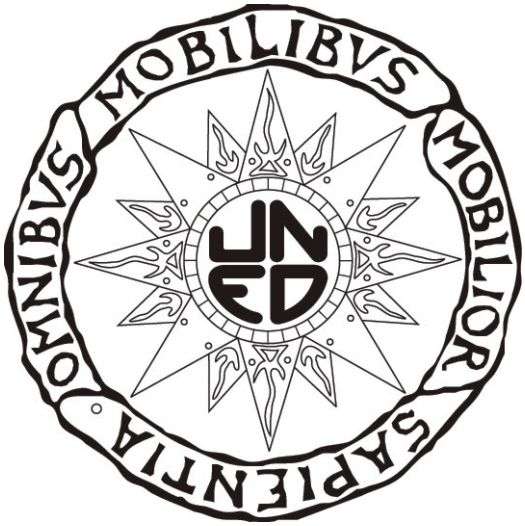


Aprendizaje Automático

Máquinas de Vectores Soporte en Tareas de Clasificación

SVM

Actividad 2

Índice

1. Introducción

Las máquinas de vectores soporte (SVM, del ingles Support Vector Machine) tienen su origen en los trabajos sobre la teoría del aprendizaje estadístico y fueron introducidas en los años 90 por Vapnik y sus colaboradores [Boser et al., 1992, Cortes & Vapnik, 1995]. Aunque originariamente las SVMs fueron pensadas para resolver problemas de clasificación binaria, actualmente se utilizan también para resolver otros tipos de problemas (regresión, agrupamiento, multiclasificación). También son diversos los campos en los que han sido utilizadas con éxito, tales como visión artificial, reconocimiento de caracteres, categorización de texto e hipertexto, clasificación de proteínas, procesamiento de lenguaje natural, análisis de series temporales. De hecho, desde su introducción, han ido ganando un merecido reconocimiento gracias a sus sólidos fundamentos teóricos. Este documento se centrará en el uso práctico de las SVMs en tareas de clasificación.

2. Requisitos previos

1. *Haber estudiado el capítulo 2 del texto base [Borrajo et al., 2006].*
2. *Haber estudiado el tutorial dedicado a Máquinas de Vectores Soporte[Carmona, 2016] (accesible en el curso virtual).*

3. Objetivos

1. *Familiarizarse con el uso del algoritmo SMV para resolver tareas de clasificación.*
2. *Experimentar con el algoritmo SVM implementado en Weka para los siguientes tipos de problemas:*
   a) Conjunto de ejemplos 2D linealmente separables (dos clases).
   b) Conjunto de ejemplos 2D no linealmente separables (dos clases).

3. *Experimentar con el valor de los diferentes parámetros del algoritmo SVM de WEKA.*
4. *Analizar diferentes formas de evaluar un modelo.*
5. *Experimentar con el software LIBSVM[l] para visualizar las fronteras de decisión asociadas al modelos SVM en conjuntos de ejemplos 2D no linealmente separables.*

4. Material

Programa Weka.
Applet LIBSVM.
Archivo de datos: "iris.arff" (accesible en el curso virtual).

5. Actividades

1. Cargar el fichero "iris.arff" desde el panel Preprocess de Explorer. El contenido de este archivo es una colección de instancias que representan tres variedades de lirio. Se compone de cinco atributos. Los cuatro primeros correspondes a diversas medidas morfológicas de la planta (anchura y longitud de sépalos y de pétalos) y el quinto corresponde al valor de la clase (tipo de lirio). Para más detalle, abra el archivo con cualquier editor de textos y consulte la información de cabecera de dicho archivo.

2. Para facilitar la interpretación del modelo obtenido por un clasificador SVM, empezaremos generando un problema artificial de dos dimensiones y de dos clases. Para ellos, vamos a transformar el fichero "iris.arff" original (se asumen que ya está cargado) en otro conjunto de datos en el que se eliminan los atributos "sepallength" y "sepalwidth" y, además, se eliminan todos los ejemplos asociados a la clase "Iris-virginica". Realice los siguientes pasos:

- *Elimine los atributos y los ejemplos pertenecientes a la clase indicada. Para ello, haga uso, respectivamente, de los filtros Weka denominados Remove (filters > unsupervised > atribute) y RemoveWithValues (filter > unsupervised > instance), cada uno de ellos accesibles desde el botón Choose del panel Preprocess. Lea la ayuda asociada al botón More de cada filtro para configurarlo convenientemente. Además, al usar el segundo filtro, no olvide también poner la opción "ModifyHeader" a true para, de esta forma, eliminar cualquier referencia a los valores excluidos por el filtro.*
- *Dado que al forma de fabricar el conjunto de entrenamiento para obtener una colección de ejemplos "D pertenecientes a dos clases ha sido un tanto artificiosa, ocurrirá que, tras aplicar los dos filtros anteriores, es muy posible que parezcan algunos ejemplos repetidos, tal y como así ocurre. Para eliminar los ejemplos repetidos, se aplicará el filtro denominado RemoveDuplicates (filters > unsupervised > instance).*
- *Utilice el botón Edit del panel Preprocess para ver el contenido de los datos filtrados, asegurándose de que el conjunto de datos resultante contiene sólo ejemplos de la clase "Iris-setosa" (22 ejemplos) e "Iris-versicolor" (36 ejemplos) y, además, sólo aparecen los atributos "petallength" y "petalwidth".*
- *Utilice el botón Save para guardar el nuevo conjunto generado. Por ejemplo, utilice el nombre "iris_2D_2Clases_Ls.arff".*
- *Utilice el panel Visualize de Explorer para visualizar el conjunto de datos filtrado. En particular, resulta de interés el plot que muestra las instancias de cada clase cuando se representa el atributo "petallength" respecto del atributo "petalwidth". Claramente, podrá observar que el conjunto de datos es linealmente separable.*

Documentación a entregar apartado 5.2.-

Vamos a mostrar las opciones de configuración de los filtros para preparar las instancias, el conjunto de ejemplos obtenidos y el plot generado.

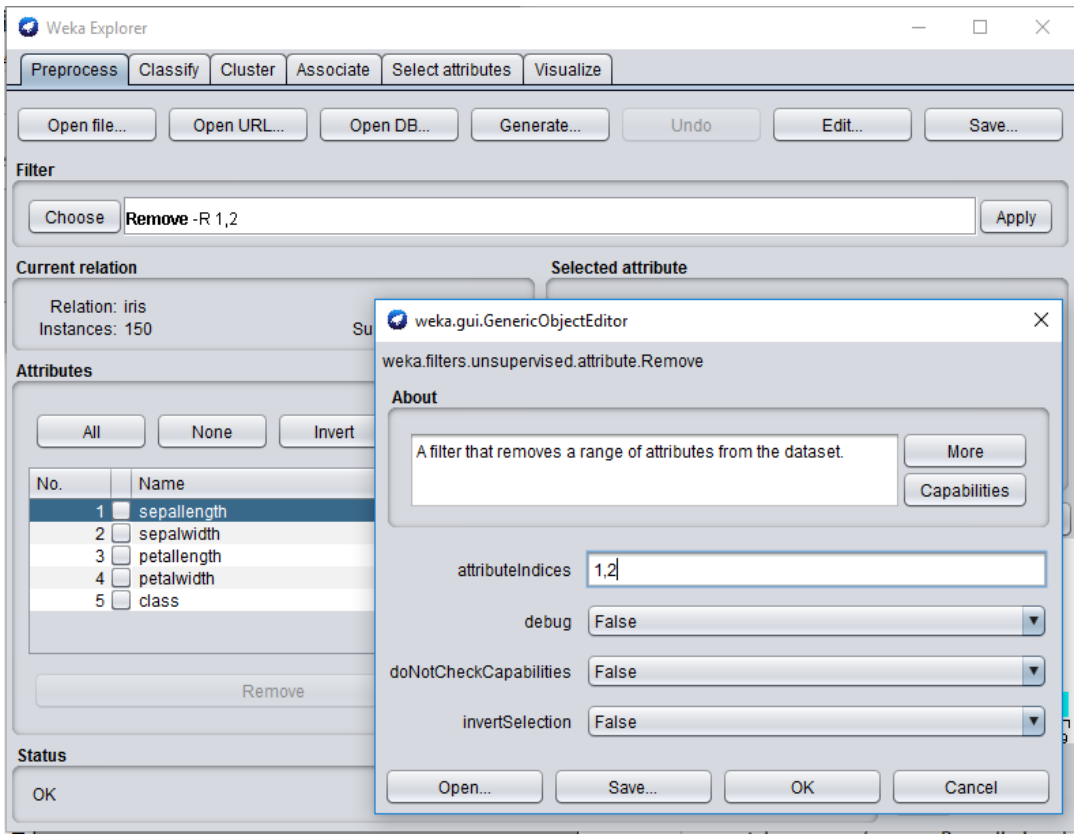


Figura 1.- Uso de filters – unsupervised – atribute -Remove con los atributos a eliminar 1,2.

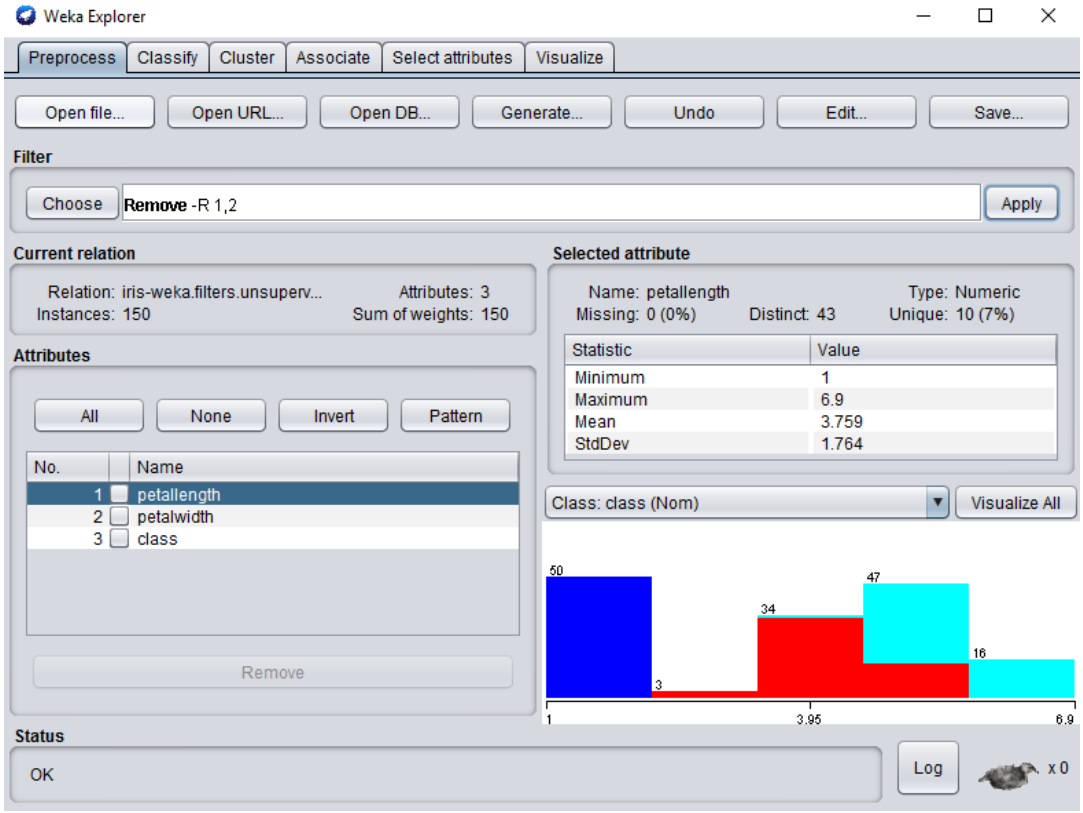


Figura 2.- Preprocess después de quitar los atributos 1 y 2.

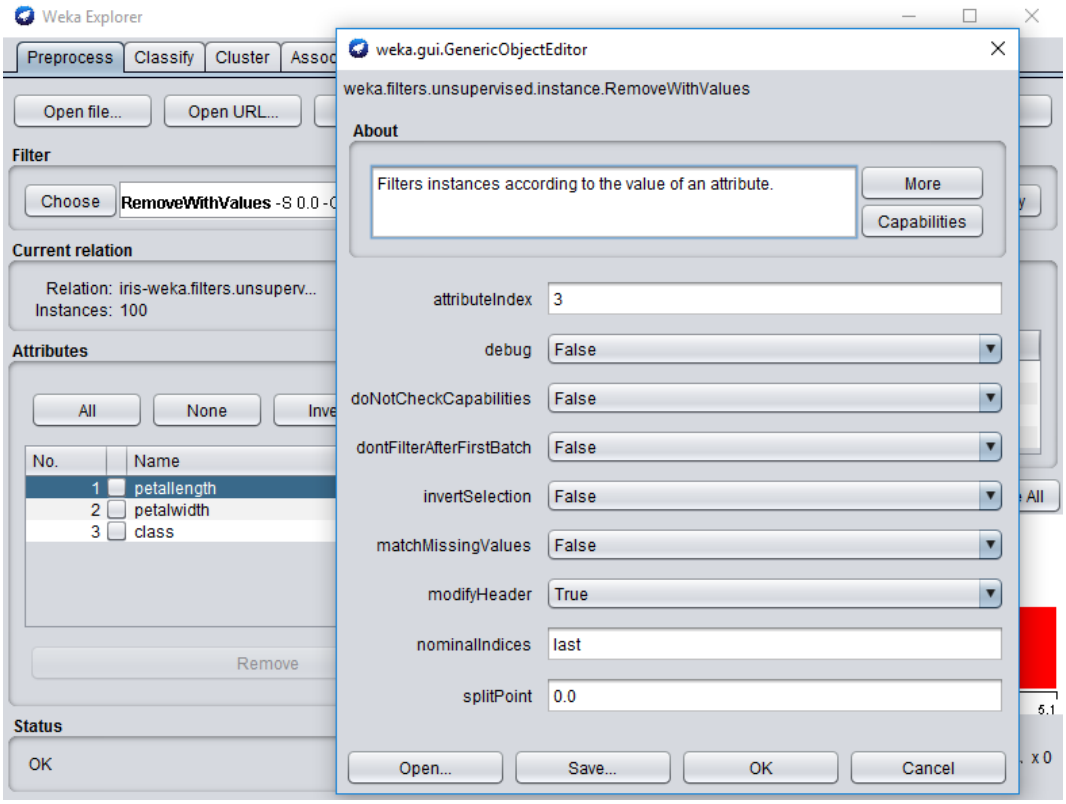


Figura 3.- Uso de filters- unsupervised – instance – RemoveWithValues con index 3 que corresponde a la clase y nominalIndices con last para que sea la última clase que parece, Iris-virginica.

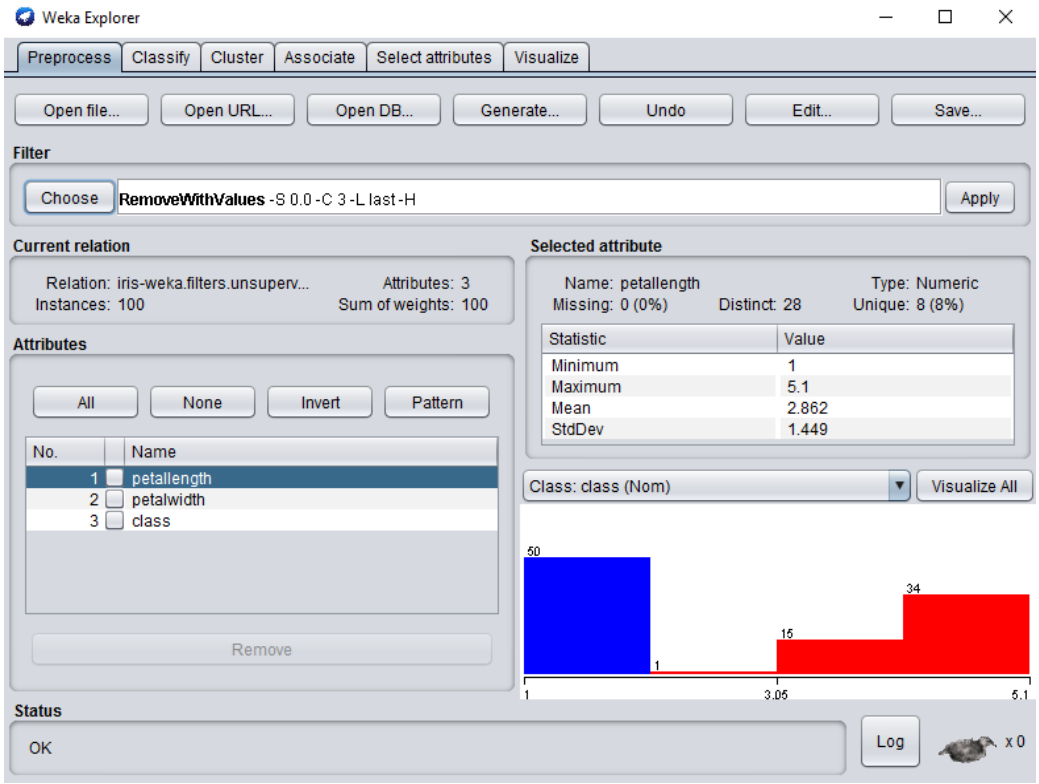


Figura 4.- Preprocess después de luso del filtro anterior.

