Sistemas Inteligentes

Practica 2. Visión Artificial y Aprendizaje

Ernesto Canales Pereña
74444845D

Contenido

Ernesto Canales Pereña
74444845D

Introducción

Objetivo de la práctica

En esta segunda práctica de la asignatura se ha implementado un sistema capaz de reconocer un conjunto de imágenes, en este caso 10 tipos de imágenes. El sistema debe ser capaz de poder distinguir entre imágenes de de distintos dígitos (0, 1, 2, 3…)

El objetivo es que un clasificador (que nosotros debemos de implementar), que tras ser entrenado, sea capaz de decirnos de qué clase es cada imagen que le pasamos.

Escenario

De igual forma que en la primera práctica, se ha utilizado en IDE Netbeans, usando el lenguaje Java.

El proyecto está compuesto principalmente por el paquete pkg1820_p2si, el cual engloba las clases MNISTLoader.java, Imagen.java, MostrarImagen.java y Main.java, además se han añadido dos clases para la implementación del clasificador, que son ClaseT.jav y Adaboost.java (más adelante se detallan de qué están formadas estas dos últimas clases).

La batería de imágenes que se utilizará para entrenar y poner a prueba el clasificador se encuentra dentro de la carpeta mnist_1000, podemos encontrar otras 10 carpetas , en cada una de ellas encontramos las imágenes clasificadas por categoría.

¿Que es ADABOOST?

Se trata de un algoritmo de aprendizaje automático formulado por Yoav Freund y Robert Schapire. Puede ser usado junto con otros tipos de algoritmos de aprendizaje para mejorar el rendimiento de estos. La salida de los otros algoritmos de aprendizaje (clasificadores débiles) se combinan en una suma ponderada que representa la salida final del clasificador potencial (clasificador fuerte).

El algoritmo es adaptable, en el sentido de que los clasificadores débiles subsiguientes se ajustan a favor de aquellos casos clasificados erróneamente por clasificadores anteriores.

Clasificador débil

Nuestro clase clasificador débil estará compuesto de la siguiente forma:

Ernesto Canales Pereña
74444845D

Sistemas Inteligentes
2019-20204

```
class ClaseT{
        public int pixel;
        public int umbral;
        public int direccion;
        public double alfa;

        public ClaseT(){
            this.pixel = (int)(Math.random()*784);
            this.umbral = (int)(Math.random()*255);

            while(this.umbral == 127){
              this.umbral = (int)(Math.random()*255);
            }
            this.direccion = (int)(Math.random()*785);

            if(this.direccion%2 == 0){
                this.direccion = 1;
            }
            else{
                this.direccion = -1;
            }

            this.alfa = 0;
        }

        public ClaseT(ClaseT t){
            this.pixel = t.pixel;
            this.umbral = t.umbral;
            this.direccion = t.direccion;
            this.alfa = t.alfa;
        }
```

Un campo pixel que se generara al azar entre 0-783, umbral de 0 a 254 y una dirección. Tambien contendra el atributo alfa para utilizarlo en el algoritmo de ADABOOST.

Para comprobar si un clasificador clasifica bien una imagen tendremos que pasarles los pixel de cualquier imagen y ver si acierta o no. Para ello tenemos los siguientes métodos:

```
public int esIgual(Imagen i){

    byte imageD[] = i.getImageData();
    imageD[this.pixel] = (byte)(255 - imageD[this.pixel]);
    int umbralI = (int)(Byte2Unsigned(imageD[this.pixel]));
    if(this.direccion == 1){
        if(umbralI > this.umbral){
            return 1;
        }
    }
    else{
        if(umbralI <= this.umbral){
            return 1;
        }
    }
    return -1;
}

public int esIgualI(int imagen[]){

    int umbralI = imagen[this.pixel];
    if(this.direccion == 1){
        if(umbralI > this.umbral){
            return 1;
        }
    }
    else{
        if(umbralI <= this.umbral){
            return 1;
        }
    }
    return -1;
}
```

