Aprendizaje Automático
Máquinas de Vectores Soporte en Tareas de Clasificación

Enrique J. Carmona Suárez
ecarmona@dia.uned

Primera versión: 29/11/2016 Última versión: 29/11/2016

Dpto. Inteligencia Artificial, ETS de Ingeniería Informática, Universidad Nacional de Educación a Distancia (UNED), C/Juan del Rosal, 16, 28040-Madrid (Spain)

Índice

1. Introducción

Las máquinas de vectores soporte (SVM, del inglés *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 SVM 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).
   c) Conjunto de ejemplos 4D no linealmente separables (multiclase).
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[1] para visualizar las fronteras de decisión asociadas al modelo 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* (ver figura 1). El contenido de este archivo es una colección de instancias que representan tres variedades de lirio. Se compone de cinco atributos. Los cuatros primeros corresponden 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.

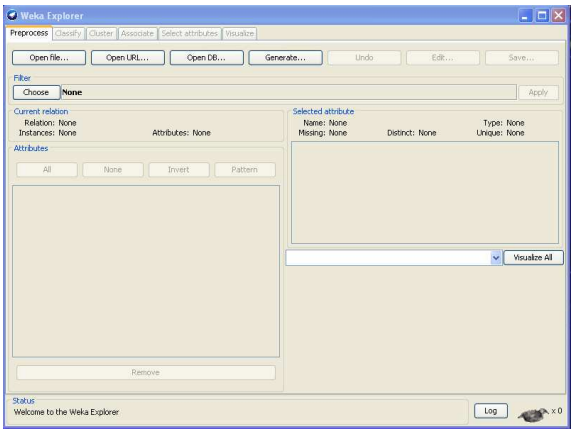
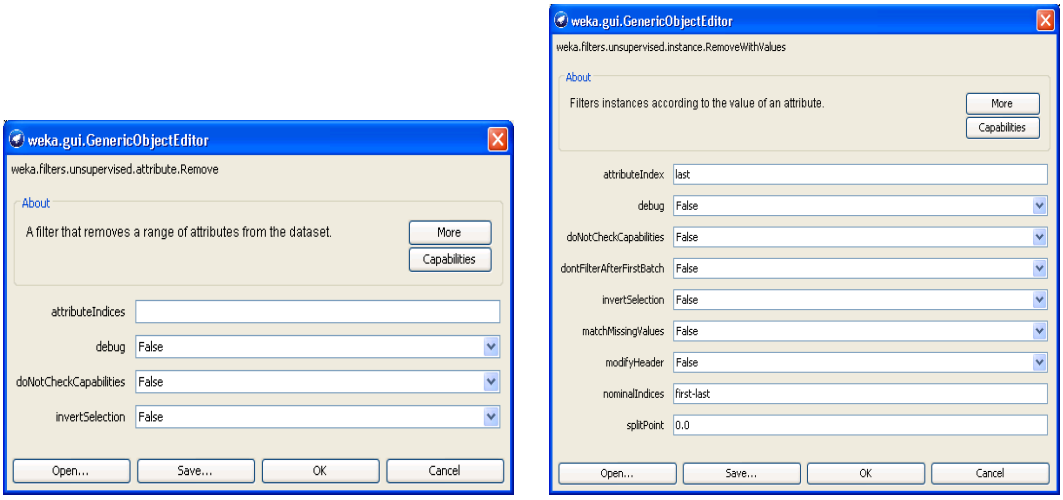


Figura 1: Panel de Preprocesado del Explorer de Weka.

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 ello, vamos a transformar el fichero "iris.arff" original (se asume 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 siguiente pasos:

[1] LIBSVM es un software integrado para máquinas de vectores soporte

- Elimine los atributos indicados y los ejemplos pertenecientes a la clase indicada. Para ello, haga uso, respectivamente, de los filtros Weka denominados *Remove* (*filters* > *unsupervised* > *attribute*) y *RemoveWithValues* (*filters* > *unsupervised* > *instance*), cada uno de ellos accesible desde el botón Choose del panel *Preprocess* (ver fig. 2). 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.



