Aprendizaje Automático
Máquinas de Vectores Soporte en Tareas de Regresión

Enrique J. Carmona Suárez
ecarmona@dia.uned.es

Primera versión: 15/12/2017 Última versión: 15/12/2017

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 y 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 regresión.

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 desde el curso virtual).

3. Objetivos

1. Familiarizarse con el uso del algoritmo SVM para resolver tareas de regresión.
2. Experimentar con el algoritmo *SMOreg* (una de las versiones Weka del algoritmo SVM para regresión) para los siguientes tipos de problemas:
   a) Los datos contienen ruido, pero se pueden ajustar mediante una función lineal.
   b) Los datos se pueden ajustar mediante una función cuadrática.
   c) Los datos contienen ruido, pero pueden ajustarse mediante una función cuadrática.
   d) Los datos son no lineales.
3. Experimentar con el valor de los diferentes parámetros del algoritmo *SMOreg*.
4. Evaluar los modelos obtenidos.
5. Experimentar con el software LIBSVM[1] para visualizar las funciones resultantes de ajustar un conjunto de datos 1D.

4. Material

- Programa Weka.
- *Applet* LIBSVM[2].
- Archivo de datos: "data_CuasiLineal.arff", "data_Cuadratica.arff", "data_CuasiCuadratica.arff" y "data_NoLineal.arff" (accesibles desde el curso virtual).

5. Actividades

1. Desde la ventana *Explorer*, seleccione el panel *Classify* y escoja el algoritmo *SMOreg* (classifiers > functions). A continuación, haga click con el botón izquierdo del ratón en el campo de la caja *Classifier* (del panel *Classify*) en la que se muestra el comando *SMOreg* con todos sus parámetros. En la ventana emergente resultante (ver Fig. 1) se pueden configurar diferentes parámetros de interés. Concretamente, el algoritmo para regresión basado en SVMs se selecciona realmente desde el campo *regOptimizer*. En esta práctica siempre utilizaremos el denominado *RegSMOImproved* [Shevade et al., 2000], el cual viene ya seleccionado por defecto. Cliqueando sobre el campo en el que se muestra *RegSMOImproved*, aparecerá una nueva ventana emergente que permite configurar los parámetros asociados. En nuestro caso, el único parámetro de interés es el denominado *epsilonParameter* (ϵ), que permite definir la anchura de la zona insensible de la función de pérdida (ver Eq. (61) en [Carmona, 2016]). El valor del resto de parámetros del optimizador se mantendrá siempre a su valor por defecto. Dado que el valor de *epsilonParameter* se modificará frecuentemente en los diferentes experimentos a realizar, es preciso señalar que éste no debe confundirse con el valor del parámetro *epsilon*, el cuál debería conservar siempre su valor por defecto. Desde la ventana correspondiente a la Fig. 1 se puede configurar también el tipo de kernel a utilizar, así como el valor de C, es decir, el conocido como *parámetro de regularización* (ver Eq. (62) en [Carmona, 2016]). Finalmente, el valor del parámetro *filterType* se mantendrá siempre a "No normalization/standardization" (¡ojo!, no es la opción por defecto), es decir, se trabajará con el valor real de los datos, sin normalizar ni estandarizar. Esto permite que el modelo de regresor obtenido pueda ser fácilmente interpretable en el dominio de los datos originales. En cada una de las ventanas mencionadas, existe un botón More que permite mostrar información de ayuda asociada a cada uno de los parámetros de configuración.
2. Cargar el fichero "data_CuasiLineal.arff" desde el panel *Preprocess* de *Explorer*. El contenido de este archivo contiene un conjunto de datos con ruido, pero ajustable mediante una función lineal. Visualize dicho conjunto desde el panel *Visualize* de *Explorer*. A continuación, realice las siguientes acciones:

[1] LIBSVM es un software integrado para máquinas de vectores soporte
[2] Disponible en https://www.csie.ntu.edu.tw/~cjlin/libsvm/index.html

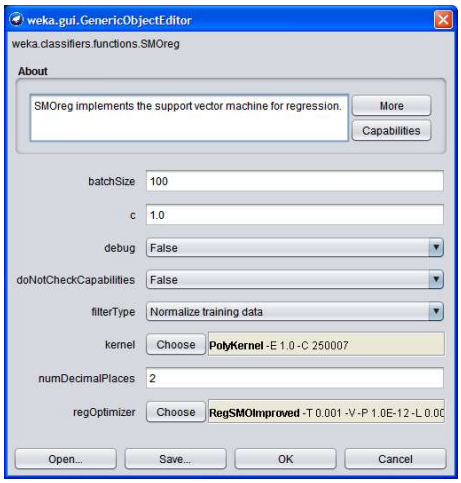

Figura 1: Conjunto de parámetros que permiten configurar el algoritmo SMO de Weka.