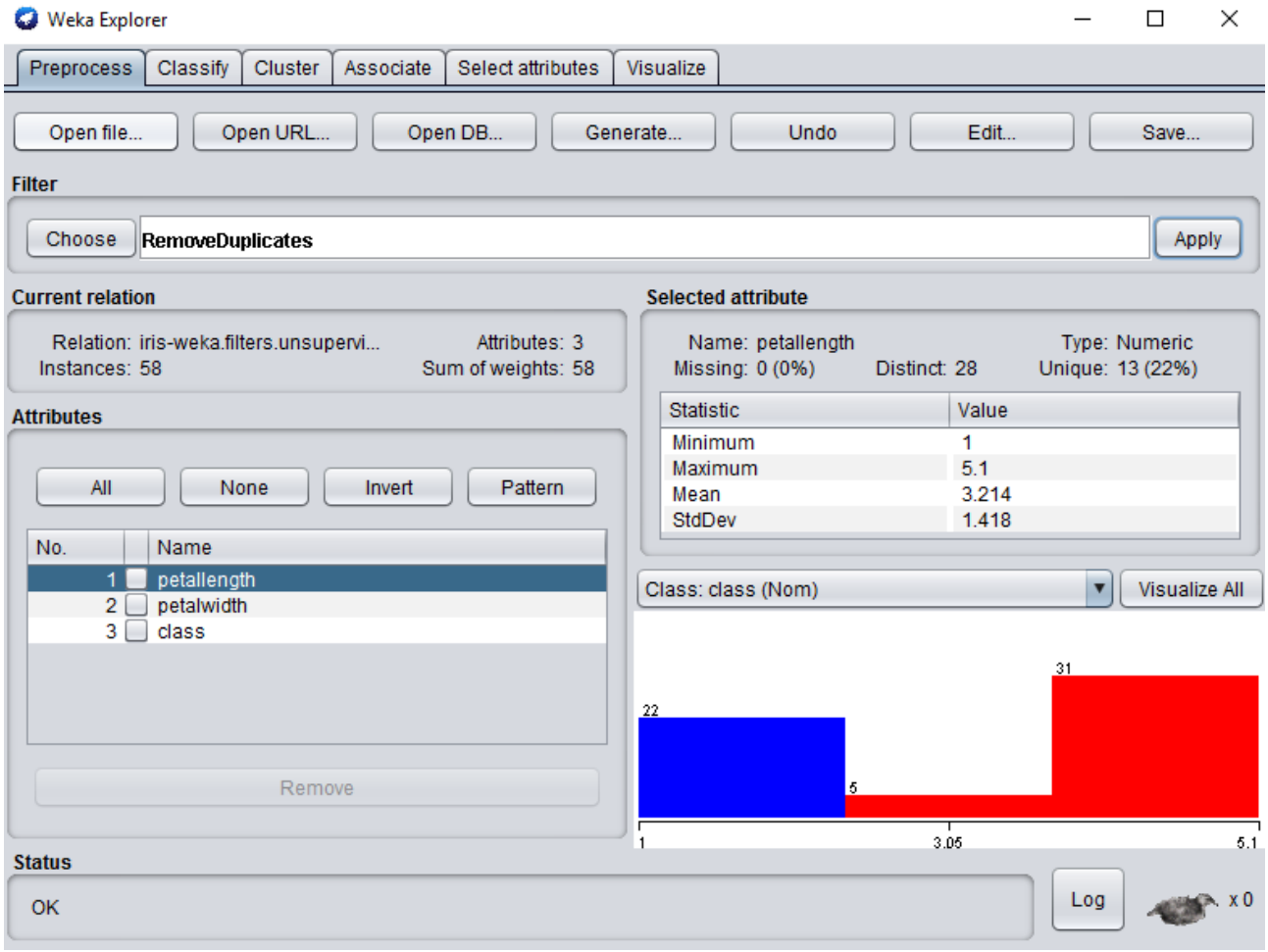


Figura 5.- Uso del atributo RemoveDuplicates

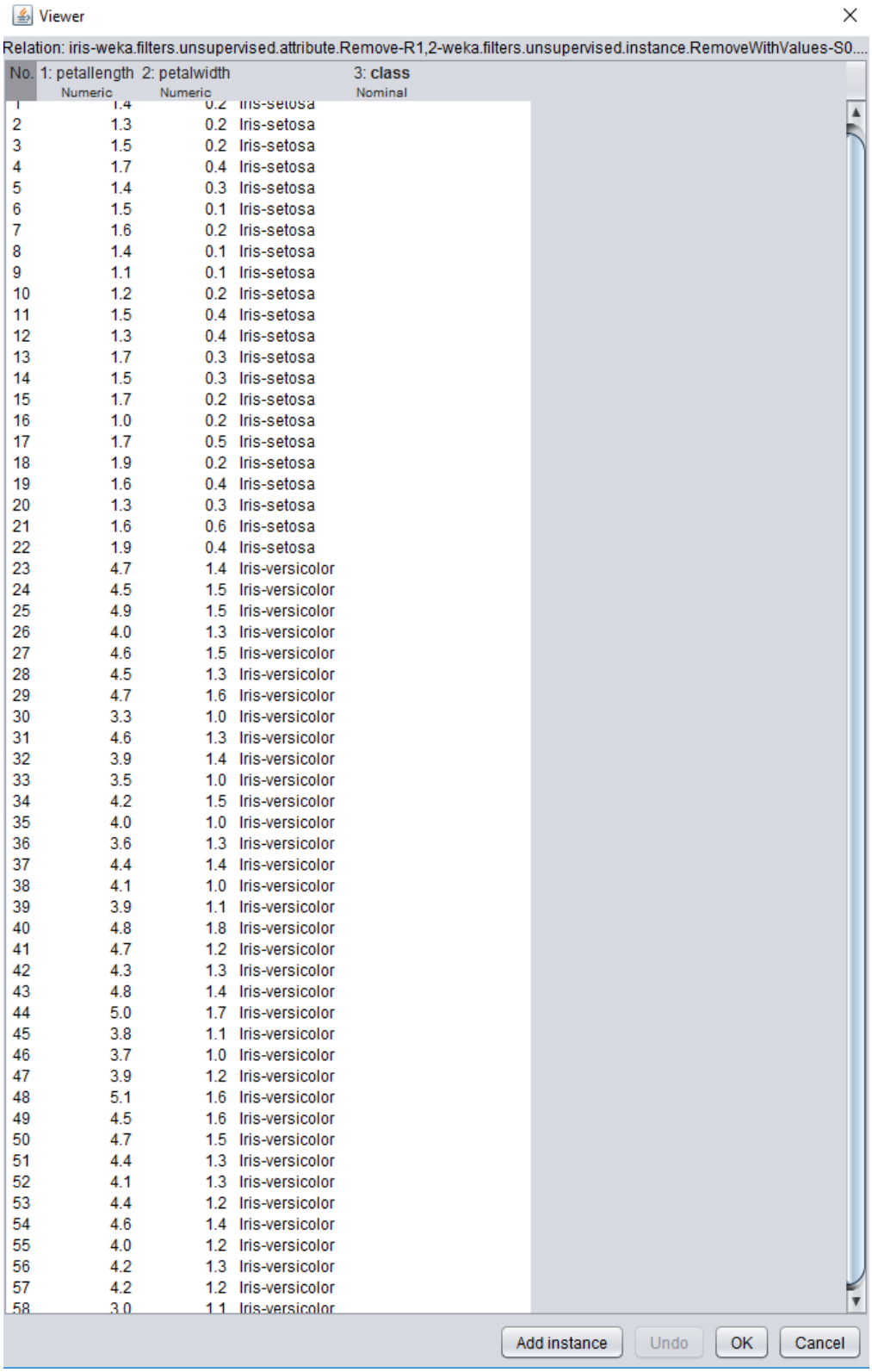


Figura 6.- Conjunto resultante de aplicar todos los filtros requeridos (58 instancias) y poder generar el fichero "iris_2D_2Clases_Ls.arff"

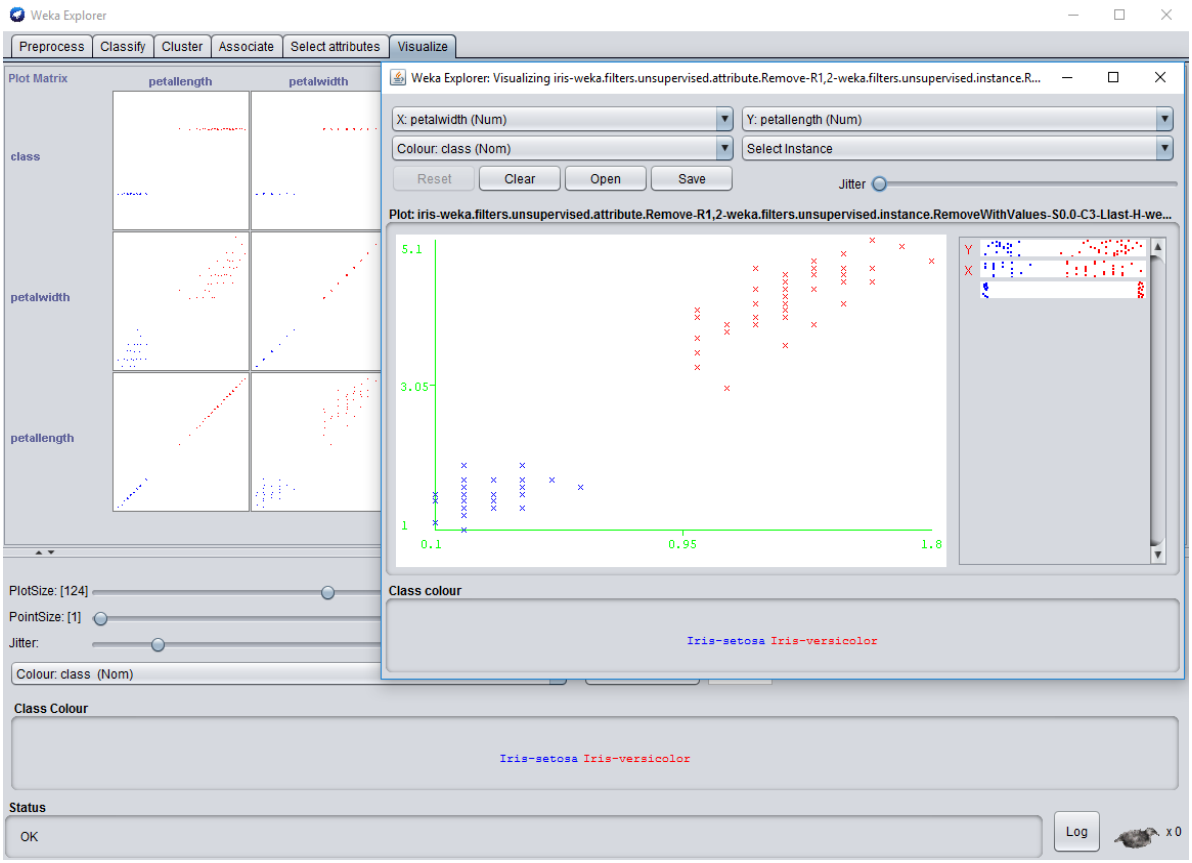


Figura 7.- Plot petallength vs petalwidth, con las 58 instancias.

No ha hecho falta introducir ruido en el plot, las instancias no se solapan, se ve perfectamente que el conjunto de datos es separable linealmente. Como ampliación solo mencionaremos de momento que los hiperplanos a utilizar pueden ser infinitos, como conocemos de la teoría de SVM dado que son separables los conjuntos de cada clase.

De cualquier forma el hiperplano óptimo es el que buscamos, corresponde al tema de Teoría de Optimización, tratado en el punto 2 de el documento de Carmona disponible en el alf.

3. Asumiendo que el conjunto de datos cargado corresponde al fichero "iris_2D_2Clases_Ls.arff", desde la venta Explorer, seleccione el panel Classify y escoja el algoritmo SMO (classifiers > functions). SMO (del inglés Sequential Minimal Optimization) corresponde a la implementación en Weka del denominado algoritmo de optimización mínima secuencial [Plat, 1998], uno de los algoritmos utilizados para el entrenamiento de clasificadores de vectores soporte. Los parámetros de configuración de este muestran en la figura. En particular, son de especial interés los parámetros c, FilterType y kernel. El primero permite al usuario actuar sobre el denominado parámetro de coste C. En particular, este parámetro permite actuar sobre la anchura del margen (ver sección 4 en [Carmona 2016]). Así, a medida que ele valor de C se hace más grande, la anchura del margen se hará más pequeña, reduciéndose así la probabilidad de que aparezcan ejemplos dentro del margen o mal clasificados. Por contra, a medida que C se hace más pequeño, la anchura del margen aumentará, a costa, normalmente, de la aparición de ejemplos situados dentro del margen o incluso mal clasificados (vectores soporte ligados). No obstante, el coste computacional asociado a la búsqueda del modelo será mayor cuanto mayor sea el valor de C. Además , en el primer caso, se generarán modelos más sobreajustados, normalmente3 definidos a partir sólo de ejemplos situados en la frontera del margen (vectores soporte), y en el segundo, modelos con mayor poder de generalización, definidos a partir tanto de vectores soporte como de

vectores soporte ligados. En cuanto al parámetro FilterType, permite establecer si los datos serán o no normalizados estandarizados. La literatura aporta evidencia de que la normalización es beneficiosa cuando el rango de variación de los diferentes atributos es muy dispar [Hsu et al., 2016]. Sin embargo, tiene el inconveniente de que la interpretación d el modelo aprendido tiene que hacerse en función del nuevo rango de valores obtenido tras realizar el proceso de normalización, disminuyendo así la interpretabilidad de dicho modelo. Finalmente, el parámetro kernel permitirá, como su propio nombre indica, elegir el tipo de kernel. La pulsación del botón More mostrará una ventana de ayuda con la función del resto de parámetros, más vinculados al funcionamiento interno del algoritmo y que, mientras no se diga lo contrario, se deberían mantener a su valor por defecto. A continuación, realice las siguientes acciones:

- *Dado que se quiere primar la interpretabilidad del modelo aprendido, el parámetro FilterType se pondrá cal valor "No normalization-standardizarion". Como kernel, se seleccionará "RBFKernel", también llamado kernel gaussiano. A su vez, esto implicará tener que realizar la configuración del mismo. En este caso, su parámetro más importante corresponde al valor de gamma, dado que matemáticamente este tipo de kernel se define como:*

$$K_G(\boldsymbol{x}, \boldsymbol{x}') = e^{-\gamma \left\| \boldsymbol{x} - \boldsymbol{x}' \right\|^2} \equiv e^{-\gamma < (\boldsymbol{x} - \boldsymbol{x}'), (\boldsymbol{x} - \boldsymbol{x}') >}, \ \gamma > 0$$

Se elegirá gamma = 10^{-2} y el resto de parámetros se dejarán a su valor por defecto. El objeto de este experimento es analizar cómo varía el tanto por ciento de instancias correctamente clasificadas y el número de vectores soporte utilizados por el modelo a medida que cambia el valor de C. Para ello, selecciones la opción Cross-validation (Folds=10) y ejecute el algoritmo para C={10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}}.

Documentación a entregar apartador 5.3.a-

i)Muestre en una tabla cómo varía el porcentaje de instancias bien clasificadas y el número de vectores soporte, |Vs|, utilizados por el modelo, en función de C.

C	% instancias ok	|Vs|
10^{-1}	62.069%	44
10^{0}	100%	30
10^{1}	100%	8
10^{2}	100%	2
10^{3}	100%	2

Tabla 1.- Porcentaje de instancias correctamente clasificadas según el valor de C y el nº de vectores soporte

ii)Justifique por que, a pesar de ser un conjunto de ejemplos linealmente separables, se producen errores de clasificación en modelos que se aprendieron con valores bajos de C.

El conjunto de ejemplos es linealmente separables, lo cual nos lleva a pensar que podemos utilizar cualquier kernel, el gausiano es el elegido en el enunciado, el lineal o polynomial también podrían estudiarse, el caso es que si el margen es pequeño no nos clasificará adecuadamente, a no ser que se alcance un umbral que nos ofrezca vectores soporte ligados. No necesitamos en este caso una función kernel tan compleja para hacer la transformación, pero también según la teoría podemos utilizarlas tratarlas y elegir la que mejor resultados obtengamos, ya que no hay un criterio exacto al respecto. De lo que se trata es de transformar nuestro espacio primal en otro mucho mayor, espacio de características que nos permita separar los conjuntos, mapeando esos puntos con el uso de la función kernel.

De todas formas no podemos encontrar un algoritmo que nos ofrezca exactamente el valor del margen de C optimo, pero nos bastará con una estimación de ese valor, atendiendo a que un valor de C pequeño aumenta la holgura permitiendo valores de margen optimo mayores y que se clasifique con errores. Por contra un valor mayor de cero y cuanto mayor sea admitirá que los conjuntos son separables, ofreciendo unos valores de clasificación mejores [Carmona 2016 pagina 11].

iii)Los modelos o funciones de decisión obtenido por Weka, se pueden expresar de forma genérica como:

$$D(\boldsymbol{x}) = \pm a_1 * < x_{11}\ x_{21} > *X] \pm ...a_n * < x_{1n}\ x_{2n} > *X] \pm a_0$$

donde el símbolo ± delante de cada sumando denota que el signo puede ser positivo o negativo. Indique qué representa: (a) el valor del signo, (b) los coeficientes a_i, i = 0, … n y (c) las tuplas (X_{1j},X_{2j}), j= 1, …, n.

El valor del signo nos dirá si el ejemplo cae a un lado o a otro del hiperplano para su clasificación, es decir, la clase a la que pertenece cada vector soporte resultante del conjunto de entrenamiento. El coeficiente corresponde a la solución que optimiza el problema dual y las tuplas son los vectores soporte.

iv) Atendiendo a las respuestas das en (iii), exprese la función de decisión obtenida por Weka para el caso C = 10^2 de acuerdo al siguiente esquema:

$$D(\boldsymbol{x}) = \alpha_0^* + \alpha_1^* y_1 K(\boldsymbol{x}, \boldsymbol{x}_1) + ... + \alpha_n^* y_n K(\boldsymbol{x}, \boldsymbol{x}_n)$$

donde, K(x,x'), en este caso, tiene que expresarse de acuerdo a la expresión (1).

La función de decisión seria:

$$D(x) = 0 + 59.3249\ \{+1\}\ K(<3\ 1.1>,X) + 59.3249\ \{-1\}\ K(<1.9\ 0.4>,X)$$

donde cada X (x' de la expresión) seria el vector en forma <a b> de cada ejemplo que sería mapeado y clasificado. Si expresamos la función kernel gaussiana K(x,x') resultaría:

$$K(<3\ 1.1>,X) = e^{-0.01||<3\ 1.1> - X||2} \equiv e^{-0.01<(<3\ 1.1> - X)\ ,(<3\ 1.1> - X)>}$$
$$K(<1.9\ 0.4>,X) = e^{-0.01||<1.9\ 0.4> - X||2} \equiv e^{-0.01<(<1.9\ 0.4> - X)\ ,(<1.9\ 0.4> - X)>}$$

donde de nuevo cada X seria el ejemplo a clasificar, que correspondería a los vectores x', de la expresión**.

** Suponemos nuestra función K cumple las dos propiedades exigidas por Aronszajn y que de la primera propiedad de simetría podemos escribir que K(x,x')=K(x',x) y que por ello no es necesario conocer el valor de los puntos que forman el vector de ejemplo X en el espacio de características, tan solo utilizar el valor de los ejemplos pertenecientes al problema primal y los vectores soporte.

v) Dado que estamos trabajando en un problema de dos clases, cuál es la etiqueta, {+1,-1}, asignada, respectivamente, a la clase "Iris-setosa" e Irisi-versicolor" para el caso $C=10^2$.

