2018

PRÁCTICA 2. Visión Artificial y Aprendizaje

SISTEMAS INTELIGENTES

JAVIER RIVILLA ARREDONDO

INDICE

¿Cuál es el número de clasificadores que se han de generar para que un clasificador débil funcione?..... 1

¿Cómo afecta el número de clasificadores generados al tiempo empleado para el proceso de aprendizaje? ¿Qué importancia le darías? Justifica tu respuesta..... 4

¿Cómo has dividido los datos en conjunto de entrenamiento y test? ¿Para qué es útil hacer esta división?..... 5

¿Has observado si se produce sobre entrenamiento? Justifica tu respuesta con una gráfica en la que se compare el error de entrenamiento y el de test a lo largo de las ejecuciones. 5

Comenta detalladamente el funcionamiento de AdaBoost teniendo en que tasa media de fallos obtienes para aprendizaje y test. 7

¿Cómo has conseguido que AdaBoost clasifique entre los 10 dígitos cuando solo tiene una salida binaria?..... 10

Práctica 2. Visión Artificial y Aprendizaje

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

En principio debo decir que he realizado una práctica que va dando los datos que nosotros necesitamos por interacciones, para que a su vez veamos el aprendizaje y podamos sacar conclusiones.

Primero explicaré los datos respecto al test y al aprendizaje:

Como bien intento hacer entender, la columna aprendizaje esta formada por {aciertos, errores, falsos positivos} y la de test mas de lo mismo. La de aprendizaje son los datos que se obtienen en las iteraciones mientras se entrena y los de test indican las pruebas que se realizan.

Aciertos Aprendizaje: Porcentaje de las imágenes que se han podido identificar.

Errores Aprendizaje: La cantidad de imágenes que son el número que buscábamos y no se han identificado.

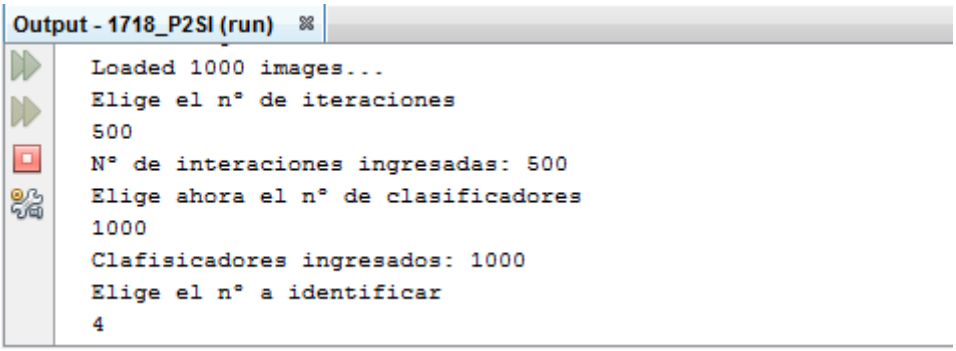
Falsos positivos: Imágenes que se identifican como tal número, pero realmente no son.

Aciertos Test: Indican los resultados que se realizan.

Para ello he realizado las siguientes pruebas:

- Pruebas realizadas con 500 iteraciones.
- Dos bloques: Aprendizaje y Test (70% para aprendizaje y 30% para test).
- Pruebas de n clasificadores {1000, 2000, 3000, 4000, 5000}.
- El número utilizado es el 4.

Y bien, una vez iniciamos el programa nos sale lo siguiente:



La imagen anterior muestra que hemos realizado pruebas con 500 iteraciones, 1000 clasificadores y el número 4.

Mostraré simplemente las 10 últimas iteraciones:

491	1	0	3	0,94	15	3	0,444
492	1	0	3	0,943	14	3	0,481
493	1	0	3	0,94	15	3	0,444
494	1	0	4	0,94	14	4	0,481
495	1	0	3	0,94	15	3	0,444
496	1	0	3	0,94	15	3	0,444
497	1	0	3	0,94	15	3	0,444
498	1	0	3	0,94	15	3	0,444
499	1	0	3	0,94	15	3	0,444
500	1	0	3	0,94	15	3	0,444

Iteración	Aciertos (Aprendizaje)	Errores (Aprendizaje)	Falsos positivos	Aciertos (Test)	Errores (Test)	F. P. (Test)	V. P. (Test)
491	100%	0	3	0,94	15	3	0,444
492	100%	0	3	0,943	14	3	0,481
493	100%	0	3	0,94	15	3	0,444
494	100%	0	4	0,94	14	4	0,481
495	100%	0	3	0,94	15	3	0,444
496	100%	0	3	0,94	15	3	0,444

497	100%	0	3	0,94	15	3	0,444
498	100%	0	3	0,94	15	3	0,444
499	100%	0	3	0,94	15	3	0,444
500	100%	0	3	0,94	15	3	0,444

Como podemos comprobar utilizando **500 iteraciones y 1000 clasificadores** obtenemos un **44%** de detección del número buscado.