(a) (b)

Figura 2: Filtros Weka: (a) Remove y (b) RemoveWithValues

- Dado que la forma de fabricar el conjunto de entrenamiento para obtener una colección de ejemplos 2D pertenecientes a dos clases ha sido un tanto artificiosa, ocurrirá que, tras aplicar los dos filtros anteriores, es muy posible que aparezcan algunos ejemplos repetidos, tal y como así ocurre. Para eliminar todas 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:** Mostrar el plot obtenido en el punto anterior.

3. Asumiendo que el conjunto de datos cargado corresponde al fichero "iris_2D_2clases_Ls.arff", desde la ventana *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* [Platt, 1998], uno de los algoritmos utilizados para el entrenamiento de clasificadores de vectores soporte. Los parámetros de configuración de este algoritmo se muestran en la fig. 3. 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 el 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, normalmente 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 o 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 del 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:

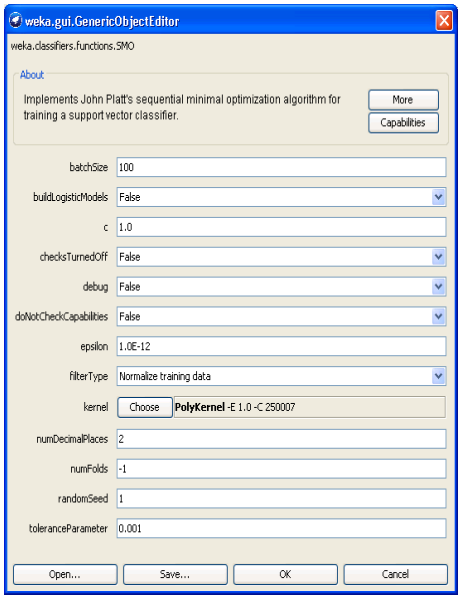


Figura 3: Conjunto de parámetros que permiten configurar el algoritmo SMO de Weka.

- Dado que se quiere primar la interpretabilidad del modelo aprendido, el parámetro *FilterType* se pondrá al valor "*No normalization-standardization*". Como kernel, se seleccionará "*RBFKernel*", también llamado *kernel gaussiano*. A su vez, esto implicará tener que realizar la configuración del mismo (ver fig. 4). En esta caso, su parámetro más importante corresponde al valor de *gamma*, dado que matemáticamente este tipo de kernel se define como:

$$K_G(\boldsymbol{x}, \boldsymbol{x'}) = e^{-\gamma\|\boldsymbol{x}-\boldsymbol{x}'\|^2} \equiv e^{-\gamma<(\boldsymbol{x}-\boldsymbol{x}'),(\boldsymbol{x}-\boldsymbol{x}')>},\ \gamma > 0 \tag{1}$$

  Se elegirá $gamma = 10^{-2}$ y el resto de parámetros se dejarán a su valor por defecto. El objeto de 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, seleccione 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**: (i) Muestre en una tabla cómo varía el porcentaje de instancias bien clasificadas y el número de vectores soporte, $|V_S|$, utilizados por el modelo, en función de C; (ii) Justifique por qué, 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; (iii) Los modelos o funciones de decisión obtenidos por

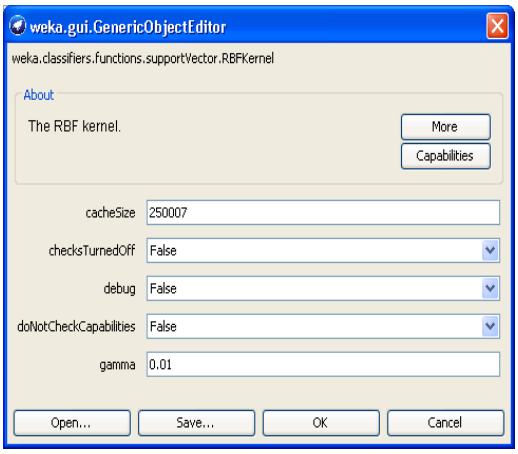


Figura 4: Parámetros de configuración del kernel gaussiano en Weka

Weka, se pueden expresar de forma genérica como:

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

donde el símbolo $\pm$ delante de cada sumando denota que el signo puede ser positivo o negativo. Indique qué representa: (a) el valor del signo, (b) los coeficientes $a_i, \quad i = 0, ..., n$ y (c) las tuplas $(x_{1j}, x_{2j}), \quad j = 1, ..., n$; (iv) Atendiendo a las respuestas dadas en (iii), exprese la función de decisión obtenida por Weka para el caso $C = 10^2$ de acuerdo al siguiente esquema:

$$D(\boldsymbol{x}) = \alpha_0^* + \alpha_1^* y_1 K(\boldsymbol{x}, \boldsymbol{x}_1) + ... + \alpha_n^* y_n K(\boldsymbol{x}, \boldsymbol{x}_n) \tag{3}$$

donde, $K(\boldsymbol{x}, \boldsymbol{x}_i)$, en este caso, tiene que expresarse de acuerdo a la expresión (1); (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 "Iris-versicolor" para el caso $C = 10^2$; (vi) Adjunte el modelo obtenido por Weka para el caso $C = 10$. A partir de dicho modelo 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; (vii) Mediante la evaluación del modelo es posible diferenciar también entre vectores soporte frontera y vectores soporte ligados. Así, si $|D(\boldsymbol{x})| = 1$, entonces $\boldsymbol{x}$ es un vector soporte frontera, siendo $|.|$ el operador valor absoluto[2]. En otro caso, $\boldsymbol{x}$ 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; (viii) **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 $D(\boldsymbol{x}) = 0$ y operar la expresión resultante convenientemente.

- 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**: (i) Construya dos tablas como la indicada en el caso anterior, una para los resultados obtenidos con $gamma = 1$ y, la otra, para $gamma = 10^{-4}$; (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 $|V_S|$ 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.

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:

[2] Puede ocurrir que, por problemas de redondeo, al evaluar $|D(\boldsymbol{x})|$, siendo $\boldsymbol{x}$ un vector soporte frontera, no se obtenga un valor exacto igual a 1, sino un valor aproximadamente igual a 1.

- Cargue el fichero "iris.arff" desde el panel *Preprocess*.
- Elimine los atributos "sepalwidht" y "sepallength".
- Elimine los ejemplos de la clase "Iris_setosa".
- Elimine ejemplos duplicados.
- Utilice el botón Edit del 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:** Mostrar el plot obtenido en el punto anterior.

5. Asumiendo que el conjunto de datos cargado corresponde al fichero "iris_2D_2clases_NLs.arff" y que el algoritmo SMO está seleccionado, realice las siguientes acciones:

   - Seleccione el valor de *FilterType* a "*No normalization-standardization*" (se quiere seguir manteniendo la interpretabilidad del modelo).
   - Seleccione el kernel "*RBFKernel*".
   - Seleccione la opción *Use Training set* como método de evaluación de los modelos aprendidos.
   - Tal y como se vio en el paso 3, 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 aquél asociado a la combinación que produzca el número de instancias mal clasificadas más bajo. Considere los siguientes conjuntos[3] de $C = \{10^{-1}, 1, 10^1, 10^2, 10^3, 10^4\}$ y $gamma = \{1, 10^{-1}, 10^{-2}\}$.
   - **Documentación a entregar:** (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 $(C, gamma)$; (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, (b) Número de vectores soporte utilizados por el modelo, (c) Indique cuáles de ellos son vectores soporte frontera y cuáles vectores soporte ligados; (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 modelo 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); (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.

6. Repetir el paso 5 con las siguientes diferencias:

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

[3] El rango y muestreo de valores elegidos aquí para C y *gamma* es tentativo, es decir, no tiene porqué garantizar el hallazgo del modelo óptimo. Ambos dependerán normalmente del problema a resolver (conjunto de ejemplos).

- **Documentación a entregar:** (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 $(C, gamma)$; (ii) De los resultados obtenidos, elija el modelo de menor error y adjunte a la memoria la siguiente información: (a) Modelo obtenido por Weka, (b) Número de vectores soporte utilizados por el modelo, (c) Indique cuáles de ellos son vectores soporte frontera y cuáles vectores soporte ligados; (iii) Por qué se dice que el porcentaje de acierto ahora obtenido es más fiable que el obtenido para el mismo modelo en el paso 5, donde se usó la opción *Use Training set* como método de evaluación.

