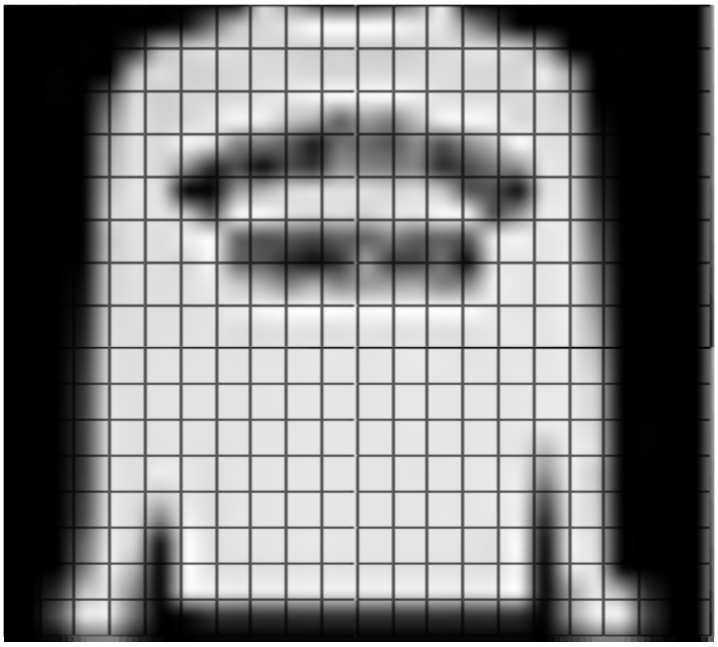
SISTEMAS INTELIGENTES

CURSO 2018-19

PRACTICA 2



VISION ARTIFICIAL Y APRENDIZAJE

IVÁN MAÑÚS MURCIA - 48729799K
GRUPO JUEVES 9:00-11:00
UNIVERSIDAD DE ALICANTE
GRADO EN INGENIERÍA INFORMÁTICA

Contenido

Documentación de la práctica

1. ¿Qué es Adaboost?

- El termino boosting hace referencia a un tipo de algoritmos cuya finalidad es encontrar una hipótesis fuerte a partir de utilizar hipótesis simples y débiles. Durante este trabajo se hablará del algoritmo AdaBoost, el cual fue creado por Freund y Schapire y es un diseño mejorado del boosting original; y de sus diferentes variantes [Freund and Schapire E., 1996].

AdaBoost es una contracción de “Adaptive Boosting”, en donde el termino Adaptive hace alusión a su principal diferencia con su predecesor. En términos de funcionalidad son iguales, ambos algoritmos buscan crear un clasificador fuerte cuya base sea la combinación lineal de clasificadores “débiles simples” ht (x). Sin embargo, AdaBoost propone entrenar una serie de clasificadores débiles de manera iterativa, de modo que cada nuevo clasificador o “weak learner” se enfoque en los datos que fueron erróneamente clasificados por su predecesor, de esta manera el algoritmo se adapta y logra obtener mejores resultados.

En este punto cabe aclarar que el tipo de AdaBoost sobre el que se trabajó en esta investigación es de píxeles; esto significa que sólo se trabaja con umbrales y píxeles cuales se representan con un número del 1-784(pixel) y un número que indica el umbral de 0-255, por lo tanto, el resultado se expresa como” pertenece a x clase/no pertenece a x clase”.

Una forma para entender este algoritmo es mediante el ejemplo proporcionado por sus creadores, donde se considera a una persona experta en apuestas de carreras de caballos, dicho personaje puede tener varias estrategias para determinar al posible ganador en una carrera específica, por ejemplo, basándose en mayor número de carreras ganadas, mejor tiempo en dar una vuelta, el caballo con mayor número de apuestas, etc. Sin embargo, para obtener una hipótesis que permita predecir con mayor seguridad el resultado de cualquier carrera, se necesita discernir cuales carreras aportan mayor información.

2. ¿Cómo trabaja Adaboost?