- Dado que el conjunto de datos asociados al fichero cargado puede ajustarse por una recta, se debería usar un kernel lineal. Para ello, bastará seleccionar un kernel polinómico (*PoliKernel*) y, desde su ventana de configuración, seleccionar *exponent*=1. El resto de parámetros se dejarán a su valor por defecto.
- Dado que el kernel lineal no depende de ningún parámetro, los únicos parámetros con los que podemos jugar en este experimento son C y ϵ. Por tanto, seleccione la opción *Cross-validation* (*Folds*=10) y construya una tabla donde se muestre el coeficiente de correlación (*correlation coefficient*) obtenido al ejecutar el algoritmo para cada par de valores posibles de C y ϵ, donde C={1, 10, 100} y ϵ={0.01, 0.1, 1}. **Nota**: No es obligatorio, pero, si lo desea, puede probar otras parejas de valores C y ϵ en un intento de obtener un modelo más conveniente.
- Observe que al ejecutar el algoritmo con un kernel lineal, el modelo resultante (ver caja *Classifier Output*) no se muestra en términos de vectores soporte sino en términos de los dos coeficiente necesarios para definir una recta (pendiente y término independiente). Sin embargo, esto no significa que no sea posible definir un modelo de regresión lineal en función de vectores soporte para un conjunto de datos ajustables mediante un kernel lineal. Simplemente, lo que ocurre es que, para el caso de usar un kernel lineal, el algoritmo utilizado construye el modelo en la forma anteriormente mencionada.
- **Documentación a entregar**: (i) Mostrar el plot que representa la variable a predecir (*class*) en función de la variable x; (ii) Muestre la tabla construida anteriormente; (iii) Justifique los resultados obtenidos en la tabla en términos del significado físico de C y ϵ; (iv) Muestre en una gráfica el conjunto de datos a ajustar y la recta de regresión resultante del mejor modelo obtenido (represente gráficamente también la zona tubular, mostrando las ecuaciones de las rectas que delimitan superior e inferiormente dicha zona). **Pista**: para obtener las ecuaciones de las rectas de la zona tubular, deberá tener en cuenta el valor de ϵ elegido para obtener el modelo.

3. Vamos a considerar ahora el caso de un conjunto de ejemplos no separables linealmente, pero que pueden ajustarse perfectamente mediante una función cuadrática. Realice las siguientes acciones:

   - Cargar el fichero "data_Cuadratica.arff" desde el panel *Preprocess*. Visualize dicho conjunto desde el panel *Visualize* de *Explorer*.
   - Como kernel, se seleccionará también el kernel polinómico (*PoliKernel*), pero en este caso con valor de *exponent*=2. Tal y como se indica en la ventana de configuración del kernel, se puede utilizar dos tipos de kernels polinómicos: $K(x,y) =< x,y >^p$ o $K(x,y) = (< x,y > +1)^p$, donde p corresponde al valor del parámetro *exponent*. Se elegirá el primer tipo de kernel. Para ello, bastará poner el valor del parámetro *useLowerOrder=False* (valor por defecto). El resto de parámetros permanecen al mismo valor

que los indicado en el experimento del epígrafe 2. Nótese que los modelos o funciones de regresión producidos por Weka se pueden expresar de forma genérica como (ver caja *Classifier Output* del panel *Classify*):

$$f(\boldsymbol{x}) = \pm a_1 * K[1] \pm ...a_n * K[n] \pm a_0 \tag{1}$$

donde el símbolo $\pm$ delante de cada sumando denota que el signo puede ser positivo o negativo y $K[i]$ es el kernel aplicado al vector soporte i-ésimo. Seleccione la opción *Cross-validation* (*Folds*=10) y ejecute el algoritmo para todas las parejas de valores posibles de C y ϵ, siendo C=$\{1, 10, 100\}$ y ϵ=$\{0.01, 0.1, 1\}$. **Nota**: No es obligatorio, pero, si lo desea, puede probar otras parejas de valores C y ϵ en un intento de obtener un modelo más conveniente.

