorremos los 10 fuertes y calculamos su h que es la multipliacacion de la confianza por el signo del clasificador debil, el que tenga mayor h será el clasificador que pertenece la clase

```
ArrayList<ClasificadorFuerte>Fuertes=LeerFichero(FILE);
Imagen img=new Imagen();
img.loadFromPath(args[1]);
int tipo=0;
double aciertos=0;
 for (int i = 0; i < 10; i++) {//recorremos los 10 clasificadores
     double h=0;
         for (int j = 0; j<Fuertes.get(i).getDebiles().size() ; j++) {//optene
             ClasificadorDebil debil=Fuertes.get(i).getDebiles().get(j);
             h+=debil.getConfianza()*debil.clasDebil(debil,img);
         }
     if(i==0){
         aciertos=h;
     }
     if(h>aciertos){//optenemos el mejor clasificador
         aciertos=h;
         tipo=i;
     }
 }
```

después tenemos un switch que dependiendo del clasificador elegido de imprime el nombre de su clase
Un ejemplo serio:
Le paso una imagen de un barco, ejecuto y de dice el tipo que es

```
Debugger Console × 2021_P2SI (run) ×
run:
Barcos
BUILD SUCCESSFUL (total time: 0 seconds)
```

Podemos observar que nos detecta correctamente, si hacemos una vista detallada de ese valor de la H, vemos los siguiente

```
run:
peso= 1.3740258307796376
peso= 0.16485952782183372
peso= -1.8594942278238324
peso= -1.3759495326727043
peso= -1.339178084635965
peso= -1.1877884196071813
peso= -3.2062465162475533
peso= -0.5152573089029913
peso= 1.4874230614250585
peso= -0.9472649316974991
Barcos
BUILD SUCCESSFUL (total time:
```

Las clase aviones y barcos son las mas puntuadas, al final se decanta por barcos porque es la que mejor nota saca, ese es el motivo por el que he encontrado ejecuciones que me confunde para esa imagen un barco por un avión,

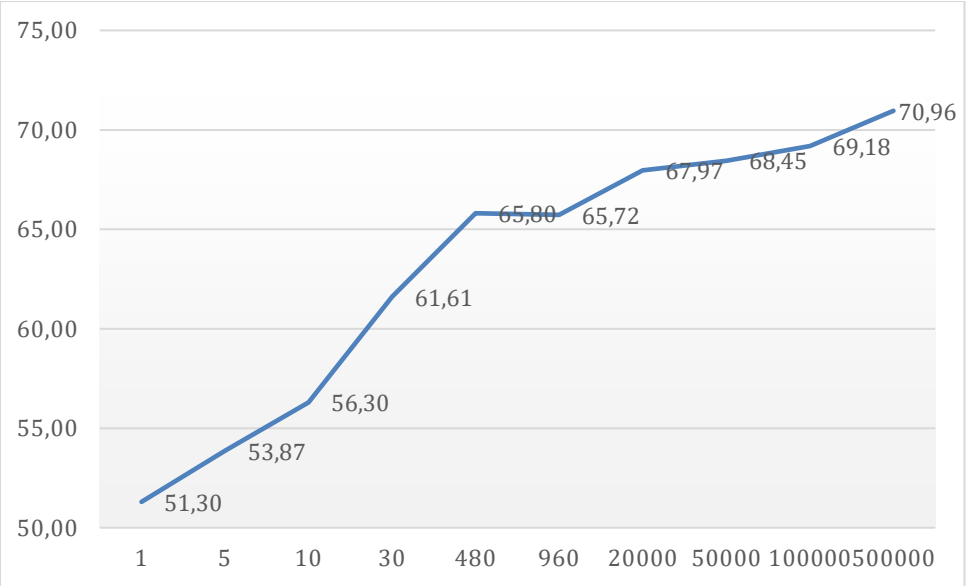
Preguntas propuestas

¿Cuál es el número de clasificadores que se han de generar para que un clasificador débil funcione? Muestra una gráfica que permita verificar lo que comentas.