Para lograr esto, primero se necesita de un “weak learner”, este es un algoritmo que nos ayuda a encontrar las hipótesis débiles.

En esta práctica el weak learner estará compuesto por 3 números, uno corresponde al número de píxel, otro al umbral de grises, y otro a una dirección (> o <), además de un valor de confianza.

```
this.pixel = (int)(Math.random()*784);
this.umbral = (int)(Math.random()*255);
this.direccion = (int)(Math.random()*784);

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

this.alfa = 0;
```

Como podemos ver, alfa es su valor de confianza, que al crear el weak learner, será 0, y lo actualizaremos más tarde.

A todos los datos se les asigna inicialmente el mismo peso de (1/N), este peso se va actualizando con cada iteración según los ejemplos mal clasificados, de esta manera se busca minimizar el error esperado y enfocarse en clasificar correctamente los datos que ahora tienen mayor peso.

```
//INICIALIZACIÓN DEL VECTOR DE PESOS
double[] D = new double[X.size()];
Arrays.fill(D,(double)1/X.size());
```

El método boosting llama t veces al weak learner, cada vez dándole una distribución diferente de los datos para el entrenamiento.

Pongamos de ejemplo…

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)

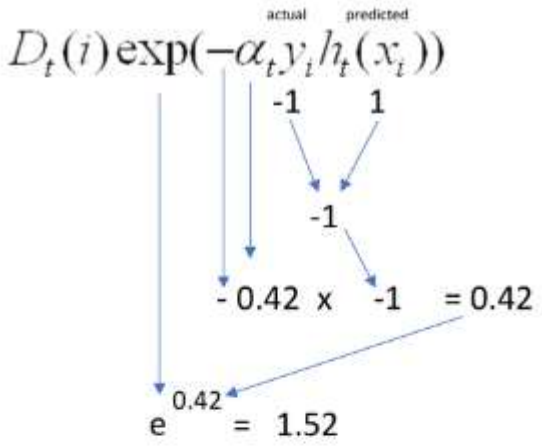
```
for(int t=1;t<=Adaboost.ClasificadoresDebilesAUsar;t++){
    double errormin = 230.0;
    CD elegido = null;
    double vector = 0.0;
    for(int k=1;k<=Adaboost.PruebasAleatorias;k++){
         double errorv = 0.0;
         CD cd = new CD();
         for(int i=0;i<D.length;i++){
             int hxy = 0;
             if(cd.estaDentro(X.get(i)) != Y[i]) hxy = 1;
             errorv += (D[i] * hxy);
         }
        if(errorv < errormin) {
           elegido = new CD(cd);
           errormin=errorv;
    }
}
```

Ahora calcularíamos la confianza del mejor clasificador débil.

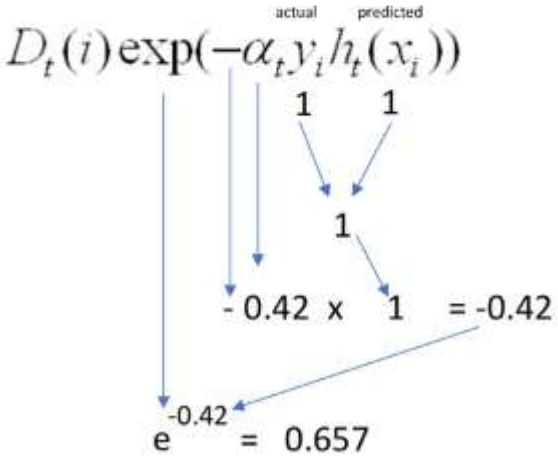
$$\alpha_t = 1/2 \log_2 \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$$

```
double alf = (double)(0.5*(Math.log((1-errormin)/errormin)));
elegido.alfa = alf;
```

