PRÁCTICA 2 SISTEMAS INTELIGENTES

ADABOOST

24 DE JUNIO DE 2020

GABRIEL DE LAMO DUTRA

45929352P

Contenido

Clasificador Débil

Un clasificador débil consiste en un tipo de clasificador, el cual su principal característica es que su tasa de acierto sea mayor del 50%

Los clasificadores están basados en un umbral o dirección.

A la hora de implementar el clasificador débil, crearemos una nueva clase, llamada "ClasificadorDebil"

```
*/
public class ClasificadorDebil {
    public int pixel;
    public int valor;
    public int direccion;
    public double confianza;
    public double error; //Colocamos el error inicial a 0
    int dimension = (28*28);

    //Constructor
    ClasificadorDebil(){
        // create instance of Random class
        Random rand = new Random();
        //Asignamos aleatoriamente el valor de pixel, valor y la dirección
        pixel =  rand.nextInt(dimension);
        //Asignamos aleatoriamente un valor a la variable valor
        valor = rand.nextInt(256); //[0,256]

        //Asignamos una dirección aleatoriamente entre 1 o -1
        direccion = rand.nextInt(2);
        if(direccion==1){
            direccion=1;
        }else{
            direccion=-1;
        }
    }
```

En la clase, creamos las variables que aparecen en la imagen, cada una se refiere a :

- Pixel (o coordenada): la posición dentro de la imagen
- Valor:
- dirección: el sentido de la comparación, si es 1 o -1
- confianza: Valor que determina el peso que toma en las decisiones
- error: tasa de error del clasificador débil.

- dimensión: Número total de pixeles que se encuentran en una imagen

Se generarán los valores al azar en el constructor clasificador Débil para luego utilizar el Adaboost .

Además, se creará una función para crear un clasificador débil bajo unas dimensiones en concreto :

```
    ClasificadorDebil generarClasifAzar(int dimesion){
        ClasificadorDebil patata = new ClasificadorDebil();
        patata.dimension=dimension;
        return patata;
    }
}
```

Clasificador Fuerte

Un clasificador fuerte es un conjunto de clasificadores débiles.

En este sentido, crearemos una clase “ClasificadorFuerte” la cual estará compuesta de Clasificadores Débiles

Una vez tengamos un Clasificador Fuerte , podremos etiquetar las imágenes que no tengamos idea de que son.

El código para un crear clasificador fuerte es el siguiente:

```
/**
 *
 * @author gabri
 */
public class ClasificadorFuerte {
    ArrayList<ClasificadorDebil> compounds = new ArrayList<>();

    ClasificadorFuerte(ArrayList<ClasificadorDebil>asd){
        this.compounds=asd;
    }
}
```

Además, añadiremos una función para obtener la confianza de la suma de todos los clasificadores débiles que componen un clasificador fuerte y poder decidir si una imagen es o no es el número que se indique

```
    public static int Byte2Unsigned(byte b) {
        return b & 0xFF;
    }

    ArrayList ObtenerSolucion(ArrayList img){
        ArrayList solucion= new ArrayList();
        //Comprobamos si el resultado del clasificador debil es correcto y actualizamos el error posteriormente
        for(int i =0 ; i<this.compounds.size(); i++){
            Imagen imgn = (Imagen) img.get(i);
            if(this.compounds.get(i).direccion == -1){
                if(Byte2Unsigned(imgn.getImageData()[this.compounds.get(i).pixel]) < this.compounds.get(i).valor){
                    solucion.add(1);
                    isSolucion += this.compounds.get(i).confianza;
                }else{
                    solucion.add(-1);
                    isSolucion -= this.compounds.get(i).confianza;
                }
            }

            if(this.compounds.get(i).direccion == 1){
                if(Byte2Unsigned(imgn.getImageData()[this.compounds.get(i).pixel]) > this.compounds.get(i).valor){
                    solucion.add(1);
                    isSolucion += this.compounds.get(i).confianza;
                }else{
                    solucion.add(-1);
                    isSolucion -= this.compounds.get(i).confianza;
                }
            }
        }
       return solucion;
    }
}
```

Adaboost

El algoritmo ADABOOST, o adaptative boosting es un sistema de aprendizaje adaptativo donde los clasificadores débiles y las muestras van siendo ligeramente modificadas en sus parámetros de error y pesos asignados para mejorar la eficacia.

El algoritmo Adaboost va a ir acompañado de un aprendizaje supervisado con unas muestras etiquetadas , que son las imágenes de las cuales sabemos el valor, para luego intentar clasificar imágenes sin clasificar.

