PRÁCTICA 2 SISTEMAS INTELIGENTES

Adaboost

José Soler Martínez 48796440P

Índice

ADABOOST

1. Introducción

Detalles de diseño e implementación de la práctica.

El objetivo de esta práctica es implementar el algoritmo Adaboost para construir un sistema de aprendizaje y decir a qué tipo de imagen pertenece una imagen que le pasemos al programa, si es un abrigo, un bolso, una camiseta, un pantalón, un suéter, un vestido, unas zapatillas o unos zapatos.

Para el desarrollo de la práctica se ha utilizado el IDE Netbeans 8.2 junto a la versión 1.8 de javajdk.

Para esta práctica se facilita una plantilla la cual nos permite coger datos de una base de datos de imágenes y procesar imágenes de la propia base de datos para evaluarla.

2. Implementación

Mi implementación ha sido la siguiente:

He utilizado 3 clases, una clase ClasificadorDebil, otra ClasificadorFuerte y otra AdaBoost, las clases de los clasificadores los creé para tener una mejor organización de los datos.

CLASE CLASIFICADOR DÉBIL

En la clase ClasificadorDebil me encargo de generar el hiperplano de 28*28 dimensiones con valores entre -1 y 1 y luego genero el valor independiente del plano con número aleatorios entre 0 y 255 unidos a la ecuación del plano, todo eso guardado en un array llamado clasificador que será el hiperplano de ClasificadorDebil.

```
public void generarClasificadorAzar(int dimension){
    int[] pl = new int[dimension];
    int independiente = 0;
    // le damos valores
    for(int i=0; i < dimension; i++){
        pl[i] = new Random().nextInt(3)-1;
    }
    // calculamos el termino independiente
    for(int i = 0; i < dimension; i++){
        independiente = pl[i] * new Random().nextInt(256);
    }
    int[] hiperd = new int[dimension+1];
    for(int i=0; i < dimension; i++){
        hiperd[i] = pl[i];
    }
    hiperd[dimension] = independiente;
    clasificador = hiperd;
}
```

Para poder evaluar una imagen introducida con un hiperplano utilizo el método aplicarClasificadorDebil, este método devolverá un Array de booleanos indicando si los puntos de la imagen a evaluar se encuentra contenida en el hiperplano o no.

```
Boolean[] aplicarClasificadorDebil(int[] clasificador, ArrayList<Imagen> datos){
```

Le paso como parámetros un hiperplano, y un ArrayList de imágenes para entrenar, este método me devuelve un Array de Booleanos que indican si los puntos de la imagen se encuentran dentro del hiperplano o no y por tanto si la imagen que estamos evaluando pertenece al mismo tipo que el clasificador que estamos utilizando.

Se evalúan los puntos de la imagen mediante la ecuación del plano, si se encuentra contenido en el plano o por encima ese punto corresponde con el tipo según ese clasificador débil, en caso contrario no pertenecería.

Así se calcula si los puntos de la imagen se encuentran contenidos en el plano.

```
for(Imagen aux2: datos){
    aux1 = 0;
    aux3 = aux2.getImageData();
    for(int i=0; i < clasificador.length-1; i++){
        aux1 = aux1 + clasificador[i] * (aux3[i] & 0xff);
    }
    aux1 = aux1 - clasificador[clasificador.length-1];
    if(aux1 >= 0){
        evaluar[x] = true;
    }else{
        evaluar[x] = false;
    }
```

El error de un clasificador lo calculo con otro método dentro de ClasificadorDebil, en el que le paso como parámetros un hiperplano, los datos correctos que debería darme el clasificador en el caso de que fuera todo perfecto, un ArrayList de imágenes y otro Array con el peso de cada uno de los puntos, en el caso de que los datos correctos no coincidan con los datos obtenidos al aplicar el clasificador débil se aumentará el error del clasificador.

```
double obtenerErrorClasificador(int[] clas, Boolean[] datos_correctos, ArrayList<Imagen> datos, double[] D){
    double e = 0;
    Boolean[] vector = aplicarClasificadorDebil(clas, datos);
    for(int i=0; i < datos_correctos.length;i++){
        if(!vector[i].equals(datos_correctos[i])){
            e += D[i];
        }
    }
    error = e;
    return e;
}
```

CLASE CLASIFICADOR FUERTE

En la clase ClasificadorFuerte creamos un clasificador fuerte, esto se realiza mediante un conjunto de clasificadores débiles agrupados.

