

2020

Practica 2

Sistemas inteligentes

Francisco Joaquín Murcia Gómez

48734281H

Universidad de Alicante

Indice

[Introducción 3](#_Toc59490868)

[Programa 3](#_Toc59490869)

[Clase ClasificadorDebil 3](#_Toc59490870)

[Clase ClasificadorFuerte 6](#_Toc59490871)

[Adaboost 7](#_Toc59490872)

[Programa principal 8](#_Toc59490873)

[Entrenamiento 8](#_Toc59490874)

[Test 11](#_Toc59490875)

[Preguntas propuestas 13](#_Toc59490876)

# Introducción

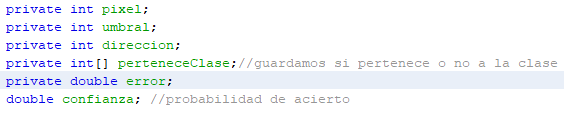
Esta práctica consiste en el reconocimiento de tipos de imágenes, hay 10 tipos (avión, coches, pájaros, gatos, ciervos…), para ello usamos el algoritmo “AdaBoost” que es un algoritmo de Machine Learning, este algoritmo aprende a diferenciar los tipos de imágenes.

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.

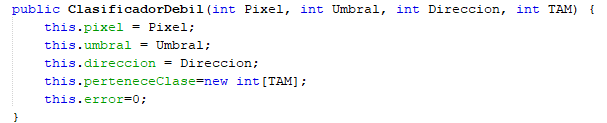
# Programa

## 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

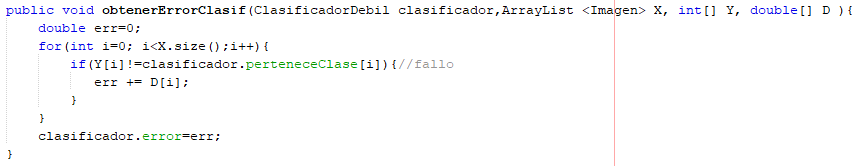


Para construir un clasificador débil, se le asigna un píxel umbral dirección y la cantidad de imágenes que ese clasificador ha de analizar, la confianza, el error se calculan aparte

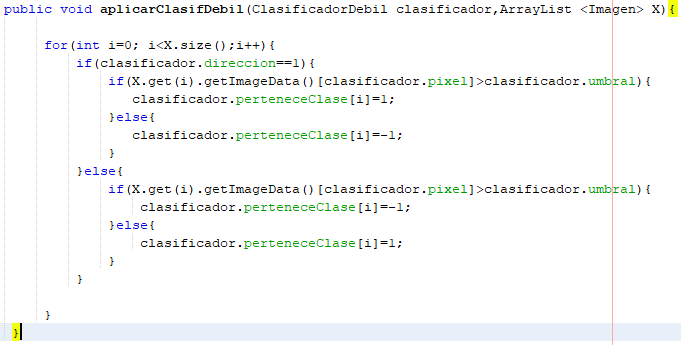


Para calcular la confianza se utilizo un método que aplica la siguiente formula

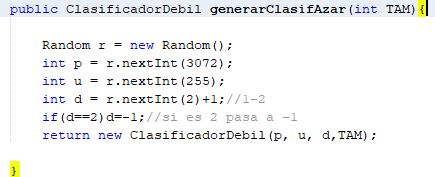
A su vez, para calcular el error comprobamos cuantas veces no pertenece a dicha clase y le sumamos el peso de esa imagen



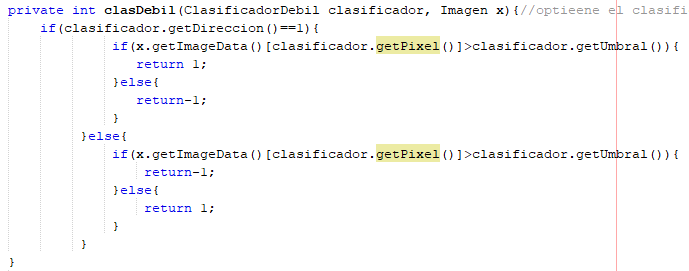
Para saber si pertenece o no a una clase, comprobamos que el pixel de la imagen entre umbral pertenece a la clase, de lo contrario no