- Ahora para 2000 clasificadores:

```
490    1    0    9    0,93    12    9    0,556
491    1    0    9    0,927   13    9    0,519
492    1    0    10   0,927   12    10   0,556
493    1    0    8    0,93    13    8    0,519
494    1    0    9    0,93    12    9    0,556
495    1    0    8    0,93    13    8    0,519
496    1    0    9    0,933   11    9    0,593
497    1    0    8    0,93    13    8    0,519
498    1    0    10   0,933   10    10   0,63
499    1    0    8    0,93    13    8    0,519
500    1    0    10   0,933   10    10   0,63
```

Iteración	Aciertos (Aprendizaje)	Errores (Aprendizaje)	Falsos positivos	Aciertos (Test)	Errores (Test)	F. P. (Test)	V. P. (Test)
491	100%	0	9	0,93	13	9	0,519
492	100%	0	10	0,927	12	10	0,556
493	100%	0	8	0,927	13	8	0,519
494	100%	0	9	0,93	12	9	0,556
495	100%	0	8	0,93	13	8	0,519
496	100%	0	9	0,93	11	9	0,593
497	100%	0	8	0,933	13	8	0,519
498	100%	0	10	0,93	10	10	0,63
499	100%	0	8	0,933	13	8	0,519
500	100%	0	10	0,93	10	10	0,6

Para **500 iteraciones y 2000 clasificadores** obtenemos un poco mejor el porcentaje de detección del número buscado**: 60%**

- Para 3000 clasificadores:

```
490    1    0    7    0,937   12    7    0,5
491    1    0    7    0,937   12    7    0,5
492    1    0    7    0,937   12    7    0,5
493    1    0    7    0,937   12    7    0,5
494    1    0    7    0,937   12    7    0,5
495    1    0    7    0,937   12    7    0,5
496    1    0    7    0,937   12    7    0,5
497    1    0    7    0,937   12    7    0,5
498    1    0    7    0,937   12    7    0,5
499    1    0    7    0,937   12    7    0,5
500    1    0    7    0,937   12    7    0,5
```

Iteración	Aciertos (Aprendizaje)	Errores (Aprendizaje)	Falsos positivos	Aciertos (Test)	Errores (Test)	F.P. (Test)	V.P. (Test)
491	100%	0	7	0,937	12	7	0,5
492	100%	0	7	0,937	12	7	0,5
493	100%	0	7	0,937	12	7	0,5

494	100%	0	7	0,937	12	7	0,5
495	100%	0	7	0,937	12	7	0,5
496	100%	0	7	0,937	12	7	0,5
497	100%	0	7	0,937	12	7	0,5
498	100%	0	7	0,937	12	7	0,5
499	100%	0	7	0,937	12	7	0,5
500	100%	0	7	0,937	12	7	0,5

Aquí ya vamos obteniendo menos detección del número buscado, tratándose de un 50% con respecto a la anterior que era 60%.

- Para 5000 clasificadores:

```
490          1          0          1          0,937          18          1          0,438
491          1          0          1          0,937          18          1          0,438
492          1          0          1          0,933          19          1          0,406
493          1          0          1          0,937          18          1          0,438
494          1          0          1          0,933          19          1          0,406
495          1          0          1          0,937          18          1          0,438
496          1          0          1          0,933          19          1          0,406
497          1          0          1          0,937          18          1          0,438
498          1          0          1          0,933          19          1          0,406
499          1          0          1          0,937          18          1          0,438
500          1          0          1          0,937          18          1          0,438
```

Iteración	Aciertos (Aprendizaje)	Errores (Aprendizaje)	Falsos positivos	Aciertos (Test)	Errores (Test)	F.P. (Test)	V.P. (Test)
491	100%	0	1	0,937	18	1	0,438
492	100%	0	1	0,933	19	1	0,406
493	100%	0	1	0,937	18	1	0,438
494	100%	0	1	0,933	19	1	0,406
495	100%	0	1	0,937	18	1	0,438
496	100%	0	1	0,933	19	1	0,406
497	100%	0	1	0,937	18	1	0,438
498	100%	0	1	0,933	19	1	0,406
499	100%	0	1	0,937	18	1	0,438
500	100%	0	1	0,937	18	1	0,438

Como podemos observar en este caso obtenemos una detección de 44% por lo que la mejor de todas será implementando **2000 clasificadores.** Hablando de un 60% de detección del número buscado.