Adaboost lo que permitirá será integrar las diferentes decisiones de los clasificadores débiles para crear tomas de decisiones más robustas y alejadas del azar que proporcionan individualmente los clasificadores débiles

Adaboost en si no es un clasificador, si no que depende de otros clasificadores , en estos casos los clasificadores débiles.

La clase Adaboost generará el algoritmo Adaboost, para ello tendremos :

```
//Función para guardar los valores reales de las imagenes
public ClasificadorFuerte AplicarPDF(ArrayList imagenes , ArrayList soluciones){
    ArrayList CDA = new ArrayList();
    double[]D;
    D = new double[imagenes.size()];
    double Z = 0.0; //Constante de normalización

    //Asignamos los pesos iniciales a cada una de las imagenes
    for(int i=0; i<imagenes.size(); i++){
        D[i]=(double)1/imagenes.size();
    }
```

Comenzamos creando un ArrayList que nos servirá para almacenar los diferentes clasificadores débiles que conformaran un clasificador fuerte, para ello, haremos uso posteriormente del constructor del clasificador fuerte.

Creamos un array D que contendrá el peso de cada una de las imágenes asociadas por su posición en el array, al principio, se inicializarán todas a:

- 1/Número de imágenes

```
        //Creamos un clasificador débil
        for(int j=0; j<clasificadoresD; j++){
            ClasificadorDebil buenCD = new ClasificadorDebil();
            ErrorClasificador(imagenes, D, soluciones, buenCD);

            for(int k=0; k<pruebas; k++){
                ClasificadorDebil rocky = new ClasificadorDebil();
                //Calculamos el Error del clasificador


                ErrorClasificador(imagenes, D, soluciones, rocky);
                System.out.println("buenCD.e=" + buenCD.error + " | rocky.e=" + rocky.error);
                if(buenCD.error > rocky.error){
                    if(rocky.error < 0.50){
                        buenCD = rocky;
                    }
                }
            }

            //Asignamos confianza con la formula
            buenCD.confianza = 1.0/2.0 * (double)((double)Math.log((double)(1.0 - buenCD.error )/buenCD.error) / (double)Math.log(2));

            //Modificamos los pesos de las imagenes
            for(int i=0;i<D.length;i++){
                if(AplicarClasifDebil((Imagen)imagenes.get(i), buenCD) == (int)soluciones.get(i)){
                    D[i] = (double)D[i] * Math.pow(Math.E, -1 * buenCD.confianza);
                }else{
                    D[i] = (double)D[i] * Math.pow(Math.E, buenCD.confianza);
                }
                Z += (double)D[i];
            }


            //Aplicamos Z para asegurarnos que la suma de todos los pesos sea igual a 1
            for(int i =0 ; i<D.length; i++){
               D[i] = D[i]/Z;
            }
            //Si podemos asegurar que el clasificador debil es mejor del 0.5, lo guardamos en el clasificador fuerte
            /*if(buenCD.error > 0.5){
                CDA.add(buenCD);
            }*/
            CDA.add(buenCD);
        }
        ClasificadorFuerte CF = new ClasificadorFuerte(CDA);
        return CF;
    }
}
```

A continuación, creamos un clasificador nuevo (buenCD) y obtenemos su error.

La función ErrorClasificador obtiene el error para un clasificador débil mediante las imágenes, el peso D, las soluciones (en este caso binarias para diferenciar entre un 0 o un 1 , se explicará en el main) y se le pasa el clasificador débil que corresponda, la función ErrorClasificador es la siguiente :

La función Error clasificador es la siguiente:

```
//Obtiene el Error de un clasificador debil
public void ErrorClasificador(ArrayList entreno, double pesos[], ArrayList IRLValue, ClasificadorDebil CD){
    ArrayList solucion = new ArrayList();

    //Comprobamos si el resultado del clasificador debil es correcto y actualizamos el error posteriormente
    for(int i =0 ; i<entreno.size(); i++){
        Imagen img = (Imagen) entreno.get(i);
        if(CD.direccion == -1){
            if(Byte2Unsigned(img.getImageData()[CD.pixel]) < CD.valor){
                solucion.add(1);
            }else{
                solucion.add(-1);
            }
        }

        else{
            if(Byte2Unsigned(img.getImageData()[CD.pixel]) > CD.valor){
                solucion.add(1);
            }else{
                solucion.add(-1);
            }
        }
    }

    //Actualizamos Error del Clasificador Debil
    double auxErr = 0;
    for(int i=0; i< IRLValue.size(); i++){
        if( solucion.get(i) != IRLValue.get(i) ){
            auxErr += (double)pesos[i];
        }
    }
    CD.error = auxErr;
}
```

Dentro de esta función, obtendremos el % de error de un clasificador débil, para ellos, iteraremos entre todas las imágenes y en base a la dirección que tome el clasificador débil, compararemos con el valor que se encuentra en la imagen y asignaremos 1 o -1 si la solución es correcta o incorrecta respectivamente para terminar asignando el error al clasificador débil en concreto.

