UNIVERSIDAD DE ALICANTE | Departamento de Ciencia de la Computación e Inteligencia Artificial

Práctica 2. Visión Artificial y Aprendizaje

sISTEMAS INTELIGENTES

Pedro Giménez Aldeguer

Clase: Viernes (09:00-11:00)

DNI: 15419933C

2018

**Contenido**

[1. Introducción: 2](#_Toc533367828)

[2. Pseudocódigo: 2](#_Toc533367829)

[3. Funcionamiento de Adaboost: 3](#_Toc533367830)

[ **ClasificadorDébil:** 3](#_Toc533367831)

[ **ClasificadorFuerte:** 6](#_Toc533367832)

[ **AdaBoost:** 8](#_Toc533367833)

[4. Pruebas: 13](#_Toc533367834)

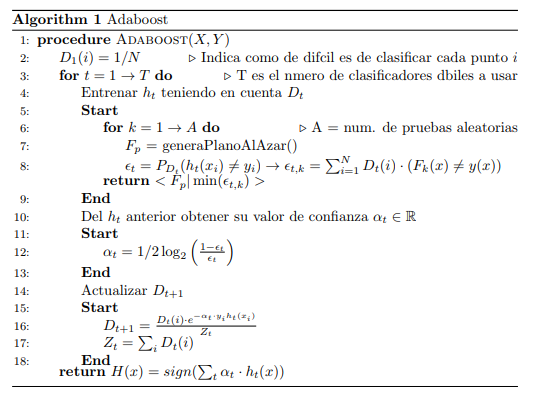
# Introducción:

En la segunda práctica de la asignatura se va a desarrollar un sistema capaz de distinguir entre distintas imágenes. Como ejemplo imaginemos que debemos desarrollar una web que permite la venta de objetos online y deseamos que los usuarios puedan subir fotos de 8 tipos de objetos para poder realizar la venta. Únicamente deben subir una imagen y el sistema tiene que clasificar los objetos entre: abrigos, bolsos, camisetas, pantalones, suéters, vestidos, zapatillas, zapatos.

Para ello se va a implementar un sistema de aprendizaje automático supervisado. La entrada al sistema consistirá en un conjunto de imágenes etiquetadas según la clase a la que pertenezcan. El objetivo de esta práctica es aprender un clasificador en base a este conjunto de entrada que permita clasificar sin problemas imágenes pertenecientes a estas clases, aunque no se hayan visto anteriormente. El objetivo final sería construir un clasificador que, tras ser entrenado, pueda decirnos a que clase corresponde una imagen.

Como ya hemos dicho, las imágenes con las que vamos a trabajar corresponden a 8 tipos: abrigos, bolsos, camisetas, pantalones, suéteres, vestidos, zapatillas, zapatos. Las imágenes están almacenadas en escala de grises. Cada imagen va a estar representada por un vector característico. El número de componentes de este vector vendrá determinado por el número de píxeles que componen la imagen y el valor de cada componente se corresponderá con el valor de gris de cada píxel.

# Pseudocódigo:



# Funcionamiento de Adaboost:

A continuación, me encargaré de explicar el funcionamiento de Adaboost paso por paso.

Para mayor detalle del funcionamiento explicaré las funciones de las diferentes clases que he generado:

## **ClasificadorDébil:**

En esta clase genero los clasificadores débiles que necesito para poder trabajar con Adaboost.

Tengo la función generarClasificadorAzar():

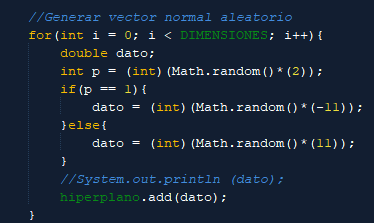
https://i.gyazo.com/bac5917807ba0bb30eba6cd995d4b050.png

En ella genero un hiperplano que utiliza una ecuación lineal para determinarse en el espacio multidimensional. Estos planos tienen 784 dimensiones ya que se trata del número de píxeles que tiene una imagen.