ADABOOST

Algorithm 1 Adaboost

procedure ADABOOST(X, Y)
  $D_1(i) = 1/N$ ▷ Indica como de difcil es de clasificar cada punto i
  for $t = 1 \rightarrow T$ **do** ▷ T es el nmero de clasificadores dbiles a usar
    Entrenar h_t teniendo en cuenta D_t
    Start
      for $k = 1 \rightarrow A$ **do** ▷ A = num. de pruebas aleatorias
        F_p = generaAlAzar()
        $\epsilon_t = P_{D_t}(h_t(x_i) \neq y_i) \rightarrow \epsilon_{t,k} = \sum_{i=1}^{N} D_t(i) \cdot (F_k(x) \neq y(x))$
      return $< F_p | \min(\epsilon_{t,k}) >$
    End
    Del h_t anterior obtener su valor de confianza $\alpha_t \in \mathbb{R}$
    Start
      $\alpha_t = 1/2 \log_2 \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
    End
    Actualizar D_{t+1}
    Start
      $D_{t+1} = \frac{D_t(i) \cdot e^{-\alpha_t \cdot y_i h_t(x_i)}}{Z_t}$
      $Z_t = \sum_i D_t(i)$
    End
  return $H(x) = sign(\sum_t \alpha_t \cdot h_t(x))$

Para aplicar AdaBoost, se debe generar un conjunto de clasificadores débiles y de ese conjunto elegir el que menos error tiene al clasificar un conjunto de imágenes similares para todos los clasificadores.

Ernesto Canales Pereña
74444845D

```
//Clasificadores debiles
static int T = 250;

//Pruebas Aleatorias
static int A = 600;
```

Aquí, determino la cantidad de clasificadores débiles a usar y las pruebas aleatorias. Hay que tener en cuenta, que si empleamos demasiadas pruebas aleatorias y clasificadores débiles corremos el peligro de sobreentrenamiento.

```
public void Adaboost(ArrayList<Imagen> X, int [] Y, String cf){
    double[] D = new double[X.size()];
    for(int i=0; i< X.size(); i++){
        double d = (double)1/X.size();
        D[i] = d;
    }
```

Creo un vector de pesos con tamaño de las imagenes con las que vamos a entrenar.

```
for(int t=0;t<Adaboost.T;t++){
    double errormin = 1.0;
    ClaseT seleccion = null;
    for(int k=0;k<Adaboost.A;k++){
        double errorv = 0.0;
        ClaseT cd = new ClaseT();

        for(int i=0;i<D.length;i++){
            int hxy = 0;
            if(cd.esIgual(X.get(i)) != Y[i]){
                errorv += D[i];
            }
        }
        if(errorv < errormin) {
            seleccion = new ClaseT(cd);
            errormin=errorv;
        }
    }
    double alf = (double)(0.5*(Math.log((1-errormin)/errormin)));
    if(alf == Double.parseDouble("Infinity")){
        break;
    }
    seleccion.alfa = alf;
    escribir(cf, seleccion);
```

Para cada clasificador hago las pruebas aleatorias que se establecieron antes, y cálculo en cada prueba aleatoria para cada imagen si el clasificador que hemos creado al azar clasifica correctamente a la imagen, si no aumentariamos el error. Una vez hecho todas las pruebas nos quedamos con el mejor clasificador, calculamos alfa y lo guardamos en un fichero texto.

```
double Z = 0.0;
for(int i=0;i<D.length;i++){
    D[i] = (double)D[i]*(Math.pow(Math.E,-seleccion.alfa*(Y[i]*seleccion.esIgual(X.get(i)))))
    Z += (double)D[i];
}

for(int i=0;i<D.length;i++){
    D[i] /= Z;
}
if(errormin == 0.0){
    break;
```

Luego tendremos que actualizar el vector de pesos para la siguiente iteración. Un ejemplo de uso es el siguiente:

CD : 1 1 -1 -1

Y : -1 1 -1 1

ERR [1 0 0 1]

D : [0.5 0.2 0.1 0.4]

Tasa de error = (0.5*1 + 0.2*0 + 0.1*0 + 0.4*1)

En resumen, se debe generar un conjunto de clasificadores débiles y de ese conjunto elegir el que menos error tiene al clasificar un conjunto de imágenes similares para todos los clasificadores.

