UNIVERSIDAD DE BURGOS
Escuela Politécnica Superior
Ingeniería Técnica en Informática de Gestión
Área de Lenguajes y Sistemas Informáticos

TRABAJO FIN DE CARRERA

TÉCNICAS DE BOOSTING

ANEXO 5. MANUALES DE USUARIO

Alumno:
Santiago David Villalba Bartolomé

Tutor:
Juan José Rodríguez Díez

Última revisión el 16/09/2002

TABLA DE CONTENIDOS

LISTA DE FIGURAS

INTRODUCCIÓN A LA SUITE OAIDTB

OAIDTB (Otra Aplicación Interactiva Demostrando Técnicas de Boosting) es una suite de programas para la aplicación y evaluación de técnicas de boosting a problemas de aprendizaje computacional. Está escrita en java y es totalmente portable entre sistemas operativos. Tiene como base a la biblioteca de algoritmos de aprendizaje computacional Weka, a la que extiende con nuevos métodos de aprendizaje basados en algoritmos de boosting.

Los programas que incluye la suite son los siguientes:

- Una biblioteca de clases java que implementan diversos algoritmos de *boosting* extendiendo a la biblioteca weka; a este conjunto de clases las llamaremos a partir de ahora *boosters*. Pueden ser utilizados desde la línea de comandos o embebidos en otro código java.

- Una aplicación gráfica para la selección, configuración, aplicación y evaluación de los métodos de aprendizaje tanto de weka como de oaidtb.

- Una aplicación gráfica docente, "Generalización Bidimensional", que permite visualizar las hipótesis lanzadas por los clasificadores de weka y oaidtb en el marco de un problema bidimensional y multiclase.

OAIDTB es software de libre distribución y se distribuye bajo la licencia GNU General Public License, de la que existe una copia en el directorio de la aplicación.

INSTALACIÓN Y CONFIGURACIÓN

REQUISITOS HARDWARE

Los requisitos hardware dependen mucho del tipo de uso que le vayamos a dar a la aplicación (docente, de investigación, clasificación de pequeños o grandes conjuntos de datos etc.).

Espacio libre en disco duro 22 MB
* 1MB para el programa en sí más
* 21 MB de documentación y conjuntos de datos para hacer pruebas que se pueden borrar en cualquier momento.

Los requisitos mínimos para correr la aplicación son:

* CPU – Pentium II 233 Mhz o equivalente
* Memoria RAM 32 MB

Aunque por regla general no vamos a necesitar una máquina de grandes prestaciones para que la aplicación corra con alegría, cuanta más potencia se tenga mejor; requisitos recomendados:

* CPU – Pentium III 600 Mhz o equivalente
* Memoria RAM 64 MB

REQUISITOS SOFTWARE

- Para los boosters:

  * Sistema operativo soportado por java
  * Máquina virtual java 1.1 ó superior
    http://java.sun.com/.
  * Biblioteca de clases Weka 3.2.x ó superior (incluida en la distribución).
    http://www.cs.waikato.ac.nz/ml/weka/

- Para las aplicaciones gráficas:

  * Sistema operativo soportado por java
  * Máquina virtual java 1.3 ó superior
    http://java.sun.com/.
  * Biblioteca de clases Weka 3.2.x ó superior (incluida en la aplicación).
    http://www.cs.waikato.ac.nz/ml/weka/
  * Biblioteca de clases Colt distribution 1.0.2 ó superior (incluida en la aplicación).
    http://nicewww.cern.ch/%7Ehoschek/colt/index.htm
  * Biblioteca de clases JCommon 0.6.4 ó superior (incluida en la aplicación).
    http://www.object-refinery.com
  * Biblioteca de clases JFreeChart 0.9.2 ó superior (incluida en la aplicación).
    http://www.object-refinery.com

INSTALACIÓN

Durante todo lo que resta de manual se supondrá que en la máquina en que se va a instalar la aplicación tiene ya corriendo una máquina virtual java compatible con la aplicación, y que la manera de invocarla es mediante el comando *java*; si esto no es así, el primer paso es instalar una máquina virtual java que podrá encontrar en http://java.sun.com/.

La suite se distribuye en un único archivo jar (en el CD del proyecto, para mayor facilidad de instalación, la he dejado sin comprimir en el directorio "Suite OAIDTB"); copie el archivo oaidtb.jar en el directorio de instalación que desee (que a partir de ahora llamaremos $OAIDTB_HOME) y descomprímalo mediante la línea de comandos:

jar –xvf oaidtb.jar

(se puede utilizar cualquier otro descompresor que soporte zip para hacerlo).

Una vez descomprimido nos encontramos con la siguiente jerarquía de directorios:

data: Contiene conjuntos de datos en formato ARFF para su utilización con los boosters.
- ****_imgs_:** Imágenes listas para ser cargadas en la aplicación "Generalización Bidimensional"
- ****agricultura:** Algunos conjuntos de datos con problemas de la agricultura neo-zelandesa
- ****conus-torus:** Conjuntos de datos conus-torus
- ****misc:** Conjuntos de datos misceláneos
- ****oaidtbGUI:** Conjuntos de datos salvados desde la aplicación "Generalización Bidimensional"
- ****uci:** Algunos conjuntos de datos utilizados provenientes de la University of Caroline at Irvine.

- **_results_:** Un directorio donde almacenar resultados de experimentos

- **bin:** En este directorio se alojan los scripts que sirven para inicia las diversas aplicaciones que forman la suite:

  cfg: Archivos de configuración de la suite
  windows: Scripts para ejecutar las aplicaciones en windows
  linux: Scripts para ejecutar las aplicaciones en linux

- **docs:** Directorio con el manual de usuario, los javadocs de la aplicación y el código fuente en formato html.

- **libs:** Las bibliotecas de funciones, incluyendo el código compilado de OAIDTB

- **licenses:** Directorio con las licencias de las diversas librerías

- **misc:** Directorio con un parche que se ha creado que solventa algunos bugs de las aplicaciones gráficas de weka, así como la proporciona nuevas funcionalidades.

- **src:** Directorio con el código fuente de la aplicación y un archivo de Ant para compilarlos.

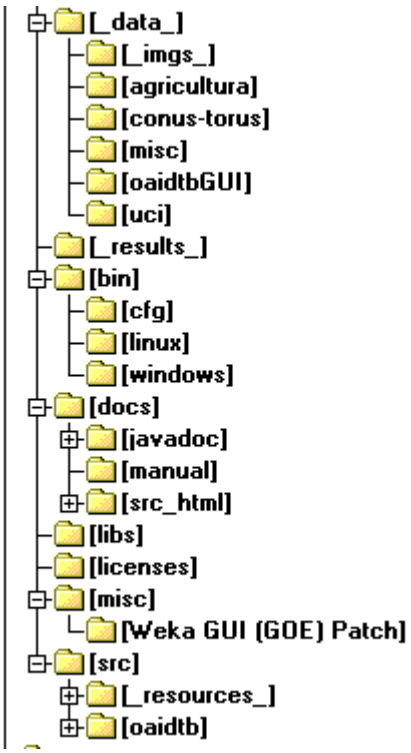

Figura 1 – Jerarquía de directorios en la aplicación

Para ejecutar cualquiera de las herramientas que forman parte de OAIDTB se deben tener en el CLASSPATH de la máquina virtual java todas los archivos *.jar que están en el directorio lib; esto lo hacen automáticamente los scripts del directorio *bin*; tan sólo debe editar el script _oaidtb_.bat en windows o el script _oaidtb_ en linux y seguir las sencillas instrucciones que se dan en los mismos para preparar el entorno de ejecución (en la mayoría de los casos bastará cambiar la línea donde se especifica la localización de la máquina virtual java):

$OAIDTB_HOME/bin/windows/_oaidtb_.bat

```
@echo off

::---------------------------------------------------------------------
:: OAIDTB Script de inicio
::---------------------------------------------------------------------
:: Este es un script genérico para ejecutar cualquiera de las aplicaciones
:: de weka o de la librería oaidtb; para usarlo:
::  -Definir la variable MAIN_CLASS_NAME (posiblemente en otro script)
::  -Llamar a este script.
::  -Si se quiere redirigir la salida a un archivo,
::   llamar a este script con el nombre de dicho archivo como parámetro.
::---------------------------------------------------------------------

:: ---------------------------------------------------------------------
:: Antes de poder ejecutar nada es necesario especificar
:: La localización de la máquina virtual java
:: ---------------------------------------------------------------------
IF EXIST "%JAVA_HOME%" goto java_home_exists
SET JAVA_HOME=d:\prog\java\j2sdk 1.3

:java_home_exists

:: ---------------------------------------------------------------------
:: Aquí se debe especificar el directorio donde se
:: ha instalado la aplicación
:: ---------------------------------------------------------------------
SET OAIDTB_HOME=..\..

:: ---------------------------------------------------------------------
:: Si se ha configurado correctamente la variable JAVA_HOME
:: entonces no necesitarás cambiar la configuraciñon
:: de la variable JAVA_EXE
:: ---------------------------------------------------------------------
SET JAVA_EXE=%JAVA_HOME%\bin\java.exe

IF NOT EXIST "%JAVA_EXE%" goto error

:: Indica aquí los parámetros opcionales para la máquina virtual java
::SET JVM_ARGS=-ms16m -mx96m

::Configura el lugar en el que se encuentra la librería weka;
::tener en cuenta que si la librería es una versión igual
::o posterior a la 3.3.1 sólo funcionarán correctamente
::sus aplicaciones gráficas si se ha aplicado el parche que viene con
::esta aplicación
:: if not "%WEKA_HOME%"=="" SET OAIDTB_WEKA_HOME=%WEKA_HOME%
::if "%OAIDTB_WEKA_HOME%"=="" SET
OAIDTB_WEKA_HOME=%OAIDTB_HOME%\libs\weka_patched.jar

:: ---------------------------------------------------------------------
::Las siguientes variables sólo es necesario configurarlas
::en el caso de usar el ClassifierPanel o el AllContainerPanel
::y de no usar las librerías que acompañan a la aplicación
```

```
:: ---------------------------------------------------------------------
::if "%COLT_HOME%"=="" SET
COLT_HOME=%OAIDTB_HOME%\libs\colt_only_used_classes.jar
::if "%JFREECHART_HOME%"=="" SET
JFREECHART_HOME=%OAIDTB_HOME\libs\jfreechart-0.9.2.jar
::if "%JCOMMON_HOME%"=="" SET
JCOMMON_HOME=%OAIDTB_HOME%\libs\jcommon-0.6.4.jar

SET OAIDTB_CLASSPATH=
for %%c in (%OAIDTB_HOME%\libs\*.zip %OAIDTB_HOME%\libs\*.jar) do call
_append_.bat %%c
call _append_.bat %OAIDTB_WEKA_HOME%
call _append_.bat %COLT_HOME%
call _append_.bat %JFREECHART_HOME%
call _append_.bat %JCOMMON_HOME%

::SET ALL_ARGS=
::if not "%*"=="*" SET ALL_ARGS = %*
::shift

::---------------------------------------------------------------------
:: Especificar la localización del archivo de propiedades para el GOE
:: No mola--> Cambiarlo en el código fuente
::---------------------------------------------------------------------

IF NOT "%1"=="" goto output_redirected
"%JAVA_EXE%" %JVM_ARGS% -cp "%OAIDTB_CLASSPATH%" -Duser.home=..\cfg
%MAIN_CLASS_NAME%
goto end

::---------------------------------------------------------------------
:: En DOS no hay manera de redirigir stderr
:: Se puede hacer de muchas maneras:
::   - Usar redir.exe del DJGPP
::   - Un programita en C de 6 líneas sirve pal caso
::--------redir.c-----------
::#include <stdio.h>
::
:: main (int argc, char ** argv){
::   dup2 (fileno (stdout), fileno (stderr));
::   execvp (argv[1], argv + 1);
:: }
::---------- end of redir.c ----------
::Usage: redir prog arg1 arg2 ... argn
::
::   - etc.
::---------------------------------------------------------------------
:output_redirected
"%JAVA_EXE%" %JVM_ARGS% -cp "%OAIDTB_CLASSPATH%" -Duser.home=..\cfg
%MAIN_CLASS_NAME% >%1
goto end

:error
echo ---------------------------------------------------------------------
echo ERROR: No se puede iniciar la máquina virtual de java
```

```
echo Por favor, especifica la variable JAVA_HOME en este archivo batch
echo ------------------------------------------------------------------
pause

:end
```

*$OAIDTB_HOME/bin/linux/_oaidtb_*

```
#!/bin/bash
#------------------------------------------------------------------
# OAIDTB Script de inicio
#------------------------------------------------------------------
# Este es un script genérico para ejecutar cualquiera de las aplicaciones
# de weka o de la librería oaidtb; para usarlo:
#  -Definir la variable MAIN_CLASS_NAME (posiblemente en otro script)
#  -Llamar a este script.
#  -Si se quiere redirigir la salida a un archivo,
#   llamar a este script con el nombre de dicho archivo como parámetro.
#------------------------------------------------------------------

#------------------------------------------------------------------
# Antes de poder ejecutar nada es necesario especificar
# la localización de la máquina virtual java
# ------------------------------------------------------------------
# Si se ha configurado correctamente la variable JAVA_HOME en
# el entorno no necesitarás cambiar la configuraciñon
# de la variable JAVA_EXE
# ------------------------------------------------------------------
if [ -z "$JAVA_HOME" ]; then
  JAVA_EXE=usr/local/java/jdk1.3.1/bin/java
else
  JAVA_EXE=$JAVA_HOME/bin/java
fi

if [ ! -x $JAVA_EXE ]; then
  echo ------------------------------------------------------------------
  echo ERROR: No se puede iniciar la máquina virtual de java
  echo Por favor, especifica la variable JAVA_EXE en este script
  echo ------------------------------------------------------------------
  return -1
fi

# ------------------------------------------------------------------
# Aquí se debe especificar el directorio donde se
# ha instalado la aplicación
# ------------------------------------------------------------------
#OAIDTB_HOME=/home/santi/pfc/app
OAIDTB_HOME=../..

OAIDTB_LIBS_HOME=$OAIDTB_HOME/libs

#Indica aquí los parámetros opcionales para la máquina virtual java
#JVM_ARGS="-ms32m -mx128m"
```

```
#Configura el lugar en el que se encuentra la librería weka;
#tener en cuenta que si la librería es una versión igual
#o posterior a la 3.3.1 sólo funcionarán correctamente
#sus aplicaciones gráficas si se ha aplicado el parche que viene con
#esta aplicación
if [ -z "$WEKA_HOME" ]; then
  if [ -z "$OAIDTB_WEKA_HOME" ]; then
    OAIDTB_WEKA_HOME=$OAIDTB_HOME/libs/weka_patched.jar
  fi
else
  OAIDTB_WEKA_HOME=$WEKA_HOME
fi

#---------------------------------------------------------------------
#Las siguientes variables sólo es necesario configurarlas
#en el caso de usar el ClassifierPanel o el AllContainerPanel
#y de no usar las librerías que acompañan a la aplicación
#---------------------------------------------------------------------
#if [ -z "$COLT_HOME" ]; then
# COLT_HOME=$OAIDTB_HOME/libs/colt_only_used_classes.jar
#fi
#if [ -z "$JFREECHART_HOME" ]; then
#  JFREECHART_HOME=$OAIDTB_HOME/libs/jfreechart-0.9.2.jar
#fi
#if [ -z "$JCOMMON_HOME" ]; then
#  JCOMMON_HOME=$OAIDTB_HOME/libs/jcommon-0.6.4.jar
#fi

AUTOMATIC_OAIDTB_CLASSPATH=""
for libfile in $OAIDTB_LIBS_HOME/*; do        # Metemos todos los archivos en el directorio
en el classpath
  AUTOMATIC_OAIDTB_CLASSPATH=$AUTOMATIC_OAIDTB_CLASSPATH$libfile:
done

OAIDTB_CLASSPATH=$OAIDTB_WEKA_HOME:$COLT_HOME:$JFREECHART_HOM
E:$JCOMMON_HOME:$AUTOMATIC_OAIDTB_CLASSPATH

if [ -z "$MAIN_CLASS_NAME" ]; then
  MAIN_CLASS_NAME="oaidtb.gui.AllContainerPanel\$AppMainFrame"
fi

#---------------------------------------------------------------------
# Especificar la localización del archivo de propiedades para el GOE
# No mola--> Cambiarlo en el código fuente
#---------------------------------------------------------------------
GOE_CFG_FILE_DIR=../cfg
#-Duser.dir=$OAIDTB_HOME
if [ -z "$1" ]; then
  exec ${JAVA_EXE} $JVM_ARGS -cp ${OAIDTB_CLASSPATH} -
Duser.home=${GOE_CFG_FILE_DIR} $MAIN_CLASS_NAME
else
  exec ${JAVA_EXE} $JVM_ARGS -cp ${OAIDTB_CLASSPATH} -
Duser.home=${GOE_CFG_FILE_DIR} $MAIN_CLASS_NAME >${1} 2>${1}
fi
```

3. MANUAL DE USUARIO

En esta parte del documento se va a explicar cómo utilizar la aplicación. Dado que el programa se compone de dos entidades diferenciadas, la parte de los alumnos y la parte de los profesores / administradores, explicaremos primeramente el uso de aquélla y posteriormente el uso de ésta. A los alumnos no les interesa saber, cómo se utiliza la interfaz de administración, y un profesor no tiene por qué saber exactamente cómo se usa la interfaz del alumno. Vamos pues a detallar el funcionamiento para el alumno:

UTILIZACIÓN DESDE LA LÍNEA DE COMANDOS

INTRODUCCIÓN

Se presupone ya en esta sección que el sistema ya está preparado para usar los clasificadores (JRE instalado y Weka y Oaidtb en el classpath).