Después solo nos quedaría actualizar los pesos para la siguiente iteración



mal clasificado



correctamente clasificado

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

 for(int i=0;i<D.length;i++) D[i] /= Z;
```

De este modo, al final se realiza la suma de todos los clasificadores previamente generados y se obtiene una hipótesis cuya predicción es más acertada.

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

3. Ajuste de Adaboost

-Adaboost necesita que ajustemos sus parámetros para que pueda clasificar mejor.

Con las pruebas ejecutadas, he determinado que un clasificador con muchos clasificadores débiles y muchas iteraciones, puede llegar a ser perjudicial, ya que corremos el peligro de sobreentrenamiento de éste.

4. Datos utilizados

-Primero de todo, utilicé el 80% de toda la base de datos para entrenamiento, considerando así, 3200 imágenes de las distintas categorías.

Debido a que los resultados no eran buenos, pasé a usar el 80% de cada categoría, para completar el 50% del vector de entrenamiento, y para el otro 50% del vector, el 20% restante dividido entre las demás categorías que quedaban.

Los resultados no han sido los esperados, por lo tanto, he vuelto a utilizar el 80% de toda la base de datos, obteniendo así un clasificador bueno.

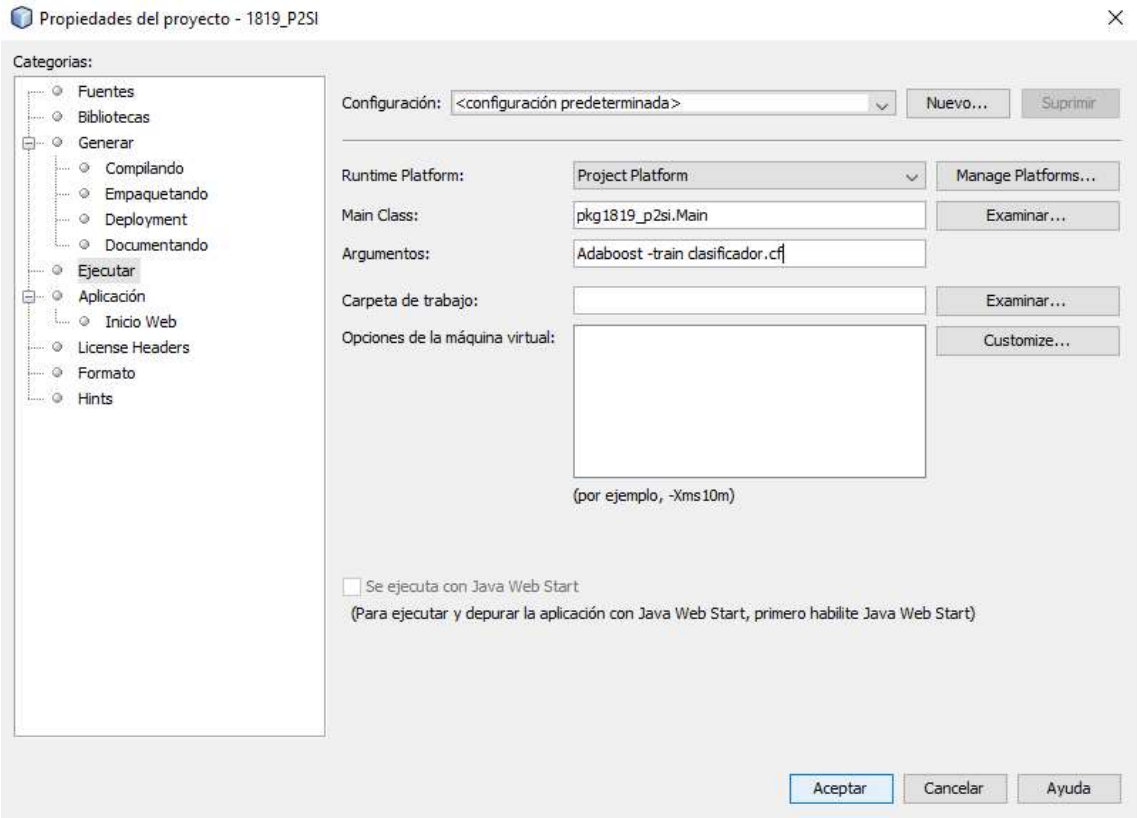
5. Mejoras para Adaboost

-He leído en Internet que para mejorar Adaboost, al calcular el error del clasificador, aparte de sumar los errores, se puede dividir entre el valor total del array de pesos.

6. Ejemplo de uso para Adaboost

6.1 Ejemplo para entrenar a Adaboost

Aquí podemos ver una ejecución del programa para entrenar a Adaboost:



Ajustamos los clasificadores débiles a usar y las pruebas aleatorias para entrenar:

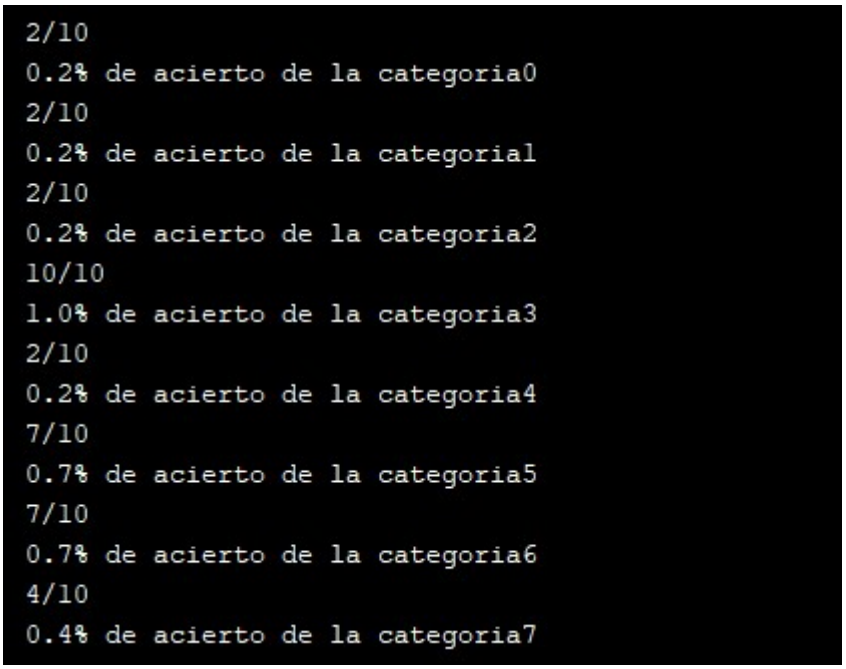
```
/**
 *
 * @author Ivan
 */
public class Adaboost {