Este conjunto sirve para entrenar a los clasificadores con el fin de que se vayan adaptando a un conjunto de imágenes de una de las clases en concreto. Habrá que emplear un porcentaje de imágenes de la base de datos para entrenar, y el restante para test. En mi caso empleo un 70% para entrenar y un 30% para test. Obtengo la cantidad de imágenes que hay en una carpeta donde se encuentra por ejemplo el 0 y lo multiplico por 0,7 para saber hasta que cantidad me guardo en el X[] para entrenar. Todo esto por categoría.

```
public static void sacarCF(String cf){

    MNISTLoader ml = new MNISTLoader();
    ml.loadDBFromPath("./mnist_1000");

    ArrayList<Imagen> X = new ArrayList<>();
    int rangos[][] = new int[10][10]; int numIm[] = new int[10];
    int aux = 0; int total = 0;

    for(int i=0; i<10; i++){
       ArrayList d0imgs = ml.getImageDatabaseForDigit(i);
       rangos[i][0] = aux;
       rangos[i][1] = (int)(d0imgs.size()*entrenamiento) + aux;
       aux = rangos[i][1];
       int max = (int)(d0imgs.size()*entrenamiento);
        for(int j=0; j<max; j++){
            X.add((Imagen) d0imgs.get(i));
            total++;
            numIm[i]++;
        }
    }
    gym(X, rangos, total, cf);
```

Más tarde lo entrenare con un orden primero entrenare los de la categoría 0 luego los del 1 por lo tanto en el array Y habrá que almacenar bien que categoría pertenece cada imagen correspondiente del array X.

```
public static void gym(ArrayList<Imagen> X, int rangos [][], int total, String cf){
    for(int s = 0; s<10; s++){
        int Y[] = new int[total];
        for(int i=0; i<total; i++){
            Y[i] = -1;
        }
        for(int i=rangos[s][0]; i<rangos[s][1]; i++){
            Y[i] = 1;
        }
        Adaboost ada = new Adaboost();
```

Como resultado tendremos un fichero con los clasficadores con menor error para cada clase.

```
434 1 -1 0.8464467963519374
522 244 1 0.7759636284287459
463 110 1 1.0272171057102017
462 45 -1 1.679441526896386
348 204 1 1.493315248422795
462 11 -1 3.384830193154732
434 14 -1 8.257353468616337
517 208 1 5.15559626408623
328 33 -1 11.489844273668808
344 89 1 20.050186968558027
434 217 1 28.172284211924712
518 65 -1 41.480488693822096
290 40 1 44.77427591166807
519 156 -1 76.62445139535625
517 54 -1 86.58056400782654
517 18 -1 86.58188693028926
427 187 1 115.97649234184325
344 87 -1 202.44155743792868
374 110 -1 57.458343561879786
545 173 1 238.0199668625328
326 247 1 164.7245815526937
489 201 1 342.7068340912721
```

Una vez hecho esto, empezaremos con los test, cogiendo desde la última imagen de cada categoría que hayamos guardado en X hasta el final de la cantidad de imágenes de esa categoría. Las guardaremos en otro array, en el que pasaremos a cada imagen los clasificadores obtenidos anteriormente.

```
///tests
Adaboost ada = new Adaboost(); aux=0; int cantidadPD=0;
int rangos2[][] = new int[10][10];
ArrayList<Imagen> testA = new ArrayList<>();
for(int i=0; i<10; i++){
    ArrayList d0imgs = ml.getImageDatabaseForDigit(i);
    rangos2[i][0] = aux;
    for(int j=numIm[i]+1; j<d0imgs.size(); j++){
        testA.add((Imagen)d0imgs.get(i));
        cantidadPD++;
    }
    rangos2[i][1] = cantidadPD + aux;
    aux = rangos2[i][1];
    cantidadPD=0;
}
```