He generado una tabla con los porcentajes que generan A clasificadores débiles, esta tabla esta pintada de colores siendo el color rojo los porcentajes más bajos y el verde el más alto. He generado hasta 500000 clasificadores porque si generaba mas java abortaba la ejecución

Clase/A	1	5	10	30	480	960	20000	50000	100000	500000
1	47,77	56,37	56,69	63,69	67,52	69,11	69,11	69,11	69,75	71,38
2	48,99	55,74	55,74	64,86	67,57	65,88	70,95	69,93	70,27	70,95
3	49,69	55,25	51,85	55,56	61,42	61,42	63,27	64,20	64,81	69,00
4	58,58	53,40	56,96	60,52	60,52	61,49	61,81	62,78	63,75	68,00
5	48,25	52,34	57,60	62,87	64,62	66,08	67,84	68,42	68,42	68,71
6	60,34	50,00	58,28	61,38	65,52	64,83	68,28	69,31	70,00	69,97
7	46,79	54,43	56,88	60,55	66,97	66,97	69,72	70,64	72,78	72,48
8	53,21	52,88	55,13	55,45	63,78	64,42	68,27	65,38	67,31	68,27
9	50,93	51,85	61,11	66,98	71,60	68,21	70,37	74,07	74,07	78,84
10	48,46	56,48	52,78	64,20	68,52	68,83	70,06	70,68	70,68	71,99

A	1	5	10	30	480	960	20000	50000	100000	500000
Media	51,30	53,87	56,30	61,61	65,80	65,72	67,97	68,45	69,18	70,96

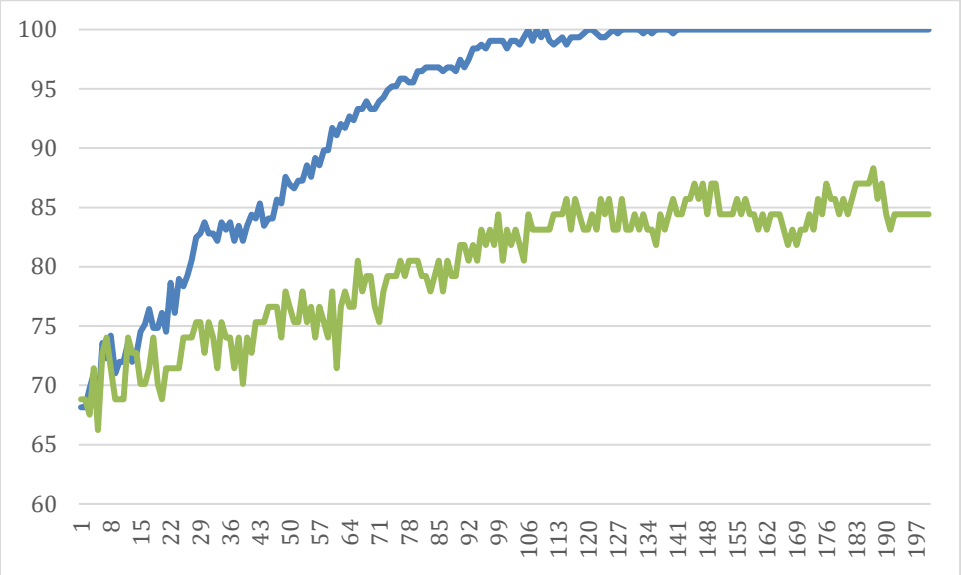


Como observamos si generamos un clasificador únicamente existen valores inferiores al 50% cosa que es inaceptable, ya con 20000 ya tenemos unos porcentajes que podemos aceptar como aceptables ya que lo ideal seria usar el de 500000 pero alargaría el tiempo de ejecución excesivamente, en mi opinión seria usar este valor si queremos usar solo un clasificador débil en el fuerte, pero lo suyo es emplear mas de un clasificador ya que así no tendríamos que generar tantos

¿Cómo afecta el número de clasificadores generados al tiempo empleado para el proceso de aprendizaje? ¿Qué importancia le darías? Justifica tu respuesta.

Conti mas generemos mas tiempo emplearemos ya que recorremos mas iteraciones del bucle, lo suyo es encontrar un equilibrio, por ejemplo

Esta es una comparación de los porcentajes de test (verde) y de entrenamiento (azul), generamos 200 clasificadores débiles, podemos ver que llega un punto que es inútil generar más clasificadores por que se estanca en 80%-85% si generamos mas seria una perdida de tiempo y de recursos, en este cao podríamos parar con 135 clasificadores



En este caso se aprecia un sobre entrenamiento, por mucho que siga entrenando no mejora

