Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial
ISSN: 1137-3601
revista@aepia.org
Asociación Española para la Inteligencia Artificial
España

Rodríguez, Juan José; Alonso, Carlos J.
Clasificación de Series: máquinas de vectores soporte y literales basados en intervalos
Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial, vol. 8, núm. 23, verano, 2004, p. 0
Asociación Española para la Inteligencia Artificial
Valencia, España

Clasificación de Series: Máquinas de Vectores Soporte y Literales basados en Intervalos*

Juan José Rodríguez

Lenguajes y Sistemas Informáticos
Universidad de Burgos, Spain
jjrodriguez@ubu.es

Carlos J. Alonso

Grupo de Sistemas Inteligentes
Departamento de Informática
Universidad de Valladolid, Spain
calonso@infor.uva.es

Resumen

En trabajos previos se ha presentado un sistema de clasificación de series. Dicho método se apoya en el método de combinación de clasificadores denominado *boosting*, utilizando clasificadores base muy simples, formados por únicamente un literal. Estos predicados se basan en intervalos temporales. Los clasificadores obtenidos son simplemente una combinación lineal de literales. Por tanto, es natural el esperar alguna mejora en los resultados si los literales se combinan de modos más complejos. En este trabajo se explora la posibilidad de utilizar los literales seleccionados mediante el método de boosting como nuevos atributos, y entonces utilizar el método SVM sobre estos metaatributos. Los resultados experimentales demuestran la validez del método propuesto.

Palabras clave: Máquinas de Vectores Soporte, Clasificación de Series, Boosting.

1. Introducción

Normalmente, el método boosting [13] se utiliza con clasificadores base bien conocidos, como árboles de decisión o redes neuronales. En consecuencia, su principal aportación es la capacidad de mejorar los resultados, en cuanto a precisión, de dichos métodos. Sin embargo, este método también permite desarrollar métodos de aprendizaje específicos para un dominio de un modo bastante simple. Es suficiente con desarrollar un método de clasificación modesto (*débil*), adecuado para dicho dominio. Entonces, utilizando boosting, es posible obtener un sistema de aprendizaje *fuerte* para dicho dominio.

El dominio que aquí se va a considerar es el de las series temporales. Dentro de este dominio hay muchas casuísticas y por lo tanto subdominios, por lo que es posible plantear clasificadores base adecuados para dichos subdominios. Aquí no nos ceñiremos a ninguna aplicación concreta. Nuestros clasificadores base, los literales basados en intervalos, consideran lo que ocurre en un determinado intervalo, e.g., cuál es el valor medio. Estos clasificadores son muy simples, pero están diseñados para este dominio en particular. En conjunción con el boosting, es posible obtener buenos resultados [10, 9].

Los clasificadores obtenidos mediante boosting son combinaciones lineales de los clasificadores

*Este trabajo ha sido financiado por el proyecto del MCyT DPI2001-4404-E y el proyecto de la Junta de Castilla y León VA101/01.

base. Entonces, es razonable cuestionar si sería posible obtener mejores resultados utilizando combinaciones de los clasificadores base más complejas. Aunque el boosting no sea tan resistente al sobreajuste como en un principio se pensaba [12], sigue siendo muy útil su capacidad para obtener una combinación de clasificadores base que son capaces de clasificar razonablemente bien. Es por tanto una idea natural el obtener clasificadores base mediante boosting, pero combinarlos utilizando un método más robusto. En este trabajo se explora el uso de las Máquinas de Vectores Soporte (SVM) [2] para la combinación de los clasificadores base obtenidos.

El resto del trabajo se organiza como sigue. El método de clasificación se presenta en la sección 2. La sección 3 presenta la validación experimental del método. Se mencionan algunos trabajos relacionados en la sección 4. Finalmente, se concluye en la sección 5.