Iris-versicolor es ejemplo clasificado como positivo {+1}
Iris- setosa es ejemplo clasificado como negativo {-1}

vi) Adjunte el modelo obtenido por Weka para el caso C=10. A partir de dicho modelos y de los vectores soporte que lo forman indique y justifique (sin evaluar el modelo) cuáles de ellos son vectores soporte pertenecientes a la frontera del margen y cuáles otros son vectores soporte ligados.

```
=== Run information ===

Scheme:       weka.classifiers.functions.SMO -C 10.0 -L 0.001 -P 1.0E-12 -N 2 -V -1 -W 1 -K
"weka.classifiers.functions.supportVector.RBFKernel -G 0.01 -C 250007" -calibrator
"weka.classifiers.functions.Logistic -R 1.0E-8 -M -1 -num-decimal-places 4"
Relation:     iris-weka.filters.unsupervised.attribute.Remove-R1,2-
weka.filters.unsupervised.instance.RemoveWithValues-S0.0-C3-Llast-H-
weka.filters.unsupervised.instance.RemoveDuplicates
Instances:    58
Attributes:   3
              petallength
              petalwidth
              class
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

SMO
```

```
Kernel used:
  RBF kernel: K(x,y) = e^-(0.01* <x-y,x-y>^2)

Classifier for classes: Iris-setosa, Iris-versicolor

BinarySMO

-       0.8858 * <1.7 0.4 > * X]
+       10      * <3.3 1 > * X]
+       10      * <3 1.1 > * X]
+       10      * <3.5 1 > * X]
-       10      * <1.7 0.5 > * X]
+       0.8858 * <3.7 1 > * X]
-       10      * <1.9 0.2 > * X]
-       10      * <1.9 0.4 > * X]
-       0.1182

Number of support vectors: 8

Number of kernel evaluations: 660 (69.216% cached)



Time taken to build model: 0.01 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances          58               100      %
Incorrectly Classified Instances         0                0      %
Kappa statistic                          1
Mean absolute error                      0
Root mean squared error                  0
Relative absolute error                  0      %
Root relative squared error              0      %
Total Number of Instances               58

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               1,000    0,000    1,000      1,000    1,000      1,000    1,000     1,000     Iris-setosa
               1,000    0,000    1,000      1,000    1,000      1,000    1,000     1,000     Iris-versicolor
Weighted Avg.  1,000    0,000    1,000      1,000    1,000      1,000    1,000     1,000

=== Confusion Matrix ===

 a  b   <-- classified as
22  0 |  a = Iris-setosa
 0 36 |  b = Iris-versicolor
```

Datos 1.- Ejecución en Weka para C=10

Según lo entendido en la teoría y ampliación de esta, los multiplicadores lagrangianos de la solución dual, que corresponden a los coeficientes α^*, nos permiten conocer si estamos ante vectores soporte frontera si ese coeficiente esta entre 0 y el valor del margen C, y son ligados si es igual a C.

O bien, podemos también decir que si $|D(x)|=1$ es vector soporte frontera, aunque para ello tendríamos, que evaluar la formula completa tal y como se hará en el siguiente apartado.

Serían vectores soporte frontera {-1}<1.7 0.4 > y {+1}<3.7 1 > , siendo el resto ligados.

vii) Mediante la evaluación del modelo es posible diferenciar también entre vectores soporte frontera y vectores soporte ligados. Así, si |D(x)|=1, entonces x es un vector soporte frontera, siendo |.| el operador valor absoluto. En otro caso, x es un vector soporte ligado. Exprese el modelo obtenido para C=10 en la forma dada por la expresión (3) y compruebe, mediante esta estrategia, que las respuestas dadas en (vi) concuerdan con los resultados ahora obtenidos.

Vamos a evaluar un par de vectores soporte para confirmar las respuestas anteriores, por ejemplo {+1}<3.7 1> como soporte frontera y {+1}<1.9 0.2>. Para ello debemos evaluar el siguiente modelo de D(x) para C=10. Para ello utilizaremos los resultados ofrecidos por weka con respecto a los valores de dichos vectores soporte que forman parte importante de la expresión de la función de decisión D(x).

```
BinarySMO

-       0.8858 * <1.7 0.4 > * X]
+       10       * <3.3 1 > * X]
+       10       * <3 1.1 > * X]
+       10       * <3.5 1 > * X]
-       10       * <1.7 0.5 > * X]
+       0.8858 * <3.7 1 > * X]
-       10       * <1.9 0.2 > * X]
-       10       * <1.9 0.4 > * X]
-       0.1182
```

Datos 2.- Valores de los vectores soporte con el valor α*y C=10.

$$D(\boldsymbol{x}) = \alpha_0^* + \alpha_1^* y_1 K(\boldsymbol{x}, \boldsymbol{x}_1) + \ldots + \alpha_n^* y_n K(\boldsymbol{x}, \boldsymbol{x}_n)$$

Va a ser la expresión que vamos a obtener, aunque vamos a sustituir los K, de la función kernel utilizando la gaussiana propuesta por el enunciado de la actividad, de forma similar al apartado iv) pero esta vez cada vector X, será el vector soporte que hemos mencionado anteriormente que íbamos a evaluar.

Para el vector soporte <3.7 1>

$$D(<3.7\ 1>) = (-0.1182) +$$
$$0.8858\ (-1)\ e^{-0.01<(<1.7\ 0.4> - <3.7\ 1>)\ ,(<1.7\ 0.4> - <3.7\ 1>)>} +$$
$$10\ (+1)\ e^{-0.01<(<3.3\ 1> - <3.7\ 1>)\ ,(<3.3\ 1> - <3.7\ 1>)>} +$$
$$10\ (+1)\ e^{-0.01<(<3\ 1.1> - <3.7\ 1>)\ ,(<3\ 1.1> - <3.7\ 1>)>} +$$
$$10\ (+1)\ e^{-0.01<(<3.5\ 1> - <3.7\ 1>)\ ,(<3.5\ 1> - <3.7\ 1>)\ >} +$$
$$10\ (-1)\ e^{-0.01<(<1.7\ 0.5> - <3.7\ 1>)\ ,(<1.7\ 0.5> - <3.7\ 1>)>} +$$
$$0.8858\ (+1)\ e^{-0.01<(<3.7\ 1> - <3.7\ 1>)\ ,(<3.7\ 1> - <3.7\ 1>)>} +$$
$$10\ (-1)\ e^{-0.01<(<1.9\ 0.2> - <3.7\ 1>)\ ,(<1.9\ 0.2> - <3.7\ 1>)>} +$$
$$10\ (-1)\ e^{-0.01<(<1.9\ 0.4> - <3.7\ 1>)\ ,(<1.9\ 0.4> - <3.7\ 1>)>} = (+0{,}99999) \approx (+1)$$

De donde:

$$|D(<3.7\ 1>)| = 1$$

lo que nos confirma que ese vector soporte es **frontera.**

Para el vector soporte <1.9 0.2>

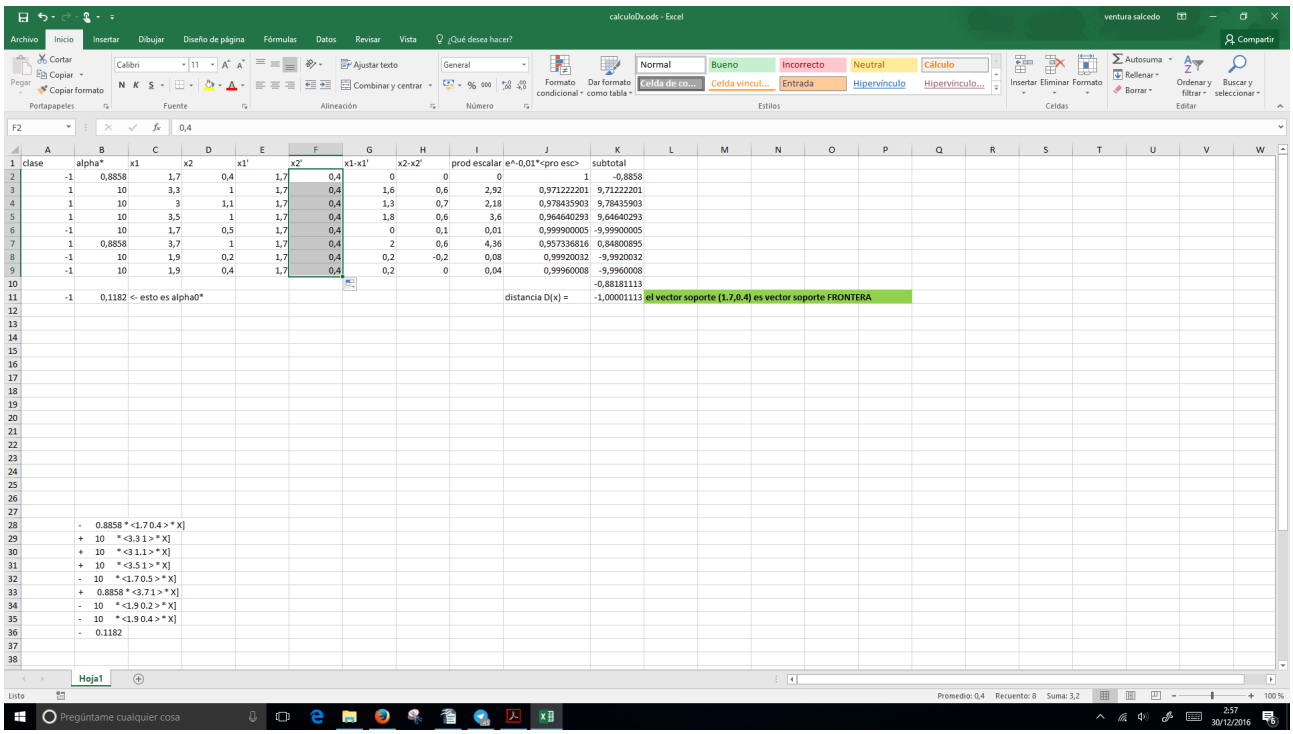
$$D(<1.9\ 0.2>) = (-0.1182) +$$
$$0.8858\ (-1)\ e^{-0.01<(<1.7\ 0.4> - <1.9\ 0.2>),(<1.7\ 0.4> - <1.9\ 0.2>)>} +$$
$$10\ (+1)\ e^{-0.01<(<3.3\ 1> - <1.9\ 0.2>),(<3.3\ 1> - <1.9\ 0.2>)>} +$$
$$10\ (+1)\ e^{-0.01<(<3\ 1.1> - <1.9\ 0.2>),(<3\ 1.1> - <1.9\ 0.2>)>} +$$
$$10\ (+1)\ e^{-0.01<(<3.5\ 1> - <1.9\ 0.2>),(<3.5\ 1> - <1.9\ 0.2>)>} +$$
$$10\ (-1)\ e^{-0.01<(<1.7\ 0.5> - <1.9\ 0.2>),(<1.7\ 0.5> - <1.9\ 0.2>)>} +$$
$$0.8858\ (+1)\ e^{-0.01<(<3.7\ 1> - <1.9\ 0.2>),(<3.7\ 1> - <1.9\ 0.2>)>} +$$
$$10\ (-1)\ e^{-0.01<(<1.9\ 0.2> - <1.9\ 0.2>),(<1.9\ 0.2> - <1.9\ 0.2>)>} +$$
$$10\ (-1)\ e^{-0.01<(<1.9\ 0.4> - <1.9\ 0.2>),(<1.9\ 0.4> - <1.9\ 0.2>)>} = (-0{,}90576)$$

De donde:

$$|D(<1.9\ 0.2>)| = 0{,}90576$$

lo que nos confirma que ese vector soporte es **ligado.**

Podríamos hacer esto mismo para evaluar cada uno de los vectores soporte. Por otro lado se adjunta un hoja de calculo en la siguiente dirección que he creado para calcular ese valor, tanto solo sustituir la columna x1' y x'2 por el valor del ejemplo a evaluar.



clase	alpha*	x1	x2	x1'	x2'	x1-x1'	x2-x2'	prod escalar	e^-0,01*<pro esc>	subtotal
-1	0,8858	1,7	0,4	1,7	0,4	0	0	0	1	-0,8858
1	10	3,3	1	1,7	0,4	1,6	0,6	2,92	0,971222201	9,71222201
1	10	3	1,1	1,7	0,4	1,3	0,7	2,18	0,978435903	9,78435903
1	10	3,5	1	1,7	0,4	1,8	0,6	3,6	0,964640293	9,64640293
-1	10	1,7	0,5	1,7	0,4	0	0,1	0,01	0,999900005	-9,99900005
1	0,8858	3,7	1	1,7	0,4	2	0,6	4,36	0,957336816	0,84800895
-1	10	1,9	0,2	1,7	0,4	0,2	-0,2	0,08	0,99920032	-9,9920032
-1	10	1,9	0,4	1,7	0,4	0,2	0	0,04	0,99960008	-9,9960008
										-0,88181113
-1	0,1182	<- esto es alpha0*						distancia D(x) =		-1,00001113 el vector soporte (1.7,0.4) es vector soporte FRONTERA

- 0.8858 * <1.7 0.4 > * X]
+ 10 * <3.3 1 > * X]
+ 10 * <3 1.1 > * X]
+ 10 * <3.5 1 > * X]
- 10 * <1.7 0.5 > * X]
+ 0.8858 * <3.7 1 > * X]
- 10 * <1.9 0.2 > * X]
- 10 * <1.9 0.4 > * X]
- 0.1182

Figura 8.- Preview de la hoja de calculo propuesta para la evaluación D(x), (usar zoom%)

Opcional: Demuestre que, para el caso $C=10^2$, la expresión que define la frontera de separación viene dada por una recta. Para ello, tendrá que hacer D(x) = 0 y operar la expresión resultante convenientemente.

BinarySMO

59.3249 * <3 1.1 > * X]
- 59.3249 * <1.9 0.4 > * X]
- 0

Datos 3.- Valores de los vectores soporte con el valor α*y C=100.

Para C=100 tendremos la expresión como comentamos en el apartado iv). Nos piden que demostremos haciendo D(x)=0, que la expresion resultante es una recta.

$$D(x) = 0 + 59.3249 \{+1\} K(<3\ 1.1>,X) + 59.3249 \{-1\} K(<1.9\ 0.4>,X) = 0$$

Sustituyendo la expresión K de la función kernel gaussiana sería:

$$59.3249 \{+1\}\ e^{-0.01||<3\ 1.1> - X||2} + 59.3249 \{-1\}\ e^{-0.01||<1.9\ 0.4> - X||2} = 0$$

Operando los valores de clase

$$59.3249\ e^{-0.01||<3\ 1.1> - X||2} - 59.3249\ e^{-0.01||<1.9\ 0.4> - X||2} = 0$$

Pasando el segundo termino a la derecha de la igualdad

$$59.3249\ e^{-0.01||<3\ 1.1> - X||2} = 59.3249\ e^{-0.01||<1.9\ 0.4> - X||2}$$

Quitamos la constante en ambos lados

$$e^{-0.01||<3\ 1.1> - X||2} = e^{-0.01||<1.9\ 0.4> - X||2}$$

Como la función e^x es inyectiva podemos escribir

$$-0.01||<3\ 1.1> - X||^2 = -0.01||<1.9\ 0.4> - X||^2$$

Quitamos la constante en ambos términos

$$||<3\ 1.1> - X||^2 = ||<1.9\ 0.4> - X||^2$$

Ponemos X en forma vectorial

$$||<3\ 1.1> - <x1\ x2>||^2 = ||<1.9\ 0.4> - <x1\ x2>||^2$$

Realizamos la diferencia en cada término

$$||(3 - x1\ ,\ 1.1 - x2)\ ||^2 = ||(1.9 - x1\ ,\ 0.4 - x2>\ ||^2$$

Operamos la norma , producto escalar de cada termino y la raiz(√ = raiz cuadrada)

$$(\sqrt{}\ (\ (3 - x1)^2 + (1.1 - x2)^2\ )\ )^2 = (\sqrt{}\ (\ (1.9 - x1)^2 + (0.4 - x2)^2))^2$$

Como la raíz siempre será positiva podemos escribir

$$(3 - x1)^2 + (1.1 - x2)^2 = (1.9 - x1)^2 + (0.4 - x2)^2$$

Agrupando las variables con los cuadrados, tendremos **una diferencia de cuadrados**

$$(3 - x1)^2 - (1.9 - x1)^2 = (0.4 - x2)^2 - (1.1 - x2)^2$$

Utilizando la identidad notable $a^2 + b^2 = (a+b)(a-b)$, para todo a,b real, y haciendo un cambio de variable, usando a,b,c y d para que se vea claro lo que estamos haciendo:

$$a^2 - b^2 = c^2 - d^2 \rightarrow (a+b)(a-b) = (c+d)(c-d)$$

Deshaciendo el cambio de variables, nos queda:

$$((3-x1)+(1.9-x1))\,((3-x1)-(1.9-x1)) = ((0.4-x2)+(1.1-x2))((0.4-x2)-(1.1-x2))$$

Operando el signo -

$$(3-x1+1.9-x1)(3-x1-1.9+x1) = (0.4-x2+1.1-x2)(0.4-x2-1.1+x2)$$

Ahora eliminamos las variables que tienen signos contrarios de dentro de los parentessis, marcados con rojo y hacemos las sumas, quedando

$$(4.9 - 2x1)(1.1) = (1.5-2x2)(-0.7)$$

Usando x1 como X y x2 como Y para verlo mas claro

$$(4.9 - 2X)(1.1) = (1.5-2Y)(-0.7)$$

Operamos para deshacer paréntesis y sale

$$5.39 - 2.2X = 1.4Y - 1.05$$

Lo que nos da pasando y a un lado y x a otro y operando que

$$5.39 - 2.2X = 1.4Y - 1.05 \rightarrow 6.44 - 2.2X = 1.4Y \rightarrow$$

$$\mathbf{Y = -1.57X + 4.6}$$

Lo cual es una recta siguiendo que Y=mX+n, con m= -1.57 es la pendiente y n= 4.6 es el corte con el eje Y, dando por demostrado lo que pide el enunciado.

Seleccione ahora gamma = 1 y ejecute el algoritmo para C={10^{-1}, 10^{0}, 10^{1}, 10^{2}, 10^{3}}. Después, repita lo mismo, pero usando gamma = 10^{4}

Documentación a entregar apartador 5.3.b-

i) Construya dos tablas como la indicada en el caso anterior, una para los resultados obtenidos con gamma = 1 y, la otra, para gamma = 10^{4}.