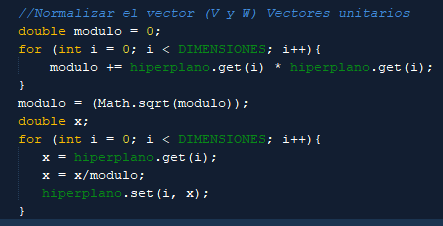
Un hiperplano afín en un espacio *n*-dimensional puede ser descrito por una ecuación lineal no degenerada con la siguiente forma: {\displaystyle a\_{1}x\_{1}+a\_{2}x\_{2}+…+a\_{n}x\_{n}=b}

https://i.gyazo.com/3c258cd55a55fc79db532f22500f8534.png

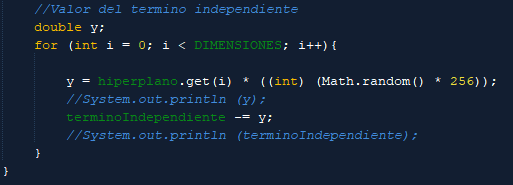
La función genera aleatoriamente un vector normal entre -10 y 10:



Se normaliza para que su módulo sea 1. Cada componente de este vector normal se corresponde con cada componente de a en la ecuación lineal del hiperplano.



Se calcula aleatoriamente un punto en el hiperespacio por el que pase el hiperplano. El punto está limitado a los valores que se puede tomar. Como son imágenes en blanco y negro, los valores que se podrán tomar son del 0 al 255.

Además, al acabar se calcula el término independiente sustituyendo ese punto aleatorio y resolviendo la incógnita D.

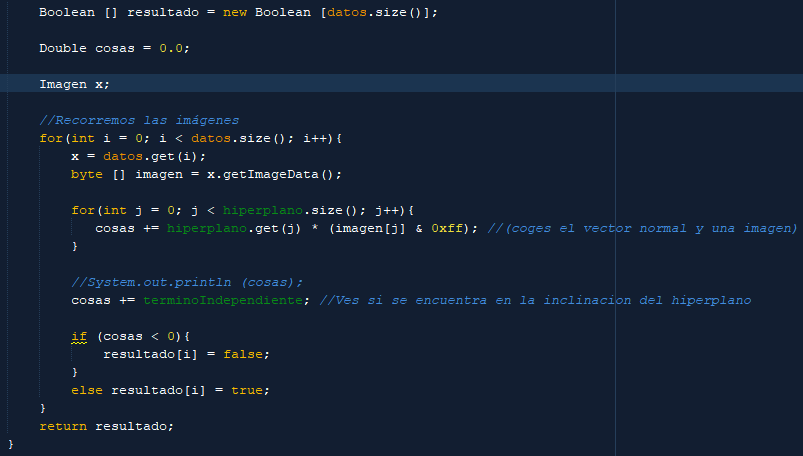
La función aplicarClasificadorDebil():

https://i.gyazo.com/0432e3c9e86bec58a1c98b189c194525.png

Esta función se encarga de comprobar si un punto se encuentra por encima o por debajo del hiperplano.

Se recorren las imágenes y se van añadiendo a la ecuación lineal cada uno de los píxeles de cada imagen para ver si cada píxel (punto) está contenido en el hiperplano:

* Si se obtiene 0 entonces la imagen está contenida en el plano.
* Si se obtiene un valor positivo, la imagen esta por encima del plano.
* Si se obtiene un valor negativo la imagen esta por debajo del plano.



https://i.gyazo.com/a4cf56d1d9fed41f01eef770f14f3517.pngLa función obtenerErrorClasificador():

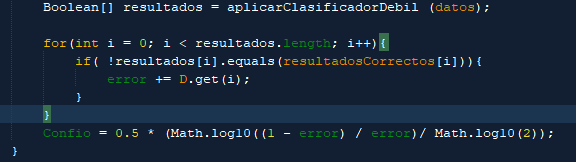
Esta función obtiene el error que tiene el clasificador débil tras haber aplicado el hiperplano a las imágenes.