2. El Método de Clasificación

2.1. Predicados basados en Intervalos

Antes de empezar con los basados en intervalos, se consideran condiciones sobre valores puntuales:

- valor-punto$_{\leq}$(Ejemplo, Variable, Punto, Umbral). Es cierto si, para el Ejemplo, el valor de la Variable en el Punto es menor o igual que el Umbral.

La Variable se incluye en el predicado porque también se consideran series multivariable. El concepto aquí utilizado de variable es el que se utiliza en series temporales, no el usado en aprendizaje automático. La combinación de una Variable (e.g., x) y un Punto (e.g., 7) se podría considerar en aprendizaje automático como un atributo (e.g., x_7). Un sistema que utilice sólo este predicado es equivalente a un sistema de aprendizaje "atributo valor" (aquellos en los que en los clasificadores obtenidos se comparan los valores de los atributos con unos determinados valores, e.g., los árboles de decisión). Este predicado se introduce con el único fin de comparar los resultados obtenidos para boosting y SVM con y sin literales basados en intervalos.

Los predicados basados en intervalos consideran lo que ocurre en un intervalo dado. En el presente trabajo nos limitaremos a considerar los siguientes:

- media$_{\leq}$(Ejemplo, Variable, Inicio, Fin, Umbral). Es cierto si, para el Ejemplo, el valor medio de la variable Variable en el intervalo delimitado por Inicio y Fin es menor o igual que el Umbral.
- desviación$_{\leq}$(Ejemplo, Variable, Inicio, Fin, Umbral). Es cierto si, para el Ejemplo, la desviación de los valores de la Variable en el intervalo delimitado por Inicio y Fin es menor o igual que el Umbral.

En [9], se describen más predicados basados en intervalos (e.g., siempre, alguna-vez), y se explica como seleccionarlos de un modo eficiente.

2.1.1. Series de Longitud Variable

Hay diversos métodos, más o menos complicados, que permiten uniformizar la longitud de un conjunto de series. Estos métodos, que preprocesan el conjunto de datos, son adecuados para diversos dominios. No obstante, el uso de estos métodos no es una solución general. Por ejemplo, sea el caso extremo en el que las series de dos clases tuvieran la misma forma pero diferentes longitudes. Por tanto, es importante que el método de aprendizaje pueda tratar directamente con series de longitud variable. Por supuesto, esto no impide que se pueda utilizar sobre conjuntos de datos en los que se ha realizado un preprocesamiento para uniformizar las longitudes.

Si las series tienen diferentes longitudes, existirán literales cuyos intervalos harán referencia a posiciones posteriores al final de la serie. Para estos casos, el resultado de la evaluación del literal no será cierto o falso, sino una abstención.

2.2. Boosting de Literales basados en Intervalos

Claramente, para obtener clasificadores precisos es necesario combinar varios literales. Por ejemplo, sería posible combinarlos usando reglas o árboles de decisión. Sin embargo, la aproximación

seguida en este trabajo es combinarlos mediante boosting. Actualmente, un área de investigación activa es el uso de agrupaciones (*ensembles*) de clasificadores. Se obtienen generando y combinando clasificadores base, construidos con cualquier otro método de aprendizaje. El objetivo de estas agrupaciones es incrementar la precisión.

Uno de los métodos más populares para la creación de agrupaciones de clasificadores es la familia de métodos conocida como *boosting* [13]. Dentro de dicha familia el miembros más conocido es AdaBoost. Su funcionamiento se basa en asignar un peso a cada ejemplo (inicialmente el mismo para todos). En cada iteración se construye un clasificador *base* (también denominado *débil*), teniendo en cuenta la distribución de pesos. Entonces los pesos de cada ejemplo se reajustan, en función de si el clasificador base lo clasificó o no correctamente. A la hora de clasificar el resultado se obtiene mediante voto ponderado de los clasificadores base.

AdaBoost trata sólo con problemas binarios, pero hay diversos métodos que permiten su extensión a problemas multiclase. El aquí utilizado es AdaBoost.MH [14].