Como conclusión podríamos decir, que, en la tabla de 500 iteraciones y 2000 clasificadores, del número 4 los resultados son mejorables respecto a falsos positivos y en la detección del número.

¿Cómo afecta el número de clasificadores generados al tiempo empleado para el proceso de aprendizaje? ¿Qué importancia le darías? Justifica tu respuesta.

Clasificadores	1500	2000	3500	4000	5500	6000	7500	8000	9500
Tiempo	5,06	6,131	10,62	11,35	15,44	17,9	21,45	23,52	27,9

Este tiempo se ha calculado con 500 iteraciones y por cada n numero de clasificadores de esta forma que si ponemos 1500 clasificadores serán 500*1500 clasificadores usados, porque son por cada iteración. En cuanto al tiempo no es excesivo, por lo que sería mínima la importancia y más hoy en día que la capacidad de procesamiento hace que las respuestas sean más rápidas.

Además, los tiempos aumentan linealmente con el número de clasificadores, es decir, cuanto mas clasificadores metamos más tiempo tardará en efectuarse.

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

Al conjunto de entrenamiento le he destinado el 70% y al de entrenamiento le he destinado 30%. Pienso que es considerable el coste del proceso porque realmente se realizan muchísimos intentos para conseguir mejoras.

De hecho, en mi práctica no se alcanza para nada la perfección de hecho los resultados no son del todo malo, pero el porcentaje de aciertos en los test son inferiores a la mitad.

Respondiendo para que fuesen útiles hacer esta división, el hecho sería minimizar el impacto que la limitación de la información para la eficacia de un clasificador, obteniendo mejores rendimientos hablando de estos.

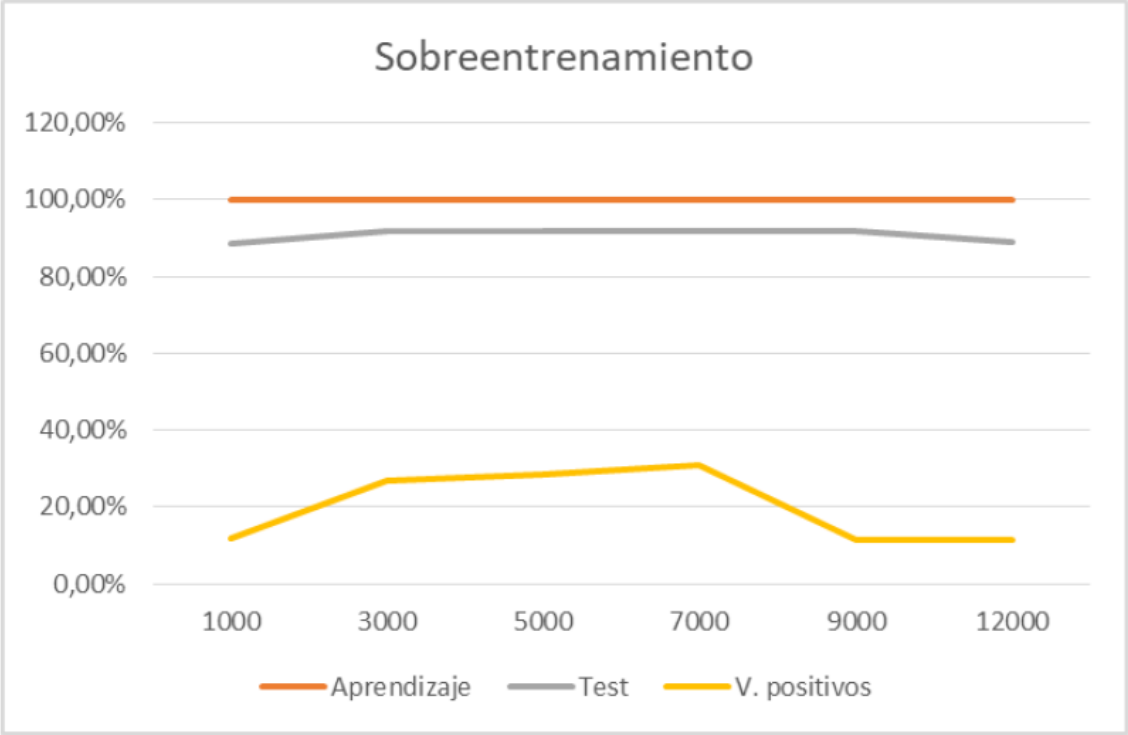
¿Has observado si se produce sobre entrenamiento? Justifica tu respuesta con una gráfica en la que se compare el error de entrenamiento y el de test a lo largo de las ejecuciones.