    static int CDaUsar = 30;
    static int PruAle = 200;
```

Ejecutamos y vemos como empieza a entrenar, usando las imágenes pasadas como entrenamiento

```
Loaded class 0
Loaded class 1
Loaded class 2
Loaded class 3
Loaded class 4
Loaded class 5
Loaded class 6
Loaded class 7
Loaded 4001 images...
-----------------------------------------------------------------------------------------------------------------------
0.124999999999999871% de error
Confianza: 0.9729550745276625
0.275892857142848674% de error
Confianza: 0.4824633886088549
0.216406664875705454% de error
Confianza: 0.643365417604288
0.155067704945953344% de error
Confianza: 0.8476973361502536
0.313829195613976% de error
Confianza: 0.39113885390777214
0.23577307930207734% de error
Confianza: 0.5879974738032513
0.307150329740958574% de error
Confianza: 0.40673787423265556
0.35866693066245364% de error
Confianza: 0.2905773714102664
0.27781065522728124% de error
Confianza: 0.4776737847772505
0.183854352384360684% de error
Confianza: 0.7452244738663819
0.151620303099126084% de error
Confianza: 0.8609744508307473
0.0687268461662455844% de error
Confianza: 1.303206367887881
0.0644538694408725744% de error
```

Al final aparece un test donde ponemos a prueba el clasificador anteriormente creado

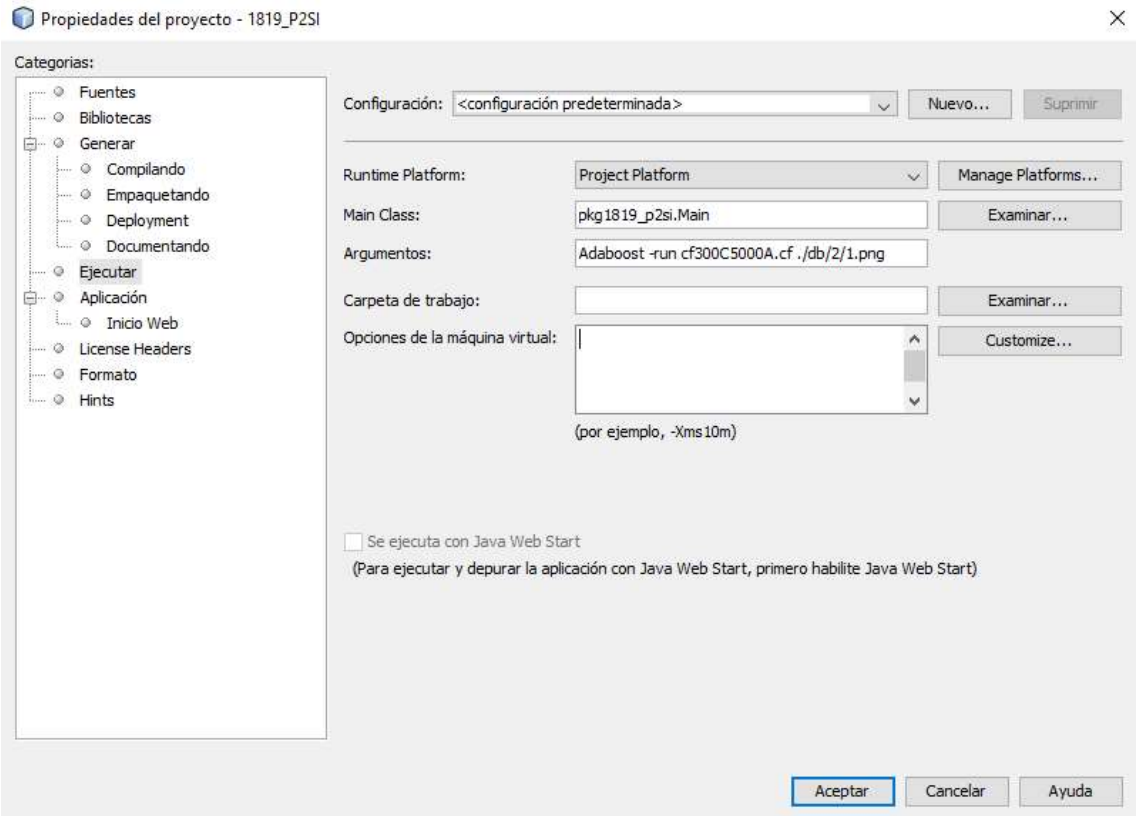


Como podemos ver, está muy poco entrenado.

6.2 Ejemplo de test para Adaboost

Ahora vamos a poner a prueba a Adaboost con esta imagen donde se ve claramente que es una camiseta.

Ponemos esto en los argumentos del programa



Y aquí podemos ver la salida:

En esta ocasión Adaboost ha acertado.

```
Loaded class 0
Loaded class 1
Loaded class 2
Loaded class 3
Loaded class 4
Loaded class 5
Loaded class 6
Loaded class 7
Loaded 4001 images...
Categoria0: -41.537583196165
Categoria1: 65.18291490775609
Categoria2: 77.93272070166367
Categoria3: -74.65370926845677
Categoria4: 0.735352248024771
Categoria5: -12.16381780796727
Categoria6: -69.08332696429211
Categoria7: -639.4130794825488
Adaboost: La imagen es una camiseta
BUILD SUCCESSFUL (total time: 1 second)
```

7. Bibliografía

https://towardsdatascience.com/adaboost-for-dummies-breaking-down-the-math-and-its-equations-into-simple-terms-87f439757dcf

https://en.wikipedia.org/wiki/AdaBoost

https://stackoverflow.com/questions/1922985/explaining-the-adaboost-algorithms-to-non-technical-people