Compara los resultados obtenidos tras haber aplicado el hiperplano con el resultado correcto que debería haber dado.

Si los resultados no coinciden entonces se va incrementando los errores junto al peso de cada imagen.

Además, en esta función almaceno el valor de confianza que tiene el clasificador débil.



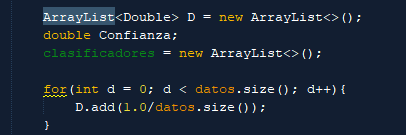


## **ClasificadorFuerte:**

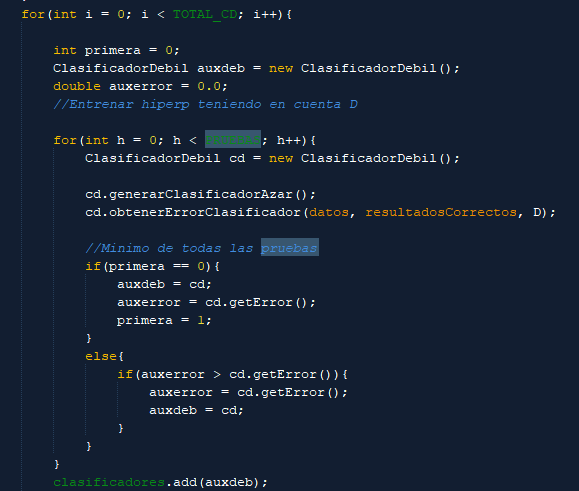
La función clasificadorFuerte():

https://i.gyazo.com/288c59d761e999128e739f594c579940.png

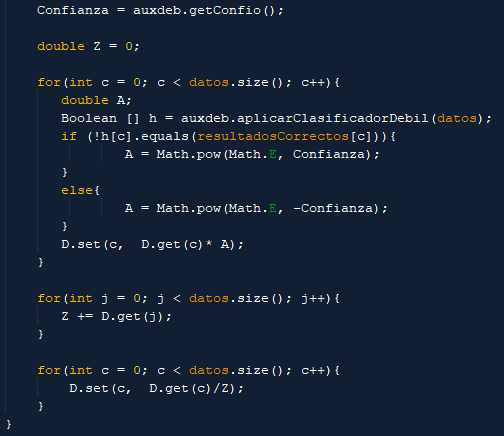
Lo primero de todo, indica como de difícil es de clasificar cada imagen.



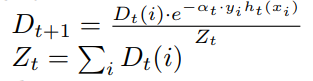
Empieza a generar Clasificadores Débiles dependiendo de lo que le pongamos. Además de generar muchas pruebas para quedarnos con los clasificadores con menor error.



Tras tener un clasificador débil, se guarda el valor de confianza que tiene y se empieza a actualizar D para los siguientes clasificadores que se generen. Si un clasificador tiene un valor de confianza alto, será más relevante a la hora de tomar la decisión.



Para cada patrón se actualiza su peso según la expresión siguiente:



En la fórmula de Dt+1, “y” es el resultado real, el verdadero y “h” es el resultado que devuelve el clasificador que se ha generado.

Cuando “y” no coincide con “h”, el exponente “e” toma un valor positivo, si no, un valor negativo. Cuando no coincide, el peso se incrementa y si coincide decrementa por ser negativo.

Además, se calcula Z como un valor de normalización, sumando el total de pesos.

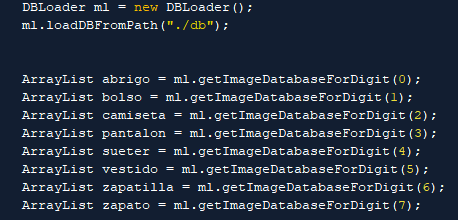
## **AdaBoost:**

La función entrenamiento():