Encontramos un método llamado ClasificadorFuerte, al cual se le pasa el número de hiperplanos que deseas generar para probar, el número de clasificadores débiles de los cuales quieres que se componga tu clasificador fuerte, un array con los datos correctos al analizar una imagen y un array de imágenes.

```
public ArrayList ClasificadorFuerte(int planos, int numClasificadores, Boolean[] datos_correctos, ArrayList<Imagen> datos){
```

En este método inicializamos los pesos de los clasificadores débiles.

```
for(int j=0; j < datos.size(); j++){
    D[j] = 1.0/datos.size();
}
```

Luego buscamos en un bucle el mejor clasificador que podemos encontrar entre los generados. El mejor clasificador será aquel que tenga menor error en la clasificación.

```
for(int i=0; i < planos; i++){
    ClasificadorDebil clasif = new ClasificadorDebil();

    clasif.generarClasificadorAzar(dimension);
    int[] clas = clasif.getClasificador();
    boolClas = clasif.aplicarClasificadorDebil(clasif.getClasificador(), datos);
    auxerror = clasif.obtenerErrorClasificador(clasif.getClasificador(), datos_correctos, datos, D);
    if(auxerror < mejorDebil){
        deb.setClasificador(clas);
        deb.setResultados(boolClas);
        deb.setError(auxerror);
        mejorDebil = auxerror;
    }
}
deb.setConfianza(0.5 * (Math.log10((1 - deb.getError())/deb.getError()))/Math.log10(2));
```

Calculamos la confianza y se la añadimos al mejor clasificador generado con la siguiente fórmula.

$$\alpha_t = 1/2 \log_2 \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$$

Tras esto, se actualizan los pesos de los clasificado

```
// se actualizan los pesos
double Z = 0;
double[] A = new double[D.length];
for(int i=0; i < D.length; i++){
    if(datos_correctos[i].equals(deb.getResultados()[i])){
        A[i] = Math.pow(Math.E, -deb.getConfianza());
    }else{
        A[i] = Math.pow(Math.E, deb.getConfianza());
    }
}
for(int i=0; i < D.length; i++){
    D[i] = D[i] * A[i];
}
for(int i=0; i < D.length;i++){
    Z = Z + D[i];
}
for(int i=0; i < D.length; i++){
    D[i] = D[i]/Z;
}
// Fin actualizar pesos
```

Y se añade dicho clasificador a un ArrayList de clasificadores débiles, el cual será devuelto. Este ArrayList será el que formará uno de los clasificadores fuertes.

A parte de generar los clasificadores fuertes, en la misma clase también se ejecutan las pruebas de los clasificadores fuertes. La ejecución se realiza con el método ejecutarFuerte, al cual se le pasa un ArrayList de clasificadores fuertes y una imagen a evaluar.

```
public void ejecutarFuerte(ArrayList<ClasificadorFuerte> fuertes, Imagen img){
```

Este método evalúa uno por uno los clasificadores débiles que componen cada clasificador fuerte con la imagen a evaluar.

```
for(int i=0; i < fuertes.size(); i++){
    ArrayList<ClasificadorDebil> arrayDebiles = new ArrayList<>();
    arrayDebiles = fuertes.get(i).getDebiles();
    Boolean[] booldebil = new Boolean[arrayDebiles.size()];
    Double[] resultdebil = new Double[arrayDebiles.size()];
    double acumulador = 0;
    for(int j=0; j < arrayDebiles.size(); j++){
        ClasificadorDebil d = new ClasificadorDebil();
        d = arrayDebiles.get(j);
        Boolean[] uno = d.aplicarClasificadorDebil(d.getClasificador(), im);
        booldebil[j] = uno[0];
        resultdebil[j] = d.getConfianza();
```

Como factor de confianza para obtener el resultado de qué tipo es una imagen he decidido utilizar el clasificador fuerte que mayor confianza tenga entre todos sus clasificadores débiles, de ese modo creo que se puede obtener el mejor resultado.

A razón de los datos obtenidos, si el hiperplano que evalúo contiene a la imagen se sumará la confianza de dicho clasificador, en caso contrario, se restará.

```
if(booldebil[j]){
    acumulador += resultdebil[j];
}else{
    acumulador += -resultdebil[j];
}
```

Se van acumulando las confianzas de dichos clasificadores débiles, de tal forma que el clasificador fuerte que más confianza tenga en total al final decidirá el tipo de imagen que es.