Guardamos imágenes para los test…

```
int acertados=0; int totalA=0; double suma=0.0;
for(int i=0; i<10; i++){
    int Y[] = new int[testA.size()];
    Arrays.fill(Y, -1);
    for(int s=rangos2[i][0]; s<rangos2[i][1]; s++){
        Y[s]=1;
    }
    for(int j=0; j<testA.size()-1; j++){
        suma=ada.AplicarClasificador(testA.get(j), cf, i, Y[j]);
        double tope = 0.0;
        if(suma > tope){
            acertados++; totalA++;
        }
    }
    double porcentaje = (double)acertados/testA.size();
    System.out.println("Porcetanje de aciertos cat " + i + " " + porcentaje*100 + "%. Num= " + ac
    acertados=0;
}
```

Y aquí,en el método AplicarClasificador irá aplicando, a cada imagen de los test, los diferentes clasificadores para cada clase, para más tarde hacer el sumatorio de la multiplicación de alfa con el signo de si pertenece o no a la clase dicha (H(x)) . Si devuelve

que es mayor que 0 querra decir que habra acertado dicho clasificador.

```
public double AplicarClasificador(Imagen imagen, String cf, int cat, int Y){
  File archivo = null;
  FileReader fr = null;
  BufferedReader br = null;
  double suma = 0.0;

  try {
     // Apertura del fichero y creacion de BufferedReader para poder
     // hacer una lectura comoda (disponer del metodo readLine()).
     archivo = new File ("./" + cf);
     fr = new FileReader (archivo);
     br = new BufferedReader(fr);

     String linea = br.readLine(); String fin = "**********";

     for(int i=0; i<cat; i++){
         while(!linea.equals(fin)){
             linea = br.readLine();
         }
         linea = br.readLine();
     }
     if(cat != 0){
        linea = br.readLine();
     }
     while(!linea.equals(fin)){
         String auxS[] = linea.split(" ");
         ClaseT auxC = new ClaseT(Integer.parseInt(auxS[0]), Integer.parseInt(auxS[1]), Integer.parse
         suma += (double)auxC.alfa*auxC.esIgual(imagen);
         linea = br.readLine();
     }
```

$$H(x) = sign(\sum_t \alpha_t \cdot h_t(x))$$

Resultados

En cuanto a los resultados obtenidos, he escogido una imagen de internet, externa a la base de datos y la he añadido al directorio para observar si llega a clasificar bien el dígito.

Pero primero, tendremos que cargar la ruta de la imagen invertirla y como hemos realizado antes, habrá que obtener H(X) y quedarnos con el valor más alto. Es decir, para cada clasificador de cada clase comprobamos en la imagen y obtenemos H(x).

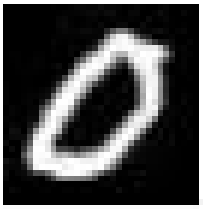
```
Imagen img = new Imagen();
img.loadFromPath(ruta);
int imagen[] = new int[784];

byte imageD[] = img.getImageData();
for (int i = 0; i < imageD.length; i++){

    imageD[i] = (byte) (255 - imageD[i]);
    int aux = Byte2Unsigned(imageD[i]);
    imagen[i] = aux;
}
System.out.print("\n");
Adaboost ada = new Adaboost();
double x[] = new double[10];
for(int i=0; i<10; i++){
    x[i] = ada.AplicarClasificadorI(imagen, cf, i);
}
double mayor = x[0];
int pos=0;
for(int i=1; i<10; i++){
    if(x[i] > mayor){
        pos = i;
        mayor = x[i];
    }
}
```