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



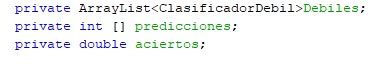
Para aplicar el clasificador débil a una sola imagen usamos una adaptación de la anterior funcion pero pasando una sola imagen



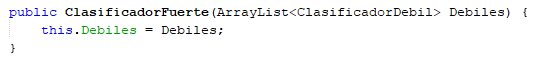
A parte de estas funciones están los gueters, seters de los datos junto a un constructor de copia y uno por defecto y un toString para la impresión por pantalla y la escritura del fichero

## Clase ClasificadorFuerte

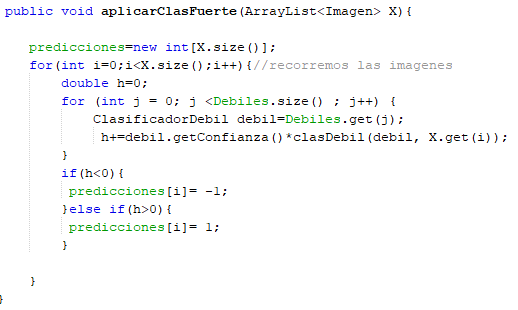
Esta clase almacena todo lo necesario para almacenar y trabajar un clasificador fuerte, como ya comentamos, un clasificador fuerte es un conjunto de clasificadores fuertes por este motivo la clase está compuesta por una lista de clasificadores débiles y dos datos auxiliares como los aciertos que ha tenido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



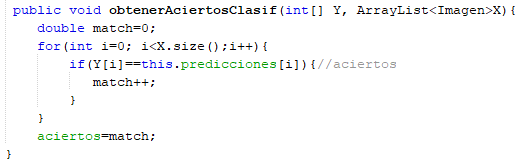
Para construir el clasificador simplemente se asignamos un conjunto de clasificadores débiles



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



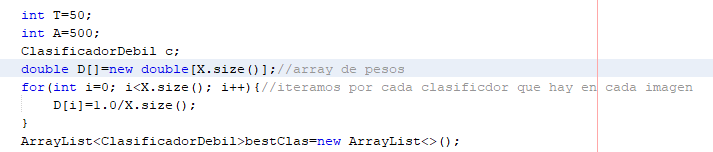
Finalmente, para obtener el error es tan sencillo como ver cuantos fallos hay comparando nuestra predicción con el resultado



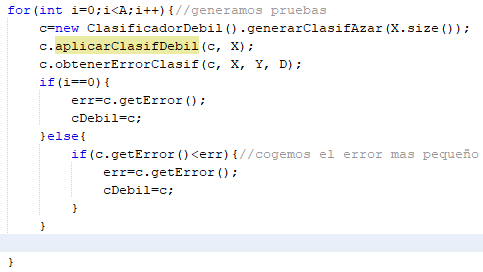
A parte de estas funciones tenemos los geters seters y toString pertinentes para la impresión por pantalla y la escritura del fichero

## 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

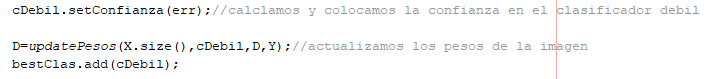


Luego generamos los T clasificadores para ese conjunto de imágenes, para cada clasificador generamos A clasificadores para escoger el mejor, para ello generamos un clasificador, lo aplicamos y obtenemos su error (con las funciones de la clase ClasificadorDebil), con estos clasificadores generados escogemos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

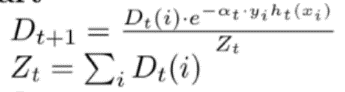


El proceso se repite A veces.