Así busco qué clasificador fuerte tiene mayor fiabilidad.

```
for(int i=1; i < arrayAcumuladorFuerte.length; i++){
    if(mejor < arrayAcumuladorFuerte[i]){
        mejor = arrayAcumuladorFuerte[i];
        pos = i;
    }
}
```

Tras esta comprobación el tipo del vencedor se obtiene de la siguiente forma:

```
if(pos == 0){
    System.out.println("Abrigos");
}else if(pos == 1){
    System.out.println("Bolsos");
}else if(pos == 2){
    System.out.println("Camisetas");
}else if(pos == 3){
    System.out.println("Pantalones");
}else if(pos == 4){
    System.out.println("Sueters");
}else if(pos == 5){
    System.out.println("Vestidos");
}else if(pos == 6){
    System.out.println("Zapatillas");
}else{
    System.out.println("Zapatos");
}
```

CLASE ADABOOST

En esta clase se ejecutará el entrenamiento de Adaboost, junto a la generación del fichero de pruebas y la ejecución del algoritmo para determinar el tipo de una imagen.

El entrenamiento de Adaboost se lleva a cabo cogiendo un conjunto de imágenes de prueba de una base de datos, un número definido de imágenes, en este caso se han utilizado 500 imágenes de las cuales para hacer una prueba válida se han utilizado la mitad de imágenes pertenecientes al clasificador fuerte que queremos entrenar y la otra mitad de imágenes que no pertenecen a dicho clasificador.

Los datos se obtienen de la base de datos del siguiente modo:

```
DBLoader ml = new DBLoader();
ml.loadDBFromPath("./db");
```

Dentro de un bucle for de 8 iteraciones generamos el conjunto de imágenes que vamos a utilizar.

Si la imagen es de la primera mitad cogemos el clasificador que incluya el tipo de imagen a evaluar y añadimos a datos_correctos el valor true como que ese clasificador pertenece a ese tipo de imagen.

```
if(j < tamano/2){
    if(i == 0){
        img = (Imagen) chaquetas.get(j);
    }else if(i == 1){
        img = (Imagen) bolsos.get(j);
    }else if(i == 2){
        img = (Imagen) camisetas.get(j);
    }else if(i == 3){
        img = (Imagen) pantalones.get(j);
    }else if(i == 4){
        img = (Imagen) sueters.get(j);
    }else if(i == 5){
        img = (Imagen) vestidos.get(j);
    }else if(i == 6){
        img = (Imagen) zapatillas.get(j);
    }else{
        img = (Imagen) zapatos.get(j);
    }
    correcto = true;
```

En caso contrario al llegar a la mitad de imágenes haremos lo mismo pero eligiendo otra imagen que no sea del mismo tipo, en el caso de que de forma aleatoria aparezca el mismo tipo que queremos evaluar, este se cambiará hasta que no coincida y se añadirá a datos_correctos como false.

```
else{
    int coger = j - (tamano/2);
    int aleatorio = new Random().nextInt(8);
    while(aleatorio == i){
        aleatorio = new Random().nextInt(8);
    }
    if(aleatorio == 0){
        img = (Imagen) chaquetas.get(coger);
    }else if(aleatorio == 1){
        img = (Imagen) bolsos.get(coger);
    }else if(aleatorio == 2){
        img = (Imagen) camisetas.get(coger);
    }else if(aleatorio == 3){
        img = (Imagen) pantalones.get(coger);
    }else if(aleatorio == 4){
        img = (Imagen) sueters.get(coger);
    }else if(aleatorio == 5){
        img = (Imagen) vestidos.get(coger);
    }else if(aleatorio == 6){
        img = (Imagen) zapatillas.get(coger);
    }else{
        img = (Imagen) zapatos.get(coger);
    }
    correcto = false;
```

Todas estas imágenes se añadirán a un ArrayList de imágenes y serán evaluadas con la función ClasificadorFuerte explicada en el apartado anterior.

```
ClasificadorFuerte fuerte = new ClasificadorFuerte();
fuerte.ClasificadorFuerte(20, 200, datos_correctos, fotos);
```

Añadimos el clasificador fuerte generado a un array de clasificadores fuertes, y pasamos a generar el archivo con todos los datos guardados.

```
    arrayFuertes.add(fuerte);
}
generarFichero(arrayFuertes, fichero);
```

Para generar el fichero con todos los datos de los clasificadores fuertes se utiliza el siguiente método:

```
void generarFichero(ArrayList<ClasificadorFuerte> fuertes, String fichero){
```

Se le pasa un ArrayList de clasificadores fuertes y el nombre del fichero a generar.