- **Documentación a entregar**: (i) Mostrar el plot que representa la variable a predecir (*class*) en función de la variable x; (ii) Justifique por qué para el tipo de datos usados es mejor usar un kernel polinómico en lugar de un kernel gaussiano o un kernel lineal; (iii) Justifique por qué el valor usado del exponente del kernel polinómico es 2; (iv) Muestre en sendas tablas el valor del coeficiente de correlación y el número de vectores soporte utilizados en cada modelo obtenido, en función de los valores de C y ϵ y justifique los resultados obtenidos en términos del significado físico de C y ϵ. Tenga en cuenta que el modelo será tanto más exacto cuanto más cercano a 1 sea el coeficiente de correlación, pero estará más sobreajustado a los datos cuanto mayor sea el número de vectores soporte que contenga; (v) Teniendo en cuenta que el regresor obtenido por una SVR se puede expresar de forma genérica como (ver Ec. (87) en [Carmona, 2016]):

$$f(\boldsymbol{x}) = (\alpha_1^{*-} - \alpha_1^{*-})K(\boldsymbol{x}, \boldsymbol{x}_1) + ... + (\alpha_n^{*-} - \alpha_n^{*-})K(\boldsymbol{x}, \boldsymbol{x}_n) + b^* \tag{2}$$

indique cómo se relacionan los valores a_i en la Ec. (1) con los valores α_1^{*-} y α_1^{*-} mostrados en la Ec. (2); (vi) Atendiendo a las respuestas dadas en (v), exprese la función de regresión obtenida por Weka para el caso $(C, \epsilon) = (10, 0.1)$ de acuerdo al formalismo expresado en la Ec. (2), siendo $K(x, x_i) =< x, x_i >^p$; (vii) Aplique la definición de producto escalar y opere la expresión obtenida en (vi) hasta obtener una expresión compacta y simplificada; (viii) Justifique si cada uno de los vectores soporte obtenidos para $(C, \epsilon) = (10, 0.1)$ son vectores soporte propiamente dichos o vectores soporte ligados; (ix) Muestre el plot de errores[3] del que considere mejor regresor obtenido y justifique por qué es el mejor.

4. Vamos a considerar ahora el caso de un conjunto de ejemplos no separables linealmente, pero que pueden ajustarse aproximadamente mediante una función cuadrática. Realice las siguientes acciones:

   - Cargar el fichero "data_ CuasiCuadratica.arff" desde el panel *Preprocess*. Visualice dicho conjunto desde el panel *Visualize* de *Explorer*.
   - Se seleccionará la misma configuración de parámetros realizada en el experimento del epígrafe 3, salvo que, en este caso, el valor del parámetro del kernel polinómico *useLowerOrder* se pondrá a *True*, indicando con esto que implícitamente se usará el kernel $K(x, y) = (< x, y > +1)^p$. Seleccione la opción *Cross-validation* (*Folds*=10) y ejecute el algoritmo para todas las parejas de valores posibles de C y ϵ, siendo C=$\{1, 10, 100\}$ y ϵ=$\{0.01, 0.1, 1\}$. **Nota**: No es obligatorio, pero, si lo desea, puede probar otras parejas de valores C y ϵ en un intento de obtener un modelo más conveniente.
   - **Documentación a entregar**: (i) Mostrar el plot que representa la variable a predecir (*class*) en función de la variable x; ; (ii) Muestre en sendas tablas el valor del coeficiente de correlación y el número de vectores soporte utilizados en cada modelo obtenido, en función de los valores de C y ϵ; (iii) Muestre el modelo de regresor obtenido por Weka para $(C, \epsilon) = (100, 1)$ y justifique si cada uno de los vectores soporte obtenidos son vectores soporte propiamente dichos o vectores soporte ligados; (iv) Muestre el plot de errores del que considere mejor regresor obtenido y justifique por qué es el mejor.

[3] Para ello, desde la caja *Result list* del panel *Classify*, haga click con el botón derecho del ratón en el modelo deseado y seleccione la opción *Visualize classifier errors*. No olvide seleccionar convenientemente las variables a representar en los ejes X e Y.

5. Vamos a considerar ahora el caso general de un conjunto de ejemplos no separables linealmente. Realice las siguientes acciones:

   - Cargar el fichero "data_NoLineal.arff" desde el panel *Preprocess*. Visualice dicho conjunto desde el panel *Visualize* de *Explorer*.
   - Como kernel, se seleccionará "*RBFKernel*", también llamado *kernel gaussiano*. En su ventana de configuración de parámetros, el parámetro a tener en cuenta corresponde al valor de *gamma*, dado que matemáticamente este tipo de kernel se define como:

   $$K_G(\boldsymbol{x}, \boldsymbol{x}') = e^{-\gamma \|\boldsymbol{x}-\boldsymbol{x}'\|^2} \equiv e^{-\gamma <(\boldsymbol{x}-\boldsymbol{x}'),(\boldsymbol{x}-\boldsymbol{x}')>}, \ \gamma > 0 \tag{3}$$