C con gamma=1	% instancias ok	|Vs|
10^{-1}	100%	31
10^{0}	100%	7
10^{1}	100%	6
10^{2}	100%	6
10^{3}	100%	6

Tabla 2.- Porcentaje de instancias correctamente clasificadas para gamma=1

C con gamma=10^{-4}	% instancias ok	|Vs|
10^{-1}	62.069%	44
10^{0}	62.069%	44
10^{1}	62.069%	44
10^{2}	100%	30
10^{3}	100%	8

Tabla 3.- Porcentaje de instancias correctamente clasificadas para gamma=0.0001

C con gamma=0.01	% instancias ok	|Vs|
10^{-1}	62.069%	44
10^{0}	100%	30
10^{1}	100%	8
10^{2}	100%	2
10^{3}	100%	2

Tabla 4.- Copia de la tabla 1. Porcentaje de instancias correctamente clasificadas para gamma=0.01

ii) Observando las dos tablas obtenidas, junto con la tabla obtenida en el paso anterior, para un valor de C fijo, ¿se puede apreciar alguna tendencia en el valor de |Vs| respecto de los tres valores de gamma considerados? Intente justificar dicha tendencia. Pista: Recuerde que el parámetro gamma está relacionado con el valor σ (desviación típica de la gaussiana), siendo este último inversamente proporcional al valor del primero.

La desviación típica es una medida de dispersión para variables, para los ejemplos en nuestro caso, un gamma mas pequeño indicará una mayor dispersión de los datos, siendo necesarios mayor número de vectores soporte, un gamma mayor indica que la desviación típica es

menor y por tanto datos más concentrados menor número de vectores soporte. Recordemos que estamos ante un problema de optimización y que el gamma es un parámetro de la propio función kernel, lo que puede hacer que la función kernel elegida no sea la adecuada. Por otro lado debemos mencionar también el margen que influye en las clasificaciones, con un mayor margen obtenemos mejores clasificaciones. No podemos dejar de lado el hecho que necesitamos un espacio de características que debe ser mucho mayor que el problema original planteado y que el el hiperplano debe ser optimo. Dicho todo esto significa que ni podemos conocer a priori el mejor valor C, ni el mejor de gamma, ni el número de vectores soporte para abordar el hiperplano optimo, ni tan siquiera la computación necesaria, si no que el sistema debe ser entrenado de varias formas para la resolución de cada problema de clasificación de instancias que se nos plantee.

4. Vamos a considerar ahora el caso de un conjunto de ejemplos no separables linealmente. Como en el caso anterior, se construirá un conjunto de ejemplos ficticio a partir del conjunto "iris.arff". A continuación realice las siguientes acciones:

- *Cargue el fichero "iris.arff" desde el panel Preprocess.*
- *Elimine los atributos "sepalwidth" y "sepallength".*
- *Elimine los ejemplos de la clase "Iris_setosa".*
- *Elimine ejemplos duplicados*
- *Utilice el botón Edirdel panel Preprocess para visualizar el contenido de los datos filtrados, asegurándose de que el conjunto de datos resultante contiene sólo ejemplos de la clase "Iris-virginica" (45 ejemplos) e "Iris-versicolor" (36 ejemplos) y, además, sólo aparecen los atributos "petallength" y "petalwidth".*
- *Utilice el botón Save para guardar el nuevo conjunto generado. Por ejemplo, utilice el nombre "iris_2D_2Clases_NLsep.arff".*
- *Utilice el panel Visualize de Explorer para ver el conjunto de datos filtrado. En particular, resulta de interés el plot que muestra las instancias de cada clase cuando se representa el atributo "petallength" respecto del atributo "petalwidth". Claramente, podrá observar que el conjunto de datos no es linealmente separable.*

Documentación a entregar apartado 5.4:

Mostrar el plot obtenido en el punto anterior.

Vamos a mostrar primero el listado de instancias después de la preparación propuesta:

No.	petallength	petalwidth	class

1	4.7	1.4	Iris-versicolor
2	4.5	1.5	Iris-versicolor
3	4.9	1.5	Iris-versicolor
4	4.0	1.3	Iris-versicolor
5	4.6	1.5	Iris-versicolor
6	4.5	1.3	Iris-versicolor
7	4.7	1.6	Iris-versicolor
8	3.3	1.0	Iris-versicolor
9	4.6	1.3	Iris-versicolor
10	3.9	1.4	Iris-versicolor
11	3.5	1.0	Iris-versicolor
12	4.2	1.5	Iris-versicolor
13	4.0	1.0	Iris-versicolor
14	3.6	1.3	Iris-versicolor
15	4.4	1.4	Iris-versicolor
16	4.1	1.0	Iris-versicolor
17	3.9	1.1	Iris-versicolor
18	4.8	1.8	Iris-versicolor
19	4.7	1.2	Iris-versicolor
20	4.3	1.3	Iris-versicolor
21	4.8	1.4	Iris-versicolor
22	5.0	1.7	Iris-versicolor
23	3.8	1.1	Iris-versicolor
24	3.7	1.0	Iris-versicolor
25	3.9	1.2	Iris-versicolor
26	5.1	1.6	Iris-versicolor
27	4.5	1.6	Iris-versicolor
28	4.7	1.5	Iris-versicolor
29	4.4	1.3	Iris-versicolor
30	4.1	1.3	Iris-versicolor
31	4.4	1.2	Iris-versicolor
32	4.6	1.4	Iris-versicolor
33	4.0	1.2	Iris-versicolor
34	4.2	1.3	Iris-versicolor
35	4.2	1.2	Iris-versicolor
36	3.0	1.1	Iris-versicolor
37	6.0	2.5	Iris-virginica
38	5.1	1.9	Iris-virginica
39	5.9	2.1	Iris-virginica
40	5.6	1.8	Iris-virginica
41	5.8	2.2	Iris-virginica
42	6.6	2.1	Iris-virginica
43	4.5	1.7	Iris-virginica
44	6.3	1.8	Iris-virginica
45	5.8	1.8	Iris-virginica
46	6.1	2.5	Iris-virginica
47	5.1	2.0	Iris-virginica

48	5.3	1.9	Iris-virginica
49	5.5	2.1	Iris-virginica
50	5.0	2.0	Iris-virginica
51	5.1	2.4	Iris-virginica
52	5.3	2.3	Iris-virginica
53	5.5	1.8	Iris-virginica
54	6.7	2.2	Iris-virginica
55	6.9	2.3	Iris-virginica
56	5.0	1.5	Iris-virginica
57	5.7	2.3	Iris-virginica
58	4.9	2.0	Iris-virginica
59	6.7	2.0	Iris-virginica
60	4.9	1.8	Iris-virginica
61	5.7	2.1	Iris-virginica
62	6.0	1.8	Iris-virginica
63	4.8	1.8	Iris-virginica
64	5.6	2.1	Iris-virginica
65	5.8	1.6	Iris-virginica
66	6.1	1.9	Iris-virginica
67	6.4	2.0	Iris-virginica
68	5.6	2.2	Iris-virginica
69	5.1	1.5	Iris-virginica
70	5.6	1.4	Iris-virginica
71	6.1	2.3	Iris-virginica
72	5.6	2.4	Iris-virginica
73	5.4	2.1	Iris-virginica
74	5.1	2.3	Iris-virginica
75	5.9	2.3	Iris-virginica
76	5.7	2.5	Iris-virginica
77	5.2	2.3	Iris-virginica
78	5.0	1.9	Iris-virginica
79	5.2	2.0	Iris-virginica
80	5.4	2.3	Iris-virginica
81	5.1	1.8	Iris-virginica

Datos 4. Listado de instancias de preparación del ejercicio eliminado sepalwidth y sepallength e Iris-setosa.

Mostramos a continuación un cuadro resumen del fichero que vamos guardar “iris_2D_2Clases_NLsep.arff”.

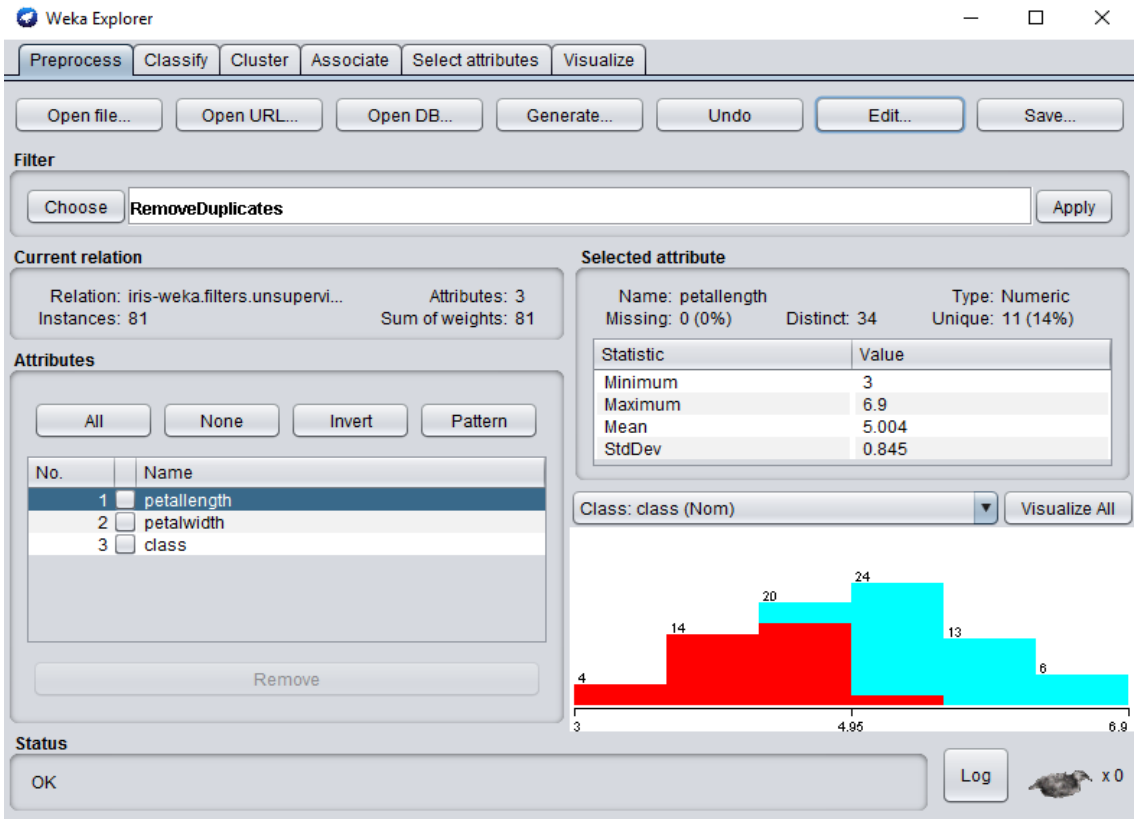


Figura 8.- Captura resumen del fichero que vamos a generar "iris_2D_2Clases_NLsep.arff"

Finalmente mostramos el plot requerido por el enunciado, que no es linealmente separable.

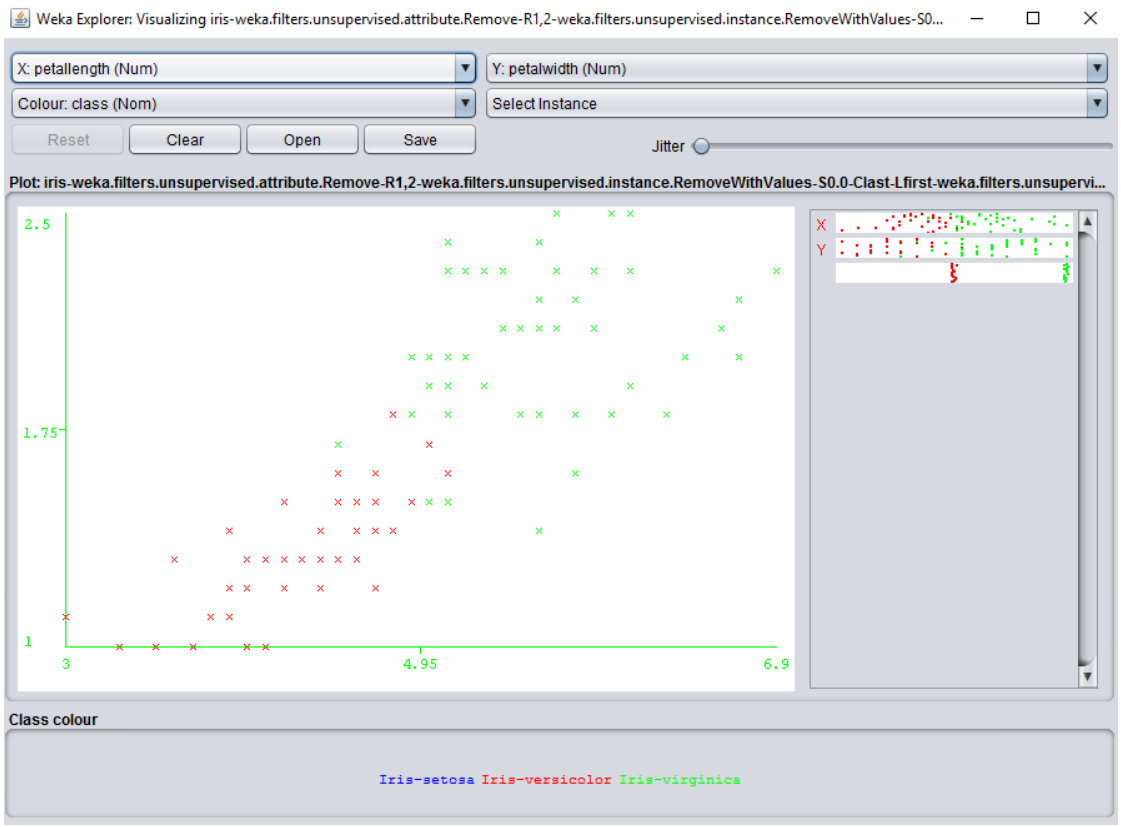


Figura 9.- Plot del fichero iris_2D_2Clases_NLsep.arff

5. Asumiendo que el conjunto de datos cargado corresponde al fichero "iris_2D_2Clases_Ls.arff" y que el algoritmo SMO está seleccionado, realice las siguiente acciones:

- *Selecciones el valor de FilterType a "No normalizan-standardization" (se quiere seguir manteniendo la interpretabilidad del modelo).*
- *Seleccione el kernel "RBFKernel".*
- *Seleccione la opción Use Trainera set como método de evaluación de los modelos aprendidos.*
- *Tal y como se vio en el paso, el comportamiento del algoritmo SMO depende en gran medida del valor de C y del valor de los parámetros que definen el kernel (parámetro gamma, en el caso del kernel gaussiano). Para resolver esta dependencia, lo normal es realizar un barrido en grid, de tal forma que el algoritmo se ejecutará para todas las combinaciones posibles de un conjunto de valores discretos elegidos para C y gamma. El modelo finalmente elegido será aquel asociado a la combinación que produzca el número de instancias mal clasificadas más bajo. Considere los siguientes conjuntos de C={10^{-1}, $1,10^{1},10^{2},10^{3},10^{4}$} y gamma = {$1,10^{-1},10^{-2}$}.*

Documentación a entregar apartado 5.5:

i) Construya una tabla en la que se muestre el número de instancias mal clasificadas obtenido por el modelo para cada par de combinaciones de valores (C,gamma).

C	gamma	Mal clasificadas
10^{-1}	1	4
10^{-1}	10^{-1}	5
10^{-1}	10^{-2}	36
10^{0}	1	5
10^{0}	10^{-1}	5
10^{0}	10^{-2}	5
10^{1}	1	6
10^{1}	10^{-1}	5
10^{1}	10^{-2}	5
10^{2}	1	4
10^{2}	10^{-1}	6
10^{2}	10^{-2}	5

10^3	1	3
10^3	10^{-1}	6
10^3	10^{-2}	5
10^4	1	1
10^4	10^{-1}	4
10^4	10^{-2}	6

Tabla 5.- Instancias mal clasificadas según configuración (C,gamma) para SMO con gaussiana y Use Training Set.

ii) De los resultados obtenidos en la tabla, elija el modelo de menor error y adjunte a la memoria la siguiente información:

a) Modelo obtenido por weka.