El fichero a generar tendrá la siguiente estructura:

Hiperplano 1
Confianza 1
Hiperplano 2
Confianza 2
…

Así tantos hiperplanos como haya en los 8 clasificadores fuertes.

Los datos se incluyen recorriendo cada uno de los clasificadores fuertes y poniendo cada uno de los hiperplanos que lo componen y sus confianzas.

```
for(int i=0; i < fuertes.size(); i++){
    ClasificadorFuerte cfuerte = fuertes.get(i);
    for(int j=0; j < cfuerte.getDebiles().size() ; j++){
        ClasificadorDebil deb = cfuerte.getDebiles().get(j);
        int[] clas = deb.getClasificador();
        for(int z=0; z < clas.length; z++){
            linea += clas[z] + " ";
        }
        escribir.write(linea);
        linea = "\n";
        linea += deb.getConfianza();
        escribir.write(linea);
        escribir.write("\n");
        linea = "";
    }
}
```

Para ejecutar Adaboost se utiliza la función leerFichero, al cual se le pasa el nombre del fichero a leer y la imagen para buscar su tipo.

```
public void leerFichero(String fichero, Imagen img){
```

Primero el archivo se lee una vez para contar el número de líneas que hay, así dividiendo entre 8 se sabe el número de líneas que pertenecen a cada clasificador.

```
while ((linea = bf.readLine())!= null) {
    nlineas++;
}
nlineas = nlineas/8;
```

Para leer cada uno de los clasificadores débiles se utiliza la función Split para dividir un string en un array de strings y luego se pasa a int para poder utilizarlo como hiperplano.

```
ClasificadorDebil deb = new ClasificadorDebil();
String[] aux = linea.split(" ");
for(int j = 0; j < aux.length; j++){
    c[j] = Integer.parseInt(aux[j]);
}
deb.setClasificador(c);
```

Se lee la línea siguiente y se guarda la confianza.

```
linea = bf.readLine();
deb.setConfianza(Double.parseDouble(linea));
cdebil.add(deb);
```

Una vez se han leído tantas líneas como pertenecen a un clasificador fuerte, este clasificador se guarda y se sigue con el siguiente.

```
if(i % nlineas == (nlineas-1) && i != 0){
   f.setDebiles(cdebil);
   clasfuerte.add(f);
   f = new ClasificadorFuerte();
   cdebil = new ArrayList<>();

}
```

Cada uno de los clasificadores fuertes se guardará en un array de clasificadores fuertes.

Finalmente se ejecuta la función ejecutarFuerte, explicada anteriormente y se pasa ala ejecución del algoritmo adaboost, pasando como parámetro el ArrayList de clasificadores fuertes y la imagen a evaluar.

3. Cuestiones adicionales

Algunas de las cuestiones que se encuentran en el pdf de la práctica han sido desarrolladas durante la explicación de Adaboost.

¿Cuál es el número de clasificadores (T) que se han de generar para que un clasificador débil funcione?

- El número de clasificadores (T) necesarios para que un clasificador débil funcione dependen del tipo de imagen a evaluar, pero con unos 50 seguro que se encuentra un clasificador débil que sea útil y que englobe una buena nube de puntos de la imagen.

¿Cómo afecta el número de clasificadores generados y su entrenamiento al tiempo empleado para el proceso de aprendizaje? ¿Qué importancia le darías?

- El número de clasificadores generados influye mucho, a mayor cantidad de clasificadores débiles mayor es el tiempo que tarda en generar cada uno de ellos y en evaluarlos, es lógico, se tarda menos en evaluar 2 clasificadores fuertes con dos clasificadores débiles cada uno (4 clasificadores en total) que 4 clasificadores fuertes con 4 clasificadores débiles cada uno (16 clasificadores en total).
  A mi parecer es muy importante que exista un número de planos considerable, porque a mayor número de clasificadores débiles, más posible es que se adapte a la imagen a evaluar.