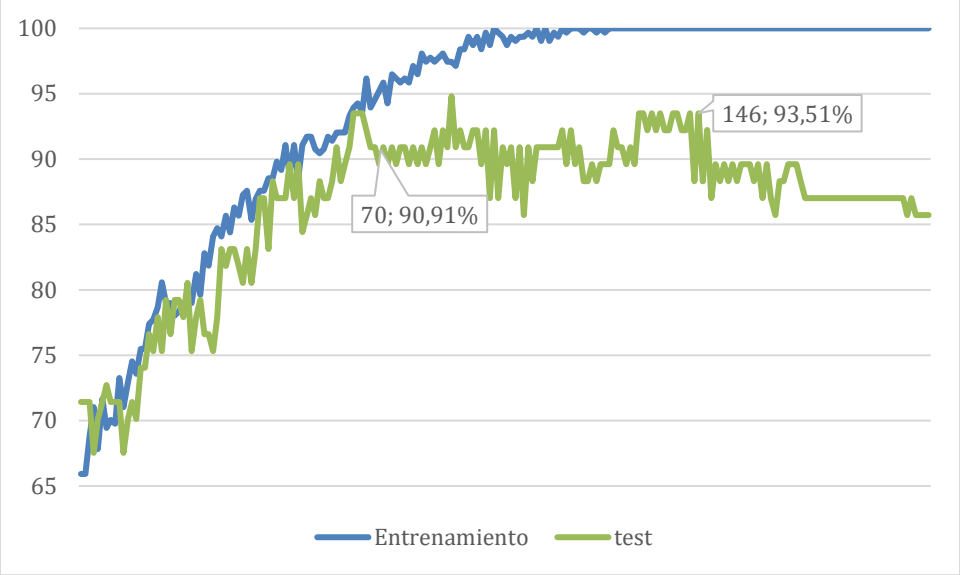
¿Cómo has dividido los datos en conjunto de entrenamiento y test? ¿Para qué es útil hacer esta división?

En mi caso he usado un 80% para entrenar y un 20% para test, se podría usar un 70-30 perfectamente, pero a mí 80-20 me daba mejores resultados.

Es importante usar unas imágenes para probar que no conozca el algoritmo, porque es donde de verdad se va a probar si está aprendiendo, si usáramos las mismas seria como hacerle un examen donde ya sabe las preguntas y se las ha estudiado

¿Has observado si se produce sobre entrenamiento? Justifica tu respuesta con una gráfica en la que se compare el error de entrenamiento y el de test a lo largo de las ejecuciones.

Si, por ejemplo, si generamos un clasificador fuerte con 200 clasificadores débiles y que pruebe cada clasificador 500 veces, si mostramos una grafica del porcentaje de aciertos tanto de test como de entrenamiento podemos ver que a partir del clasificador 70 se mantiene estable en valores de alrededor del 90% hasta llegar al clasificador 146 que observamos que empieza a ser contraproducente seguir entrenando ya que empieza a bajar el porcentaje de acierto.



Comenta detalladamente el funcionamiento de AdaBoost teniendo en cuenta que tasa media de fallos obtienes para aprendizaje y test.

Como ya he mencionado más detalladamente en la parte de Adaboost, el funcionamiento seria que genera un conjunto de clasificadores débiles, cada clasificador de débil es el mejor que los A generados, siendo A el número de clasificadores débiles que se han de generar para obtener un clasificador débil. Un ejemplo de la ejecución que se ve un promedio de las 10 clase es en Entrenamiento otros ejemplos serian los siguientes, voy a obviar las ejecuciones individuales de las clases y mostrar solo los promedios:

```
-------------------------------------
          Promedios totales
-------------------------------------

********** Entrenamiento **********
Nº imagenes: 3162
Nº negativos: 67
Porcentaje de acierto Promedio: 97,89%

********** Test **********
Nº imagenes: 764
Nº negativos: 140
Porcentaje de acierto Promedio: 81,62%
```

```
-------------------------------------
          Promedios totales
-------------------------------------

********** Entrenamiento **********
Nº imagenes: 3162
Nº negativos: 75
Porcentaje de acierto Promedio: 97,64%

********** Test **********
Nº imagenes: 764
Nº negativos: 146
Porcentaje de acierto Promedio: 80,90%
```

Si hacemos la dedia de las tres ejecuciones nos da aproximadamente un 81% de aciertos lo que es un 19% de fallos

¿Cómo has conseguido que Adaboost clasifique entre los 10 dígitos cuando solo tiene una salida binaria?

En el main empleo un bucle que me ejecuta el adaboost 10 veces 1 por cada clase, como comento en la parte de la memoria de Entrenamiento