Se muestra pantalla de weka y listado de datos para C=10000 y gamma= 1

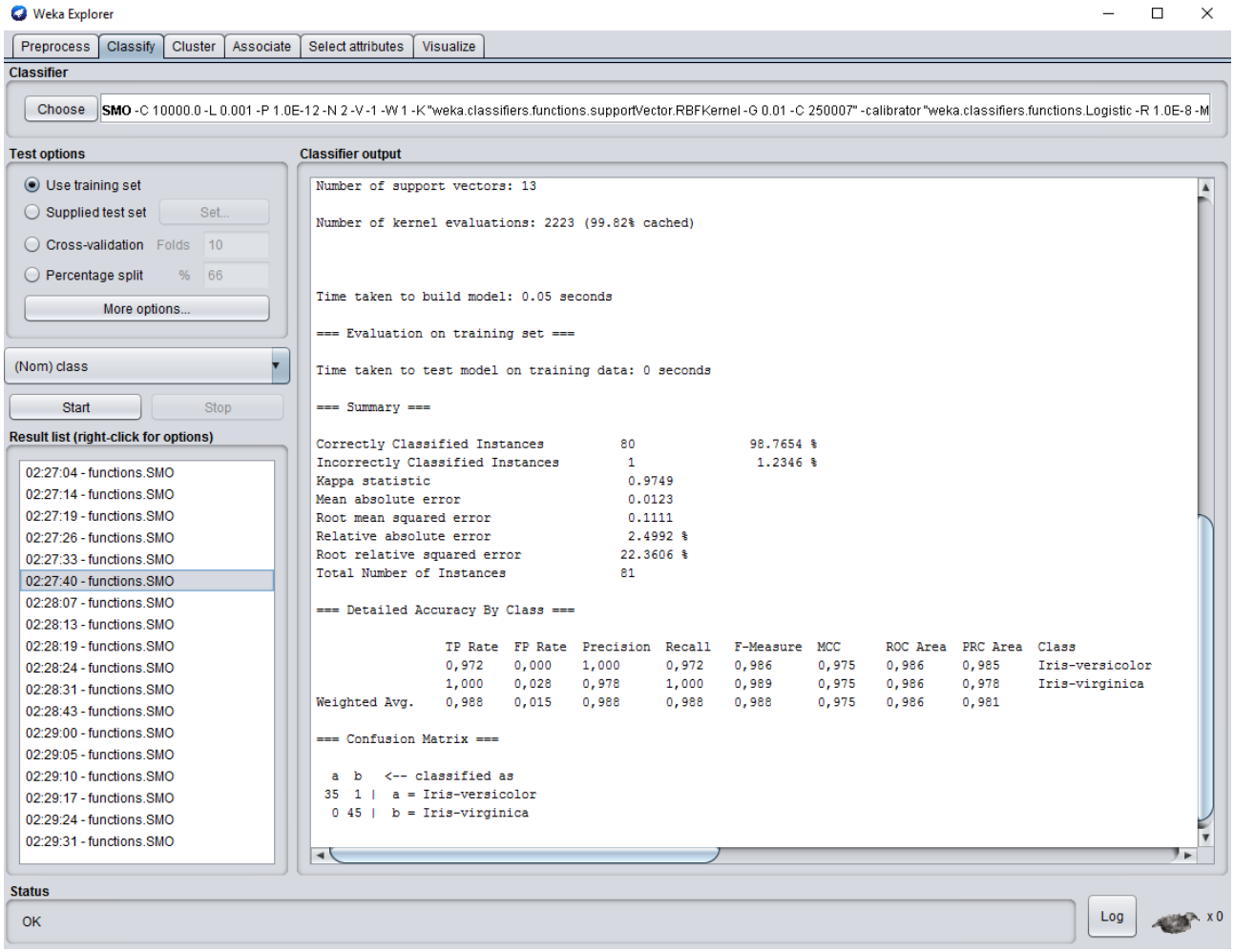


Figura 10.- Captura de weka del modelo con un error de clasificación para C=10000 y gamma=1

```
=== Run information ===

Scheme:       weka.classifiers.functions.SMO -C 10000.0 -L 0.001 -P 1.0E-12 -N 2 -V -1 -W 1 -K
"weka.classifiers.functions.supportVector.RBFKernel -G 1.0 -C 250007" -calibrator
"weka.classifiers.functions.Logistic -R 1.0E-8 -M -1 -num-decimal-places 4"
Relation:     iris-weka.filters.unsupervised.attribute.Remove-R1,2-
weka.filters.unsupervised.instance.RemoveWithValues-S0.0-Clast-Lfirst-H-
weka.filters.unsupervised.instance.RemoveDuplicates
Instances:    81
Attributes:   3
              petallength
              petalwidth
              class
Test mode:    evaluate on training data

=== Classifier model (full training set) ===

SMO

Kernel used:
  RBF kernel: K(x,y) = e^-(1.0* <x-y,x-y>^2)

Classifier for classes: Iris-versicolor, Iris-virginica

BinarySMO

-    4444.3086 * <4.9 1.5 > * X]
-    7482.4141 * <5.1 1.6 > * X]
+   10000      * <5 1.5 > * X]
-   10000      * <4.8 1.8 > * X]
+     170.5244 * <5.5 1.8 > * X]
-      6.4789 * <3 1.1 > * X]
+    1414.4552 * <4.8 1.8 > * X]
-    361.5373 * <4.7 1.2 > * X]
+    1775.9135 * <4.5 1.7 > * X]
+     94.955  * <5.8 1.6 > * X]
+    9350.0469 * <4.9 1.8 > * X]
-    393.056  * <4.8 1.4 > * X]
-    118.1002 * <3.9 1.4 > * X]
+     24.878

Number of support vectors: 13

Number of kernel evaluations: 2223 (99.82% cached)
```

```
Time taken to build model: 0.05 seconds

=== Evaluation on training set ===

Time taken to test model on training data: 0 seconds

=== Summary ===

Correctly Classified Instances          80               98.7654 %
Incorrectly Classified Instances         1                1.2346 %
Kappa statistic                          0.9749
Mean absolute error                      0.0123
Root mean squared error                  0.1111
Relative absolute error                  2.4992 %
Root relative squared error             22.3606 %
Total Number of Instances               81

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure  MCC      ROC Area  PRC Area  Class
               0,972    0,000    1,000      0,972    0,986      0,975    0,986     0,985     Iris-versicolor
               1,000    0,028    0,978      1,000    0,989      0,975    0,986     0,978     Iris-virginica
Weighted Avg.  0,988    0,015    0,988      0,988    0,988      0,975    0,986     0,981

=== Confusion Matrix ===

 a  b   <-- classified as
35  1 |  a = Iris-versicolor
 0 45 |  b = Iris-virginica
```

Datos 5.- Listado para configuración C=10000 y gamma 1 para SMO con gaussiana y Use Training Set

b) Numero de vectores utilizados por el modelo

```
BinarySMO

-    4444.3086 * <4.9 1.5 > * X]
-    7482.4141 * <5.1 1.6 > * X]
+   10000      * <5 1.5 > * X]
-   10000      * <4.8 1.8 > * X]
+     170.5244 * <5.5 1.8 > * X]
-      6.4789 * <3 1.1 > * X]
+    1414.4552 * <4.8 1.8 > * X]
-     361.5373 * <4.7 1.2 > * X]
+    1775.9135 * <4.5 1.7 > * X]
+      94.955  * <5.8 1.6 > * X]
+    9350.0469 * <4.9 1.8 > * X]
-     393.056  * <4.8 1.4 > * X]
```

```
-    118.1002 * <3.9 1.4 > * X]
+     24.878

Number of support vectors: 13
```

Datos 6.- Vectores soporte para C=10000 y gamma=1

c) Indique cuales de ellos son vectores soporte frontera y cuáles vectores ligados.

```
+   10000      * <5 1.5 > * X] LIGADO
-   10000      * <4.8 1.8 > * X] LIGADO
```

Datos 7.- Vectores soporte ligados para C=10000 y gamma=1

```
-   4444.3086 * <4.9 1.5 > * X] FRONTERA
-   7482.4141 * <5.1 1.6 > * X] FRONTERA
+    170.5244 * <5.5 1.8 > * X] FRONTERA
-     6.4789 * <3 1.1 > * X] FRONTERA
+   1414.4552 * <4.8 1.8 > * X] FRONTERA
-    361.5373 * <4.7 1.2 > * X] FRONTERA
+   1775.9135 * <4.5 1.7 > * X] FRONTERA
+     94.955  * <5.8 1.6 > * X] FRONTERA
+   9350.0469 * <4.9 1.8 > * X] FRONTERA
-    393.056  * <4.8 1.4 > * X] FRONTERA
-    118.1002 * <3.9 1.4 > * X] FRONTERA
```

Datos 8.- Vectores soporte frontera para C=10000 y gamma=1

iii) Seleccionando la opción Visualize Classifier Error, accesible desde la ventana emergente obtenida como resultado de hacer clic con el botón derecho del ratón sobre el modelo elegido, muestre la captura de pantalla del plot obtenido e indique si existe alguna relación entre los ejemplos mal clasificados por el modelos y los denominados vectores soporte ligados (Nota: No olvide actuar sobre la barra de control "Jitter" de la ventana de plot para poder visualizar instancias que estén muy próximas entre sí.

Si enfrentamos petallength frente a petalwitdh el plot de errores de clasificación nos ofrece:

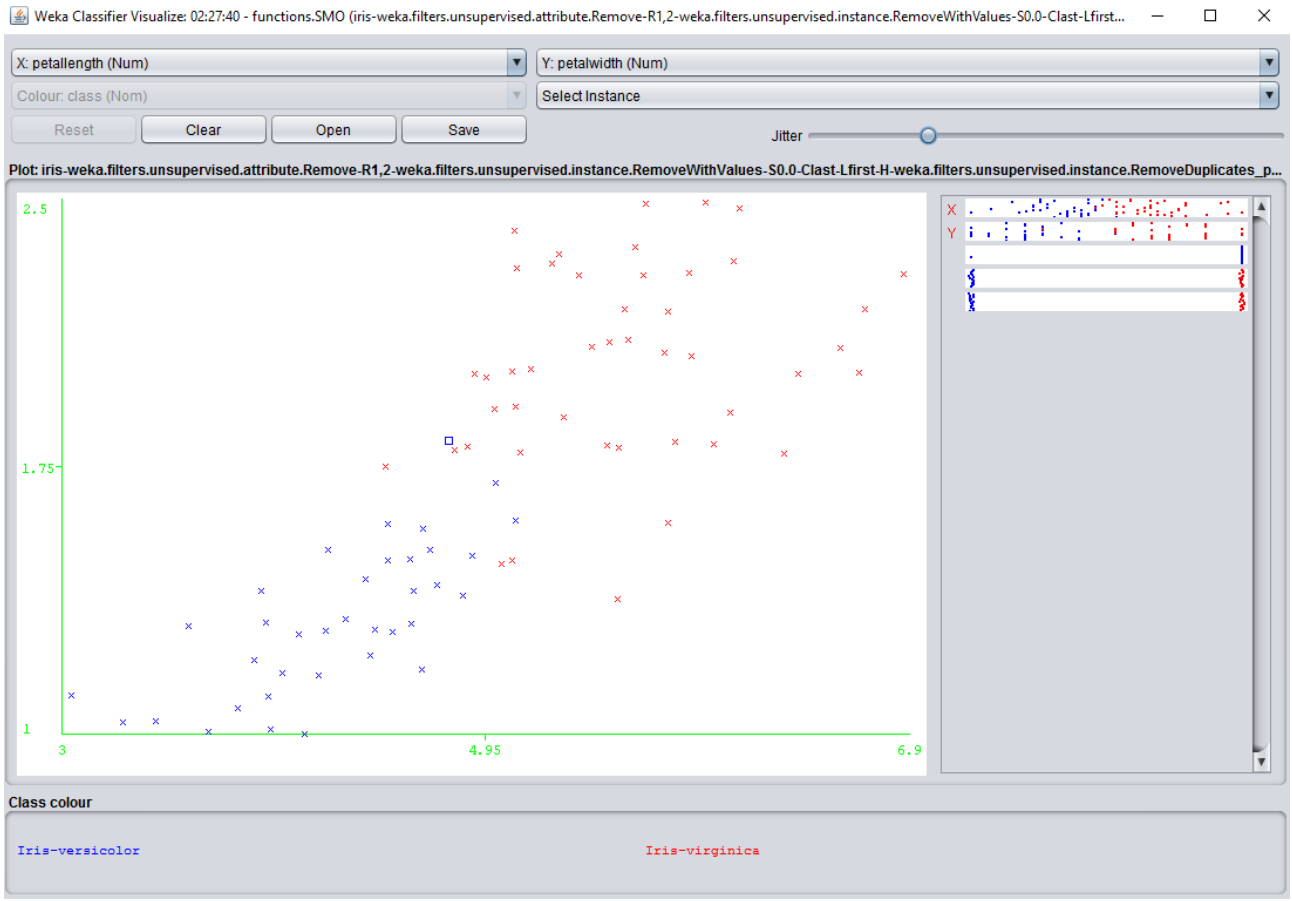


Figura 12.- Plot petallegth frente a petalwidth, se ha introducido un poco de ruido, con C=10000 y gamma=1

iv)Opcional: A partir de la captura de pantalla del punto anterior y cualquier editor de gráficos, identifique los vectores soporte frontera de cada clase y, a partir de la posición de dichos vectores, trace de forma aproximada la frontera de separación de las dos clases. Adjunte el gráfico resultante.

He utilizado paint para modificar el dibujo, los vectores soporte se han adjuntado en la imagen y los ligados se han marcado, el resto son frontera. Cada punto lo identifica el propio weka pinchando en cada punto, la escala de graduación que ofrece weka para X e Y no es demasiado eficiente y sólo orientativa.

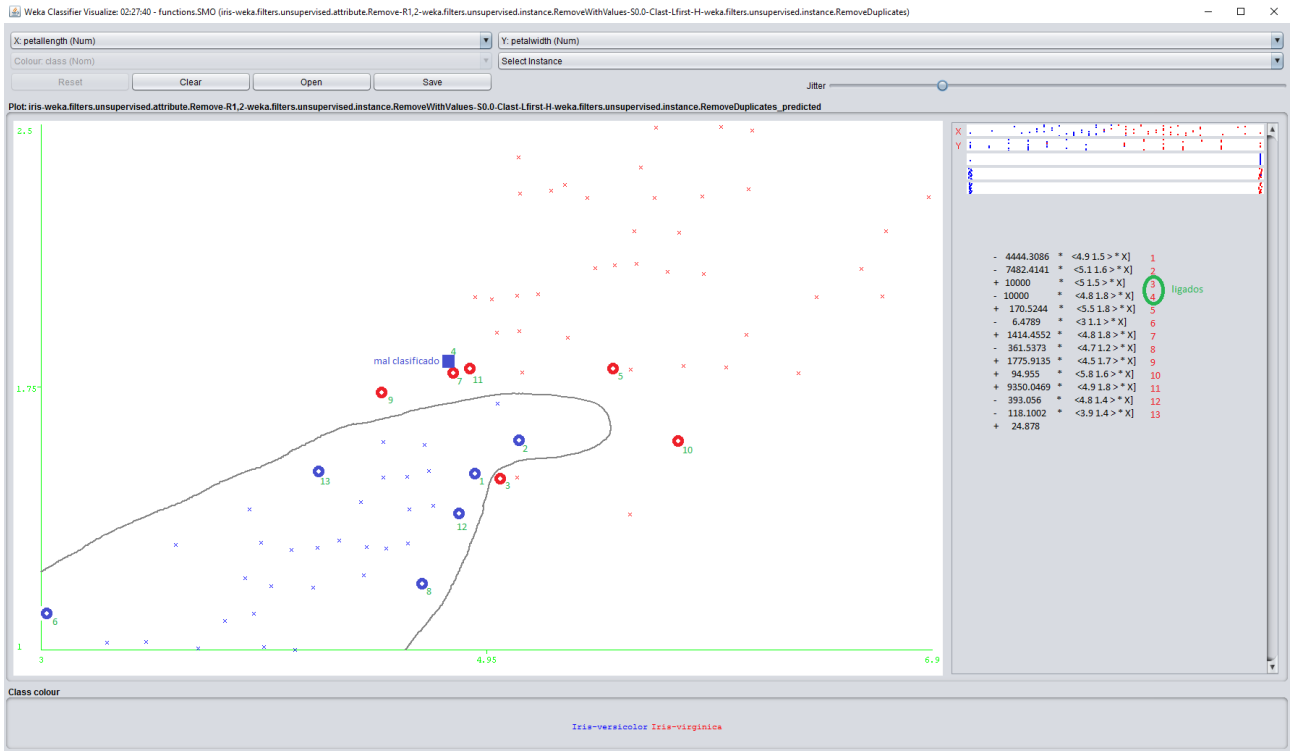


Figura 13.- Plot anterior con los vectores soporte frontera identificados.

6. Repetir el paso 5 con las siguientes diferencias:

Selecciones ahora la opción Cross-validation (Folds=10) como método de evaluación de cada modelo.

Documentación a entregar apartado 5.6:

i) Construya una tabla en la que se muestre el número de instancias mal clasificadas obtenido por el modelo al ejecutar el algoritmo con cada par de combinaciones de valores (C, gamma).

C	gamma	Mal clasificadas
10^{-1}	1	4
10^{-1}	10^{-1}	8
10^{-1}	10^{-2}	36
10^{0}	1	5
10^{0}	10^{-1}	6
10^{0}	10^{-2}	6
10^{1}	1	7

10^1	10^{-1}	5
10^1	10^{-2}	6
10^2	1	8
10^2	10^{-1}	7
10^2	10^{-2}	5
10^3	1	10
10^3	10^{-1}	8
10^3	10^{-2}	6
10^4	1	9
10^4	10^{-1}	8
10^4	10^{-2}	8

Tabla 5.- Instancias mal clasificadas según configuración (C,gamma) para SMO con gaussiana y Cross-Validation.

ii)De los resultados obtenidos, elija el modelo de menor error y adjunto a la memoria la siguiente información:

a) Modelo obtenido por Weka.

El menor número de clasificación obtenidos es 4, para la configuración C=0.1 y gamma=1. Obteniendo el siguiente modelo:

```
=== Run information ===

Scheme:       weka.classifiers.functions.SMO -C 0.1 -L 0.001 -P 1.0E-12 -N 2 -V -1 -W 1 -K
"weka.classifiers.functions.supportVector.RBFKernel -G 1.0 -C 250007" -calibrator
"weka.classifiers.functions.Logistic -R 1.0E-8 -M -1 -num-decimal-places 4"
Relation:     iris-weka.filters.unsupervised.attribute.Remove-R1,2-
weka.filters.unsupervised.instance.RemoveWithValues-S0.0-Clast-Lfirst-H-
weka.filters.unsupervised.instance.RemoveDuplicates
Instances:    81
Attributes:   3
              petallength
              petalwidth
              class
Test mode:    10-fold cross-validation
=== Classifier model (full training set) ===

SMO
Kernel used:
  RBF kernel: K(x,y) = e^-(1.0* <x-y,x-y>^2)
```

Classifier for classes: Iris-versicolor, Iris-virginica

BinarySMO

- 0.1 * <4.5 1.6 > * X]
- 0.1 * <4.7 1.4 > * X]
- 0.1 * <4.5 1.3 > * X]
- 0.1 * <4.9 1.5 > * X]
+ 0.1 * <5.1 2.3 > * X]
- 0.1 * <5.1 1.6 > * X]
- 0.1 * <4.5 1.5 > * X]
- 0.1 * <4.7 1.6 > * X]
+ 0.1 * <5 1.5 > * X]
+ 0.1 * <6.6 2.1 > * X]
+ 0.1 * <6.9 2.3 > * X]
+ 0.0181 * <6 2.5 > * X]
+ 0.1 * <5.3 1.9 > * X]
- 0.1 * <4.2 1.5 > * X]
- 0.1 * <4.6 1.4 > * X]
+ 0.1 * <5.6 1.4 > * X]
- 0.1 * <4.6 1.3 > * X]
- 0.1 * <4.8 1.8 > * X]
- 0.1 * <3.8 1.1 > * X]
+ 0.1 * <5.5 1.8 > * X]
+ 0.1 * <6.3 1.8 > * X]
+ 0.1 * <6.1 2.5 > * X]
- 0.1 * <3 1.1 > * X]
+ 0.1 * <5.1 2.4 > * X]
+ 0.1 * <5.2 2.3 > * X]
- 0.1 * <4.4 1.4 > * X]
- 0.1 * <3.5 1 > * X]
+ 0.1 * <5.1 1.8 > * X]
+ 0.1 * <5 1.9 > * X]
+ 0.1 * <4.8 1.8 > * X]
+ 0.1 * <5.1 2 > * X]
+ 0.1 * <4.9 2 > * X]
- 0.1 * <4.6 1.5 > * X]
+ 0.1 * <5 2 > * X]
- 0.1 * <4.7 1.2 > * X]
- 0.0972 * <4.4 1.2 > * X]
+ 0.1 * <6.7 2.2 > * X]
- 0.1 * <3.7 1 > * X]
+ 0.1 * <5.1 1.9 > * X]
+ 0.1 * <5.1 1.5 > * X]
- 0.1 * <3.6 1.3 > * X]
- 0.1 * <5 1.7 > * X]

```
+       0.0071 * <6.4 2 > * X]
+       0.1    * <4.5 1.7 > * X]
-       0.1    * <4.7 1.5 > * X]
+       0.1    * <5.8 1.6 > * X]
-       0.1    * <3.3 1 > * X]
+       0.1    * <6.7 2 > * X]
+       0.1    * <4.9 1.8 > * X]
-       0.1    * <4.8 1.4 > * X]
-       0.028  * <3.9 1.4 > * X]
-       0.1    * <4.4 1.3 > * X]
+       0.1    * <5.2 2 > * X]
+       0.1675

Number of support vectors: 53

Number of kernel evaluations: 3139 (80.662% cached)



Time taken to build model: 0.01 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances          77               95.0617 %
Incorrectly Classified Instances         4                4.9383 %
Kappa statistic                          0.8994
Mean absolute error                      0.0494
Root mean squared error                  0.2222
Relative absolute error                  9.9819 %
Root relative squared error             44.6515 %
Total Number of Instances               81

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure  MCC      ROC Area  PRC Area  Class
               0,917    0,022    0,971      0,917    0,943      0,901    0,947     0,927     Iris-versicolor
               0,978    0,083    0,936      0,978    0,957      0,901    0,947     0,928     Iris-virginica
Weighted Avg.  0,951    0,056    0,951      0,951    0,950      0,901    0,947     0,927

=== Confusion Matrix ===

  a  b   <-- classified as
 33  3 |  a = Iris-versicolor
  1 44 |  b = Iris-virginica
```

Datos 9.- Listado para configuración C=0.1 y gamma 1 para SMO con gaussiana y Cross Validation

b) Número de vectores soporte utilizados por el modelo.

