# Práctica de Técnicas de Inteligencia Artificial. Segunda práctica: Aprendizaje

Patricia Compañ, Otto Colomina, Francisco Escolano y Diego Viejo
Departamento de Ciencia de la Computación e Inteligencia Artificial
Universidad de Alicante

Curso 2009 - 2010

## 1. Descripción de la práctica

La práctica de aprendizaje para el curso 2009/10 consistirá en el desarrollo de la implementación en Java del algoritmo *Adaboost*. Este algoritmo se basa en el hecho de que la combinación de varios clasificadores débiles, cada uno con su propio valor de confianza, permite obtener un clasificador más preciso.

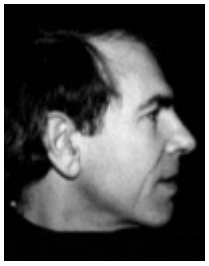
En el caso concreto de la práctica, disponemos de una base de datos de fotografías de distintas personas vistas de frente, de perfil y de medio perfil. Se trata de desarrollar un algoritmo de aprendizaje que distinga entre imágenes de frente e imágenes de perfil.

La base de datos suministrada esta compuesta por más de 300 imágenes: el nombre de cada imagen está codificado de la siguiente manera:

- El primer carácter es una $f$ si se trata de una mujer o una $m$ si se trata de un hombre.
- A continuación un número que indica el individuo en cuestión.
- El siguiente carácter es una $v$.
- Un número: 1 para imágenes de frente, 2 y 3 para imágenes de perfil o medio perfil.
- El carácter $e$ y un número que hace referencia a la imagen concreta de las varias que hay del mismo individuo.

En este problema, las imágenes de las personas constituyen el conjunto de patrones. Vamos a considerar dos clases: frente y perfil (incluye medio perfil también). Lo primero que hay que determinar es qué primitiva vamos a utilizar como clasificador. Ya que en realidad, una imagen se representa por un vector $N$ -dimensional, necesitamos un elemento que permita dividir un espacio $N$-dimensional

(a) Imagen de frente (*f1v1e1)*



(b) Imagen de perfil (*m5v2e1)*

Figura 1: Ejemplos de imágenes

en dos subespacios. Para ello vamos a emplear un hiperplano, es decir, cada clasificador débil será un hiperplano.

Una parte de la base de datos se empleará como conjunto de test y el resto como conjunto de aprendizaje.

## 1.1. Caracterización de un hiperplano

Para generar un hiperplano de manera sencilla, se puede encontrar el valor máximo y el valor mínimo en cada dimensión. Con esto se genera al azar un punto dentro de estos límites y se genera también al azar un vector con el mismo número de dimensiones. Este vector está normalizado y representa la normal del hiperplano.

## 1.2. Algoritmo Adaboost

El proceso se resume de la siguiente manera:

1. Crear un clasificador débil
   - Entrenar el clasificador. Para ello se generan $T$ hiperplanos aleatoriamente y se selecciona aquel que tiene menor error de clasificación.
   - Calcular la confianza del clasificador.

     $$\alpha_t = \frac{1}{2} \log \left( \frac{1 - \epsilon_t}{\epsilon_t} \right) ,$$

     siendo $\epsilon_t$ es el error del clasificador $h_t$.
2. Actualizar la distribución de probabilidad de los ejemplos.
3. Calcular el error del clasificador combinado, o lo que es equivalente, la tasa de aciertos

4. Todo este proceso se repite hasta que la tasa de aciertos es igual a 1 o bien se ha generado el número máximo de clasificadores que se ha puesto como parámetro. El número máximo $c$ de clasificadores deberá fijarse atendiendo a evitar el *overfitting*, esto es, si con $c' > cT$ se consigue la misma tasa de acierto, es conveniente usar $c$ clasificadores.