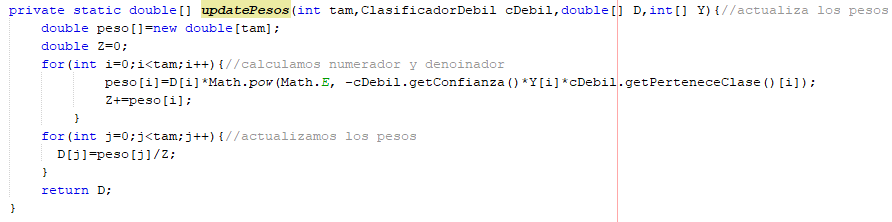
Después, calculamos la confianza de el clasificador elegido y actualizamos el pedo



Para actualizar los pesos se aplica la siguiente formula:



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



Este proceso lo repetimos hasta generar los T clasificadores débiles

Finalmente devolvemos el clasificador fuerte

## Programa principal

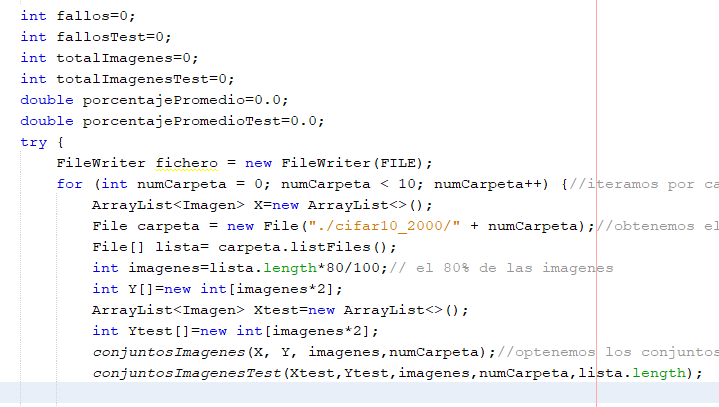
En el main de programa tenemos dos partes, una de entrenamiento y otra de prueba

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

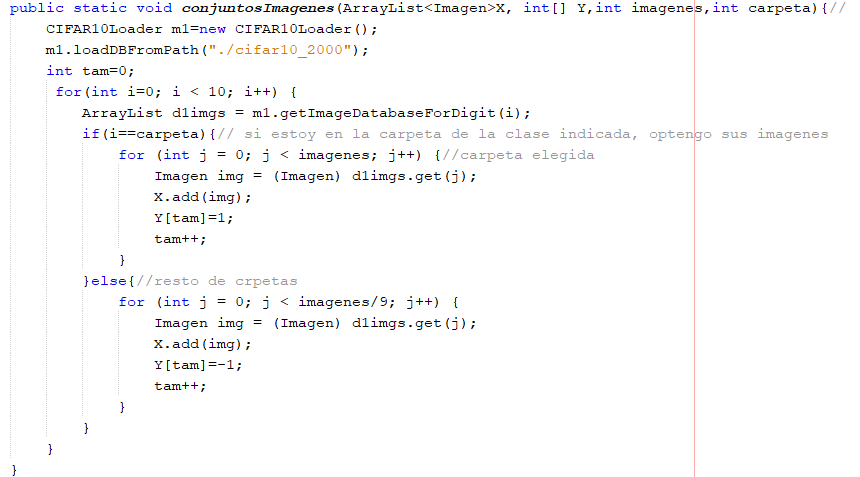
La parte de test lee el fichero con los clasificadores fuertes guardados por la parte de test y lo aplica a una sola imagen para decir su tipo (avión coche…)

### Entrenamiento

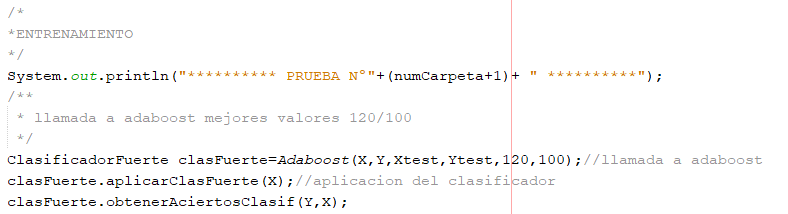
Al principio inicializo variable útiles ara la ejecución como los fallos del test el array de las imágenes (X) el array de las carpetas (Y) y genero los conjuntos de imágenes tanto de test como de entrenamiento estas están dentro de un bucle que me recorre las 10 clases con las que estamos trabajando,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