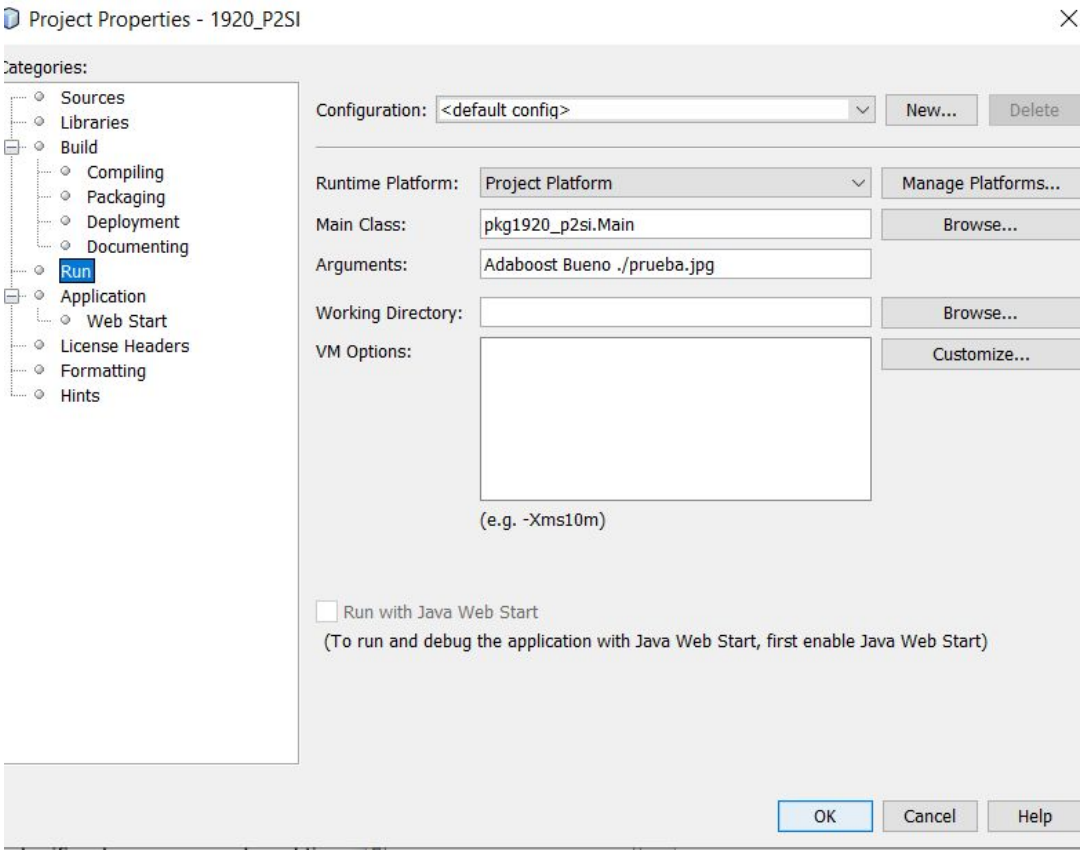
```
        while(!linea.equals(fin)){
            String auxS[] = linea.split(" ");
            ClaseT auxC = new ClaseT(Integer.parseInt(auxS[0]), Integer.parseInt(auxS[1]), Integer.parse
            double aux = (double)auxC.alfa*auxC.esIgualI(imagen);
            suma = suma + aux;
            linea = br.readLine();
        }
    }catch(Exception e){
    e.printStackTrace();
    }finally{
        // En el finally cerramos el fichero, para asegurarnos
        // que se cierra tanto si todo va bien como si salta
        // una excepcion.
        try{
            if( null != fr ){
                fr.close();
            }
                return suma;
        }catch (Exception e2){
            e2.printStackTrace();
        }
    }
    return 0;
}
```