El resultado de los vectores soporte es:

BinarySMO

- 0.1 * <4.5 1.6 > * X]
- 0.1 * <4.7 1.4 > * X]
- 0.1 * <4.5 1.3 > * X]
- 0.1 * <4.9 1.5 > * X]
+ 0.1 * <5.1 2.3 > * X]
- 0.1 * <5.1 1.6 > * X]
- 0.1 * <4.5 1.5 > * X]
- 0.1 * <4.7 1.6 > * X]
+ 0.1 * <5 1.5 > * X]
+ 0.1 * <6.6 2.1 > * X]
+ 0.1 * <6.9 2.3 > * X]
+ 0.0181 * <6 2.5 > * X]
+ 0.1 * <5.3 1.9 > * X]
- 0.1 * <4.2 1.5 > * X]
- 0.1 * <4.6 1.4 > * X]
+ 0.1 * <5.6 1.4 > * X]
- 0.1 * <4.6 1.3 > * X]
- 0.1 * <4.8 1.8 > * X]
- 0.1 * <3.8 1.1 > * X]
+ 0.1 * <5.5 1.8 > * X]
+ 0.1 * <6.3 1.8 > * X]
+ 0.1 * <6.1 2.5 > * X]
- 0.1 * <3 1.1 > * X]
+ 0.1 * <5.1 2.4 > * X]
+ 0.1 * <5.2 2.3 > * X]
- 0.1 * <4.4 1.4 > * X]
- 0.1 * <3.5 1 > * X]
+ 0.1 * <5.1 1.8 > * X]
+ 0.1 * <5 1.9 > * X]
+ 0.1 * <4.8 1.8 > * X]
+ 0.1 * <5.1 2 > * X]
+ 0.1 * <4.9 2 > * X]
- 0.1 * <4.6 1.5 > * X]
+ 0.1 * <5 2 > * X]
- 0.1 * <4.7 1.2 > * X]
- 0.0972 * <4.4 1.2 > * X]
+ 0.1 * <6.7 2.2 > * X]
- 0.1 * <3.7 1 > * X]
+ 0.1 * <5.1 1.9 > * X]
+ 0.1 * <5.1 1.5 > * X]
- 0.1 * <3.6 1.3 > * X]

```
-      0.1    * <5 1.7 > * X]
+      0.0071 * <6.4 2 > * X]
+      0.1    * <4.5 1.7 > * X]
-      0.1    * <4.7 1.5 > * X]
+      0.1    * <5.8 1.6 > * X]
-      0.1    * <3.3 1 > * X]
+      0.1    * <6.7 2 > * X]
+      0.1    * <4.9 1.8 > * X]
-      0.1    * <4.8 1.4 > * X]
-      0.028  * <3.9 1.4 > * X]
-      0.1    * <4.4 1.3 > * X]
+      0.1    * <5.2 2 > * X]
+      0.1675

Number of support vectors: 53
```

Datos 10.- Vectores soporte para C=0.1 y gamma=1

c) Indique cuáles de ellos son vectores soporte frontera y cuáles vectores soporte ligados.

```
-      0.1    * <4.5 1.6 > * X] LIGADO
-      0.1    * <4.7 1.4 > * X] LIGADO
-      0.1    * <4.5 1.3 > * X] LIGADO
-      0.1    * <4.9 1.5 > * X] LIGADO
+      0.1    * <5.1 2.3 > * X] LIGADO
-      0.1    * <5.1 1.6 > * X] LIGADO
-      0.1    * <4.5 1.5 > * X] LIGADO
-      0.1    * <4.7 1.6 > * X] LIGADO
+      0.1    * <5 1.5 > * X] LIGADO
+      0.1    * <6.6 2.1 > * X] LIGADO
+      0.1    * <6.9 2.3 > * X]LIGADO
+      0.0181 * <6 2.5 > * X] FRONTERA
+      0.1    * <5.3 1.9 > * X] LIGADO
-      0.1    * <4.2 1.5 > * X] LIGADO
-      0.1    * <4.6 1.4 > * X] LIGADO
+      0.1    * <5.6 1.4 > * X] LIGADO
-      0.1    * <4.6 1.3 > * X] LIGADO
-      0.1    * <4.8 1.8 > * X] LIGADO
-      0.1    * <3.8 1.1 > * X] LIGADO
+      0.1    * <5.5 1.8 > * X] LIGADO
+      0.1    * <6.3 1.8 > * X] LIGADO
+      0.1    * <6.1 2.5 > * X] LIGADO
-      0.1    * <3 1.1 > * X] LIGADO
+      0.1    * <5.1 2.4 > * X] LIGADO
+      0.1    * <5.2 2.3 > * X] LIGADO
-      0.1    * <4.4 1.4 > * X] LIGADO
-      0.1    * <3.5 1 > * X] LIGADO
+      0.1    * <5.1 1.8 > * X] LIGADO
```

```
+       0.1    * <5 1.9 > * X] LIGADO
+       0.1    * <4.8 1.8 > * X] LIGADO
+       0.1    * <5.1 2 > * X] LIGADO
+       0.1    * <4.9 2 > * X] LIGADO
-       0.1    * <4.6 1.5 > * X] LIGADO
+       0.1    * <5 2 > * X] LIGADO
-       0.1    * <4.7 1.2 > * X] LIGADO
-       0.0972 * <4.4 1.2 > * X] FRONTERA
+       0.1    * <6.7 2.2 > * X] LIGADO
-       0.1    * <3.7 1 > * X] LIGADO
+       0.1    * <5.1 1.9 > * X] LIGADO
+       0.1    * <5.1 1.5 > * X] LIGADO
-       0.1    * <3.6 1.3 > * X] LIGADO
-       0.1    * <5 1.7 > * X] LIGADO
+       0.0071 * <6.4 2 > * X] FRONTERA
+       0.1    * <4.5 1.7 > * X] LIGADO
-       0.1    * <4.7 1.5 > * X] LIGADO
+       0.1    * <5.8 1.6 > * X] LIGADO
-       0.1    * <3.3 1 > * X] LIGADO
+       0.1    * <6.7 2 > * X] LIGADO
+       0.1    * <4.9 1.8 > * X] LIGADO
-       0.1    * <4.8 1.4 > * X] LIGADO
-       0.028  * <3.9 1.4 > * X] FRONTERA
-       0.1    * <4.4 1.3 > * X] LIGADO
+       0.1    * <5.2 2 > * X] LIGADO
+       0.1675
```

Datos 11.- Vectores soporte frontera y ligados para C=0.1 y gamma=1

iii) Por qué se dice que el porcentaje de acierto ahora obtenido es más fiable que el obtenido para el mismo modelo en el paso5, donde se usó la opción Use Training set como método de evaluación.

En el experimento con Use Training Set, se ofrecen siempre los resultados más optimistas ya que se entrena con todo el conjunto para las pruebas, realmente lo que nos interesa es tener un cierto grado de incertidumbre para la clasificación con un alto grado de acierto, como sabemos Cross Validation, toma subgrupos de entrenamiento y el resto pruebas en cada iteración, lo que permite introducir esa incertidumbre en la clasificaciones con alto grado de acierto.

7. Tal y como se menciona en la introducción, las SVM también se pueden aplicar al caso multiclase. La aproximación más usada para abordar el problema multiclase mediante SVMs es transformar el problema original en diferentes problemas de clasificación binaria. Existen varias estrategias para realizar dicha transformación. Por ejemplo, una de ellas consiste en construir diferentes clasificadores binarios que distingan entre cada clase y el resto de clases. Otra opción, por ejemplo, es construir tantos clasificadores binarios como pares de clase se puedan formar. El algoritmo SMO implementado en Weka utiliza esta última opción. Aquí, como problema multiclase, vamos a considerar el conjunto de ejemplos "iris.arff" (problema de tres clases y atributos de 4 dimensiones). A continuación realice las siguientes acciones:

- *Desde el panel Preprocess, cargue los datos del fichero "iris.arff".*
- *Desde el panel Clasify selecciones el algoritmo SMO y configure los siguientes parámetros (el resto se mantendrán al valor por defecto): Seleccione FilterType igual a "No normalizarion-standardization" y kernel igual a "RBFKernel".*
- *Selecciones ahora la opción Cross-Validation (Folds=10) como método de evaluación del modelo.*
- *Considere los siguientes conjuntos de C={10^{-1}, 1,10^{1},10^{2},10^{3},10^{4}} y gamma = {1,10^{-1},10^{-2}}.*

Documentación a entregar apartado 5.7:

i) Construya una tabla en la que se muestre el número de instancias mal clasificadas obtenido por el método al ejecutar el algoritmo con cada par de combinaciones de valores (C, gamma).

C	gamma	Mal clasificadas
10^{-1}	1	6
10^{-1}	10^{-1}	8
10^{-1}	10^{-2}	17
10^{0}	1	6
10^{0}	10^{-1}	6
10^{0}	10^{-2}	10
10^{1}	1	7
10^{1}	10^{-1}	5
10^{1}	10^{-2}	4
10^{2}	1	11
10^{2}	10^{-1}	7
10^{2}	10^{-2}	5
10^{3}	1	11
10^{3}	10^{-1}	8
10^{3}	10^{-2}	6
10^{4}	1	11
10^{4}	10^{-1}	12
10^{4}	10^{-2}	6

Tabla 6.- Instancias mal clasificadas con (C,gamma) para SMO con gaussiana y Cross-Validation, con 4D.

ii) De los resultados obtenidos, elija el modelo de menor error y adjunte a la memoria la siguiente información

a) Modelo obtenido por Weka para cada par de clases, indicando el número de vectores soporte utilizado en cada caso.

Los mejores resultados han sido con 4 errores de clasificación para C=10 y gamma = 0.01 con el siguiente modelo:

```
=== Run information ===

Scheme:       weka.classifiers.functions.SMO -C 10.0 -L 0.001 -P 1.0E-12 -N 2 -V -1 -W 1 -K
"weka.classifiers.functions.supportVector.RBFKernel -G 0.01 -C 250007" -calibrator
"weka.classifiers.functions.Logistic -R 1.0E-8 -M -1 -num-decimal-places 4"
Relation:     iris
Instances:    150
Attributes:   5
              sepallength
              sepalwidth
              petallength
              petalwidth
              class
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

SMO

Kernel used:
  RBF kernel: K(x,y) = e^-(0.01* <x-y,x-y>^2)

Classifier for classes: Iris-setosa, Iris-versicolor

BinarySMO

-      10      * <5.1 3.3 1.7 0.5 > * X]
+      10      * <5.1 2.5 3 1.1 > * X]
-      7.0634 * <4.5 2.3 1.3 0.3 > * X]
+      10      * <4.9 2.4 3.3 1 > * X]
-      1.4819 * <5.1 3.8 1.9 0.4 > * X]
+      8.5453 * <5 2.3 3.3 1 > * X]
-      10      * <4.8 3.4 1.9 0.2 > * X]
-      0.1282

Number of support vectors: 7
```

Number of kernel evaluations: 1276 (49.184% cached)

Classifier for classes: Iris-setosa, Iris-virginica

BinarySMO

```
   10      * <4.9 2.5 4.5 1.7 > * X]
+      0.2155 * <6 3 4.8 1.8 > * X]
-      7.7783 * <4.8 3.4 1.9 0.2 > * X]
-      2.4372 * <5.1 3.3 1.7 0.5 > * X]
-      0.0445
```

Number of support vectors: 4

Number of kernel evaluations: 1147 (54.138% cached)

Classifier for classes: Iris-versicolor, Iris-virginica

BinarySMO

```
   10      * <6.3 2.5 5 1.9 > * X]
-     10      * <5.4 3 4.5 1.5 > * X]
-     10      * <6.7 3.1 4.7 1.5 > * X]
+     10      * <6.9 3.1 5.1 2.3 > * X]
-     10      * <6 3.4 4.5 1.6 > * X]
+     10      * <5.9 3 5.1 1.8 > * X]
-     10      * <6.2 2.2 4.5 1.5 > * X]
+     10      * <5.8 2.7 5.1 1.9 > * X]
+     10      * <6.5 3 5.2 2 > * X]
-     10      * <5.7 2.8 4.5 1.3 > * X]
-     10      * <6 2.7 5.1 1.6 > * X]
-     10      * <6.7 3 5 1.7 > * X]
-      7.8208 * <7 3.2 4.7 1.4 > * X]
-     10      * <6.1 2.9 4.7 1.4 > * X]
+     10      * <6.4 2.7 5.3 1.9 > * X]
-     10      * <6.9 3.1 4.9 1.5 > * X]
+     10      * <6 2.2 5 1.5 > * X]
+     10      * <5.7 2.5 5 2 > * X]
+      6.8239 * <6.4 3.1 5.5 1.8 > * X]
+     10      * <6.5 3.2 5.1 2 > * X]
+     10      * <6.1 3 4.9 1.8 > * X]
-     10      * <6.8 2.8 4.8 1.4 > * X]
-     10      * <6.1 3 4.6 1.4 > * X]
+     10      * <5.8 2.7 5.1 1.9 > * X]
+     10      * <6.2 2.8 4.8 1.8 > * X]
-     10      * <5.9 3.2 4.8 1.8 > * X]
```

```
+      10      * <6.1 2.6 5.6 1.4 > * X]
+      10      * <6.3 2.7 4.9 1.8 > * X]
+      10      * <5.6 2.8 4.9 2 > * X]
-      10      * <6.3 2.5 4.9 1.5 > * X]
-      9.003   * <6.3 2.3 4.4 1.3 > * X]
+      10      * <6.3 2.8 5.1 1.5 > * X]
+      10      * <4.9 2.5 4.5 1.7 > * X]
+      10      * <7.2 3 5.8 1.6 > * X]
-      10      * <6.3 3.3 4.7 1.6 > * X]
+      10      * <6 3 4.8 1.8 > * X]
-      10      * <5.6 3 4.5 1.5 > * X]
-      10      * <6.1 2.8 4.7 1.2 > * X]
-      10      * <6 2.9 4.5 1.5 > * X]
-      10      * <6.5 2.8 4.6 1.5 > * X]
+      0.2039

Number of support vectors: 40

Number of kernel evaluations: 3744 (80.708% cached)



Time taken to build model: 0.01 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances         146               97.3333 %
Incorrectly Classified Instances         4               2.6667 %
Kappa statistic                          0.96
Mean absolute error                      0.2281
Root mean squared error                  0.2828
Relative absolute error                 51.3333 %
Root relative squared error             60     %
Total Number of Instances              150

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure  MCC      ROC Area  PRC Area  Class
               1,000    0,000    1,000      1,000    1,000      1,000    1,000     1,000     Iris-setosa
               0,960    0,020    0,960      0,960    0,960      0,940    0,970     0,935     Iris-versicolor
               0,960    0,020    0,960      0,960    0,960      0,940    0,980     0,942     Iris-virginica
Weighted Avg.  0,973    0,013    0,973      0,973    0,973      0,960    0,983     0,959

=== Confusion Matrix ===

  a  b  c   <-- classified as
```

```
50  0  0 |  a = Iris-setosa
 0 48  2 |  b = Iris-versicolor
 0  2 48 |  c = Iris-virginica
```

Datos 12.- Listado para configuración C=0.1 y gamma 1 para SMO con gaussiana y Cross Validation

El número de vectores para cada par es:

- Iris-setosa, Iris-versicolor 7 vectores.
- Iris-setosa, Iris-virginica 4 vectores.
- Iris-versicolor, Iris-virginica 40 vectores.

iii) Indique cómo se utiliza la información de cada uno de los tres clasificadores para asignar el valor de clase a una instancia dada. Es decir, dado que existen 3 hiperplanos D11(x), D12(x),D23(x) indique, por ejemplo, a qué clase se asignaría un ejemplo una vez que éste es evaluado por cada una de las tres funciones de decisión indicadas.

Se realiza una evaluación una clase frente a una clase. La evaluación utiliza cada Hiperplano para realizar cada clasificación, se toma los ejemplos que caen dentro del margen produciendo que vuelvan a ser evaluados ofreciendo los mejores resultados y con un coste computacional mayor, nos permite dar la mejor opción teniendo en cuenta la distancia en la que queda con respecto a cada hiperplano.