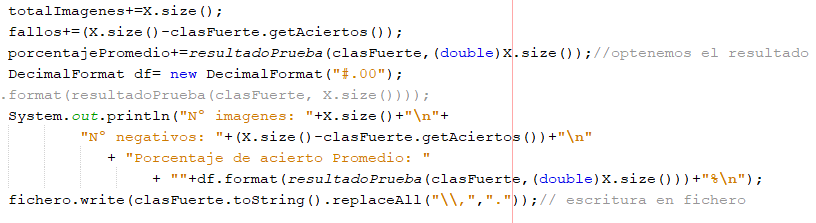
Para el conjunto de test y de entrenamiento uso dos funciones parecidas, las cuales me recorren la carpeta de la clase en la que este y me lo añaden al array de imágenes y al array de las clases que dependiendo si estoy en la clase elegida valdrá 1 o si no valdrá -1. Para test es identico solo que me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



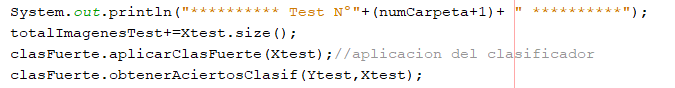
Generados los conjuntos, aplicamos adaboost para ese conjunto, en mi caso mis mejores datos son 120 clasificadores generados 100 veces , una vez optenido el clasificador fuerte obtenemos sus aciertos



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

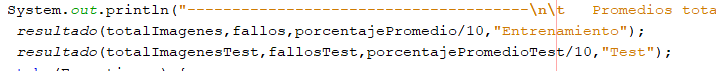


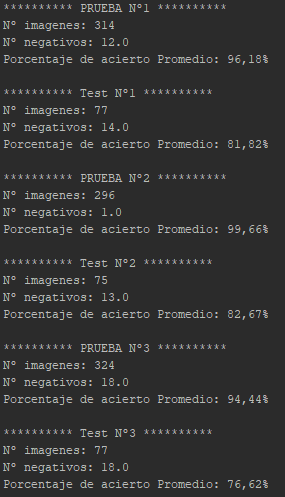
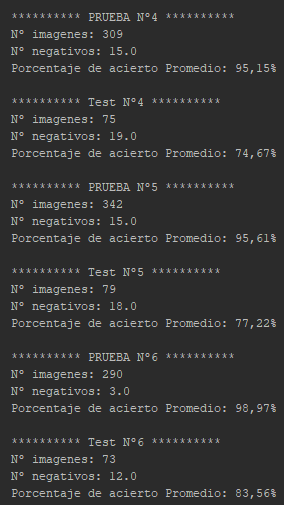
Para pasar las pruebas con el conjunto de testes igual, pero sin aplicar adaboost, se aplica directamente a el conjunto de test

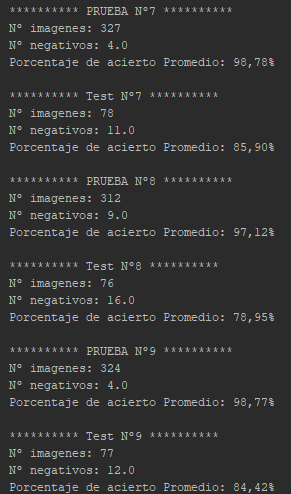
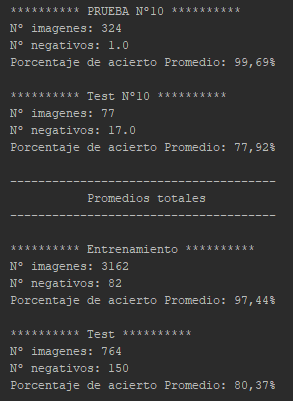


Por último, tendríamos otra vez la parte estadística y de impresión que es igual que en entenamiento