A continuación se describen los parámetros que toman las diversas clases desde la línea de comandos; para ejecutarlas de manera sencilla, se pueden utilizar los scripts *SchemeTemplate*:

$OAIDTB_HOME/bin/linux/SchemeTemplate

```
#!/bin/bash
#Plantilla de ejemplo que muestra cómo construir y evaluar un clasificador desde la línea de comandos
#Se puede utilizar en conjunción con el GOE (copiar y pegar la configuración de los clasificadores)
#Con pequeñas modificaciones, con este script se pueden programar experimentos en modo batch

#Especificamos las instancias de entrenamiento
TRAIN_DATA_FILE=../_data_/uci/audiology.arff
#Especificamos el nombre del clasificador y sus opciones
SCHEME_NAME="oaidtb.boosters.AdaBoostECC"
SCHEME_OPTIONS="-x 2 -F
oaidtb.filters.NominalToRandomPermutationOfEvenSplitOCFilter -U 1 -I 10 -S 1 -N 0 -W
weka.classifiers.trees.DecisionStump --"

#Lo juntamos todo
MAIN_CLASS_NAME="$SCHEME_NAME -t $TRAIN_DATA_FILE
$SCHEME_OPTIONS"
export MAIN_CLASS_NAME

#Especificamos un archivo donde guardar los resultados
OUTPUT_FILE=../../_results_/ECC_Audiology.txt

#Llamamos al script de ejecución
./_oaidtb_ $OUTPUT_FILE
```

$OAIDTB_HOME/bin/windows/SchemeTemplate.bat

```
:: Ejemplo de uso de los clasificadores desde la línea de comandos
:: Se puede utilizar en conjunción con el GOE (copiar y pegar la configuración de los clasificadores)
:: Con pequeñas modificaciones, con este script se pueden programar experimentos en modo batch
@echo off

SET TRAIN_DATA=..\..\_data_\uci\iris.arff
SET SCHEME_NAME=oaidtb.boosters.AdaBoostECC
SET SCHEME_OPTIONS=-F oaidtb.filters.NominalToRandomPermutationOfEvenSplitOCFilter -U 1 -I 100 -S 1 -N 0 -W weka.classifiers.trees.DecisionStump --

SET MAIN_CLASS_NAME= %SCHEME_NAME% -t %TRAIN_DATA% %SCHEME_OPTIONS%

::SET OAIDTB_OUTPUT_FILE=..\..\_results_\ECC_Iris.txt
SET OAIDTB_OUTPUT_FILE=

call _oaidtb_.bat %OAIDTB_OUTPUT_FILE%
```

A continuación se enumeran todos los boosters disponibles junto con las opciones que los gobiernan. Recordar que las opciones son "case sensitive", es decir, no es lo mismo –V que –v.

OPCIONES COMUNES A TODOS LOS CLASIFICADORES

Todos los clasificadores de weka pueden ser llamados desde la línea de comandos, y todos comparten una serie de opciones comunes; son las siguientes:

-t <nombreDeArchivo> (requerido)
Nombre del archivo, en formato ARFF, donde están guardados los datos de entrenamiento

-T <nombreDeArchivo>
Nombre del archivo, en formato ARFF, en el que se encuentran los datos de prueba; si no se especifican, se realiza una validación cruzada.

-c <índice>
Índice del atributo clase (1, 2, ...; por defecto, el último).

-x <número DeConjuntos>
El número de conjuntos para la validación cruzada (por defecto 10).

-s <semilla>
La semilla del generador de números aleatorios para construir los conjuntos para la validación cruzada (por defecto 1). <p>

-m <nombreDeArchivo>
El nombre de un archivo que contenga una matriz de costes.

-l <nombreDeArchivo>
Carga el clasificador desde el archivo especificado

-d <nombreDeArchivo>
Salva el clasificador construido con los datos de entrenamiento en el archivo especificado

-v
No produce estadísticas para los datos de entrenamiento

-o
Sólo muestra las estadísticas, no el clasificador

-i
Produce estadísticas de las estadísticas de recuperación de información por clase

-k
Produce estadísticas sobre la teoría de la información

-p <rango>
Produce predicciones para las instancias de test junto con los atributos en el rango especificado (y nada más). Utilice '-p 0' si no desea nostrar ningún atributo

-r
Muestra la distribución marginal acumulada (y nada más)

-g
Sólo para clasificadores que implementen el interfaz "*Graphable"* , muestra la representación gráfica del clasificador (y nada más).

OPCIONES COMUNES A TODOS LOS BOOSTERS

Todos los boosters de oaidtb pueden ser llamados desde la línea de comandos, y todos (salvo AdaBoostMH) comparten una serie de opciones comunes; son las siguientes:

-W <*clasificador base*> (requerido)
Especificar el clasificador base que, dependiendo del algoritmo, será un Classifier o un DistributionClassifier de weka.

Las opciones específicas del clasificador base se pasan después del símbolo –identifican

Por ejemplo, en:

"-W weka.classifiers.trees.j48.J48 -- -C 0.25 -M 2".

Las opciones –C 0.25 y –M 2 son parseadas por J48, no por el booster en cuestión.

-D
Mostrar información relativa al proceso de construcción del clasificador durante el mismo

-N <*factorDeNormalización*>
Configurar la política de normalización de la suma de los pesos del conjunto de entrenamiento tras terminar cada iteración; las opciones disponibles son:

*****factorDeNormalización*** < 0
à No se normalizarán los pesos

*******factorDeNormalización* == 0

à Tras cada iteración, la suma de pesos será la misma que tuviera el conjunto original de instancias (por defecto)

**factorDeNormalización > 0

à Tras cada iteración, la suma de pesos será el propio valor de *factorDeNormalización*

-I <*numIteraciones*>
Especificar el número de iteraciones a realizar como máximo.

-Q
Forzar la utilización del remuestreo incluso si el clasificador base puede manejar instancias con peso asociado.

-S <*semillaDeMuestreo*>
Especificar la semilla que se utilizará para el generador de números aleatorios que se usa en la construcción de cada muestra (por defecto 0).

NOTAS SOBRE EL RENDIMIENTO

- *La opción –N influye notablemente en la velocidad de iteración, sobre todo en la medida que el conjunto de entrenamiento sea mayor o menor; llamarla con parámetro <0 (deshabilitar la normalización) puede acelerar notablemente las cosas.*

- *Por otro lado, hacer que los pesos de las instancias evolucionen sin control puede hacer que ciertos clasificadores base de weka lleguen a fallar abortando así el proceso de construcción del ensemble; por ejemplo, esto pasa con los clasificadores DecisionStump e IBk en los casos en que los pesos de las instancias de una cierta clase tienden a hacerse muy pequeños debido a que son muy fáciles de clasificar, dando errores del tipo "Can't normalize, sum is zero"; en estos casos, hacer que se normalicen los pesos de las instancias a un número elevado, pongamos, por ejemplo 100000, puede retrasar que se produzca este caso degenerado y permitir así construir ensembles con un mayor número de clasificadores base.*

- *Este último error también puede corregirse en ocasiones forzando, con la opción –Q,, la utilización de remuestreo, lo que hace que en todas las iteraciones las instancias que se pasan al clasificador base tengan un peso de 1, suficiente en la mayoría de los casos para que los clasificadores de weka trabajen de modo seguro; no obstante, esta aproximación tiene dos inconvenientes bastante importantes: el remuestreo es una operación costosa tanto en tiempo como en espacio para conjuntos de datos de cardinalidad elevada, y además puede surgir el problema de que, al llegar a excluirse las clases más sencillas de clasificar por tener un peso pequeño en proporción a las instancias del resto de las clases, se produzca nuevamente el error por ser el peso de dicha clase tan pequeño, por lo que nos encontramos otra vez con el mismo problema.*

- *La última opción es meterse con el código del clasificador base para blindarlo ante estos errores o mandar un mensaje a los desarrolladores del mismo con la esperanza de que los solucionen…*

ADABOOSTM1 Y ADABOOSTM1W

Las clases *oaidtb.boosters.AdaBoostM1* y *oaidtb.boosters.AdaBoostM1W* implementan los algoritmos AdaBoostM1 y AdaBoostM1W, cuya descripción puede encontrar el lector en los artículos:

Yoav Freund and Robert E. Schapire: *A decision-theoretic generalization of on-line learning and an application to boosting*. Journal of Computer and System Sciences, 55(1):119-139, 1997.

y

Günther Eibl & Karl Peter Pfeiffer *How to make AdaBoost.M1 work for weak base classifiers by changing only one line of the code.* ECML'02 - European Conference on Machine Learning

respectivamente.

Las opciones específicas para estos boosters son las siguientes:

-B
Calcular el límite superior del error de entrenamiento (en AdaBoostM1) o del error de adivinación, que indica la proporción de ejemplares de entrenamiento en los que el ensemble actúa peor que la clasificación aleatoria (en AdaBoostM1W), garantizados de manera teórica.

-E <*bigErrorValue*>
Configurar el valor que será considerado como un error "demasiado elevado" en el proceso de entrenamiento de los clasificadores base y que será, por tanto, usado como criterio de parada del algoritmo en combinación con las opciones –C y –V; posibles opciones

- Si *bigErrorValue* es menor o igual a cero, se reseteará al valor por defecto (0.5)
- Si *bigErrorValue* es mayor que 1, no se aplicará ningún criterio de parada

-C <*maxNumOfBigErrors*>
Configurar el número de veces (por defecto 1) en que permitimos que se produzca un error del clasificador base menor o igual que el error configurado con la opción –E antes de para el proceso de iteración; posibles opciones.

- Si es menor o igual a cero, no se aplicará criterio de parada alguno

-V
Indica que las ocurrencias de los "errores demasiado elevados" no necesitan ser consecutivas para provocar la parada.

EJEMPLO DE USO

La siguiente línea de comandos

```
java oaidtb.boosters.AdaBoostM1 –t iris.arff  -D -B -C 1 -I 10  -N 0 -W weka.classifiers.lazy.IBk -- -K 4
```

- Construye un clasificador utilizando el archivo de datos "*iris.arff*" para leer las instancias de entrenamiento,
- -D **à** Saca información detallada del proceso de iteración a través del stream de error del sistema operativo -por lo general la consola-
- -B **à** Calcula el límite superior del error de entrenamiento según la fórmula probada en los teoremas del artículo

- -C 1 **à** Sólo permite que se produzca un error considerado como demasiado elevado, en este caso el valor por defecto
- -I 10 **à** El booster hará 10 iteraciones
- -N 0 **à** La suma de pesos de las instancias se normalizará a la suma en el conjunto inicial tras cada iteración
- -W weka.classifiers.lazy.IBk -- -K 4 **à** El clasificador base será un "*Instance Based*" con 4 vecinos
- Hace una validación cruzada del clasificador utilizando 10 conjuntos (por defecto)

ADABOOSTOC Y ADABOOSTECC

Las clases *oaidtb.boosters.AdaBoostOC* y *oaidtb.boosters.AdaBoosECC* implementan los algoritmos AdaBoostOC y AdaBoostECC, cuya descripción puede encontrar el lector en los artículos:

Robert E. Schapire. *Using output codes to boost multiclass learning problems*. In Machine Learning: Proceedings of the Fourteenth Internatinal Conferences, pages 313-321, 1997.

y

Venkatesan Guruswami and Amit Sahai. *Multiclass Learning, Boosting and Error Correcting Codes*. Proceedings of COLT'99

respectivamente.

Las opciones específicas para estos boosters son las siguientes:

-F <*filtroOC*>
Especifica el nombre de la clase (AbstractNominalToOCFilter) que se utilizará para generar el coloring

-U <*numIterations*>
Configura el número de reintentos en la búsqueda de un coloring capaz de producir un parámetro U >= 1/2 (por defecto 1)

-B
Calcular el límite superior del error de entrenamiento garantizado por los teoremas en los artículos (por defecto falso).

-A
Opción específica de AdaBoostECC, especifica que se use la versión simétrica en vez de la asimétrica (esta última es la que funciona mejor) (por defecto falso)

GENTLEADABOOST Y REALADABOOST

Las clases *oaidtb.boosters.GentleAdaBoost* y *oaidtb.boosters.RealAdaBoost* implementan los algoritmos GentleAdaBoost y RealAdaBoost, cuya descripción puede encontrar el lector en el artículo:

Jerome Friedman, Trevor Hastie and Robert Tibshirani: *Additive logistic regression: a statistical view of boosting*. The Annals of Statistics, 38(2): 337-374, April 2000.

Estos boosters no tienen opciones de configuración particulares, pero sí particularidades:

- Sólo pueden actuar problemas binarios (de dos clases), aunque pueden ser extendidos al caso multiclase utilizando *AdaBoostMH.*
- El clasificador base de GentleAdaBoost debe ser capaz de hacer predicciones numéricas.
- El clasificador base de RealAdaBoost debe ser capaz de devolver distribuciones de probabilidades (ser un *DistributionClassifier*).

ADABOOSTMH

La clase *oaidtb.boostersAdaBoostMHt* implementa el algoritmo AdaBoostMH, cuya descripción puede encontrar el lector en el artículo:

Jerome Friedman, Trevor Hastie and Robert Tibshirani: *Additive logistic regression: a statistical view of boosting*. The Annals of Statistics, 38(2): 337-374, April 2000.

Esta clase NO es un booster, por lo que las opciones comunes a todos los boosters no son aceptadas. Sin embargo, usa un Booster para construir un ensemble para cada clase del conjunto original. Dicho Booster debe implementar el interfaz de oaidtb *MulticlassExtensibleBooster*, que actualmente implementan en la librería las clases *AdaBoostM1, AdaBoostM1W, GentleAdaBoost y RealAdaBoost*.

Las opciones específicas para este clasificador son:

-B <multiclassExtensibleBooster>(requerido)
Especificar una clase (debe ser un MulticlassExtensibleBooster) para usar como el booster base.

Los parámetros se pasan al "booster base" de manera análoga a cómo se pasan los parámetros al clasificador base en los boosters normales: después de --.

EJEMPLO DE USO

La siguiente línea de comandos

```
java oaidtb.boosters.AdaBoostMH –t iris.arff -B oaidtb.boosters.RealAdaBoost -- -I 10 -S 1 -N 0 -W
weka.classifiers.trees.DecisionStump --
```

- Construye un clasificador utilizando el archivo de datos "*iris.arff*" para leer las instancias de entrenamiento,
- -B oaidtb.boosters.RealAdaBoost **à** Utiliza RealAdaBoost como clasificador base
- -I 10 **à** Cada booster hará 10 iteraciones (por tanto se harán 10xnúmero de clases de iris iteraciones de un RealAdaBoost)
- -N 0 **à** La suma de pesos de las instancias se normalizará a la suma en el conjunto inicial tras cada iteración
- -W weka.classifiers.trees.DecisionStump -- **à** El clasificador base de cada RealAdaBoost será un DecisionStump sin parámetros
- Hace una validación cruzada del clasificador utilizando 10 conjuntos (por defecto)

BOOSTERS SENSIBLES AL COSTE

Las clases del paquete *oaidtb.boosterscostSensitive:*

* *CSB0, CSB1, CSB2*
* *AdaCost, AdaCostB1 y AdaCostB2*

Son los boosters sensibles al coste implementados actualmente en oaidtb, y su descripción se encuentra en:

Ting, K.M., *Cost Sensitive Classification Using Decision Trees, Boosting and MetaCost*. Book chapter in Heuristic and Optimization for Knowledge Discovery. Edited by Sarker, R., Abbass, H. & Newton, C. Idea Group Publishing. 2002.

Estos algoritmos sólo pueden usar clasificadores base que sean DistributionClassifier; las opciones comunes son:

-M <*combined prediction selection model*>
Especificar cómo se deben combinar los clasificadores base para formar la hipótesis combinada final; las opciones válidas son:
* MVC = Criterio del máximo voto
* MVC_UCL = Criterio del máximo voto utilizando los niveles de confianza del clasificador base.
* MECC = Criterio del mínimo coste esperado
* MECC_UCL = Criterio del mínimo coste esperado utilizando los niveles de confianza del clasificador base (por defecto).

-C <archivo con la matriz de costes>
Nombre del archivo que se utilizará para cargar la matriz de costes. Si esta opción no es suministrada se intentará cargar una matriz de costes por defecto, cuyo nombre es el nombre de la relación de las instancias de entrenamiento con extensión ".cost", y la ruta a dicho archivo se especifica con la opción -O

-O <directorio>
Ruta del directorio donde se buscará la matriz de costes por defecto; si no se especifica se buscará en el directorio actual.

-U
Inicializar los pesos de las instancias de entrenamiento utilizando los costes.

-F
Utilizar la matriz de costes por defecto (todos los costes == 1 salvo los costes de clasificar incorrectamente la clase minoritaria, que valdrá el valor especificado por la opción –X.

-X <*costFactor*>
El coste de clasificar erróneamente la clase minoritaria cuando se usa la matriz de costes por defecto.

CSADABOOSTMH

Esta clase implementa una modificación de *AdaBoostMH* específica para la clasificación sensible al coste; utiliza un booster *AbstractCSB* para construir una hipótesis por cada clase, proporcionándole una matriz de costes de 2x2 donde el elemento (0,1) es el coste de clasificar (mal) cualquier otra clase como la clase en cuestión, y el elemento (1,0) es el coste de clasificar (mal) la clase en cuestión como cualquier otra clase.

-C <archivo con la matriz de costes> **à** Ver *Boosters Sensibles al Coste*
-D <directorio> **à** Ver *Boosters Sensibles al Coste*
-F **à** Ver *Boosters Sensibles al Coste*
-X <*costFactor*> **à** Ver *Boosters Sensibles al Coste*

-B classname
Especificar el "booster base" (debe ser un *AbstractCSB*, es decir, uno de los boosters explicados en la sección *Boosters Sensibles al Coste*).

LAS HERRAMIENTAS GRÁFICAS

EL EDITOR GENÉRICO DE OBJETOS (GOE)

Este programa es un editor visual de las propiedades de objetos que se hayan definido a sí mismos como editables en un archivo de configuración *"GenericObjectEditor.props"*, en el que se listan los posibles valores que pueden ser seleccionados para su edición.

Concretamente este es el componente que nos servirá para configurar visualmente cualquiera de los clasificadores de weka y de oaidtb. Está pensado para ser utilizado dentro de las aplicaciones visuales creadas, pero por sí solo tiene una utilidad muy clara: crear la línea de comandos a utilizar para realizar un experimento a través del SHELL del sistema operativo.

- ¿Cómo se ejecuta?

Si se quiere ejecutar usando la línea de comandos:
java oaidtb.gui.customizedWeka.GenericObjectEditor

Usando los scripts provistos:
Windows:
%OAIDTB_HOME%\bin\windows\GOE.bat
Linux
$OAIDTB_HOME/bin/linux/GOE

El archivo de configuración *GenericObjectEditor.props* debe encontrarse al menos en uno de los directorios siguientes: user.home ó el directorio actual (este último tiene mayor preferencia); además siempre se lee un archivo de configuración por defecto de la distribución de weka. El archivo *GenericObjectEditor.props* que se distribuye con este producto reside en el directorio $OAIDTB_HOME/bin/cfg.