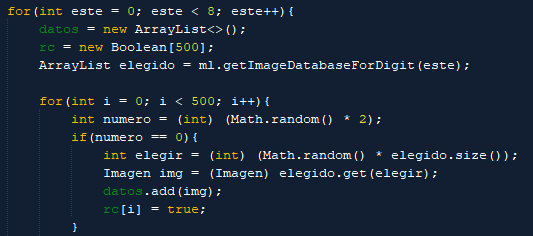
https://i.gyazo.com/a826da8cc95433fc7b614f8f3b150404.png

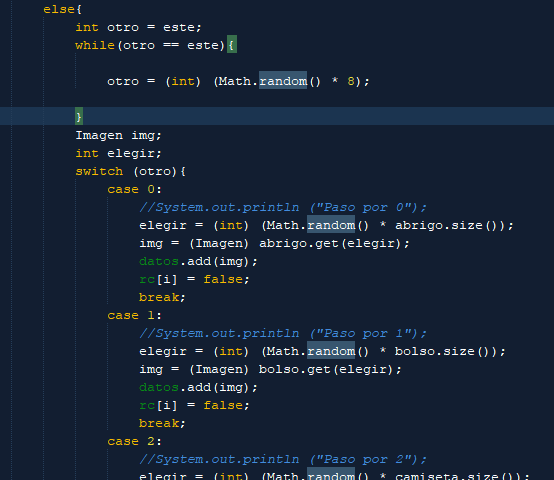
En esta función se entrenará adaboost para cada tipo de imagen.

Se carga la base de datos y empiezo a generar los diferentes entrenamientos para cada tipo de imagen. (8 tipos de imagen)



Voy seleccionando un 50% del tipo de imagen del que se va a hacer el entrenamiento y almacenando en un Boolean los resultados correctos para luego, al generar un clasificador, poder obtener el error y la confianza.



Y el 50% restante con otras imágenes diferentes al tipo del que se quiere entrenar. El entrenamiento tiene que estar equilibrado para que entrene bien. 

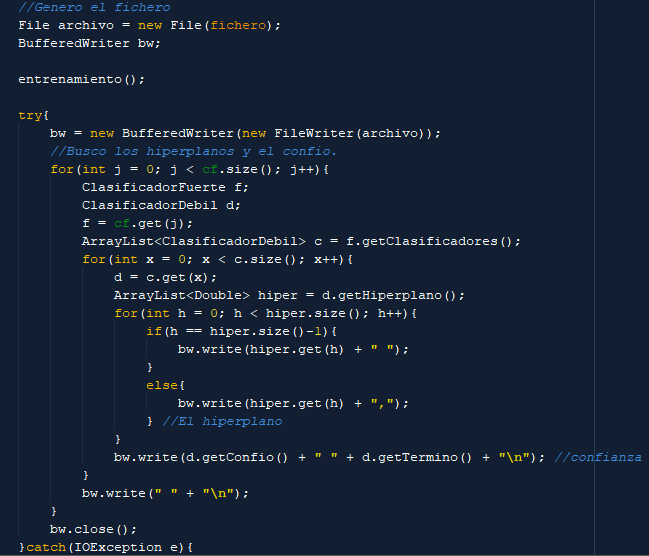
Tras tener toda esa distribución de imágenes equilibrada de un tipo de imagen, se irá generando el entrenamiento y almacenando.

https://i.gyazo.com/f2c95389e6ad22035934fdee05bbac7f.png

La función crearFichero():

https://i.gyazo.com/94569e82ef8d73666fa08842ec044d0f.png

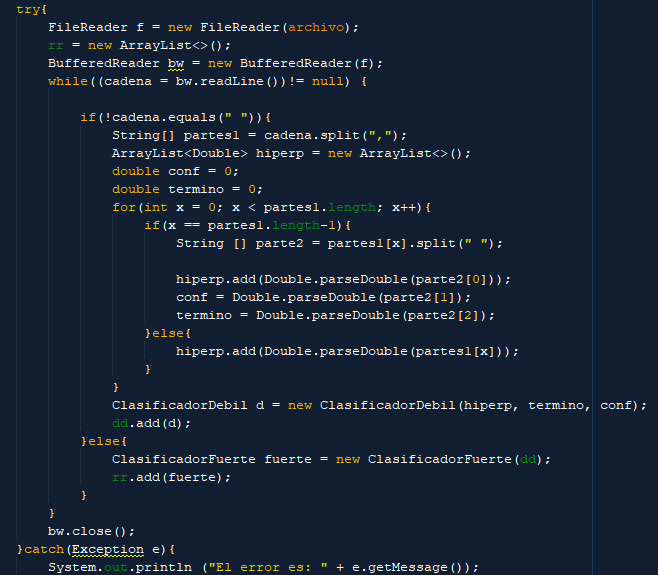
Genera un fichero guardando todos los entrenamientos generados.