Por ultimo imprimimos el resultado promedio de test y de entrenamiento



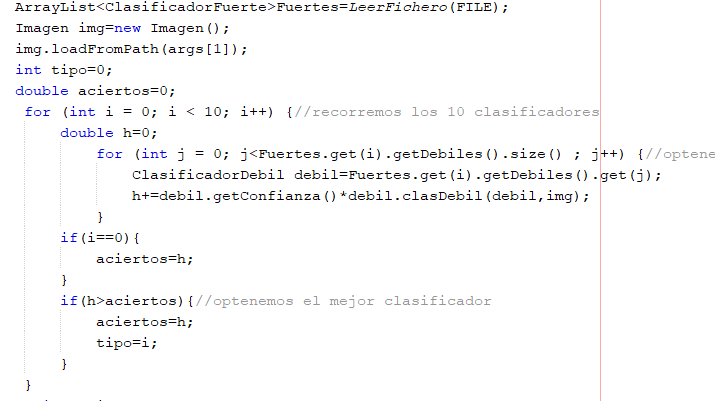
La impresión por pantalla quedaría así:



### Test

Para la parte de test lo primero que hacemos es leer el fichero generado por entrenamiento,

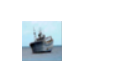
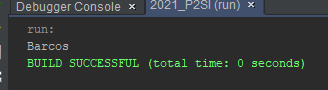
recorremos los 10 fuertes y calculamos su h que es la multipliacacion de la confianza por el signo del clasificador debil, el que tenga mayor h será el clasificador que pertenece la clase



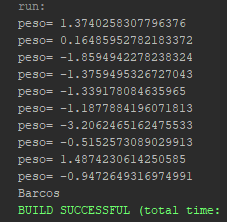
después tenemos un switch que dependiendo del clasificador elegido de imprime el nombre de su clase

Un ejemplo serio:

Le paso una imagen de un barco, ejecuto y de dice el tipo que es



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

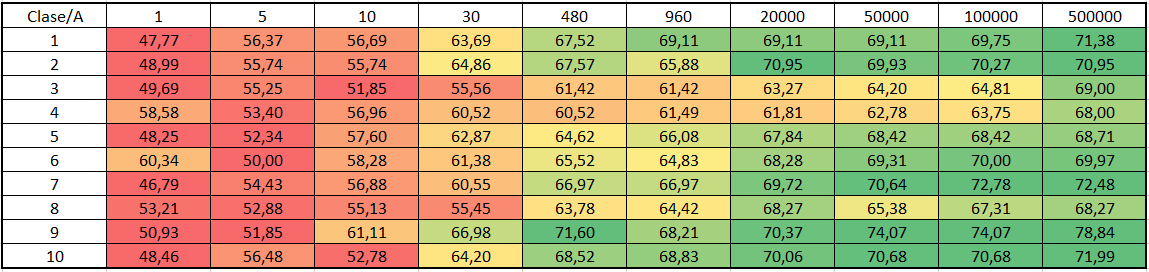


Las clase aviones y barcos son las mas puntuadas, al final se decanta por barcos porque es la que mejor nota saca, ese es el motivo por el que he encontrado ejecuciones que me confunde para esa imagen un barco por un avión,

# 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





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?

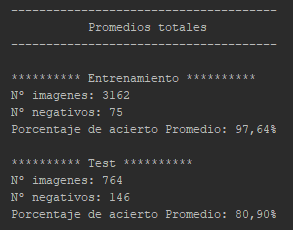
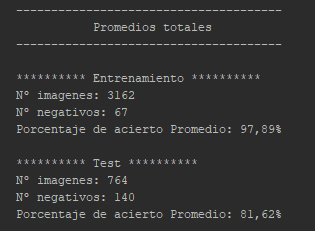
En mi caso he usado un 80% para entrenar y un 20% para test, se podría usar un 70-30 perfectamente, pero a mí 80-20 me daba mejores resultados.

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](#_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](#_Entrenamiento) otros ejemplos serian los siguientes, voy a obviar las ejecuciones individuales de las clases y mostrar solo los promedios:

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](#_Entrenamiento)