Para añadir un nuevo tipo de clases a configurar basta con editar dicho archivo; por ejemplo, la siguiente sección añadida a dicho archivo le dice al GOE cuáles son las posibles clases a elegir si el tipo de la superclase que se está editando es *oaidtb.filters.AbstractNominalToOCFilter*:

```
# Lists the AbstractNominalToOCFilters I want to choose from
oaidtb.filters.AbstractNominalToOCFilter = \
  oaidtb.filters.NominalToRandomEvenSplitOCFilter, \
  oaidtb.filters.NominalToRandomOCFilter, \
  oaidtb.filters.NominalToRandomPermutationOfEvenSplitOCFilter
```

Si lo que queremos es, por ejemplo, crear una lista de las clases que debe añadir al conjunto de clases editables cuando se encuentre una propiedad de tipo *Booster* al editar un objeto de la clase AdaBoostMH las siguientes líneas se encargan de hacer el trabajo:

```
# Lists the Boosters I want to choose from to use with de AdaBoostMH class
oaidtb.boosters.Booster@oaidtb.boosters.AdaBoostMH=\
 oaidtb.boosters.AdaBoostM1,\
 oaidtb.boosters.AdaBoostM1W,\
 oaidtb.boosters.RealAdaBoost,\
 oaidtb.boosters.GentleAdaBoost
```

Para ver un ejemplo detallado de este archivo de configuración mirar en $OAIDTB_HOME/bin/cfg/GenericObjectEditor.props

La ventana del GOE tiene el aspecto que se muestra en la figura; como vemos, da facilidades para configurar cada una de las propiedades de la clase que se edita (en la figura es el Booster AdaBoostECC).

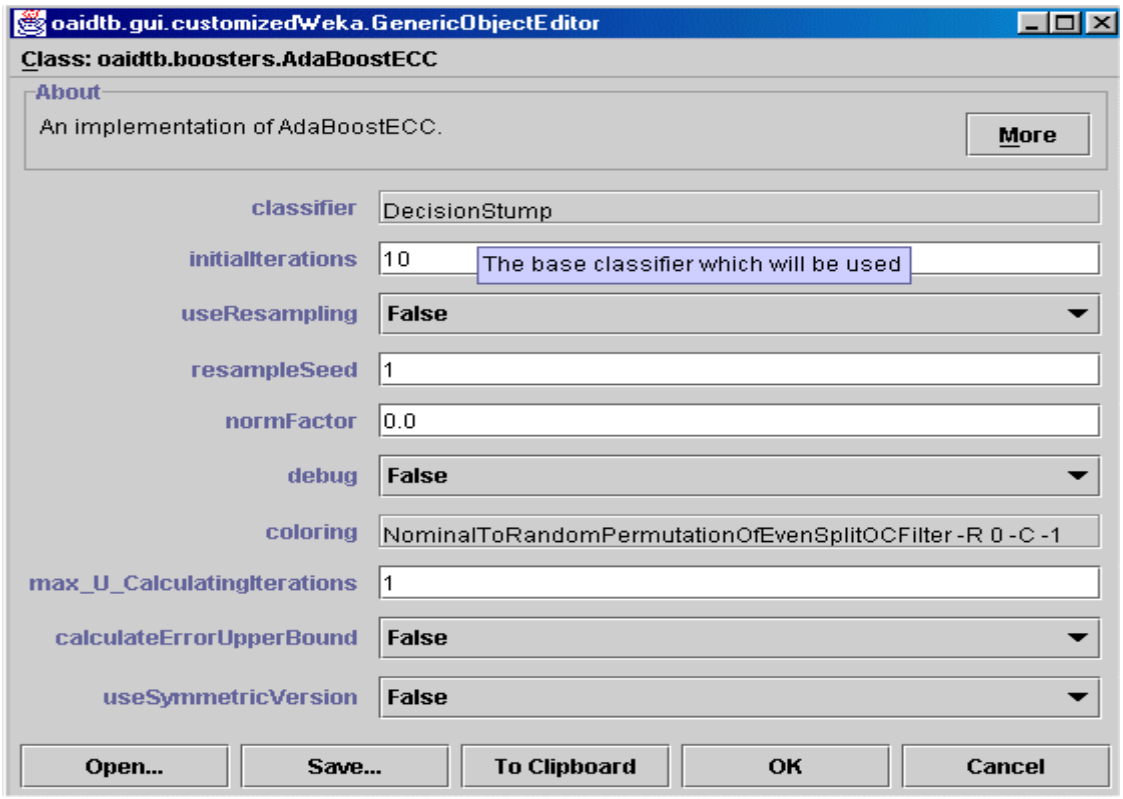

Figura 2 - Apariencia del editor genérico de objetos

Si se hace click en la línea que pone “Class: oaidtb.boosters.AdaBoostECC”, o se pulsa Alt+C, se despliega un menú en el que se pueden elegir las clases a editar.

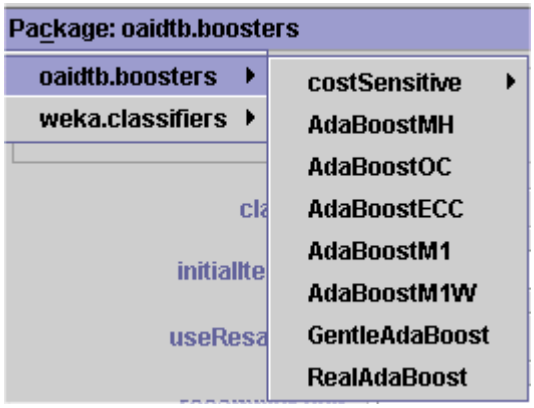

Figura 3 - Menú desplegable del GOE

En la sección de *About* se muestra, si el objeto que se está editando la provee, una pequeña descripción de la clase; además hay un botón *"More"* que si se pulse se mostrará la información que proporcione el objeto sobre lo que es y las opciones que tienen cada una de las propiedades que se pueden editar.

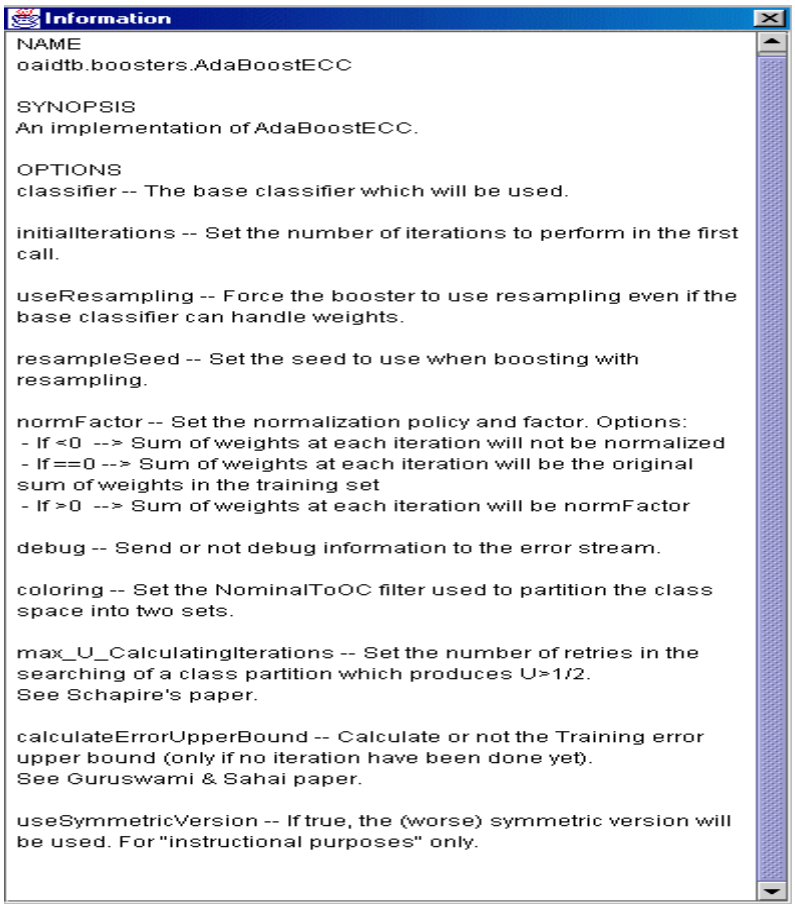

Figura 4 - Ventana de información extra del GOE

Cada una de las líneas de la región central del panel muestra la configuración de alguna de las propiedades editables (encontradas mediante un proceso de introspección) de la clase que se esté editando: a la izquierda el nombre de la propiedad y a la derecha su valor actual; algunas clases (por ejemplo, todos los boosters de oaidtb) proveen al GOE de descripciones cortas de las acciones para ser mostradas como "tooltips" (pequeña ayuda que se muestra al dejar el puntero del ratón sobre alguna región). Pulsando con el ratón sobre la propiedad se puede editar su

valor; dependiendo del tipo de la propiedad, la edición se hará introduciendo un valor (si es un número), eligiendo de un conjunto de opciones, lanzando otra ventana del GOE o con cualquier otro método según proceda; por ejemplo, esto es lo que sucede cuando decidimos editar el clasificador base para que sea un *Ibk.*

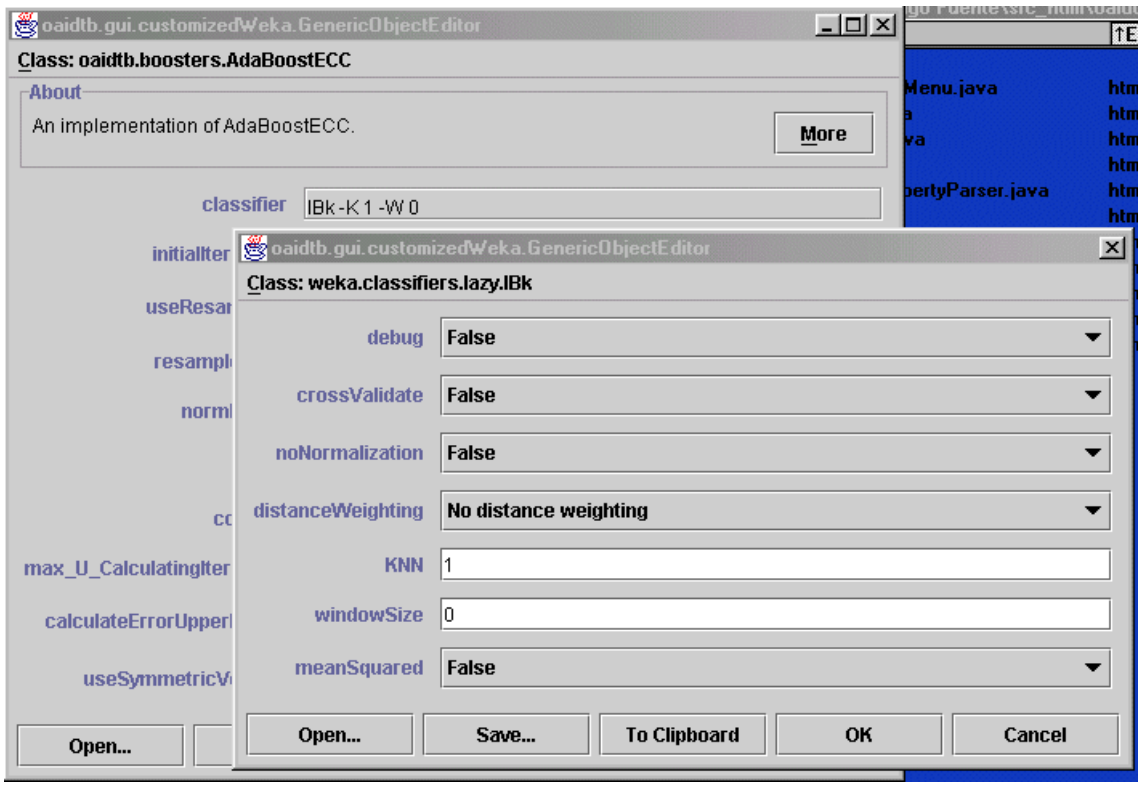

Figura 5 - Un nuevo GOE se abre para editar las propiedades del clasificador base

Por último, los botones inferiores sirven para:

- **Open**: Cargar un objeto guardado en un archivo.
- **Save**: Salvar el objeto que se está editando (junto con su configuración) a un archivo.
- **To Clipboard*:*** copiar el nombre del objeto al portapapeles del sistema; si, además, el objeto que se está configurando es un *OptionHandler* de weka (todos los clasificadores de weka y de oaidtb lo son) también se copian los argumentos que produce la misma configuración desde la línea de comandos.
- **OK**: Aceptar la selección actual (para la integración en otras herramientas visuales).
- **Cancel**: Cancelar el proceso de selección / configuración (para la integración en otras herramientas visuales).

Por el momento la función que nos interesa es la de copiar al portapapeles la descripción del objeto, pues permite crear de una manera cómoda y sin la necesidad de consultar guías de referencia las complicadas expresiones que pueden aparecer en la línea de comandos para configurar de manera adecuada un clasificador; por ejemplo, para el clasificador *AdaBoostECC* de la imagen lo que se copia al portapapeles es:

oaidtb.boosters.AdaBoostECC -F
oaidtb.filters.NominalToRandomPermutationOfEvenSplitOCFilter -U 1 -I 10 -S 1 -N 0
-W weka.classifiers.lazy.IBk -- -K 1 -W 0

EL PANEL DE CLASIFICACIÓN

Panel que permite seleccionar un clasificador, configurarlo, entrenarlo con los datos que se desee, evaluarlo y guardarlo.

- ¿Cómo se ejecuta?

Si se quiere ejecutar usando la línea de comandos:
java oaidtb.gui.classifierPanel$Test
(en unix/linux, java oaidtb.gui.classifierPanel\$Test)

Usando los scripts provistos:
Windows:
%OAIDTB_HOME%\bin\windows\ClassifierPanel.bat
Linux
$OAIDTB_HOME/bin/linux/ClassifierPanel

Una vez que lo hemos ejecutado aparece la siguiente pantalla:

Figura 6 - El panel de clasificación

La aplicación es muy sencilla, y cada componente muestra un "tooltip" que explica su función.

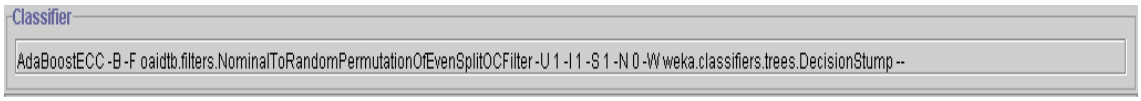

Figura 7 – panel que muestra el clasificador de trabajo

En la parte superior hay un botón en el que se muestra el clasificador sobre el que estamos trabajando; si se pulsa sobre él aparecerá un GOE que permite cambiarlo y/o configurarlo.

Figura 8 – Botones de manejo de instancias y de "About"

En la parte inferior del panel hay cinco botones:

- **Load train data:** Este botón muestra un selector de ficheros que permite seleccionar un archivo en formato ARFF que contenga las instancias de entrenamiento
- **Load test data:** Este botón muestra un selector de ficheros que permite seleccionar un archivo en formato ARFF que contenga las instancias de prueba
- **Reset train data:** Libera la memoria reservada para las instancias de entrenamiento; resetea el clasificador, las estadísticas y, si el clasificador es un booster, la gráfica de errores.
- **Reset test data:** libera la memoria reservada para los datos de test y los experimentos que se hagan ya no las tendrán en cuenta, eliminándose los datos de la gráfica de errores, del módulo de evaluación etc.
- **About:** Muestra la típica ventana "acerca de…" con información sobre la aplicación, la licencia, las bibliotecas que usa, el autor y el sistema sobre el que se está corriendo la aplicación para poder hacer informes de errores fácilmente

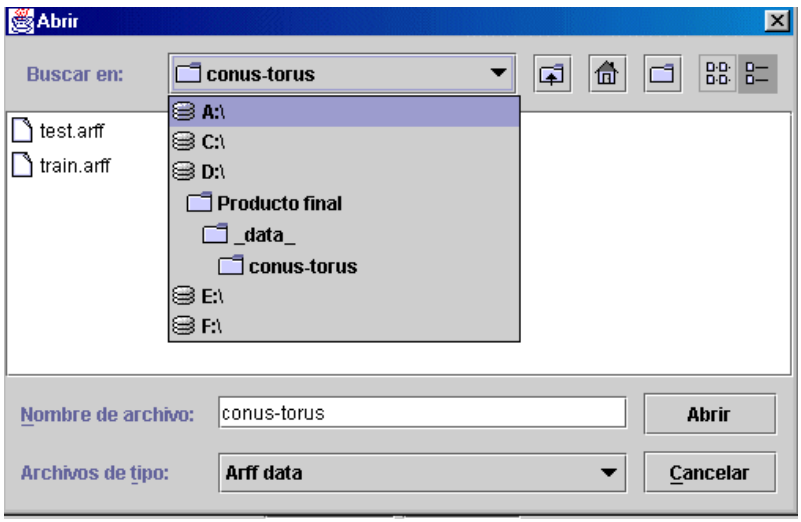

Figura 9 – Diálogo de selección de un archivo ARFF

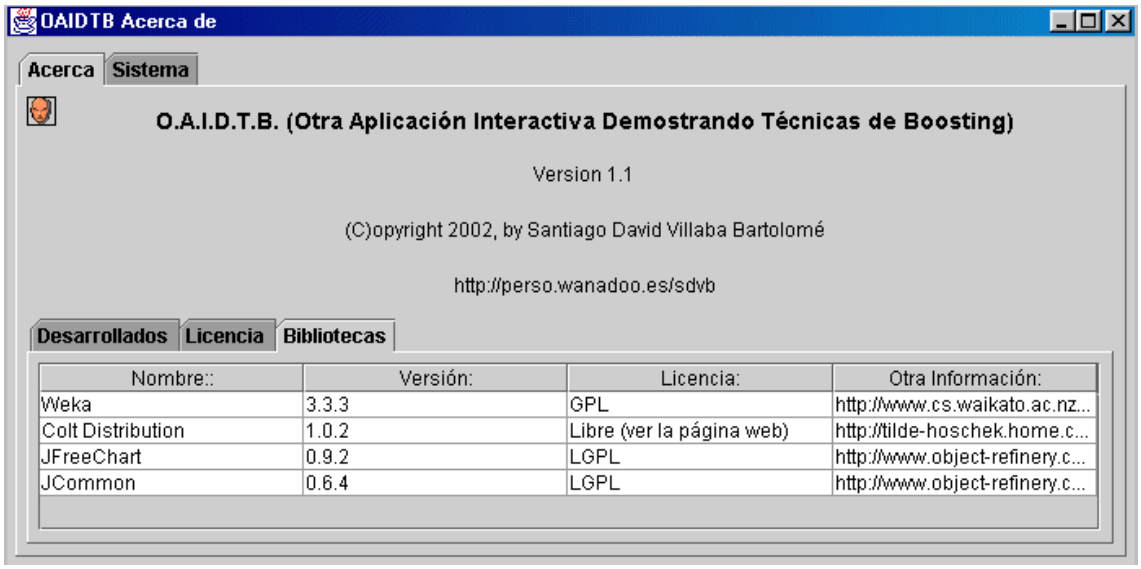

Figura 10 – Ventana "Acerca de…" de la aplicación

Encima de estos botones aparecen unas líneas de texto en las que se muestra el nombre de la relación de trabajo y el número de instancias que contiene.