Sí, como bien he comentado anteriormente, cuando se hacen mas iteraciones de las que se deberían los porcentajes de aciertos disminuyen considerablemente.

Como bien hemos visto antes, el entrenamiento en aprendizaje, los valores son los siguientes:

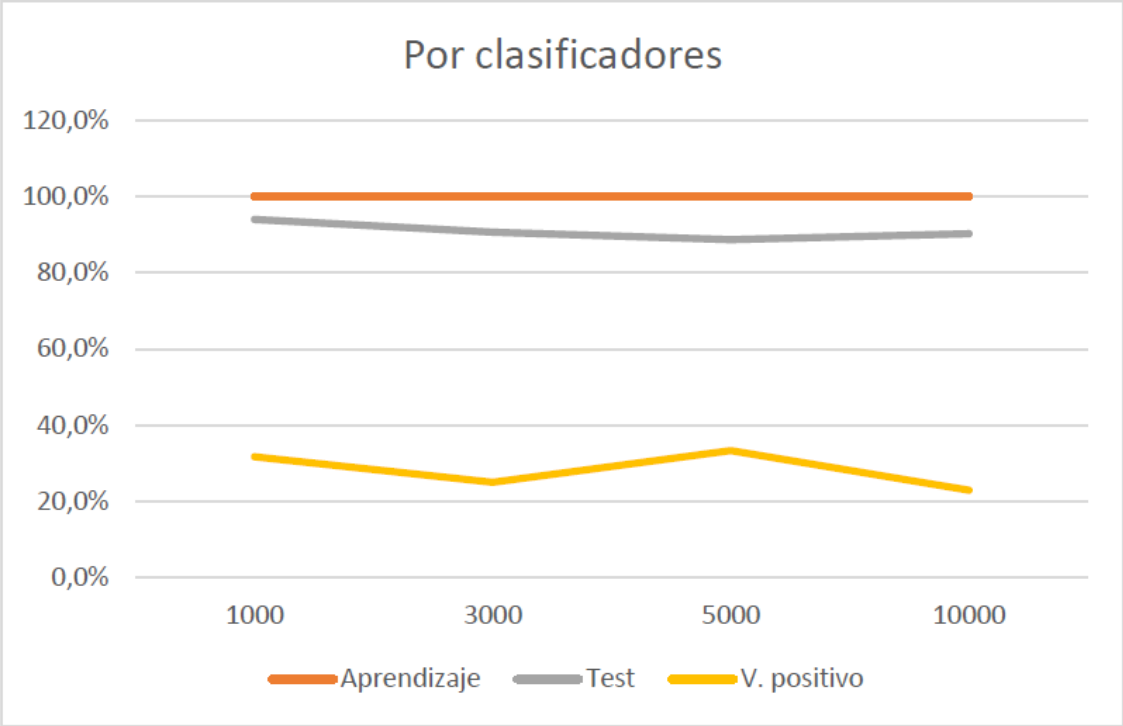
Iteraciones	Aprendizaje	Test
1000	99,90%	88%
3000	99,90%	92%
5000	99,90%	92%
7000	100%	92%
9000	99,90%	91%
10000	100%	89%

Como podemos comprobar en el aprendizaje los valores son mínimamente significativos, en cambio, en los de test a medida que aumentamos las iteraciones los % de test van disminuyendo poco a poco.



Anteriormente explicaba en cuanto a iteraciones, pero en clasificadores sería el siguiente:

Clasificadores	Aprendizaje	Test	Aciertos
1000	100%	92,2%	30,1%
3000	100%	90,7%	60%
5000	100%	87,8%	50%



Comenta detalladamente el funcionamiento de AdaBoost teniendo en que tasa media de fallos obtienes para aprendizaje y test.

El algoritmo realiza una serie de iteraciones en las que genera un hiperplano. Para ello se encargará de modificar los pesos de las imágenes en cada iteración con el objetivo final de que este coloque la mayoría de los dígitos a cada lado que este partido el espacio. Por un lado, tendríamos los dígitos positivos (los buenos) y por otro lado tendríamos los dígitos negativos (los malos).

El clasificador débil genera un número elevado de hiperplanos siempre quedándose con el mejor por lo que con este podremos generar el hiperplano.