## 2. Implementación

El código estará estructurado en un proyecto Eclipse. Para facilitar las tareas del manejo de las imágenes, se suministra un proyecto con algunas clases ya creadas (algunas con código ya incorporado y otras que hay que rellenar por completo):

- Main.java: contiene métodos que permiten el paso de los parámetros.
- ImageFilter.java: contiene los métodos que permiten filtrar los ficheros gráficos.
- Hiperplano.java
- ClasificadorDebil.java: un clasificador débil debe incluir por lo menos un hiperplano y un valor de confianza y debe incorporar el método entrenaClasificador.
- AdaBoost.java: debe incorporarse el código del algoritmo.

El programa debe tener los siguientes parámetros:

- c: número máximo de clasificadores
- T: número de candidatos para entrenar cada clasificador
- d: directorio donde está la BD de imágenes

## 3. Parte experimental

Aparte del proceso de implementación del algoritmo Adaboost, el alumno deberá desarrollar un proceso de experimentación. Habrá que ajustar pues tanto el número de hiperplanos $T$ necesarios para inferir cada clasificador débil, así como el número $c$ de clasificadores necesarios, siguiendo el principio de evitar el sobreentrenamiento. A continuación, los experimentos realizados por el alumno deberán buscar los límites del Adaboost para, razonadamente, explicar bajo qué condiciones (distribución de los datos de entrada) el error obtenido no puede reducirse más. En caso de que se haya implementado una parte opcional, el alumno deberá demostrar mediante experimentos cómo la parte opcional incorporada mejora las tasas de reconocimiento.

## 4. Documentación

La práctica irá acompañada de la correspondiente documentación. En dicho documento, el alumno explicará y justificará razonadamente los detalles de diseño de la función implementada, experimentación, y resultados.

## 5. Parte optativa

Entrenar los hiperplanos usando la regla delta para perceptrones de una única neurona vista en clase de teoría (transparencia 7 del tema de perceptrones multicapa). El hiperplano del perceptrón se inicializará aleatoriamente, igual que en la parte obligatoria. El entrenamiento terminará cuando se consiga una tasa de aciertos objetivo o bien cuando se supere un número máximo de iteraciones.

En la documentación se deben hacer distintas pruebas, variando el valor de la constante de aprendizaje con valores entre 0 y 1. Recoger datos sobre la velocidad de convergencia y sobre la tasa de aciertos con los distintos valores. Para que el resultado sea fiable hay que hacer varias pruebas con el mismo valor de la cte. y quedarse con la media.

La implementación se hará con una clase PerceptronSimple que heredará de ClasificadorDebil. Esta clase sobreescribirá el método entrenaClasificador(), donde se implementará el entrenamiento del perceptrón. En el Main se añadirán los parámetros necesarios para configurar el perceptrón (cte de aprendizaje, tasa de aciertos e iteraciones máximas). Cuando al menos uno de estos tres valores se proporcione en línea de comandos se usará el perceptrón. Debéis tener valores por defecto para los parámetros no especificados.

## 6. Normas de entrega y evaluación de la práctica

La práctica es individual. Cualquier indicio de plagio supondrá el suspenso de la parte de prácticas y, por consiguiente, de la presente convocatoria de la asignatura. Se avisa al alumno que existe la posibilidad de utilizar sistemas automáticos de detección de copia.

La práctica se valorará de la siguiente manera:

- 25 % por código y funcionamiento. Se probarán varios ejemplos y se valorará el funcionamiento final. Si la práctica no cumple con un mínimo de funcionamiento, se suspenderá por completo sin tener en cuenta los apartados siguientes. En cuanto al código, deberá estar debidamente estructurado e indentado y haciendo un uso adecuado del lenguaje de programación. El código deberá estar debidamente comentado (incluso para la generación de javadoc).

- 25 % por la experimentación.

- 30 % por documentación. La documentación se podrá entregar en cualquier formato. Será responsabilidad del alumno asegurar que la documentación pueda ser leída en cualquier sistema operativo.
- 20 % por la parte optativa.

Fechas de entrega de la práctica:

- El plazo de entrega de la práctica finaliza a las 24 horas del viernes 21 de Mayo del 2010.
- La entrega se realizará a través del sitio web moodle de la asignatura.