Figura 11 – Etiquetas de información del conjunto de datos

Encima de estas etiquetas aparece la barra de botones de clasificación, que sirven para controlar el proceso de construcción de los clasificadores.

- **Campo de entrada del número de iteraciones:** Si el clasificador con el que estemos trabajando es uno de los boosters de la librería *oaidtb* se puede controlar el número de iteraciones a realizar cada vez que se pulse el botón de iterar mediante el valor de este campo, si no lo es el campo aparecerá deshabilitado; sólo se permite la entrada de números naturales.

- **Botón de construir el clasificador / iterar:** Este botón permite iniciar el proceso de construcción de un clasificador de weka o realizar las siguientes iteraciones del proceso de construcción de un booster, dependiendo de la naturaleza del clasificador con el que estemos trabajando en cada momento.

- **Botón de cancelación:** Cancelar el proceso de construcción / iteración de un clasificador.

- **Botón de reset:** Permite resetear el estado de un booster que se esté iterando, eliminando las estadísticas recolectadas y reiniciando el gráfico de errores; es útil si, por ejemplo, queremos cambiar alguna de las opciones del booster que requieran que éste no haya realizado ninguna iteración, como por ejemplo cambiar la matriz de costes de uno de los boosters sensibles al coste o ordenar a *AdaBoostECC* que calcule la cota superior teórica del error.

Figura 12 – Controles del proceso de construcción de un clasificador

Encima de estos botones y ocupando prácticamente la totalidad de la pantalla se encuentra el área de información estadística, encargada de mostrar las diversas medidas que de la bondad y la evolución de los clasificadores se estén tomando.

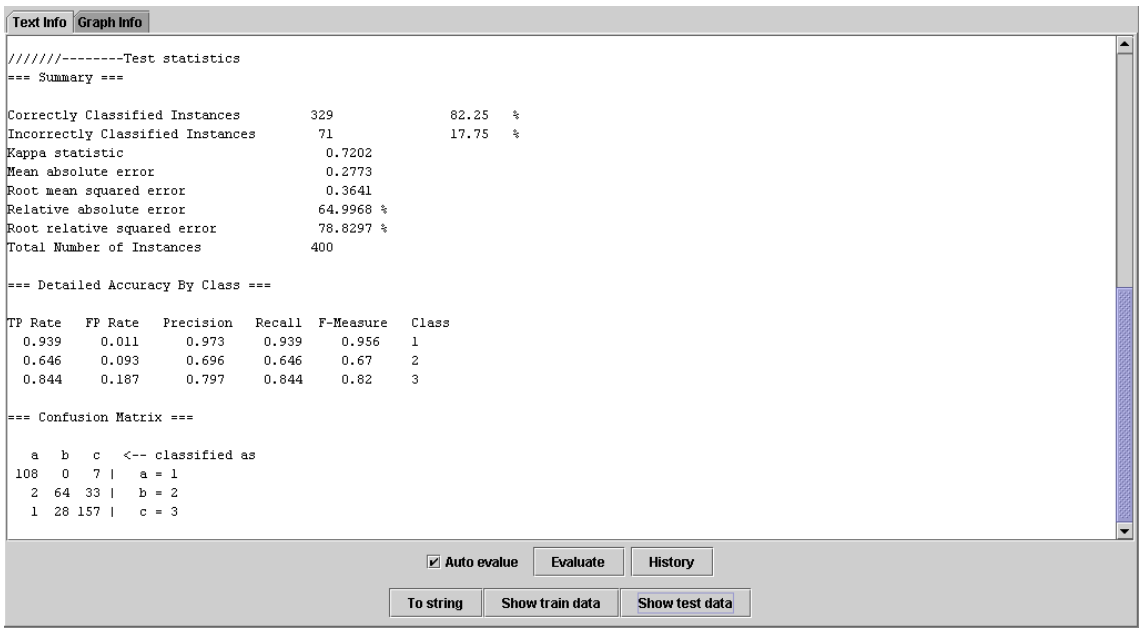

Figura 13 – El área de información estadística, modo texto

Como se aprecia en la figura existen dos pestañas en la parte superior de esta área que permiten elegir entre mostrar información de manera textual o gráfica.

En el primer de los casos, se puede recuperar la información en modo texto de que se mostrará en el área de texto central. Si pulsamos el botón derecho del ratón sobre esta área aparecerá un menú contextual como el de la siguiente figura que permite borrar el texto del área o copiar todo el texto o aquel seleccionado (esto también se puede hacer usando las facilidades del sistema operativo, como pulsar Ctrl-C en Windows) al portapapeles del sistema.

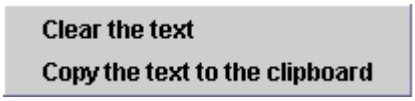

Figura 14 – Menú emergente del área de texto

En la parte inferior del panel de información en modo texto aparecen varios controles que representan las acciones de recolección de información en modo texto que actualmente se pueden realizar con el programa:

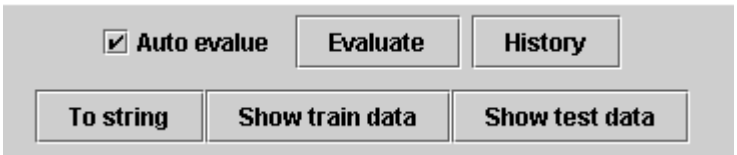

Figura 15 – Controles para seleccionar lo que se representa en el área de texto

- **Auto evalue:** Si está seleccionada esta opción, cada vez que se termine de construir un clasificador o de hacer el número de iteraciones de un booster se evaluará la eficiencia de dicho clasificador sobre los conjuntos de entrenamiento y prueba (si existe) y se volcarán los resultados al área de texto.

- **Evaluate:** Evaluar la eficiencia del clasificador actual sobre los conjuntos de entrenamiento y prueba (si existe) y volcar los resultados al área de texto.

- **History:** Si el clasificador actual es uno de los boosters de la librería oaidtb se mostrará en modo texto cuál ha sido la evolución de dicho booster a lo largo del proceso de iteración.

- **To string:** Mostrar en el área de texto una descripción del clasificador actual

- **Show train data:** Mostrar en el área el conjunto de instancias de entrenamiento (en formato ARFF).

- **Show test data:** Mostrar en el área el conjunto de instancias de prueba (en formato ARFF)

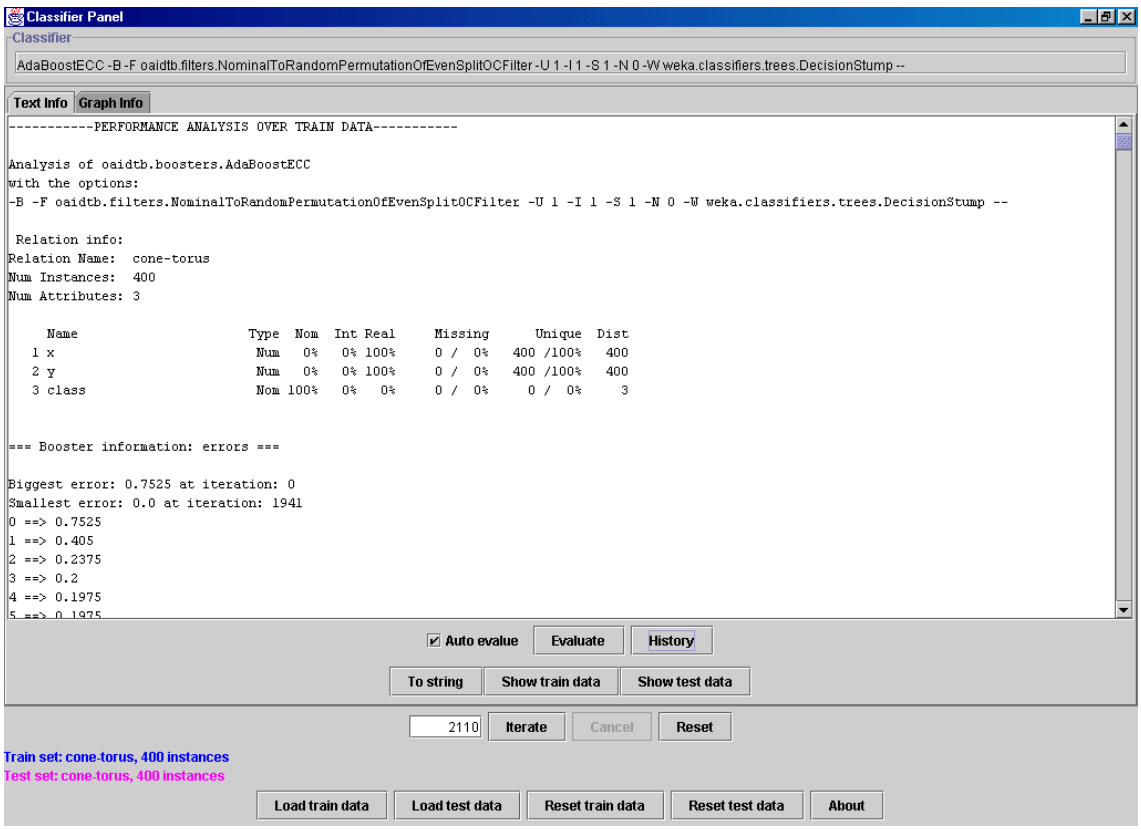

Figura 16 – Información textual del historial de errores de un booster

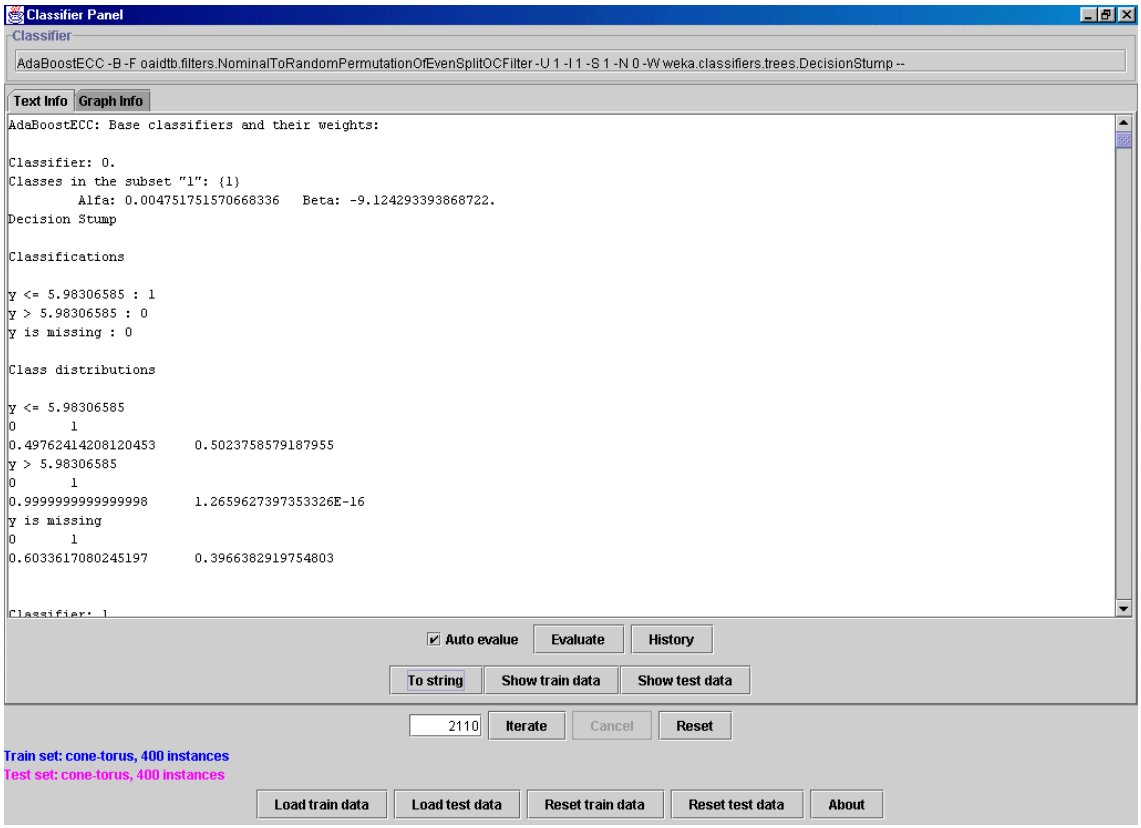

Figura 17 – Representación textual de un clasificador

Este es el aspecto que tiene el panel de información gráfica, que sólo está disponible si el clasificador es uno de los boosters implementados en la biblioteca oaidtb:

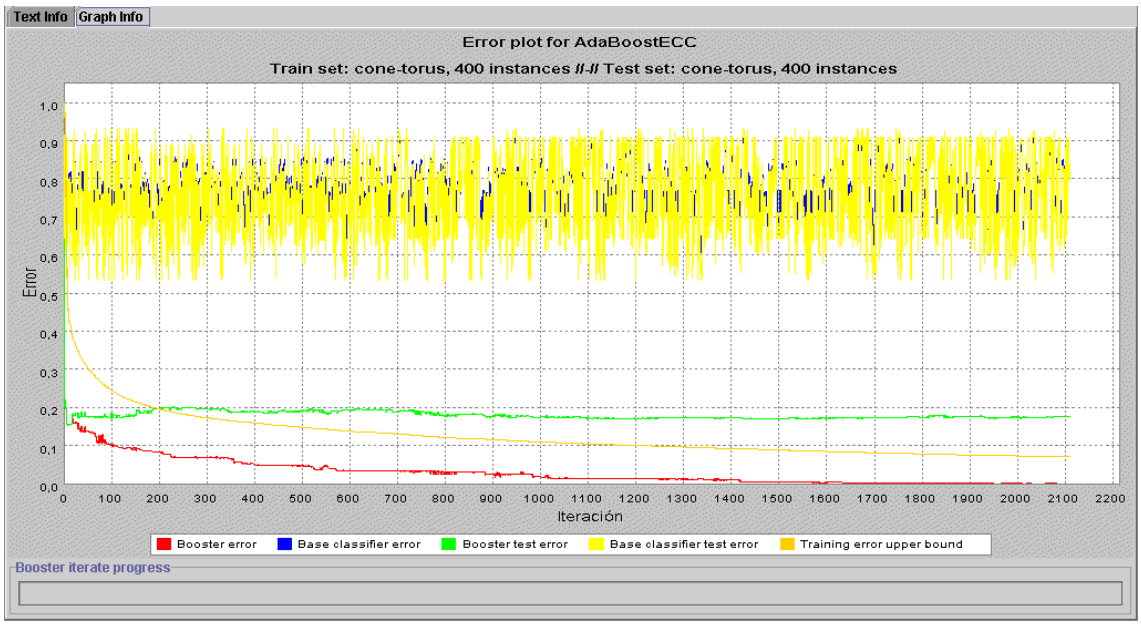

Figura 18 – Gráfica de errores de un booster

En la parte inferior del panel de información gráfica aparece una barra de progreso en la que se muestra qué porcentaje de iteraciones se han realizado durante el proceso de iteración de un booster.

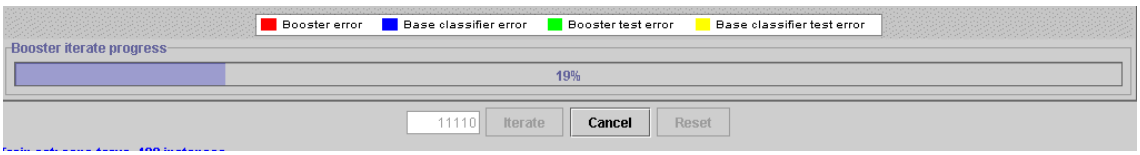

Figura 19 – Barra de progreso del proceso de iteración de un booster

En el centro se muestra un gráfico que contiene la evolución de los errores del booster a lo largo el proceso de iteración. Para elegir qué líneas de las disponibles, dependiendo del tipo de booster y de los datos que haya basta con pulsar el botón izquierdo del ratón sobre cualquier zona del mismo, apareciendo el siguiente diálogo modal.

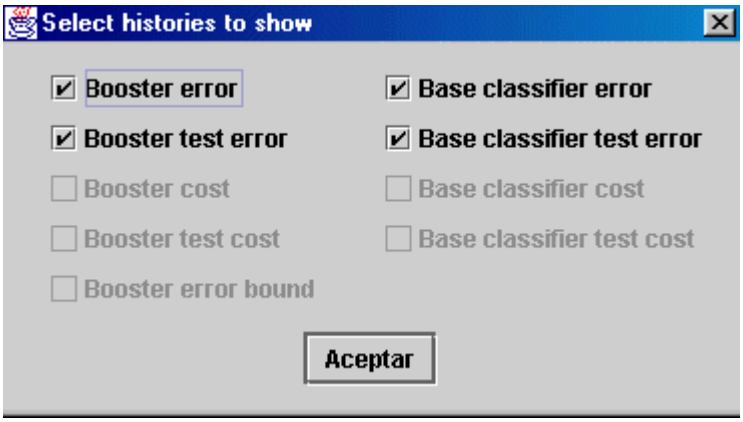

Figura 20 – Selección de las líneas del gráfico a representar

Por otro lado, también es posible configurar la apariencia del gráfico; pulsando el botón derecho del ratón sobre el mismo aparece el siguiente menú emergente:

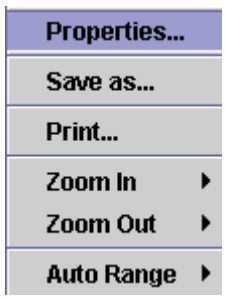

Figura 21 – Menú emergente del gráfico de errores

- **Properties:** Permite configurar (color, fuente etc.) el aspecto de la leyenda, de los ejes, de la rejilla, habilitar el anti-aliased etc.
- **Save as…:** Permite guardar el gráfico como una imagen en formato png (Portable Network Graphics)
- **Print…:** Permite imprimir el gráfico
- **Zoom In:** Permite hacer zoom sobre una región del gráfico

- **Zoom Out:** Permite deshacer el zoom

- **Auto Range:** Permite volver a ver la gráfica a escala natural

PREGUNTAS Y RESPUESTAS

- *¿Dónde se notifican los errores?*

Si se produce algún error en los procesos de carga de instancias, clasificación y/o evaluación este es notificado a través del *errorstream* estándar de java, que por defecto está dirigido a la consola (al menos en windows & linux); en las siguientes versiones del panel estos mensajes se mostrarán a través de un área de log similar a la utilizada en la aplicación "Bidimensional Generalization"; el porqué de esto se encuentra en el hecho de que este panel no fue concebido en principio para ser una aplicación por sí mismo, sino para estar integrado en "Bidimensional Generalization" que sí hace control de los errores, y no ha sido hasta entrar en una fase muy avanzada del proyecto que se ha decidido, dada su gran utilidad, convertirlo en una aplicación individual.