     El resto de parámetros del kernel deben permanecer a su valor por defecto.
   - El resto de parámetros de configuración no mencionados aquí se mantendrán a los valores indicados en el experimento del epígrafe 3.
   - Seleccione la opción *Cross-validation* (*Folds*=10) y ejecute el algoritmo para todas las parejas de valores posibles de C y ϵ, siendo C={10, 100, 1000}, ϵ={0.01, 0.1, 1} y $\gamma = \{0.1, 1, 100\}$. **Nota**: No es obligatorio, pero, si lo desea, puede probar otras parejas de valores C y ϵ en un intento de obtener un modelo más conveniente.
   - **Documentación a entregar**: (i) Mostrar el plot que representa la variable a predecir (*class*) en función de la variable x; (ii) Justifique por qué para el tipo de datos usados es mejor usar un kernel gaussiano en lugar de un kernel polinómico o un kernel lineal; (iii) Muestre en sendas tablas el valor del coeficiente de correlación y el número de vectores soporte utilizados en cada modelo obtenido, en función de los valores de C, ϵ y γ (justifique los resultados obtenidos en términos del significado físico de C, ϵ y γ). **Nota**: el parámetro γ está relacionado con el valor σ (desviación típica de la gaussiana), siendo este último inversamente proporcional al valor del primero; (iv) Muestre el plot de errores del que considere mejor regresor obtenido y justifique por qué es el mejor.

6. Acceda a la página web de LIBSVM[4]. Desde dicha web, acceda 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 regresión, como el caso que nos ocupa, se puede entrenar una SVR para conjuntos de ejemplos 1D (proporcionados manualmente por el usuario). Aunque la interfaz permite al usuario trabajar con diferentes tipos de SVRs (*epsilon-SVR*, *nu-SVR*), aquí nos centraremos en el uso de *epsilon-SVR*, es decir, el mismo tipo de SVR que el que hemos estado usando hasta ahora. A continuación realice las siguientes acciones:

   - En primer lugar se va a crear un problema de regresión no lineal 1D. 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 2(a), con el objeto de crear una curva en "diente de sierra"; (ii) Continúe añadiendo los ejemplos indicados en la fig. 2(b) hasta obtener una curva parecida.
   - Ejecute el algoritmo *epsilon-SVC* (opción "$-s$ 3") con kernel gaussiano (opción "$-t$ 2") para un barrido adecuado de diferentes valores de los parámetros ϵ (opción $-e$), C (opción $-c$) y *gamma* (opción $-g$). 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 y, por consiguiente, tendrá que volverlos a introducir manualmente.
   - **Documentación a entregar:** (i) Del barrido realizado, indique los valores de ϵ, C y *gamma* para los que se produjo la mejor función de ajuste; (ii) Incluya una captura de pantalla que muestre el ajuste de dicha función; (iii) Investigue si es posible realizar un ajuste de los mismos datos usando un kernel polinómico. En este caso, deberá indicar expresamente este tipo de kernel (opción $-t$ 1) y jugar con el grado del polinomio (opción $-d$), además de con los parámetros ϵ (opción $-e$) y C (opción $-c$). El resto de parámetros del kernel polinómico *coef0* y *gamma* los puede mantener fijos a los valores

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

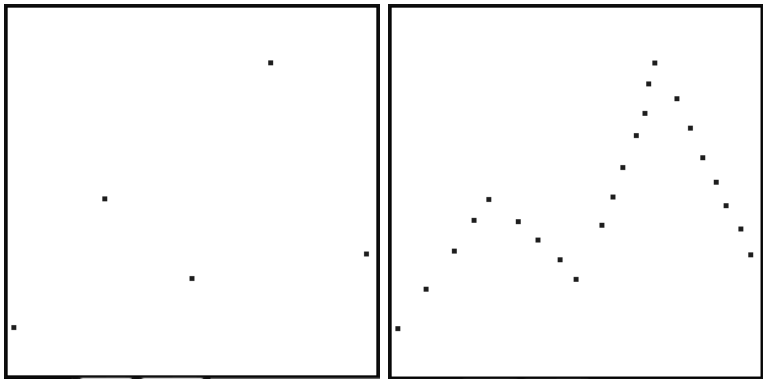

Figura 2: Construcción en dos pasos, (a) y (b), de un problema de regresión no lineal 1D (curva en "diente de sierra").

"$-r$ 1" y "$-g$ 1", respectivamente; (iv) Indique los valores de ϵ, C, *gamma* y grado del polinomio para los que se produjo el mejor ajuste; (v) Incluya una captura de pantalla que muestre el mejor regresor obtenido; (vi) A la vista de los resultados obtenidos, ¿cuál es el kernel con mayor potencial para ajustar al conjunto de datos utilizado?

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.

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.

[Shevade et al., 2000] Shevade, S. K., Keerthi, S. S., Bhattacharyya, C., & Murthy, K. R. K. (2000). Improvements to the smo algorithm for svm regression. *IEEE Transactions on Neural Networks*, 11(5), 1188–1193.