En el entrenamiento del clasificador fuerte, hemos conseguido los siguientes porcentajes para T=250 y A=650.

```
Porcetanje de aciertos cat 0 33.898305084745764%. Num= 100
Porcetanje de aciertos cat 1 21.355932203389834%. Num= 63
Porcetanje de aciertos cat 2 33.898305084745764%. Num= 100
Porcetanje de aciertos cat 3 34.57627118644068%. Num= 102
Porcetanje de aciertos cat 4 31.186440677966104%. Num= 92
Porcetanje de aciertos cat 5 18.644067796610174%. Num= 55
Porcetanje de aciertos cat 6 46.779661016949156%. Num= 138
Porcetanje de aciertos cat 7 10.508474576271185%. Num= 31
Porcetanje de aciertos cat 8 34.57627118644068%. Num= 102
Porcetanje de aciertos cat 9 45.084745762711864%. Num= 133
```

Observamos que no son porcentajes muy altos pero puede clasificar algunos dígitos. El digito a clasificar es el siguiente:





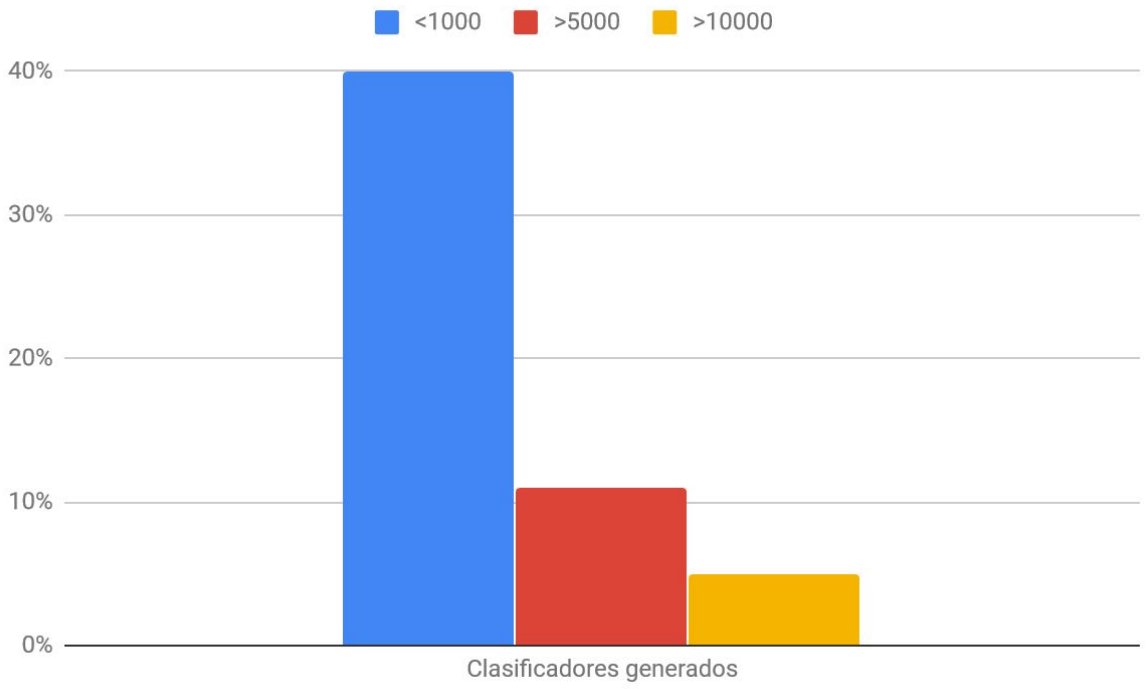

```
run:

Es el digito 0
BUILD SUCCESSFUL (total time: 0 seconds)
```

Vemos que lo clasifica correctamente.

Preguntas Propuesta

¿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.

Como hemos nonbrado antes, no habra que generar demasiados clasifiadores debiles ni demasiadas pruebas aleatorias…



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

He dividido cada clase de la base de datos en un 70% y el restante para los test. Es útil porque si comprobaramos toda la

base de datos lo único que hariamos seria memorizar las imágenes.