- *¿Se pueden cambiar las opciones de configuración de un booster durante el proceso de iteración?*

En la mayoría de los casos la respuesta es sí. Esto puede ser útil para, por ejemplo, crear boosters con clasificadores base de distinta naturaleza, investigando así la importancia de la diversidad, pudiendo salir de atractores infinitos…

Por otro lado, hay algunas opciones, como la de calcular el límite superior teórico del error, que sólo pueden ser cambiadas si el booster no ha realizado aún iteración alguna; en estos casos, sólo se puede cambiar dicha opción pulsando el botón de reset.

- *¿Por qué Cuando la barra de progreso del booster llega al 100% aún se tarda un buen rato en acabar de iterar?*

La barra de progreso sólo indica cuántas iteraciones del booster se han hecho ya, pero tras acabar el proceso de iteración el sistema automáticamente actualiza las estadísticas de evolución para todas las nuevas iteraciones realizadas –por cada nueva iteración debe evaluar la hipótesis final y el clasificador base para todas las instancias de entrenamiento y test -; además, si está configurada la auto evaluación también se estimará la bondad de la hipótesis final por parte del módulo de evaluación de Weka, y este no conoce de las bondades de las implementaciones de métodos de boosting programadas, por lo que este proceso es aún más lento, en proporción, que el análisis del historial del clasificador. En definitiva, que todo esto se hará opcional en las próximas versiones del programa, y si no se ha hecho antes es por el mismo motivo referente a la "intención inicial" que se ha expuesto en la primera pregunta.

LA APLICACIÓN "GENERALIZACIÓN BIDIMENSIONAL"

Esta aplicación permite la visualización de las hipótesis lanzadas por diversos algoritmos de aprendizaje computacional en el marco de un problema de datos bidimensional y multiclase. En concreto, se trata de clasificar instancias que representan puntos en un espacio bidimensional cuyo atributo clase es el color de dicho punto. Con ella es posible enseñar de una manera práctica y visual cómo funcionan dichos algoritmos, demostrando propiedades teóricas de los mismos y conectando las hipótesis que lanzan con un problema fácil de entender y que "entra por los ojos". Hacer notar que el imprimir este manual en blanco y negro le resta claridad a las explicaciones, por estar la aplicación basada precisamente en el color, por lo que se recomienda su lectura desde el CD.

- ¿Cómo se ejecuta?

Si se quiere ejecutar usando la línea de comandos:
java oaidtb.gui.AllContainerPanel$AppMainFrame
(en unix/linux, java oaidtb.gui.AllContainerPanel\$AppMainFrame)

Usando los scripts provistos:
Windows:
%OAIDTB_HOME%\bin\windows\AllContainerPanel.bat
Linux
$OAIDTB_HOME/bin/linux/AllContainerPanel

En líneas generales la aplicación es muy fácil de usar y los conceptos son muy simples - otra cosa es conocer los fundamentos de los algoritmos de aprendizaje, pero eso se escapa del ámbito de este pequeño manual -. Todos y cada uno de los componentes que la forman es capaz de mostrar un "tooltip" auto describiéndose; además, en cada momento sólo estarán habilitados aquellos componentes a los que se les permita, o se den las condiciones para, que la acción que representan sea ejecutada.

Básicamente el método de uso es siempre el mismo:

- Insertar unas instancias de entrenamiento (con pulsaciones del ratón o desde un archivo)
- Construir un clasificador con dichas instancias
- Crear y visualizar una imagen que represente la hipótesis del clasificador

Sin más preámbulos, pasemos ahora a describir las funcionalidades del programa.

Una vez que lo hemos ejecutado aparece la pantalla de la figura.

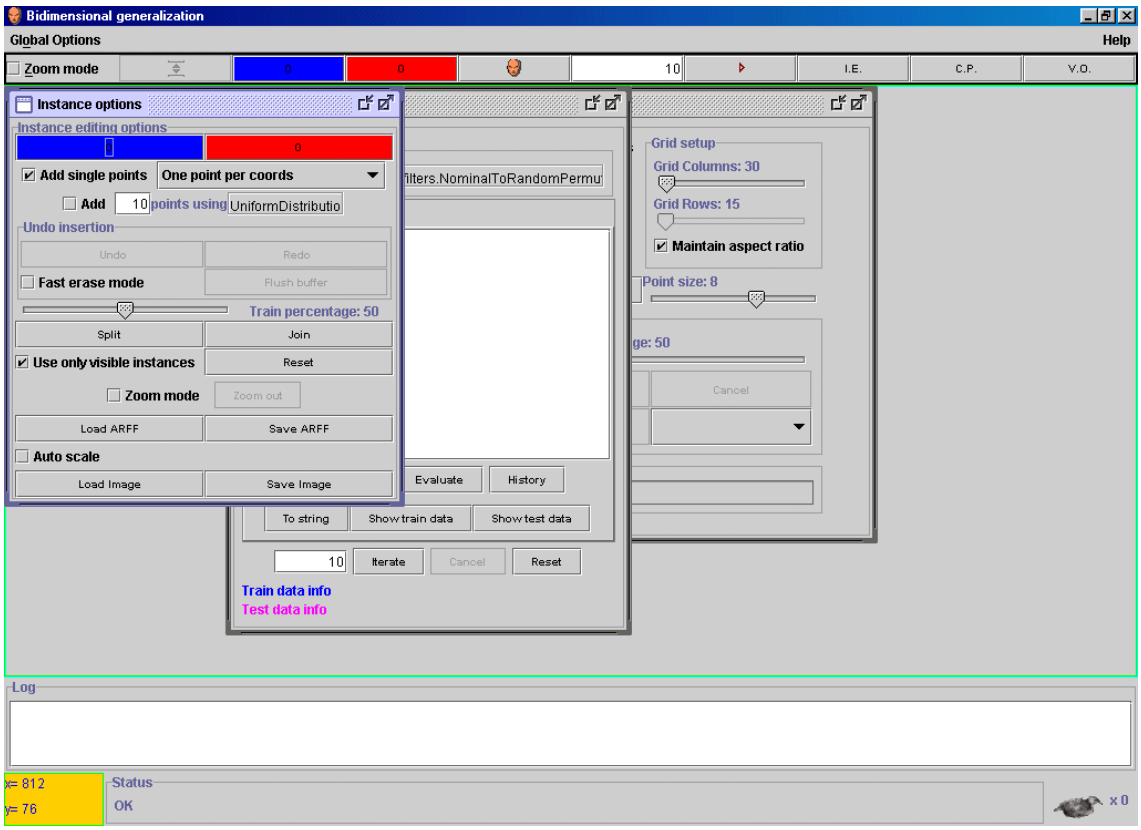

Figura 22 – Pantalla de la aplicación "Generalización Bidimensional"

La pantalla está dividida en varias regiones.

- La región central, delimitada con un borde verde, es el "panel de dibujado", y en él se dibujarán las instancias que formen el problema y las imágenes que representen el clasificador

- Al sur está el panel de información, formado por un panel de log donde se mostrarán todos los mensajes para el usuario generados por la aplicación, una barra de estado en la que se muestran las tareas "pesadas" que se estén realizando en cada momento, un "pajarito inquisitivo" que se anima indicando que una o más de estas tareas "pesadas" se están ejecutando, y dos etiquetas en las que se muestran las coordenadas del dominio del problema que representa el punto del panel de dibujado sobre el que se encuentra el puntero del ratón.

- En la parte superior nos encontramos con una barra de menú, donde se pueden configurar las opciones globales de la aplicación, y con una barra de herramientas con botones de acceso rápido a algunas de las funciones más utilizadas.

- Hay tres paneles flotando sobre el área de dibujado para permitir que ésta sea lo más grande posible y no interferir con ella. En estos paneles se concentran todas las funcionalidades de la aplicación, y son: el panel de edición de instancias, el panel de configuración del clasificador (que ya conocemos) y el panel de opciones visuales.

Veamos los detalles de uso / utilidad de cada componente.

LA BARRA DE MENÚ

En la parte superior de la ventana aparece una barra de menú que contiene las opciones generales del programa y una acceso a la ventana "*About"* ya explicada en el apartado del Panel de Clasificación (dicha ventana se comparte entre ambas aplicaciones).

Si desplegamos el menú de opciones globales del programa nos encontramos con el siguiente panorama:

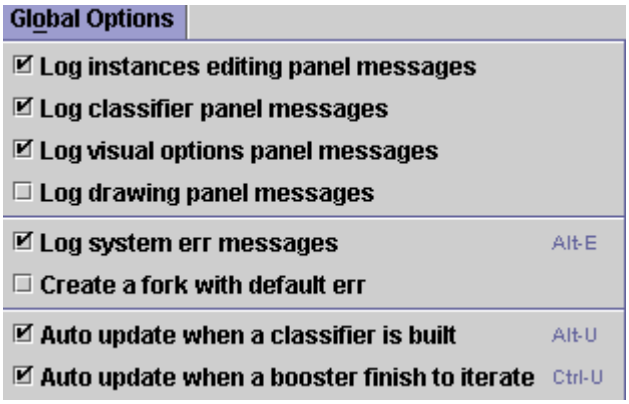

Figura 23 – Menú de opciones globales

- **Log instances editing panel messages:** Mandar los mensajes generados por el panel de edición de instancias al área de log.

- **Log classifier panel messages:** Mandar los mensajes generados por el panel de configuración del clasificador al área de log.

- **Log visual options panel messages:** Mandar los mensajes generados por el panel de opciones visuales al área de log

- **Log drawing panel messages:** Mandar los mensajes generados por el panel de dibujado al área de log

- **Log system err messages** (**Alt-E**)**:** Redirigir los mensajes que se manden a System.err – en los clasificadores de weka y en los boosters de oaidtb son numerosos, tanto los que avisan de excepciones como de información concerniente al proceso de construcción y/o clasificación -- al área de log. Estos mensajes se diferencian de los normales en el área de log porque se escriben en azul y versalita.

- **Create a fork with default err:** Sólo si "Log system err messages" está seleccionado, mandar los mensajes emitidos a través de System.err al stream de error estándar, generalmente la consola o un archivo al que se haya redirigido dicho stream.

- **Auto update when a classifier is built (Alt-U):** Ordenar la acción de crear la imagen de hipótesis inmediatamente después de haber terminado la construcción del clasificador.

- **Auto update when a booster finish to iterate (Ctrl-U):** Ordenar la acción de crear la imagen de hipótesis inmediatamente después de haber terminado la iteración de uno de los boosters de oaidtb.

LA BARRA DE HERRAMIENTAS

El programa proporciona una barra de herramientas con controles de acceso rápido a algunas de las acciones más comúnmente utilizadas en el programa; aunque sea echar piedras contra mi propio tejado, he de decir que esta barra está muy incompleta, ya que durante los largos ratos que me he pasado probando el programa he notado que la frecuencia de mostrar / ocultar las ventanas de herramientas flotantes para ejecutar una sola acción es bastante elevada; esto, sin duda alguna, se solucionará en la próxima versión del programa.

Figura 24 – Barra de herramientas de Generalización Bidimensional

- **Zoom mode (Alt-Z):** Permite entrar en modo zoom. Para más información, ver la sección del Panel de Edición de Instancias.

- **Zoom out (Ctrl-Z):** Permite volver a ver todo el dominio del problema, es decir, el panel de dibujado a escala natural. Para más información, ver la sección del Panel de Edición de instancias.

- **Botón de configuración del color asignado al botón izquierdo (Alt-1):** Permite cambiar el color asignado al botón izquierdo del ratón en el modo de edición de instancias.

- **Botón de configuración del color asignado al botón derecho (Alt-2):** Permite cambiar el color asignado al botón derecho del ratón en el modo de edición de instancias.

- **Selección y configuración del clasificador (Alt-Espacio):** Permite abrir un Editor Genérico de Objetos para cambiar y/o configurar el clasificador a utilizar. Para más información consultar la sección del Panel de Clasificación.

- **Edición del número de iteraciones a realizar:** Es un campo que permite configurar el número de iteraciones a realizar en la siguiente llamada al método “iterar”. Se mantiene sincronizado con el correspondiente campo del Panel de Clasificación.

- **Construir un clasificador / iterar un booster (Alt-Enter):** Permite lanzar la construcción de un clasificador de weka o las siguientes iteraciones de un booster de oaidtb. Para más información consultar la sección del Panel de Clasificación.

- **Mostrar / Ocultar el panel de edición de instancias (Alt-Izquierda):** Permite mostrar / ocultar el panel de Edición de instancias.

- **Mostrar / Ocultar el panel de configuración del clasificador (Alt-Abajo):** Permite mostrar / ocultar el Panel de Configuración del Clasificador.

- **Mostrar / Ocultar el panel de opciones visuales (Alt-Derecha):** Permite mostrar / ocultar el Panel de Opciones Visuales.

EL PANEL DE EDICIÓN DE INSTANCIAS

Este es el panel que se encarga de gestionar todo lo referente a la manipulación de introducción y eliminación de instancias. Su aspecto es, a día de hoy, el siguiente:

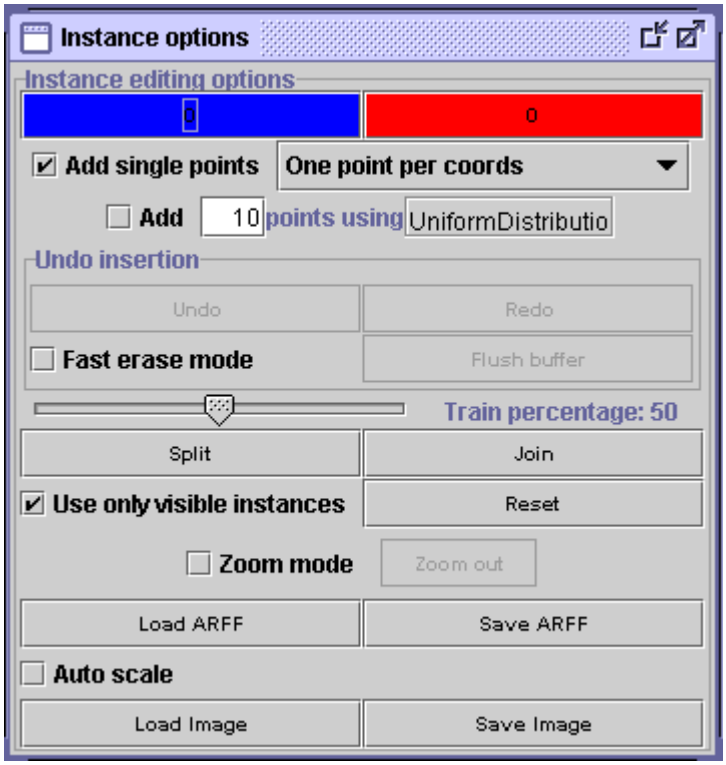

Figura 25 – Ventana de edición de instancias

- **Botón de configuración del color asignado al botón izquierdo (Alt-1):** Permite cambiar el color asignado al botón izquierdo del ratón en el modo de edición de instancias.

- **Botón de configuración del color asignado al botón derecho (Alt-2):** Permite cambiar el color asignado al botón derecho del ratón en el modo de edición de instancias.

El color de fondo de los botones muestra qué color está asignado a cada botón del ratón y su texto el número de instancias de dicho color que existen actualmente en la base de datos.

Una vez pulsado cualquiera de los dos botones se abre un diálogo de selección de color que permite seleccionar un nuevo color o cancelar la operación; el diálogo, como se muestra en las dos figuras siguientes, tiene las pestañas habituales de este tipo de componentes en Java más una nueva pestaña *Colors History* que permite seleccionar de nuevo un color para el que ya existan instancias en la base de datos, además de permitir saber qué color está asignado a qué clase (las clases valen 0, 1,…número de colores – 1).