Después de obtener el error de un clasificador débil, creamos otro y calculamos su error.

Si el error de este último fuese mayor que el primero, y, además , menor de 50%, requisito necesario para que sea un clasificador débil válido, procederíamos a calcular la confianza del clasificador y actualizar los pesos para cada una de las imágenes y posteriormente usar Z para normalizar el peso dentro del espacio de probabilidad que debe de ser 1.

Para actualizar los pesos, necesitamos saber si el clasificador ha acertado o no, para ello usamos la función AplicarClasifDebil:

```
}

//Comprobamos si el valor de un clasificador coincide o no con el valor de la imagen
public int AplicarClasifDebil(Imagen img, ClasificadorDebil CD){
    byte datosPixeles[] = img.getImageData();
    if(CD.direccion == -1){
        if(Byte2Unsigned(datosPixeles[CD.pixel])< CD.valor){
            return 1;
        }else{
            return -1;
        }
    }
    else{
         if(Byte2Unsigned(datosPixeles[CD.pixel]) > CD.valor){
             return 1;
         }else{
             return -1;
         }
    }
}
```

En ella, determinamos si hemos acertado o no devolviendo 1 o -1 y comparando con el vector de las soluciones en la posición en concreto para determinar si es o no es la imagen realmente.

Al acabar, añadiríamos el clasificador a CDA y procederíamos a crear un clasificador fuerte

Main del programa

En el main del programa, guardaremos las imágenes de la base de datos que coincide con un 0 y con un 1 para tener la posibilidad de error entre varios números

```
public static void main(String[] args) {

    //Cargador MNIST de SI
    MNISTLoader ml = new MNISTLoader();
    ml.loadDBFromPath("./mnist_1000");

    //Accedo a las imagenes de dígito 0
    ArrayList d0imgs = ml.getImageDatabaseForDigit(0);
    ArrayList d1imgs = ml.getImageDatabaseForDigit(1);



    Adaboost ada = new Adaboost();
    ArrayList entrenamiento = new ArrayList();
    ArrayList es0 = new ArrayList();


    //Almacenamos las imagenes y asignamos true a todas las imagenes.
    for(int i=0; i<d0imgs.size(); i++){
        Imagen img = (Imagen) d0imgs.get(i);
        entrenamiento.add(img);
        es0.add(1);
        Imagen img2 = (Imagen) d1imgs.get(i);
        entrenamiento.add(img2);
        es0.add(-1);
    }
```

Comenzaremos cargando desde la base de datos las imágenes, y almacenando en dos ArrayList todas las imágenes tanto de 0's como de 1's.

Crearemos una instancia de la clase Adaboost, crearemos un ArrayList para almacenar las imágenes que utilizaremos para entrenar a los clasificadores y otra para asignar el valor de una imagen y decidir si lo que devuelve un clasificador débil coincide con la imagen o no

A continuación, determinaremos a mano que es un 0 y que no es un 0 mediante el uso del vector de soluciones, colocando todas las imágenes que no son 0 a -1 y los 0's a 1

```
        //Iteramos para el número de pruebas del adaboost
        ClasificadorFuerte CF = ada.AplicarPDF(entrenamiento, es0);
        ArrayList isIt = CF.ObtenerSolucion(entrenamiento);
        //Siendo optimistas, si es 0 la confianza, pues vamos a pensar que lo es, una palmadita en la espalda
        int acierto = 0, total = 0;
        for(int i = 0; i < isIt.size(); i++)
        {
            if((double)isIt.get(i) > 0 && i % 2 != 0)
                acierto++;

            total++;
        }

        System.out.println("Porcentaje de acierto " + (double)acierto/total);
                MostrarImagen imgss = new MostrarImagen();
                imgss.setImage((Imagen)d0imgs.get(1));
                imgss.mostrar();
    }
}
```

A continuación, crearemos un Clasificador fuerte como el resultado de aplicar el algoritmo Adaboost bajo el número de clasificadores débiles y pruebas , pasando como parámetro las soluciones y las imágenes

```
int acierto = 0, total = 0;
for(int i = 0; i < isIt.size(); i++)
{
    if((double)isIt.get(i) > 0 && i % 2 != 0)
        acierto++;

    total++;
}
System.out.println("Porcentaje de acierto " + (double)acierto/total);
```

Por último, determinaríamos el % de aciertos y mostraríamos la imagen de la que se trata