Una vez generado el Hiperplano, recalculamos los pesos:

```
auxiliar = imagenes.stream().map((num) -> {
    num.setWeight(num.getWeight() * Math.pow(Math.E, - clasideb.getconfidencialactual() *
            (double) num.getType() * (double)clasideb.pointLocation(num)));
    return num;
}).map((num) -> num.getWeight()).reduce(auxiliar, (accumulator, _item) -> accumulator + _item);

for (Imagen img: imagenes){
    img.setWeight(img.getWeight() / auxiliar);
}
```

Con esto conseguimos una gran mejora, porque es por cada iteración y si cada una mejora será mayor.

Ahora bien, vamos con el algoritmo AdaBoost:

```
public ClasificadorFuerte(ArrayList<Imagen> imagenes, ArrayList<Imagen> testImagenes, int
iteraciones, int numClasificadoresDebiles) {
      //Con el objetivo de ir añadiendo los clasificadores debiles vamos a añadirlos
      //siguiente vector
 cld = new ArrayList<>();

      //3 decimales almenos!
      NumberFormat format = NumberFormat.getInstance(new Locale("es", "ES"));
      format.setMaximumFractionDigits(3);
  //recorremos las imagenes con un forEach y vamos actualizando los pesos de las imagenes.
      imagenes.forEach((f) -> {
         f.setWeight(1d/imagenes.size());
      });
      if (Adaboost.booleana == true){
         System.out.print("Iteracion | Acier Apren(%) | NºError Apren | ");
         System.out.print("NoNumError | Aprendidos | Acier Test(%) | NúmError Test | ");
         System.out.println("NoNúmError Test-Aciertos(%) | Verd Positivos ");
      }

      for (int cl = 0; cl < iteraciones; cl++) {
         String saltosLinea = "\t" + "\t";
         //Ya tiene por defecto el nivel de confianza de un clasificador debil
         ClasificadorDebil clasideb;
         clasideb = new ClasificadorDebil(numClasificadoresDebiles, imagenes);
         //inicializamos el clasificador
         cld.add(clasideb);
         double auxiliar = 0d;
         auxiliar = imagenes.stream().map((num) -> {
            num.setWeight(num.getWeight() * Math.pow(Math.E, -
clasideb.getconfidencialactual() *
                  (double) num.getType() * (double)clasideb.pointLocation(num)));
            return num;
```

```
        }).map((num) -> num.getWeight()).reduce(auxiliar, (accumulator, _item) -> accumulator
+ _item);


        for (Imagen img: imagenes){
           img.setWeight(img.getWeight() / auxiliar);
        }


        if (Adaboost.booleana == true) {
           boolean doIt = true;
           int[] resultado1;
           int[] resultado2;
           String correctly = "Vamos a ver si esto funciona correctamente";


           resultado1 = obtieneResultados(imagenes);
           resultado2 = obtieneResultados(testImagenes);


           if (doIt == true) {
              System.out.print((cl+1));
              System.out.print(saltosLinea);


              System.out.print(format.format((double)resultado1[0]/imagenes.size()));
              System.out.print(saltosLinea);


              System.out.print(resultado1[1] + "\t" + resultado2[2]);
              System.out.print(saltosLinea);


              System.out.print(format.format((double)resultado2[0]/testImagenes.size()));
              System.out.print(saltosLinea);


              System.out.print(resultado2[1] + saltosLinea + resultado2[2]);
              System.out.print(saltosLinea);
```

```
            double aux = (double)resultado2[3]/(double)(resultado2[3]+ resultado2[4]);
            System.out.println(format.format(aux));
        }
        Else {
            System.out.print(correctly);
        }
    }
  }
}
```

¿Cómo has conseguido que AdaBoost clasifique entre los 10 dígitos cuando solo tiene una salida binaria?

Lo que he hecho ha sido clasificar número a número. Cada vez que se aprende se ejecuta el siguiente código:

```
public boolean [] transformarImagen(MNISTLoader ml,Imagen imagen){

    boolean total[] = new boolean [10];
    int digito = 0;
    boolean encontrado = true;

    while (digito <= 9 && encontrado == true){
        enseñoNumeroDigito(ml,digito);

        total[digito] = cf.puntoLocalizacionImagen(imagen) > 0;

        digito = digito + 1;
    }
    return total;
}
```

Como se puede apreciar se crea un array de booleanos y si la imagen pertenece al espacio de ese número guardamos en el vector true, si no guardamos false. Realmente si te fijas bien estamos asignando a ese digito el valor de esa condición.

Una vez tenemos construido ese array de booleanos, cada vez que un número es aprendido se modifica el dígito por lo que se le asignará si el número es positivo(significa que es número) su correspondiente etiqueta, si no pues la negativa.