8. Acceder a la página web de LIBSVM. Desde dicha web, acceder al applet que muestra una interfaz gráfica, la cual permite usar LIBSVM para resolver tareas de clasificación y regresión. En el caso de problemas de clasificación, como el caso que nos ocupa, se puede entrenar una SVM para conjuntos de ejemplos 2D (proporcionados manualmente por el usuario), pudiendo manejar hasta tres clases. Aunque la interfaz permite al usuario trabajar con diferentes tipos de SVMs (C-SVC, nu-SVC, one-class SVM, epsilon SVR, nu-SVR), sólo nos centraremos en las C-SVC (clasificadores vectores soporte con parámetro C), es decir, el mismo tipo que hemos estado usando hasta ahora. A continuación realice las siguiente acciones:

- ***En primer lugar se va a crear un problema de clasificación de dos clases. Para ello, en la interfaz gráfica y cliqueando con el ratón, introduzca el conjunto de ejemplos de entrenamiento indicados en la siguiente secuencia: i) Introduzca de forma aproximada los ejemplos indicados en la figura 5(a), es decir, se trata de simular una frontera de separación en "diente de sierra"; ii) Continúe añadiendo los ejemplos indicados en la figura 5b hasta cerrar el perímetro de cada clase;iii) Finalmente, acabe rellenando, con ejemplos situados al azar, el interior de las dos clases delimitadas anterior de las dos clases delimitadas anteriormente, tal y como se indica en la figura 5c.***

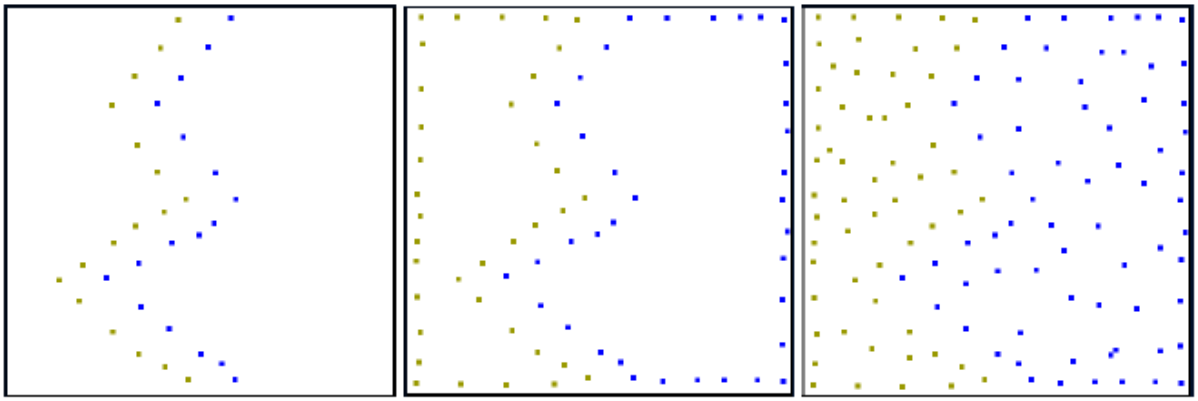


Figura 5: Construcción en tres pasos de un problema de clasificación de dos clases no separables linealmente (frontera de separación en "diente de sierra"). Visualice directamente la versión electrónica de este documento para ver correctamente los colores asociados a cada uno de los ejemplos

- **Ejecute el algoritmo C-SVC con kernel gaussiano para un barrido adecuado de diferentes valores del parámetro C (opción -c) y del parámetro gamma (opción -g). Observe que C-SVC y el kernel gaussiano son, respectivamente, el tipo de algoritmo y el tipo de kernel seleccionados por defecto. Por tanto, no es necesario especificarlos en la caja de comandos. De otro lado, tenga en cuenta que mientras no pulse el botón Clear, podrá hacer tantas ejecuciones diferentes (botón Run) como estime necesarias sin que se modifique el conjunto de ejemplos. En cambio, el uso del botón Clear provocará el borrado de los datos.**

Documentación a entregar apartado 5.8a:

i) Del barrido realizado, indique los valores de C y gamma para los que se produjo la mejor frontera de separación.

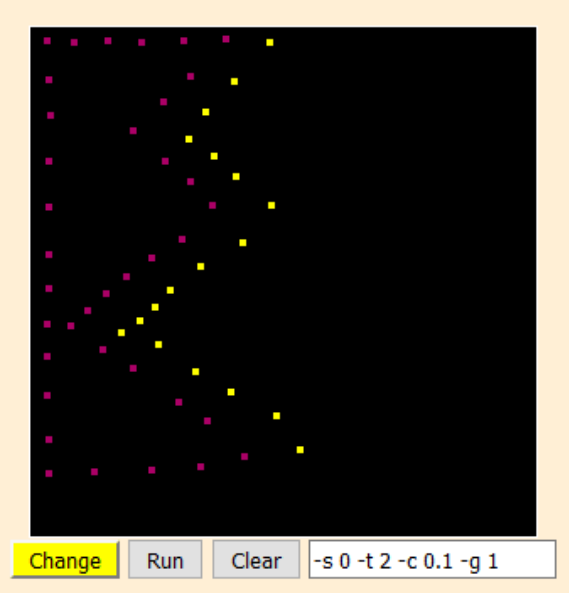


Figura 14.- Creación del ejemplo propuesto en el applet con 5a)

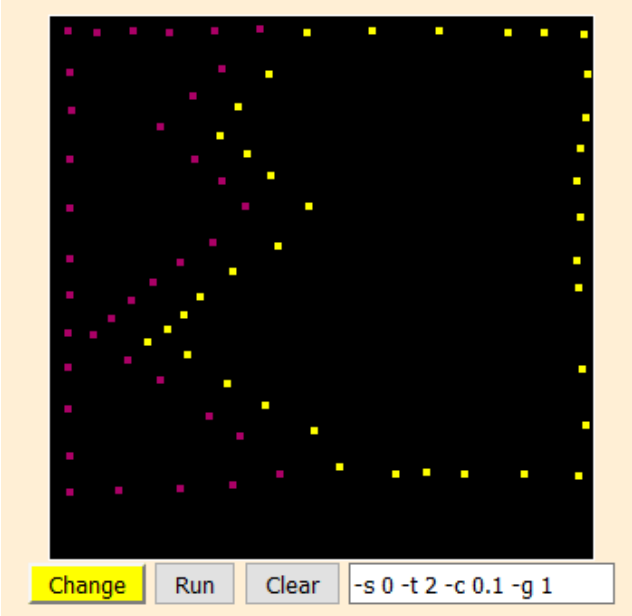


Figura 15.- Creación del ejemplo propuesto en el applet con 5b)

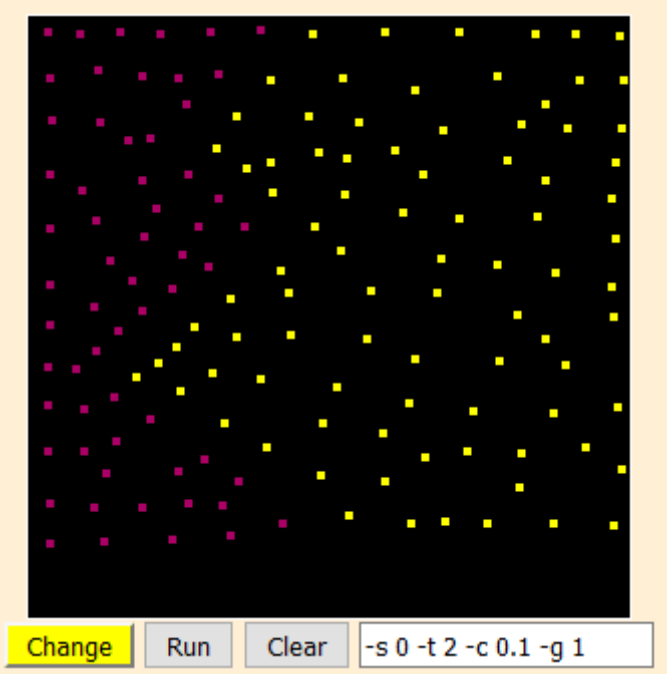
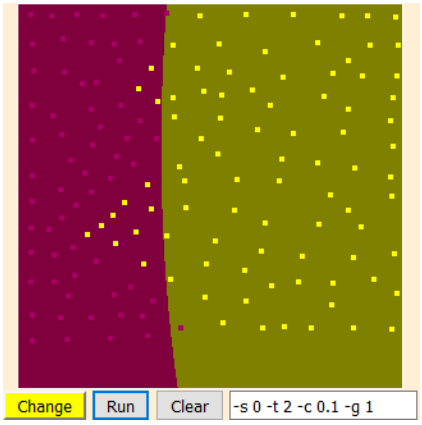
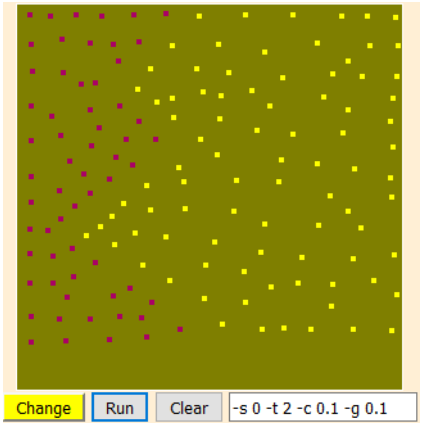
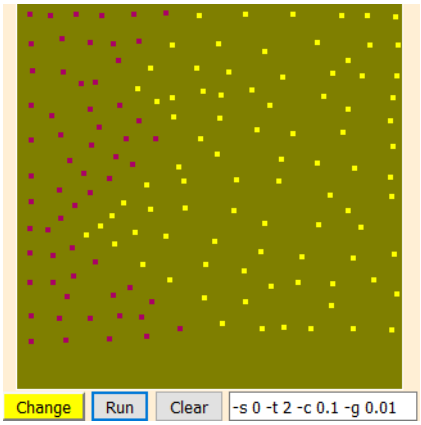
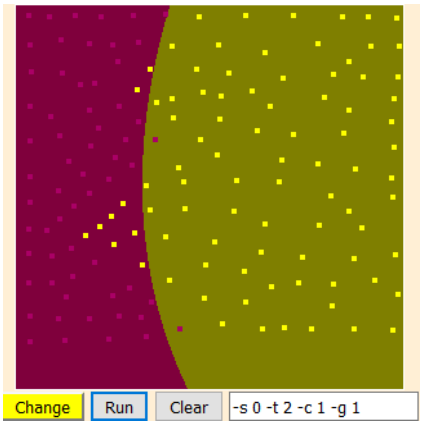


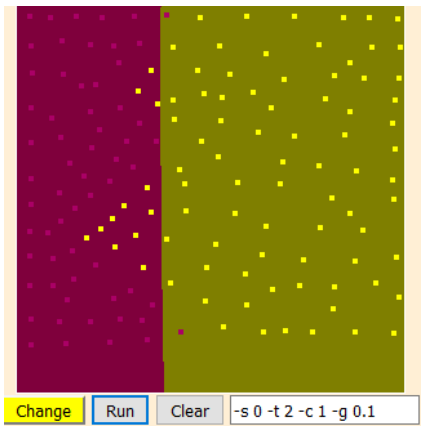
Figura 16.- Creación del ejemplo propuesto en el applet con 5c)

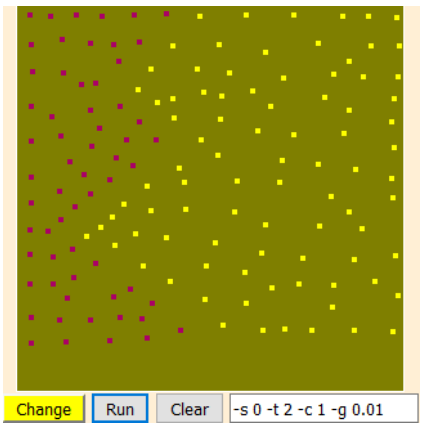

Change
Run
Clear
-s 0 -t 2 -c 0.1 -g 1

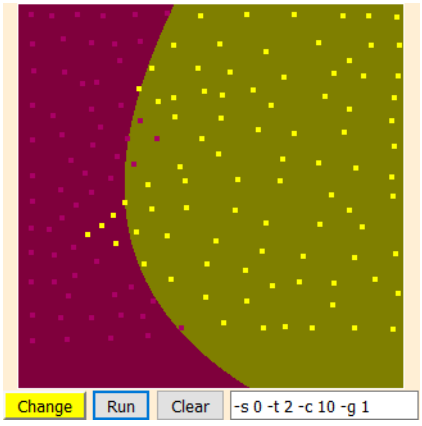

Change
Run
Clear
-s 0 -t 2 -c 0.1 -g 0.1

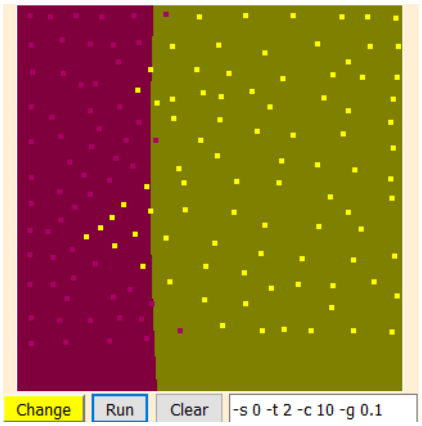

Change
Run
Clear
-s 0 -t 2 -c 0.1 -g 0.01

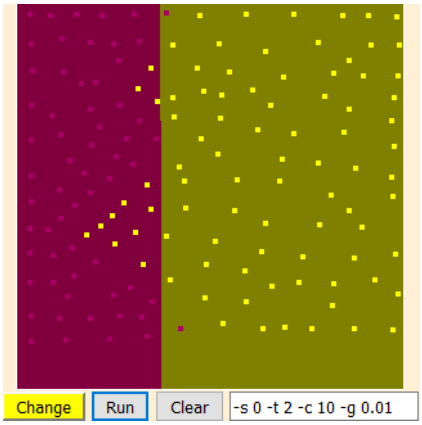

Change
Run
Clear
-s 0 -t 2 -c 1 -g 1

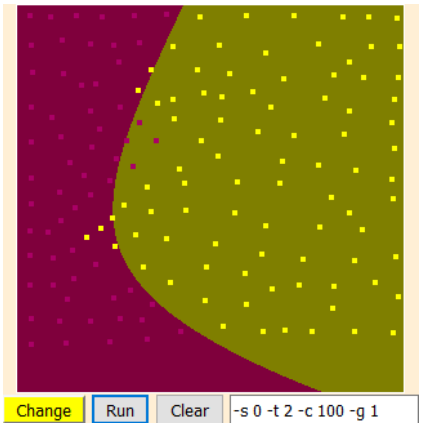

Change
Run
Clear
-s 0 -t 2 -c 1 -g 0.1

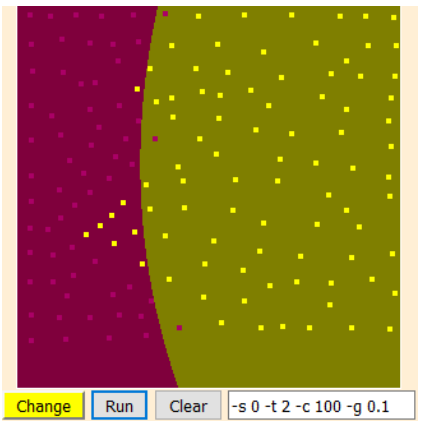

Change
Run
Clear
-s 0 -t 2 -c 1 -g 0.01

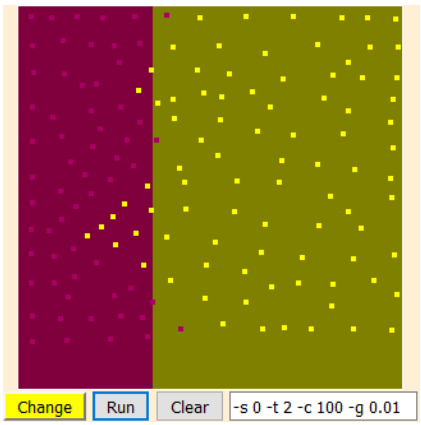

Change
Run
Clear
-s 0 -t 2 -c 10 -g 1


Change
Run
Clear
-s 0 -t 2 -c 10 -g 0.1


Change
Run
Clear
-s 0 -t 2 -c 10 -g 0.01


Change
Run
Clear
-s 0 -t 2 -c 100 -g 1


Change
Run
Clear
-s 0 -t 2 -c 100 -g 0.1


Change
Run
Clear
-s 0 -t 2 -c 100 -g 0.01

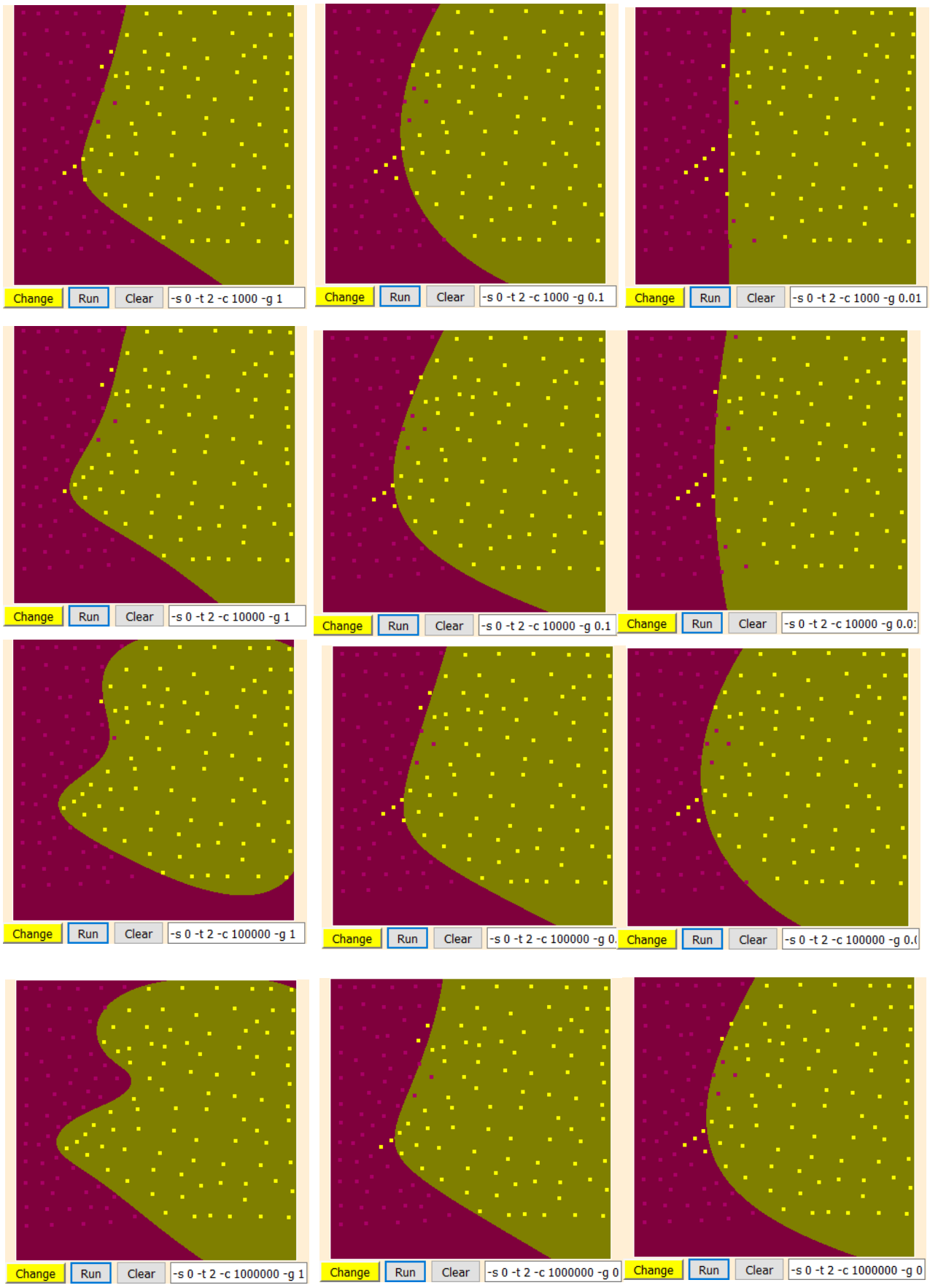


Figura 17.- Evolución para (C,gamma) con LIBSVM en un barrido con función kernel gaussiana.