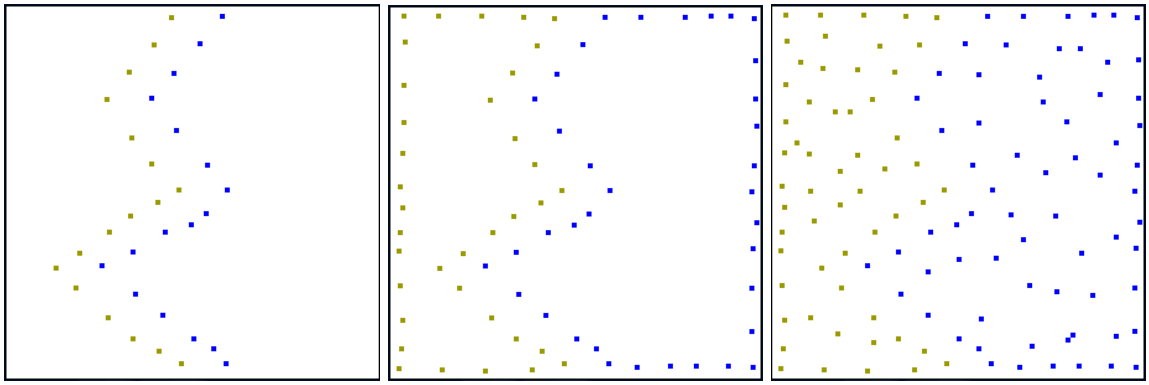
7. Tal y como se mencionada en la introducción, las SVM también se pueden aplicar al caso multiclase. La aproximación más usada para abordar el problema multiclase mediante SVMs es transformar el problema original en diferentes problemas de clasificación binaria. Existen varias estrategias para realizar 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 clases 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 *Classify* seleccione el algoritmo *SMO* y configure los siguientes parámetros (el resto se mantendrán al valor por defecto): Selecione *FilterType* igual a "*No normalization-standardization*" y kernel igual a "*RBFKernel*".
   - Seleccione 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:** (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 $(C, gamma)$; (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; (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 tres hiperplanos $D_{11}(\boldsymbol{x})$, $D_{12}(\boldsymbol{x})$ y $D_{23}(\boldsymbol{x})$ 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.

8. Acceder a la página web de LIBSVM[4]. 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 siguientes 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 fig. 5(b) hasta cerrar el perímetro de cada clase; (iii) Finalmente, acabe rellenando, con ejemplos situados al azar, el interior de las dos clases delimitadas anteriormente, tal y como se indica en la fig. 5(c).

[4] http://www.csie.ntu.edu.tw/~cjlin/libsvm/



(a) (b) (c)

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 parámetros 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:** (i) Del barrido realizado, indique los valores de C y *gamma* para los que se produjo la mejor frontera de separación; (ii) Incluya una captura de pantalla que muestre dicha frontera; (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 $-g$) y grado del polinomio (opción $-d$), además de con el parámetro C (opción $-c$); (iv) Indique los valores de C, *gamma* y grado del polinomio para los que se produjo la mejor frontera de separación; (v) Incluya una captura de pantalla que muestre dicha frontera.
- 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 fig. 6. La idea es crear tres clases dispuestas en capas concéntricas.
- 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:** (i) Justifique el tipo de kernel elegido; (ii) Indique los valores de parámetros para los que se produjo la mejor frontera de separación; (iii) Incluya una captura de pantalla que muestre dicha frontera.

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.

7. Evaluación

Consultar el documento "Rúbrica AA.pdf", accesible en el curso virtual, donde se detalla la rúbrica que se utilizará para evaluar esta práctica.

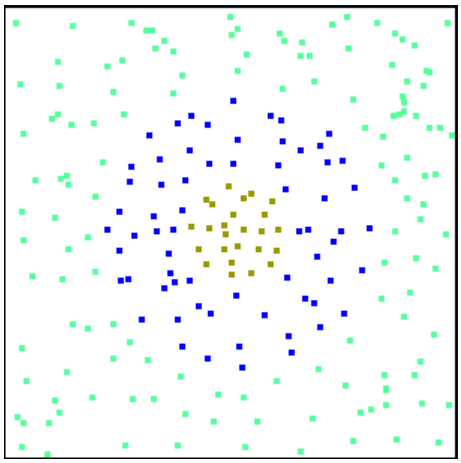


Figura 6: Problema de clasificación de tres clases no separables linealmente (fronteras de separación concéntricas). Visualice directamente la versión electrónica de este documento para ver correctamente los colores asociados a cada uno de los ejemplos

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.