La función abrirFichero():

https://i.gyazo.com/a44c14c03992602a22ddc0a1d9e9a27b.png

Abre el fichero donde se encuentra los entrenamientos guardados anteriormente y los almacena en un Array.



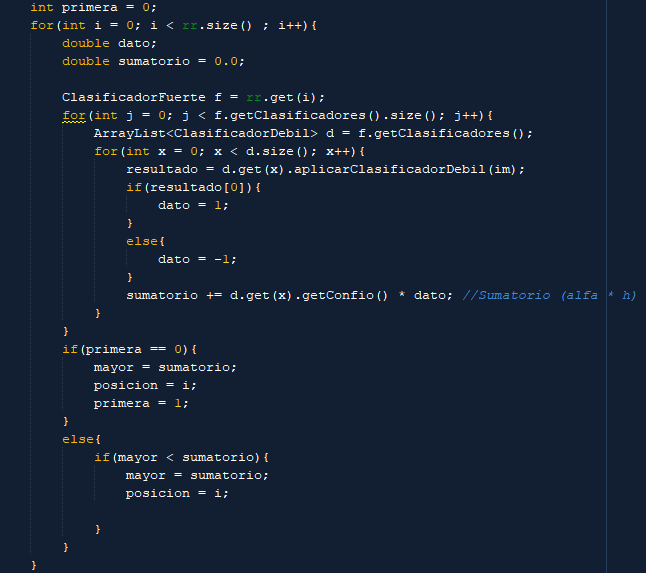
La función Conclusion():

https://i.gyazo.com/b857c699ac6bf6c0b786acc984806d38.png

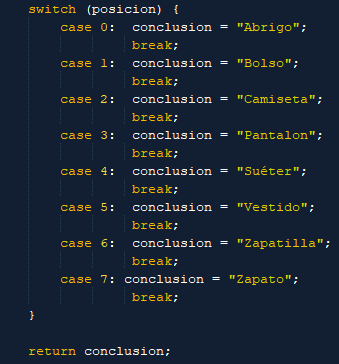
Esta función se encarga de cargar los clasificadores fuertes almacenador tras cargar el fichero de los entrenamientos y se ejecutarán sobre la imagen de prueba que se pasará por parámetro.

Tras aplicar cada uno de los clasificadores fuertes a la imagen pasada, tendrá prioridad el que tenga mayor sumatorio de las variables de confianza y el dato resultante de sus clasificadores débiles.

https://i.gyazo.com/93ceb44dc5245fa46032c8c13a278012.png



Después de saber que clasificador fuerte es el ganador, imprime el tipo de imagen correspondiente.



Este resultado que imprime debe coincidir con el tipo de imagen que se ha pasado por parámetro. Si no es así, deberemos probar muchas veces el algoritmo Adaboost para conseguir ajustar sus parámetros de entrada y así obtener el mejor resultado.

# Pruebas:

Para generar las pruebas, he tenido problemas.

Al generar en un fichero los entrenamientos y después su respectiva ejecución para un tipo de imagen, el tiempo de ejecución es excesivo.

Solo he podido sacar algunos datos, entonces no he podido conseguir el mejor resultado.

Si pongo que genere 50 clasificadores débiles con 50 pruebas aleatorias para cada clasificador fuerte, el resultado es de un 75 % de error del total de todas las imágenes de la base de datos.