He creído interesante interesante incluir la evolución del barrido para valores de C={10^{-1}, 1,10^{1},10^{2},10^{3},10^{4},10^{5},10^{6}} y gamma = {1,10^{-1},10^{-2}}.

Podemos observar que para gamma=1 y con C=10000, obtenemos unos resultados buenos, sin embargo haciendo más grande el margen en el espacio obtenemos para C=1000000 unos resultados de clasificación íntegros, a costa de un coste computacional mayor. Aún así pensamos que nuestro modelo a elegir seria C=10000, con apenas 5 errores de clasificación visibles.

ii) Incluya una captura de pantalla que muestre dicha frontera.

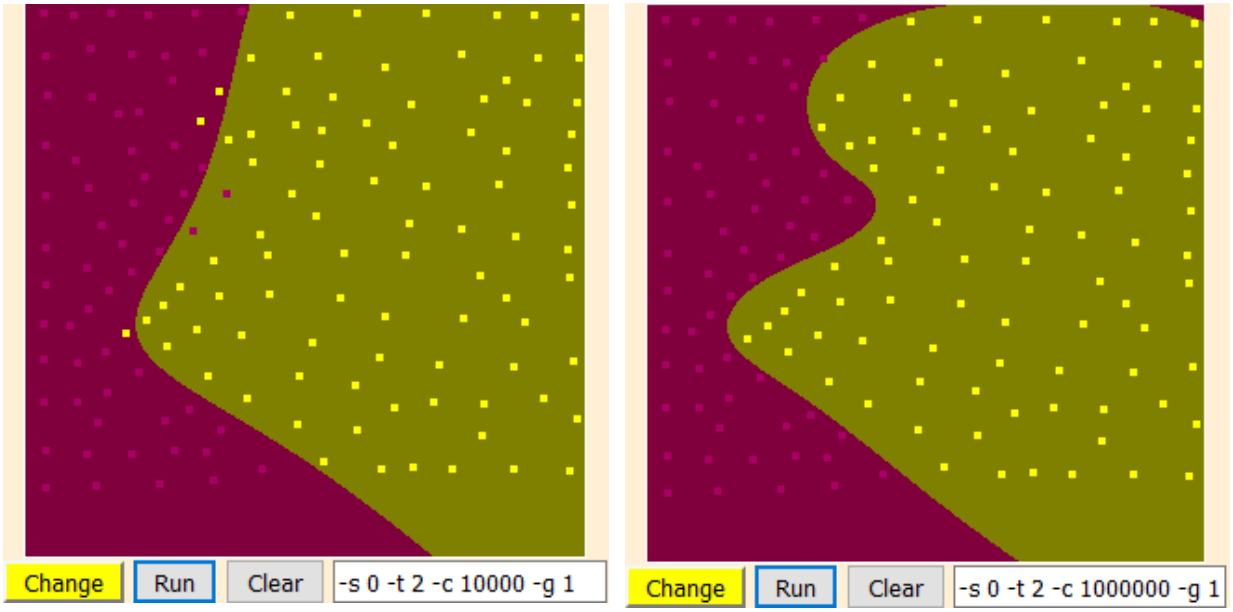


Figura 18.- Mejores resultados obtenidos con función kernel gaussiana.

iii) Investigue si es posible encontrar una nueva frontera de separación usando un kernel polinómico. En este caso, deberá indicar expresamente este tipo de kernel (opción -t 1) y jugar con sus parámetros, gamma (opción -b) y grado del polinomio (opción -d), además de con el parámetro C (opción -c).

En la actividad, solamente hemos hablado de la función kernel gaussiana, y hemos estudiado su valor gamma (que es inversamente proporcional a la desviación típica) y del valor de C como margen.

De la teoría podemos extraer que podremos utilizar cualquier función kernel, lo que deberíamos estudiar es la influencia que tiene los parámetros, que podemos ver en la siguiente figura, pues bien para un C=10000 y un gamma=1 y la ayuda del parámetro coef0 de LIBSVM del parámetro -r = 1, nos permite desplazar ese modelo y obtener para este caso una curva similar a la solicitada. Parece que ese parámetro influye desplazando la curva de la frontera, acercándose a la campana. Hemos tomado el valor del grado del polinomio con -d 3, ya que es su valor por defecto, y se supone que ofrece un compromiso, entre el coste computacional y los resultados. Si aumentamos -r=2 obtenemos una clasificación ideal.

Ejemplos de funciones kernel

Se presentan aquí algunos ejemplos de funciones kernel:

- Kernel lineal:

$$K_L(\boldsymbol{x}, \boldsymbol{x}') =< \boldsymbol{x}, \boldsymbol{x}' > \tag{52}$$

- kernel polinómico de grado-p:

$$K_P(\boldsymbol{x}, \boldsymbol{x}') = \left[\gamma < \boldsymbol{x}, \boldsymbol{x}' > +\tau\right]^p, \ \gamma > 0 \tag{53}$$

- kernel gaussiano:

$$K_G(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\gamma \left\|\boldsymbol{x} - \boldsymbol{x}'\right\|^2\right) \equiv \exp\left(-\gamma < (\boldsymbol{x} - \boldsymbol{x}'), (\boldsymbol{x} - \boldsymbol{x}') >\right), \ \gamma > 0 \tag{54}$$

- kernel sigmoidal:

$$K_S(\boldsymbol{x}, \boldsymbol{x}') = \tanh(\gamma < \boldsymbol{x}, \boldsymbol{x}' > +\tau) \tag{55}$$

A los parámetros γ, τ y p se les denomina parámetros del kernel.

v) Incluya una captura de pantalla que muestre dicha frontera.

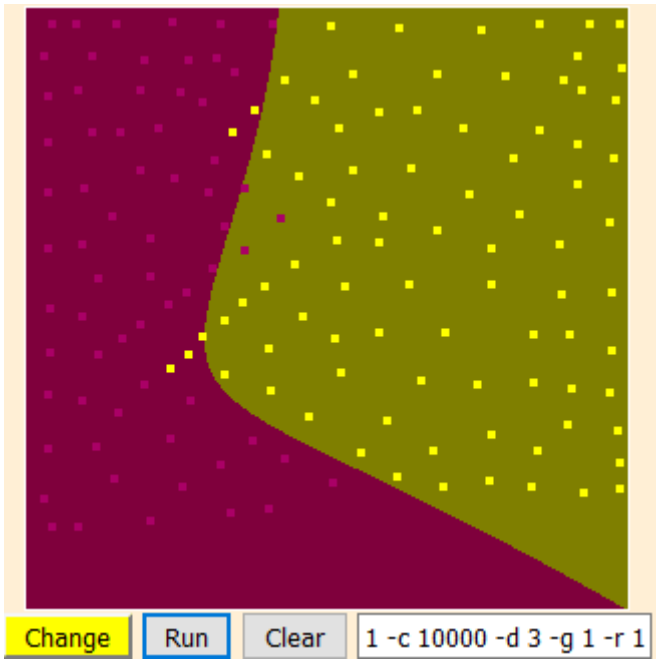


Figura 19.- Mejores resultados obtenidos con función kernel polinomio, hemos añadido el parámetro -r.

- *Finalmente, se creará un nuevo problema de clasificación, pero esta vez de tres clases. Para ello, introduzca una configuración de ejemplos parecida a la indicada en la figura. La idea es crear tres clases dispuestas en capas concéntricas.*

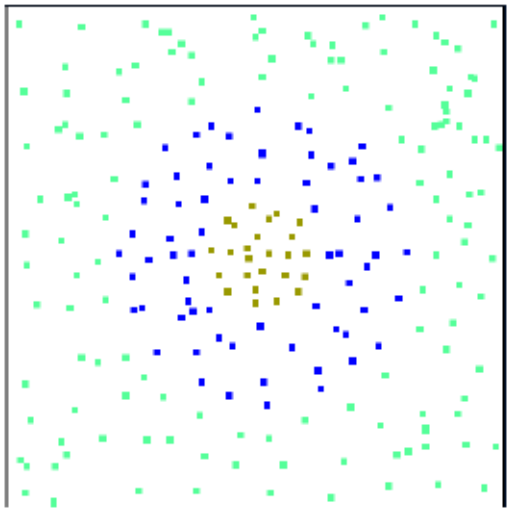


Figura 6: Problema de clasificación de tres clases no separables linealmente (fronteras de separación concéntricas). Visualice directamente la versión electrónica de este documento para ver correctamente los colores asociados a cada uno de los ejemplos

- *Elija el kernel más adecuado para este problema. Investigue y sintonice adecuadamente los valores de los parámetros del kernel elegido y del parámetro C.*

Documentación a entregar apartado 5.8b:

i)Justifique el tipo de kernel elegido.

Se ha elegido el kernel gaussiano que ha sido estudiado en toda la práctica, por sus resultados evidentemente aceptables en los modelos propuestos de los experimentos realizados.

ii)Indique los valores de parámetros para los que se produjo la mejor frontera de separación.

En el barrido realizado como nos han pedido en los enunciados C={10^1, 1, 10^1, 10^2, 10^3, 10^4} y gamma = {1,10^{-1},10^{-2}}, enseguida hemos visto que el mejor resultado ha sido para C=10000 y gamma=1.

iii) Incluya una captura de pantalla que muestre dicha frontera.

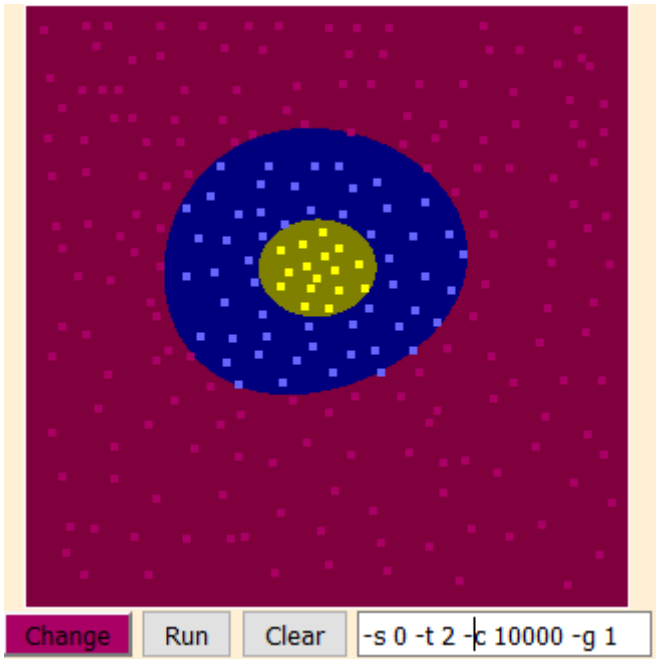


Figura 20.- Resultado obtenido función kernel gaussiana, con un barrido simple paramos en C=10000 y gamma=1.

6. Conclusiones.

En la documentación a entregar, incluya también las conclusiones derivadas de los resultados prácticos obtenidos con la realización de esta práctica y que están relacionados con los objetivos planteados inicialmente.

Primero de todo comentar que en el esquema seguido en la elaboración de la práctica he optado por reescribir TODO el enunciado de la práctica, ya que me parece inseparable el enunciado, dada su descripción teórica y descripción de ejercicios, con las respuestas de la realización de la actividad, ya que ambas partes juntas, hacen si esta bien realizada la actividad, un documento que se puede seguir para estudiar y preparar este Modelo de SVM. Espero que el corregidor no tenga en cuenta la amplitud de la práctica por incluir los enunciados.

Para exponer conclusiones, tomaremos como punto de partida, el mismo que el utilizado en el documento que el profesor nos ha presentado para el estudio de las SVMs [Carmona 2016]. Partimos de una tarea de clasificación (binaria o multiclase) en la cual se nos pueden presentar distintos escenarios que sean separables linealmente o no linealmente separables. Este tarea podemos plantearla como un problema matemático, estadístico y geométrico, en la cual entra a trabajar la Teoría de Optimización, la cual pretende minimizar una función la cual podemos identificar como problema primal (original) y que pretendemos resolver transformando en una solución dual. Esta solución dual nos permitirá utilizando multiplicadores de Lagrange formar una solución más sencilla que la original, ampliando el espacio de trabajo, **espacio de características**,

que es geométricamente mayor que el espacio de trabajo original, y que nos va a permitir separar las instancias que queremos clasificar mediante Hiperplanos, los cuales vas a llevarnos una expresión que nos permitirá clasificarlas. Estos hiperplanos de separación tomaran como datos los vectores soporte de las instancias que mejor optimicen la solución de clasificación.

Para realizar la transformación del problema primal y poder abordarlo en un espacio mayor, nos apoyaremos en la funciones kernel, ya sean lineal, polinómica, gaussiana, sigmoidea, o que podamos realizar o encontrar. Estas funciones nos permitirán dar la solución al problema dual a partir de los vectores soportes que serán instancias del problema primal, y que al ser soluciones del problema dual, serán también solución de dicho problema primal. En dichas funciones kernel existen una serie de parámetros, gamma, ro o tao, que nos permiten probar distintas soluciones que se adapten a nuestra necesidad por encontrar una frontera en la separación de las clases.

Los hiperplanos, por su parte, nos permitirán maximizar el margen para clasificar las instancias, ese margen, que en este documento hemos simbolizado con la letra C, nos permite ampliar el margen para la clasificación aunque a veces algunas instancias podemos clasificar mal. No existe un algoritmo para encontrar un C correcto. Por lo cual debemos encontrar un valor para cada tarea que nos permita realizar clasificaciones aceptables.

Pudimos ver en la base teórica como podíamos hacer uso de los multiplicadores de Lagrange, nombrados con la letra alpha, y que si formaban parte de la solución los representábamos con alpha*. Si los alpha* tenían un valor entre 0 y C, nos permitían definir los vectores soporte frontera, que también daban como solución de la ecuación del hiperplano en valor absoluto un valor próximo a 1. Por otro lado si alpha* coincidía con el valor de C estábamos hablando de vectores soporte ligados o bien que la distancia D(x) fuera distinta de 1.

En la actividad hemos podido probar como Weka nos permite seleccionar atributos y clases a partir de un conjunto de instancias, para preparar modelos linealmente separables y no linealmente separables, en los cuales hemos abordado la búsqueda de soluciones de forma similar, hemos realizado barridos con $C=\{10^{-1}, 1, 10^{1}, 10^{2}, 10^{3}, 10^{4}\}$ y gamma = $\{1,10^{-1},10^{-2}\}$, eligiendo los experimentos que menores errores obteníamos, lo cual viene a confirmar que no podemos conocer a priori cuales son esos valores, pero si que debemos elegir los que minimicen el error de clasificación.

Finalmente pudimos observar con LIBSVM, como influyen gráficamente la elección de los valores de C y g que mejor resultados nos daban, pudiendo ver que cuanto mayor era el margen C, menos errores obteníamos, pudimos ver que el kernel gaussiano, nos ofrece unos resultados muy buenos, aunque no es el único que podemos utilizar, para conseguir definir las fronteras y poder clasificar las instancias.

Bibliografía

- Tutorial sobre Máquinas de Vectores Soporte [Carmona, 2016]
- Apredizaje Automático, texto base [Borrajo et al., 2006]
- Foro y diapositivas de las asignatura de Aprendizaje Automático
- Programación cuadrática http://psmmetodosdeoptimi.wixsite.com/optimizacion/programacion-cuadratica

- Artículo de fundamentos de optimización http://nuyoo.utm.mx/~jjf/rna/guia_foe.pdf
- Explicaciones sobre multiplicadores de Lagrange http://profe-alexz.blogspot.com.es/2011/12/multiplicadores-de-lagrange-problemas.html
- Trabajo sobre Support Vector Machine http://www.cs.columbia.edu/~kathy/cs4701/documents/jason_svm_tutorial.pdf
- Documento sobre Funciones Kernel https://www.dspace.espol.edu.ec/bitstream/123456789/25019/1/CONSTRUCCION_DE_KERNELS_Y_FUNCIONES_DE_DENSIDAD_DE_PROBABILIDAD.pdf
- Libro de la asignatura de Estadística de la UNED

Referencias del enunciado de la actividad

Referencias

[Borrajo et al., 2006] Borrajo, D., González, J., & Isasi, P. (2006). *Aprendizaje Automático*. Madrid (España): Sanz y Torres.

[Boser et al., 1992] Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92 (pp. 144–152). New York, NY, USA: ACM.

[Carmona, 2016] Carmona, E. J. (2016). *Tutorial sobre Máquinas de Vectores Soporte*. Technical report, Dpto. Inteligencia Artificial, Universidad Nacional de Educación a Distancia (UNED), Madrid (Spain).

[Cortes & Vapnik, 1995] Cortes, C. & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3), 273–297.

[Hsu et al., 2016] Hsu, C.-W., Chang, C.-C., & Lin., C.-J. (2016). *A practical guide to support vector classification*. Technical report, Dept. of Computer Science, National Taiwan University, Taipei (Taiwan), (`http://www.csie.ntu.edu.tw/~cjlin/papers/quadworkset.pdf`).

[Platt, 1998] Platt, J. (1998). Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods - Support Vector Learning*: MIT Press.

* * *