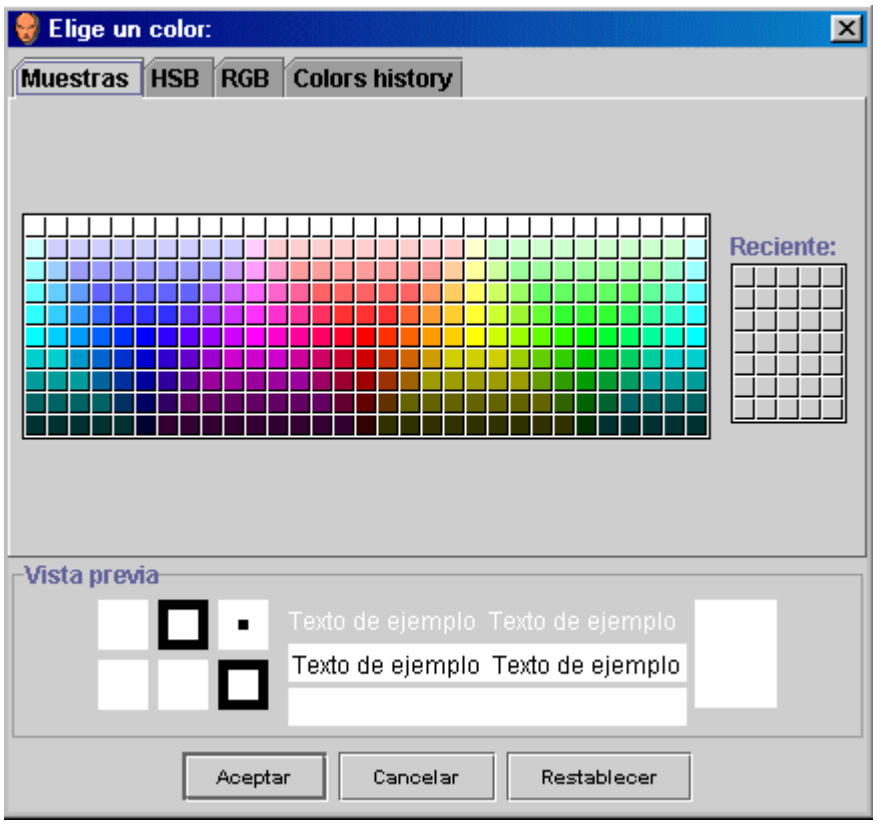

Figura 26 – Diálogo de selección de color

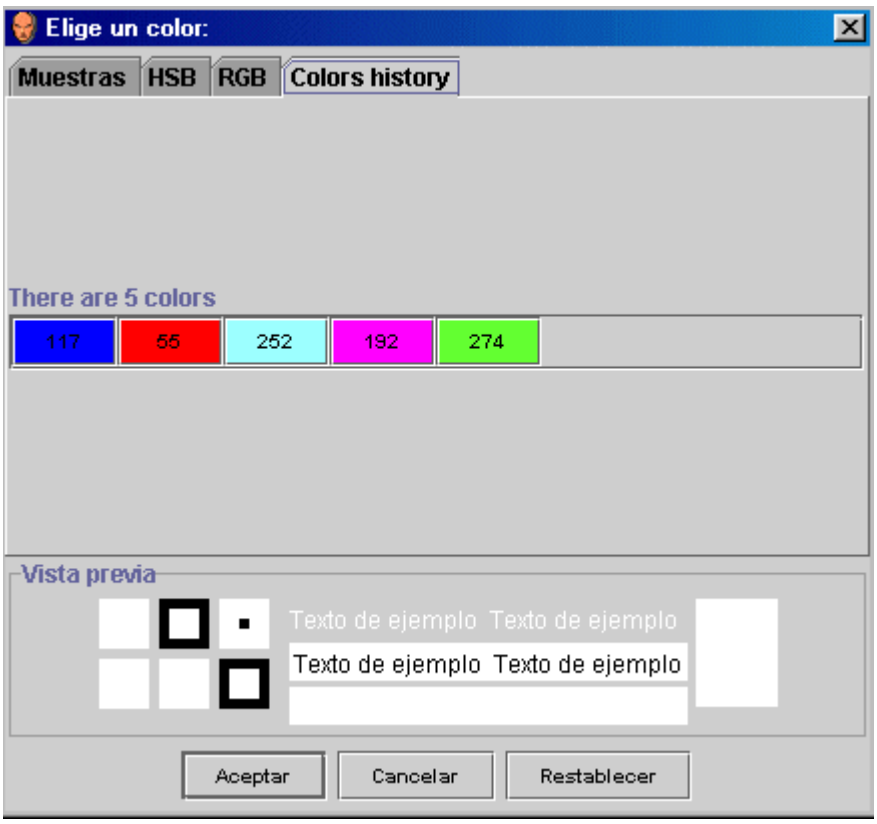

Figura 27 – Historial de colores

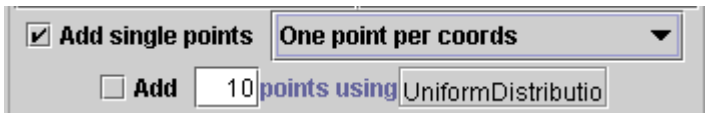

Figura 28 – Modo de introducción de instancias

- **Add Single Points:** Indica que con cada pulsación de ratón sobre el panel de dibujado se añadirá una única instancia en el punto que represente en ese momento la coordenada en pantalla del panel de dibujado sobre la que esté el puntero del ratón, respetando siempre las restricciones de inserción.

- **ComboBox de restricciones de inserción:** Permite seleccionar cuál debe ser la política a seguir cuando se introduzcan instancias a la base de datos a través de pulsaciones de los botones del ratón. Las posibles elecciones son: permitir todas las instancias, un punto por coordenadas y un punto por coordenadas y clase.

- **Modo de inserción múltiple:** Si esta checkbox está seleccionada se añadirán con cada pulsación de ratón sobre el panel de dibujado tantas instancias como haya seleccionado el usuario en el campo editable que está a su derecha –en estos momentos lo he limitado a 999 instancias como mucho por pulsación-, utilizando una función que se puede seleccionar mediante un Editor Genérico de Objetos. Sólo se introducirán los puntos generados por dicha función que

    (a) No incumplan las restricciones de inserción
    (b) Estén dentro de la región del dominio del problema que represente en cada momento el panel de dibujado, incluso si se ha hecho "zoom-in".

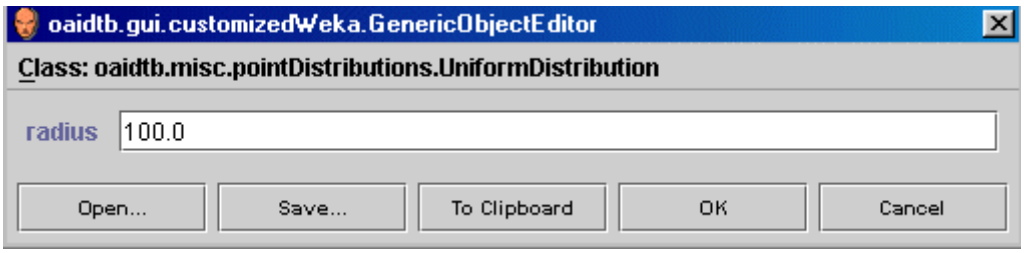

Figura 29 – Edición de la función a utilizar en el modo de "inserción múltiple"

En estos momentos hay disponibles "de fábrica" cinco funciones para la introducción múltiple de instancias, que permiten crear problemas con una estructura subyacente conocida e introducir ruido a dichas instancias. Éstas son:

- Distribución Normal
- Distribución Uniforme
- Distribución de Poisson
- Distribución Chi-Cuadrado
- Una función que pinta puntos en una circunferencia de centro el punto donde se pulse y radio configurable.

Si el lector es conocedor del lenguaje de programación Java puede crear sus propias funciones para la introducción de instancias que creen un problema interesante (o no); la manera de hacer esto es análoga a la de introducir nuevos clasificadores a utilizar:

- Crear una clase que implemente el interfaz
  oaidtb.misc.pointDistributions.PointDistribution

- Añadir dicha clase al fichero de configuración del Editor Genérico de Objetos
  GenericObjectEditor.props
  En la seccción:
  oaidtb.misc.pointDistributions.PointDistribution

El GOE se encargará de averiguar qué parámetros se pueden configurar de la nueva clase, sólo recuerda que aquellos parámetros que quieras que sean editables deben tener sus métodos *getNombreProp* y *setNombreProp* y que no deben ser *null* tras llamar al constructor.

Figura 30 – Facilidades de undo & redo

- **Botón Undo:** Permite deshacer la última operación de introducción de instancias, ya sea de una única instancia o de varias, provocada por una pulsación del ratón sobre el panel de dibujado.

- **Botón Redo:** Permite rehacer la última operación de introducción de instancias deshecha.

- **Botón Flush buffer:** Libera la memoria reservada para las operaciones de undo & redo, deshabilitando obviamente dichas acciones.

- **Checkbox Fast erase mode:** Si esta checkbox está seleccionada, las instancias que se eliminen de la base de datos mediante una operación de undo serán borradas del panel de dibujado en vez de repintar todas aquellas instancias que queden en la base de datos; esto puede ser mucho más rápido en el caso de que el número de instancias en la base de datos sea muy elevado, pero al borrar las instancias eliminadas de la base de datos pueden borrar también la representación de otras instancias que se solapen con aquellas borradas y que aún pertenezcan a la base de datos, y que no serán representadas hasta se repinte el panel de dibujado; es por ello que este modo sólo debe ser utilizado en los casos en que el repintado de todas las instancias sea lento debido a su número.

El programa permite hacer una división de manera aleatoria del conjunto total de instancias en dos subconjuntos disjuntos, valga la redundancia, de entrenamiento y prueba.

Figura 31 – Facilidades de partición del conjunto de instancias

- **Selección del porcentaje de instancias de entrenamiento:** Permite seleccionar cuál será la probabilidad de que una instancia cualquiera caiga en el conjunto de entrenamiento –y por ende, la probabilidad de que caiga en el de prueba-; el máximo valor, 100, asegura que el total de las instancias serán de entrenamiento (esto nos puede ser útil como veremos más adelante).

- **Botón Split:** Realiza la partición aleatoria de las instancias en conjuntos de entrenamiento y prueba; las instancias de ´prueba se representarán ahora como círculos no rellenos

Figura 32 – Representación de instancias de entrenamiento (rellena) y de prueba (vacía)

- **Botón Join:** Destruye los conjuntos de entrenamiento y test, unificando todas las instancias en un único conjunto de entrenamiento.

- **Checkbox Use only visible instances:** Permite que la partición de instancias se haga considerando sólo aquellas instancias que pertenezcan a la región que se esté representando en el panel de dibujado. Así, podemos entrenar al clasificador usando algunas o incluso todas (configurando el porcentaje de instancias de entrenamiento al 100% antes de pulsar el botón de split) las instancias que existen en una subregión del dominio del problema, usando otras instancias de la misma región para hacer las pruebas.

- **Botón reset:** Elimina todas las instancias de la base de datos, provocando un "reseteo" en cascada para el resto de paneles de la aplicación (Clasificación, Opciones Visuales y Área de Dibujado).

Como ya se ha venido comentando el programa proporciona la posibilidad de hacer zoom sobre cualquier región del dominio del problema, con la restricción auto impuesta de que ni la altura ni la anchura de la región sobre la que vayamos a hacer zoom sean menores que 0.0002 medido en píxeles.

Figura 33 – Botones para manejar la función zoom

- **CheckBox Zoom mode:** Permite entrar / salir del modo zoom; en el modo zoom no se pueden introducir instancias mediante la pulsación de ratón; en su lugar, se permite arrastrar el ratón para seleccionar una región dentro de la propia región que se esté representando en el panel de dibujado - se permite hacer zoom recursivamente -; para cancelar la selección de una región basta con pulsar el botón derecho.

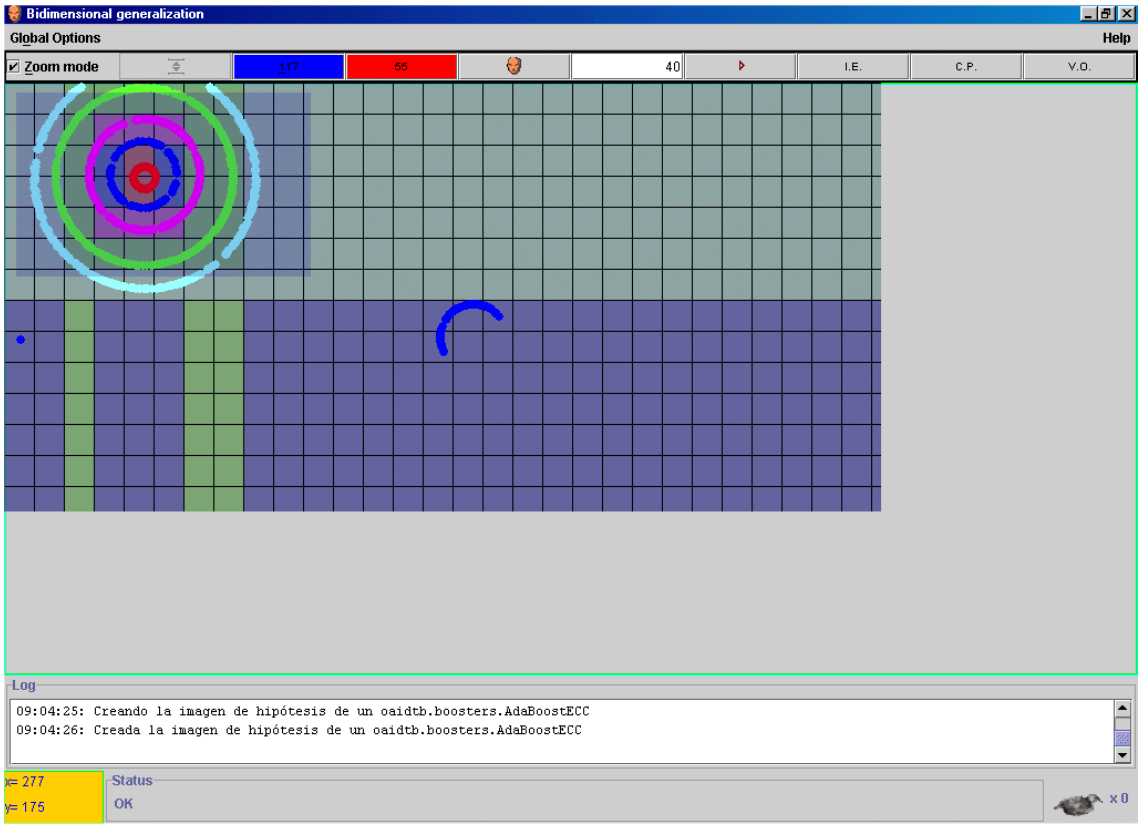

Figura 34 – El área que se está seleccionando aparece como un rectángulo de color semitransparente

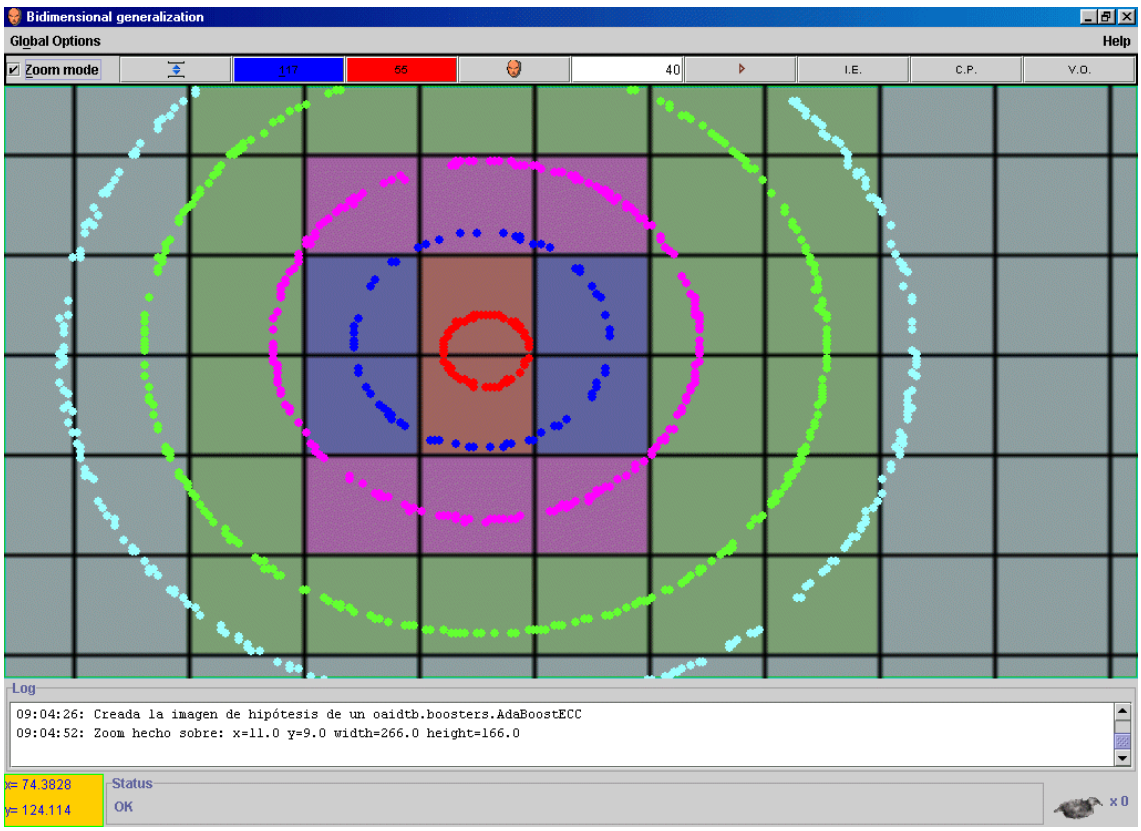

Figura 35 – Se representa en todo el panel de dibujado el área sobre la que se ha hecho zoom

Cuando se ha hecho zoom in sobre una región y se recrea la imagen de hipótesis esta se hace en relación sólo a dicha región, con lo que se puede ver con más precisión dicha hipótesis; además, al hacer zoom sobre una región pueden aparecer detalles, instancias, que antes eran inapreciables; por lo tanto, si alguna vez algo no te "cuadra", haz zoom y recrea la imagen de hipótesis.

- **Botón Zoom out:** Sirve para volver a representar el problema a "escala natural". Al salir de una situación en la que se han creado los conjuntos de entrenamiento y de test dentro de una región específica dichos conjuntos se destruyen para volver a representar todas las instancias. El programa volverá a situar la imagen de hipótesis que existiera antes de hace zoom si y sólo si el clasificador no ha cambiado desde entonces.

El programa permite salvar y cargar los conjuntos de instancias desde ficheros, bloqueando el acceso a la base de datos de puntos durante el trascurso de dichas operaciones. Estos son los controles de dichas funciones.

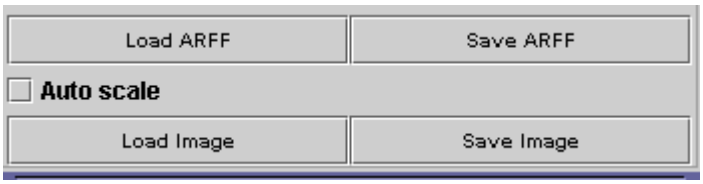

Figura 36 – Controles de entrada y salida a ficheros

- **Checkbox Auto scale:** Si esta checkbox está seleccionada las instancias que se carguen desde archivos serán trasladadas y escaladas en el espacio bidimensional para que aparezcan ocupando todo el panel.

- **Load ARFF:** Permite cargar un nuevo conjunto de instancias desde un archivo con formato ARFF; el programa espera que las instancias tengan el formato {real, real, nominal}. El programa marca las instancias que salva anteponiendo al nombre de la relación el prefijo "oaidtb_", y dichos conjuntos de datos deben tener como posibles valores para el atributo clase el valor RGB del color que representan; si el nombre de la relación que almacena el archivo ARFF no tiene ese formato (ver el Anexo A descripción del formato ARFF) el programa aún intenta cargar "a ciegas" las instancias, comprobando que los tres primeros atributos de la relación sean {real, real, nominal} y asignando de manera aleatoria colores a cada posible valor del atributo clase –esto permite cargar, por ejemplo, los conjuntos de datos conus-torus utilizados en numerosos experimentos de algoritmos de aprendizaje, y que el lector puede encontrar en el directorio $OAIDTB_HOME/_data_/conus-torus-

- **Save ARFF**: permite salvar todas las instancias que se encuentren en la base de datos a un archivo con formato ARFF cumpliendo con las especificaciones explicadas en el botón Load ARFF.

- **Load Image**: Permite cargar las instancias desde un archivo de imagen (gif, jpeg y tiff). Esta característica aún está en fase de prueba, y en particular no debe ser usada con imágenes que tengan, aproximadamente y dependiendo de la máquina sobre la que se esté ejecutando el programa, más de 5000 puntos (el sistema avisa de ello). Actualmente las estructuras de índice de la base de datos están optimizadas para la búsqueda y la ordenación, no para la inserción masiva, por lo que la carga de este tipo de imágenes puede resultar de una lentitud exasperante a medida que van creciendo dichos índices; en desarrollo se encuentra una versión de la aplicación que (con calma,

que todo llega). pensada para manejar más eficientemente este problema. En el directorio $OAIDTB_HOME/_data_/_imgs _ he incluido algunas imágenes que se pueden utilizar con el programa.

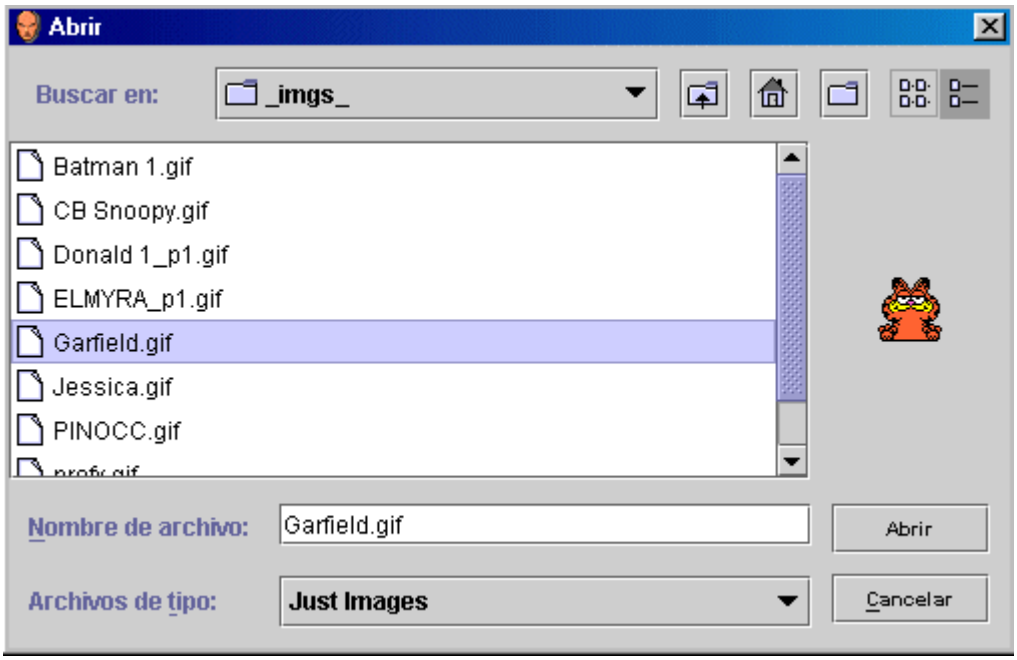

Figura 37 – Selección de un archivo de imagen

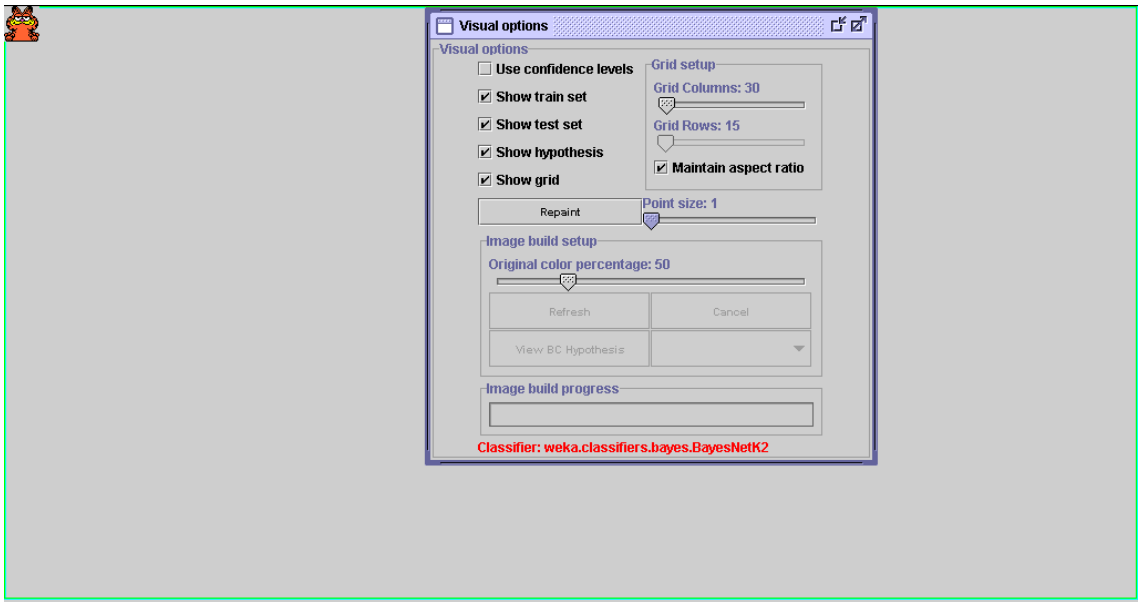

Figura 38 – Carga del archivo "garfield.gif"

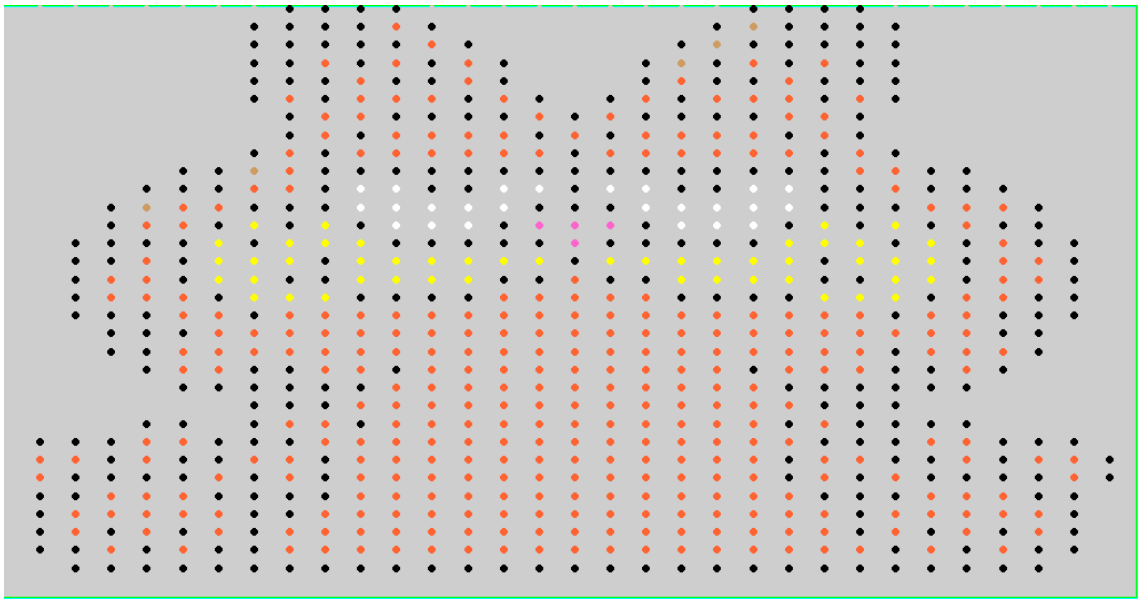

Figura 39 – Zoom-in sobre las instancias que constituyen la imagen

Figura 40 – Hipótesis lanzada por AdaBoostECC, nuestro amigo Garfield es inconfundible

- **Save Image:** Permite salvar, en formato JPEG a máxima calidad, el contenido actual (lo que se ve y no se ve por estar tras una de las ventanas de herramientas) del panel de dibujado. Al contrario de las otras tres funciones de entrada / salida a ficheros, esta acción no bloquea el acceso a la base de datos

EL PANEL DE OPCIONES DEL CLASIFICADOR

Este panel ya está explicado, así que el lector debe visitar la sección "El Panel de Clasificación" si quiere conocer los detalles.

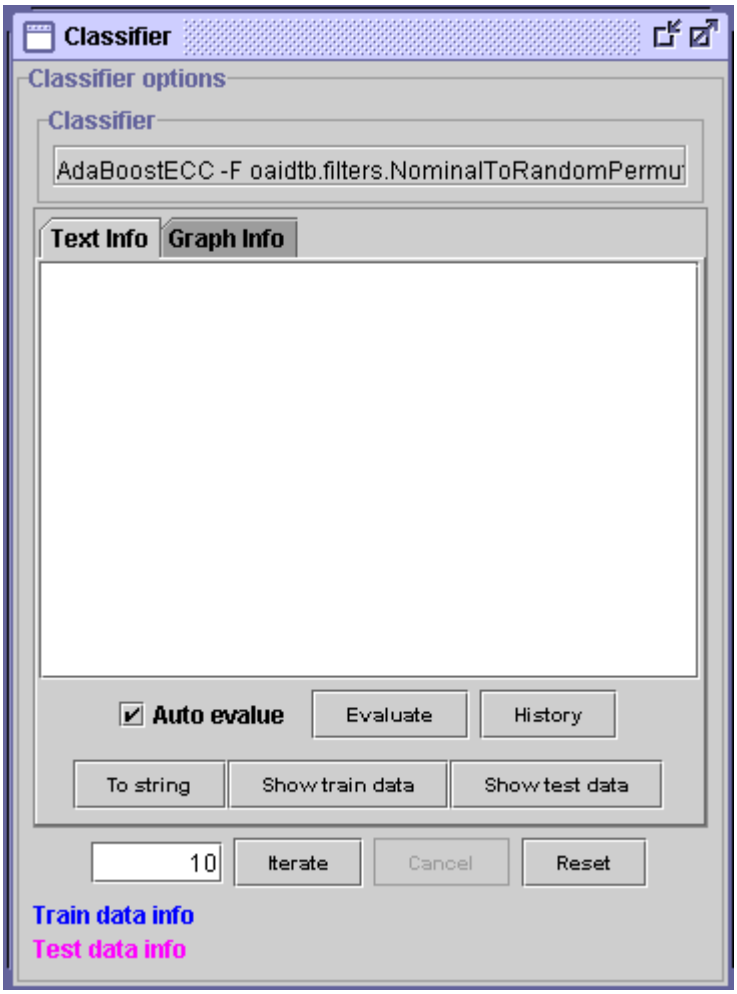

Figura 41 – El frame de opciones del clasificador muestra un "panel de clasificación"

Sólo dos consideraciones acerca e los datos que se recogen deben ser hechas:

- Cada vez que se ordena la construcción de un nuevo clasificador se recogen las instancias de entrenamiento y de prueba que existan en la base de datos, que pueden no ser las totales sino sólo las que pertenecen a una región.

- Cuando un booster se encuentra en proceso de iteración no cambiará las instancias de entrenamiento, construyendo un nuevo ensemble, hasta que se pulse el botón de reset (o se cambie el propio clasificador); debe recordar hacer esto tras introducir nuevas instancias o la imagen de hipótesis que se cree corresponderá al clasificador construido con los anteriores datos de entrenamiento (sí, en la próxima versión habrá un botón de acceso a esta acción en la barra de herramientas).

Además, no hace falta ni decir que muchos de los clasificadores de weka no pueden clasificar el problema aquí planteado (no pueden manejar atributos numéricos, clases nominales etc.); quizás en una futura versión se haga una criba en *GenericObjectEditor.props* para permitir sólo la selección de los clasificadores apropiados, pero hasta entonces el usuario no debe sorprenderse de que un clasificador que quiera probar no funcione con estos datos.

EL PANEL DE OPCIONES VISUALES

En este panel se pueden configurar las opciones de visualización del área de dibujado y de la imagen de hipótesis. Éste es su aspecto:

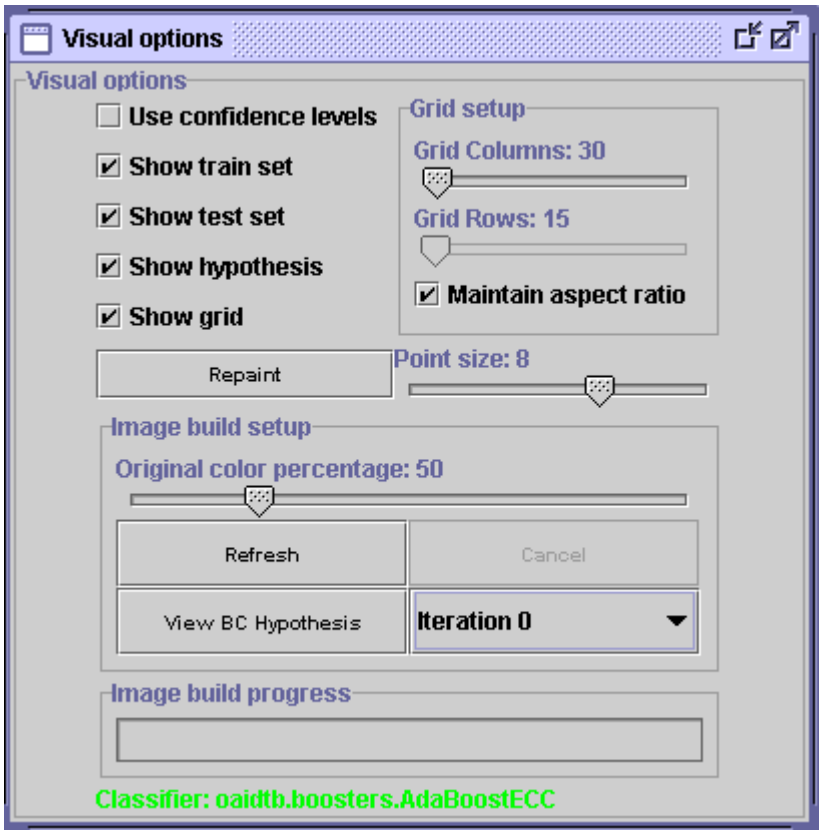

Figura 42 – El panel de opciones visuales

Los controles de este panel sirven para:

- **Use confidence levels (Alt-L):** Si el clasificador que se ha usado para construir la imagen de hipótesis puede proveer de grados de confianza para cada una de las clases que forman el problema esta checkbox hará que la imagen de hipótesis que se muestre en el panel de dibujado sea la construida utilizando dichos niveles de confianza; si se piensa utilizar esta opción es conveniente que se aumente el porcentaje del color original que representa a cada clase en la imagen de hipótesis.

- **Show train set (Alt-T):** Mostrar o no el conjunto de entrenamiento.

- **Show train data (Ctrl-T):** Mostrar o no el conjunto de prueba

- **Show hypothesis (Alt-H):** Mostrar o no la imagen de hipótesis.
- **Show grid (Alt-G):** Mostrar o no la rejilla empleada para crear la imagen de hipótesis
- **Repaint (Ctrl-Alt-R):** Repintar el panel de dibujado.
- **Point size:** Radio, en píxeles, de las circunferencias que representan a las instancias.

En este panel se encuentran los controles que permiten configurar y lanzar el proceso de construcción de la imagen de hipótesis de un clasificador. Dicho clasificador será siempre el mismo que se encuentre configurado en el panel de clasificación, y la siguiente etiqueta muestra su nombre (en verde si se puede construir una imagen de hipótesis con él y en rojo si no).

Figura 43 – Nombre del clasificador que se usa para construir la imagen de hipótesis

La manera de construir la imagen de hipótesis es, grosso modo, dividir el panel de dibujado en celdas mediante una rejilla y pintar cada celda de la rejilla del color que el clasificador predizca debe tener su punto intermedio; así pues, es posible configurar dicha rejilla.

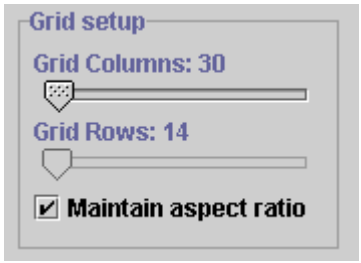

Figura 44 – Configuración de la rejilla

- **Grid Columns:** Número de columnas de la rejilla
- **Grid Rows:** Número de filas de la rejilla
- **Maintain aspect ratio:** Mantener la relación de aspecto X/Y de las dimensiones del panel de dibujado en las dimensiones de la celda de la rejilla.

Se debe tener en cuenta que a menor tamaño de celda en la rejilla mayor número de instancias se deben clasificar y más tardará el proceso de construcción de la imagen de hipótesis, aunque también será más fiel la representación de dicha hipótesis; en muchos casos, en vez de aumentar el número de columnas y/o de filas de la rejilla será más útil hacer zoom sobre las regiones que se quieran ver con más detalle y reconstruir en ellas las imágenes de hipótesis con un tamaño de rejilla menor.

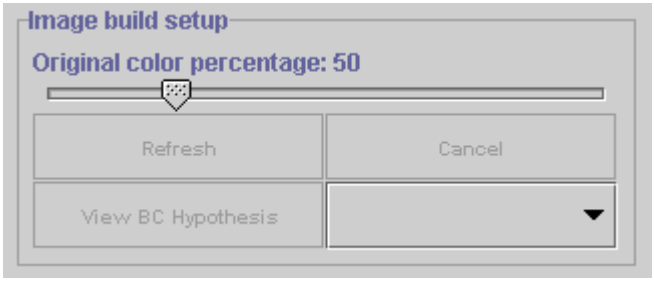

Figura 45 – Controles de construcción de la imagen de hipótesis

- **Original color percentage:** Indica qué porcentaje del color original se utilizará en la creación de la imagen de hipótesis; mi recomendación es dejarlo al 50% si sólo se va a ver la imagen de hipótesis y subirlo (incluso hasta el 100%) si se va a ver la imagen de hipótesis construida utilizando los nioveles de confianza del clasificador; para que se vean los cambios se debe reconstruir la imagen (con los boosters programados en oaidtb esta transición de color es inmediata, no así con el resto de los clasificadores de weka).

- **Refresh (Alt-R):** Lanzar el proceso de construcción de la imagen de hipótesis

- **Cancel (Ctrl-R):** Cancelar el proceso de construcción de la imagen de hipótesis.

- **View BC Hypothesis (Ctrl-B):** Lanzar el proceso de construcción de la imagen de hipótesis para uno de los clasificadores base de un booster.

- **ComboBox de selección del clasificador base:** Seleccionar para qué clasificador base del booster se construirá la imagen de hipótesis cuando se pulse el botón View BC Hypothesis.

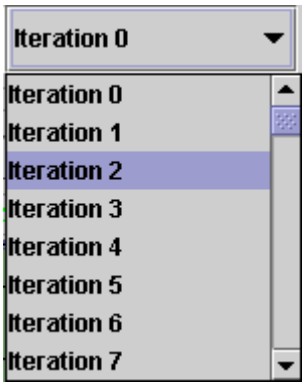

Figura 46 – selección e un clasificador base

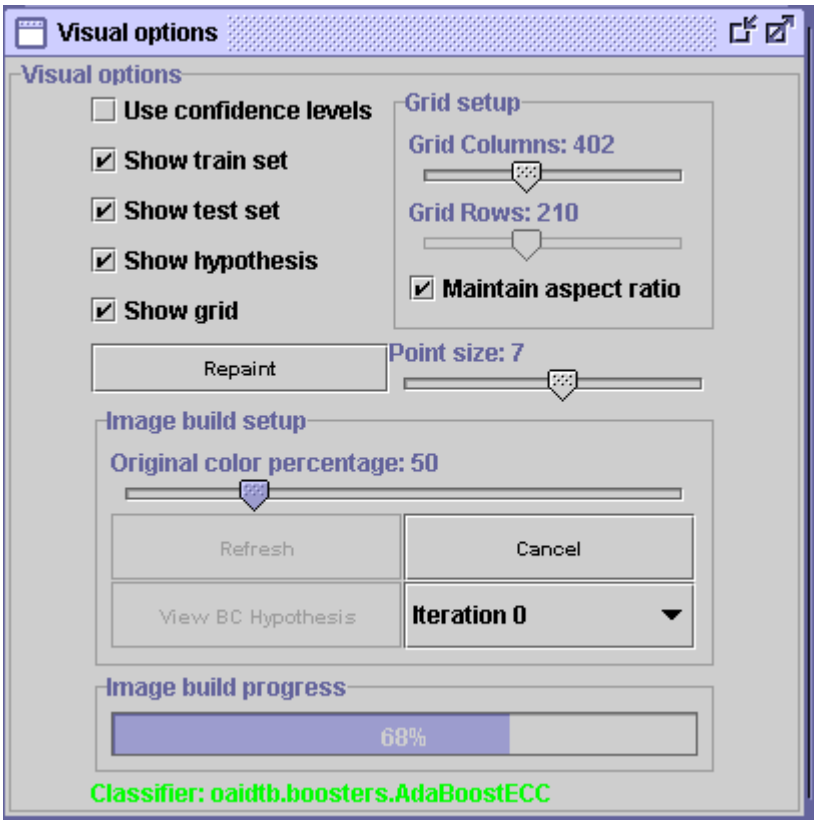

Figura 47 – Seguimiento de la construcción de la imagen de hipótesis

EL PANEL DE INFORMACIÓN AL USUARIO

Por último vamos a ver el área que sirve para informar al usuario de lo que se está haciendo en cada momento.

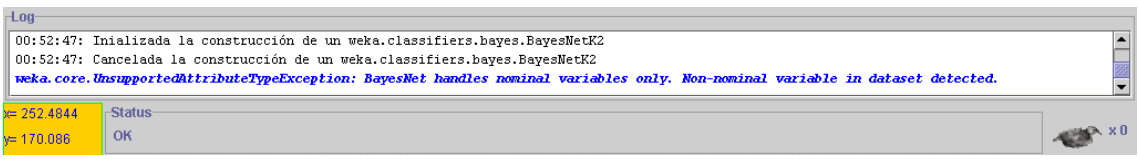

Figura 48 – Área de información al usuario

En la parte superior aparece el **panel de log** en el que aparecen los mensajes indicando al usuario las acciones iniciadas y finalizadas, con marca de tiempo, así como las excepciones e informaciones generadas por los clasificadores y las excepciones, mostradas con color azul y en versalita. Si pulsamos con el botón derecho sobre este panel aparece el siguiente menú emergente:

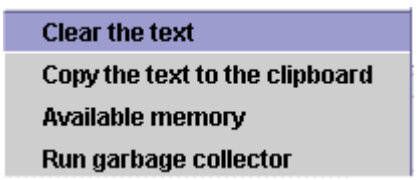

Figura 49 – Menú emergente del área de información al usuario

Este menú nos brinda la posibilidad de:

- **Clear the text:** Borra el texto del área de log
- **Copy the text to the clipboard:** Copia el texto de log al portapapeles del sistema.
- **Available memory:** Muestra la memoria libre en la pila de memoria de java (puede crecer dinámicamente).
- **Run garbage collector:** Lanza el recolector de basura de java para liberar toda la memoria posible.

En naranja se ven en la imagen dos etiquetas que muestran las coordenadas, en términos del dominio del problema, que representa el punto del panel de dibujado sobre el que se encuentra en cada momento el puntero del ratón.

Al lado de estas etiquetas está la barra de estado, que muestra si algún proceso "pesado" de los que se procesan en paralelo está en marcha - dichos procesos son el de entrada/salida a fichero, el de construcción / iteración de un clasificador / booster y el de construcción de la imagen de hipótesis; como mucho un proceso de cada tipo se ejecuta a la vez, pudiéndose estar realizando las tres tareas concurrentemente, aunque esto no se recomienda en ningún caso salvo que se tenga una gran máquina-. Al lado de esta barra de estado aparece una imagen del logo de weka que se anima si cualquiera de estas tareas se está realizando, mostrando el número de dichas tareas que están en marcha en cada momento.

UTILIZACIÓN DE LAS HERRAMIENTAS GRÁFICAS DE WEKA

Los boosters de OAIDTB pueden ser utilizados con las herramientas gráficas que ofrece la biblioteca weka. Esto se puede hacer ejecutando los scripts *"wekaXXX"* del directorio $OAIDTB_HOME/bin

Se ha creado un parche para weka que soluciona numerosos problemas que surgían al utilizar clases no pertenecientes a la misma con las aplicaciones gráficas, además de añadir un sinfín de nuevas e interesantes características. Las instrucciones para instalarlo, así como su código fuente y las instrucciones para añadir los boosters de oaidtb a cualquier distribución de weka se encuentran en el directorio ***$OAIDTB***_HOME/misc/Weka GUI (GOE) Patch.

ANEXO A. EL FORMATO DE ARCHIVOS ARFF

Las clases de weka trabajan sobre el formato de archivos ARFF (Attribute-Relation File Format); los datos pueden ser adquiridos también utilizandos otras fuentes, como archivos CSV (Comma Separated Values), conexiones a bases de datos, con pulsaciones de ratón sobre el panel de dibujado de la aplicación gráfica creada etc., pero los archivos ARFF siguen siendo la manera más utilizada de realizar esta labor, por lo que a continuación traduzco un artículo (http://www.cs.waikato.ac.nz/~ml/weka/arff.html) que describe dicho formato:

ATTRIBUTE-RELATION FILE FORMAT (ARFF)

1 de Abril, 2002

Un archivo ARFF (Attribute-Relation File Format) es un archivo de texto ASCII que describe una lista de instancias que comparten un conjunto de atributos. Los archivos ARFF han sido desarrollados por el "Machine Learning Project" en el departamento de Ciencias de la Computación en la universidad de Waikato (Nueva Zelanda) para ser usado por el software de aprendizaje computacional Weka:
http://www.cs.waikato.ac.nz/~ml/weka/

Este documento describe la versión de ARFF utilizada con las versiones de weka 3.2 y 3.3, que es una extensión del formato descrito en el libro "Minería de datos" de Ian H. Witten y Eibe Frank (las nuevas características son los atributos de tipo cadena (String), los atributos de tipo fecha (Date) y las instancias dispersas (Sparse).

Esta explicación ha sido realizada por Gordon Paynter (gordon.paynter@ucr.edu) utilizando la descripción "Weka 2.1 ARFF description", un correo de Len Trigg (lenbok@myrealbox.com) y Eibe Frank (eibe@cs.waikato.ac.nz), y algunos conjuntos de datos. Ha sido editado por Richard Kirkby (rkirkby@cs.waikato.ac.nz) y la presente traducción ha corrido a cargo de Santiago David Villalba (sdvb@wanadoo.es). Si está interesado en la propuesta para el formato ARFF 3 contacte con Len.

INTRODUCCIÓN

Los archivos ARFF tienen dos secciones distintas. La primera es la sección de información de cabecera (**Header)**, a la que sigue la sección de datos (**Data)**.

La **cabecera** contiene el nombre de la relación, una lista de atributos (las columna en los datos) y sus tipos. Como ejemplo, veamos la sección de cabecera del conjunto de instancias iris:

```
% 1. Title: Iris Plants Database
%
% 2. Sources:
%      (a) Creator: R.A. Fisher
%      (b) Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
%      (c) Date: July, 1988
%
@RELATION iris
```

```
@ATTRIBUTE sepallength  NUMERIC
@ATTRIBUTE sepalwidth   NUMERIC
@ATTRIBUTE petallength  NUMERIC
@ATTRIBUTE petalwidth   NUMERIC
@ATTRIBUTE class        {Iris-setosa,Iris-versicolor,Iris-virginica}
```

La sección de datos del mismo conjunto tiene el siguiente aspecto:

```
@DATA
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
```

Las líneas que comienzan con un % son comentarios. Las declaraciones de **@RELATION**, **@ATTRIBUTE** y **@DATA** no son sensibles al caso.

OTROS EJEMPLOS

Algunos conocidos conjuntos de datos en el aprendizaje computacional son distribuidos junto a Weka en el directorio $WEKAHOME/data como archivos ARFF.

LA SECCIÓN DE CABECERA (HEADER)

La sección de cabecera contiene la declaración de la relación y de los atributos que la forman.

LA DECLARACIÓN @RELATION

El nombre de la relación está definido en la primera línea (no comentada) del archivo.

El formato es:

```
@relation <nombre-de-la-relación>
```

donde <nombre-de-la-relación> es una cadena. La cadena debe ser encerrada entre comillas si el nombre incluye espacios

LAS DECLARACIONES @ATTRIBUTE

Las declaraciones de atributos toman la forma de una secuencia ordenada de sentencias **@attribute**. Cada atributo en el conjunto de datos tiene su propia sentencia **@attribute** que define de manera unívoca su nombre y su tipo de dato. El orden en que los atributos son declarados indica la posición de la columna en la sección de datos del archivo. Por ejemplo, si un atributo es declarado en tercer lugar entonces Weka espera que todos los valores de dicho atributo se encontrará en la tercera columna, delimitada con comas.

El formato de las sentencias **@attribute** es:

```
@attribute <attribute-name> <datatype>
```

donde:

<*attribute-name*> debe comenzar con un caracter alfabético, y si incluye espacios debe ser delimitafo mediante comillas

<*datatype*> puede ser uno de los cuatro tipos actualmente (versión 3.2.1) soportados:

- numeric
- <nominal-specification>
- string
- date [<date-format>]

donde <nominal-specification> y <date-format> están definidos más adelante. Las palabras reservadas **numeric**, **string** y **date** son sensibles al caso.

ATRIBUTOS NUMÉRICOS

Los atributos numéricos puden ser números reales o enteros.

ATRIBUTOS NOMINALES

Los valores de los atributos nominales se definen proporcionando una lista <nominal-specification> de los posibles valores que puede tomar el atributo: {<nominal-name1>, <nominal-name2>, <nominal-name3>, ...}

Por ejemplo, el atributo clase del conjunto de datos Iris puede ser definido de la siguiente forma:

```
@ATTRIBUTE class       {Iris-setosa,Iris-versicolor,Iris-virginica}
```

Los valores que contengan espacios deben ser puestos entre comillas.

ATRIBUTOS DE CADENA

Los atributos de cadena permiten crear atributos que contengan valores de texto arbitrarios. Esto es muy útil en aplicaciones de minería de texto, dado que se pueden crear conjuntos de datos con atributos de cadena y escribir filtros de Weka para manipular cadenas (como StringToWordVectorFilter). Los atributos de cadena se declaran como sigue:

```
@ATTRIBUTE LCC    string
```

ATRIBUTOS DE FECHA

Las declaraciones de los atributos de fecha toman la forma:

```
@attribute <name> date [<date-format>]
```

donde <name> is el nombre del atributo y <date-format> es una cadena de texto opciona que especifica cómo se deben parsear e imprimir (este es el mismo formato que utiliza SimpleDateFormat). El cadena de formato por defecto accepta la especificación ISO-8601 que combina la fecha y la hora en el formato: "yyyy-MM-dd'T'HH:mm:ss".

Las fechas deben ser especificadas en la sección de datos como la representación correspondiente de la cadena fecha/hora (ver el ejemplo más abajo).

LA SECCIÓN DE DATOS

La sección de datos del archivo contiene la línea de la declaración **@data** y las líneas que representan las instancias.

LA DECLARACIÓN @DATA

La declaración **@data** se compone de una única línea que indica el comienzo del segmento de datos en el archivo. El formato es

```
@data
```

LOS DATOS DE INSTANCIAS

Cada instancia está representada en una única línea, con el carácter de retorno de carro indicando el final de las instancias.

Los valores de cada atributo en una instancia están delimitados por comas. Deben aparecer en el orden que fueron declarados en la sección de cabecera (i.e. el valor correspondiente a la enésima declaración **@attribute** es siempre el enésimo campo de la instancia).

Los valores desconocidos son representados por un signo de cierre de interrogación, de la siguiente manera:

```
@data
4.4,?,1.5,?,Iris-setosa
```

Los valores de los atributos nominales y de cadena son sensibles al caso, y cualquiera que contenga espacios debe ser puesto entre comillas, como en el siguiente ejemplo:

```
@relation LCCvsLCSH

@attribute LCC string
@attribute LCSH string

@data
AG5,   'Encyclopedias and dictionaries.;Twentieth century.'
AS262, 'Science -- Soviet Union -- History.'
AE5,   'Encyclopedias and dictionaries.'
AS281, 'Astronomy, Assyro-Babylonian.;Moon -- Phases.'
AS281, 'Astronomy, Assyro-Babylonian.;Moon -- Tables.'
```

Las fechas deben ser representadas en la sección de datos utilizando la representación especificada en la declaración del atributo. Por ejemplo:

```
@RELATION Timestamps

@ATTRIBUTE timestamp DATE "yyyy-MM-dd HH:mm:ss"

@DATA
"2001-04-03 12:12:12"
"2001-05-03 12:59:55"
```

ARCHIVOS ARFF DISPERSOS (SPARSE)

Los archivos ARFF dispersos son muy parecidos a los archivos ARFF, pero los datos con valor 0 no son representados explícitamente.

Los archivos ARFF dispersos tienen la misma cabecera (i.e las etiquetas **@relation** y **@attribute**), pero la sección de datos es diferente. En vez de representar cada valor en orden, como aquí:

```
@data
0, X, 0, Y, "class A"
0, 0, W, 0, "class B"
```

Los atributos cuyo valor es cero son explícitamente identificados por el número de atributo y su valor, como en este ejemplo:

```
@data
{1 X, 3 Y, 4 "class A"}
{2 W, 4 "class B"}
```

Cada instancia es encerrada entre llaves, y el formato para cada entrada es <index> <espacio> <value>, donde index es el índice del atributo (empezando desde 0).

Téngase en cuenta que los valores omitidos en una instancia dispersa son **0**, no son valores desconocidos (missing). Si un valor es desconocido se debe representar explícitamente con el símbolo "?".

Atención: Se conoce un problema cuando se salvan objetos "SparseInstance" desde conjuntos de datos que contengan atributos de cadena. En Weka los datos de cadena y nominales son almacenados, por eficiencia, como números que representan el índice dentro de un vector que contiene los posibles valores del atributo. Por lo tanto, el primer valor de un atributo como de cadena es asignado al índice 0: esto significa que, internamente, este valor es almacenado como un 0. Cuando se escribe una SparseInstance, los atributos de cadena de las instancias que tengan el valor interno 0 no son sacados, y por lo tanto dichos valores son perdidos (y cuando se vuelva a releer de nuevo el archivo ARFF el valor por defecto 0 es el índice de un valor de cadena distinto, por lo que al atributo cambia). Para evitar este problema basta con añadir un valor de cadena vacío, nunca utilizado, en el índice 0 del atributo cada vez que se declaren atributos de cadena que puedan ser utilizados en objetos SparseInstance y salvados con archivos ARFF dispersos.

ANEXO B. EL FORMATO DE LAS MATRICES DE COSTES

Una de las entradas de los algoritmos sensibles al coste es la matriz de costes, que indica cuál es el coste de clasificar incorrectamente una instancia.

Para describir el formato lo haré a través de un pequeño ejemplo; esta es una matriz de costes para el conjunto de datos iris:

```
% Rows   Columns
3        3
% Matrix elements
0.0      1.0      1.0
2.0      0.0      1.0
0.8      1.7      0.0
```

Sean *i*=número de fila $0 \leq i <$número de filas y *j*=número de columna $0 \leq$ j <número de columnas; el elemento (*j*, *i*) de la matriz indica el coste de clasificar incorrectamente una instancia de clase *i* como clase *j*; así, el elemento (0,1), en la matriz anterior 2.0, indica el coste de clasificar una instancia de clase 1 como clase 0.

Weka espera que las matrices de costes, si se cargan desde ficheros, tengan el formato anterior, es decir:

- Una primera línea en la que se indican el número de filas y de columnas separadas por al menos un espacio en blanco; ambos deben ser números naturales e iguales, pues las matrices de costes deben ser cuadradas.

- En las siguientes líneas se especifican los costes de clasificar mal una instancia de una clase (línea 0 == clase 0, línea 1 == clase 1…), separando cada coste por al menos un espacio en blanco; se admiten números enteros y reales.

- El carácter “%” marca el comienzo de un comentario.