El cuadro 1 muestra un ejemplo de clasificador. Se corresponde con un problema de 3 clases. Está formado por 10 clasificadores base. La primera columna muestra el literal. Para cada clase hay otra columna, con el peso asociado al literal para esa clase.

Para clasificar un nuevo ejemplo, se calcula un peso para cada clase y se asigna al ejemplo la clase con mayor peso. Inicialmente el peso de cada clase es 0. Se evalúa cada literal. Si es cierto, el peso de cada clase se ve incrementado con el peso asociado al literal y la clase. Si es falso en vez de incrementar se decrementa.

En la versión original de AdaBoost, los clasificadores base se limitaban a devolver $+1$ or -1. Sin embargo, hay variantes que usan predicciones con grados de confianza [14]. En este caso el clasificador base retorna un valor real. El signo indica la clasificación y el valor absoluto el grado de confianza en la predicción. Al evaluar un literal hay 3 posibles resultados: falso, cierto y abstenciones. A estos resultados se les asignan, respectivamente, los valores numéricos -1, 1 y 0.

2.3. SVM de Literales basados en Intervalos

Una vez que se ha obtenido una combinación de literales, es posible su uso como clasificador. Sin embargo, aquí se considerará otra alternativa. Cada literal puede verse como un nuevo (meta)atributo. Entonces, cualquier otro método de clasificación puede usarse sobre estos nuevos atributos.

Sería posible utilizar directamente los literales como atributos booleanos, pero los aquí considerados comparan el valor de una función sobre un intervalo (e.g., la media) con un umbral. Si el método de clasificación a utilizar soporta atributos numéricos, es razonable usar los valores de las funciones en vez de los valores de los literales.

Boosting combina los clasificadores base mediante un voto ponderado. Nuestra hipótesis es que quizás los clasificadores base podrían ser combinados de algún modo mejor. En este trabajo se estudia esta hipótesis utilizando como clasificadores las Máquinas de Vectores Soporte.

3. Validación Experimental

3.1. Conjuntos de Datos

El cuadro 2 resume las características de los conjuntos de datos. El primero es artificial, los otros dos son reales. Para los dos primeros existe una partición especificada de los ejemplos en entrenamiento y test. Para el último no existe dicha partición, por lo que se usa validación cruzada.

3.1.1. Trace

Este conjunto de datos se introduce en [11]. Se propone como banco de pruebas de sistemas de clasificación de patrones temporales en la industria de procesos. Se generó de modo artificial. Hay 4 variables, y cada una de ellas tiene dos comportamientos, como se muestra en la figura 1. La combinación de los comportamientos de las distintas variables da lugar a 16 clases diferentes. Se generaron 1600 ejemplos, 100 de cada clase. La mitad son para entrenamiento y la otra mitad para test.

Literal	Clase 1	Clase 2	Clase 3
desviación$_{\leq}$(E, x, 63, 126, 1.813266)	-0.399084	-0.421169	1.037954
desviación$_{\leq}$(E, x, 48, 111, 1.889214)	-0.232020	-0.153754	0.606268
media$_{\leq}$(E, x, 38, 101, 0.770881)	0.096480	0.072766	0.715047
¬media$_{\leq}$(E, x, 30, 33, 3.349725)	0.746371	-1.641331	0.664797
media$_{\leq}$(E, x, 55, 58, 4.280890)	-0.906306	0.352215	0.471577
desviación$_{\leq}$(E, x, 3, 34, 1.653625)	-0.334412	2.162192	-0.114016
¬media$_{\leq}$(E, x, 49, 52, 5.508630)	0.743638	0.517645	-0.195655
desviación$_{\leq}$(E, x, 25, 56, 1.107998)	0.634717	0.540074	0.000214
¬media$_{\leq}$(E, x, 44, 47, 3.720549)	0.484320	-0.932335	-0.155742
¬desviación$_{\leq}$(E, x, 26, 33, 3.355703)	-0.149486	0.090624	0.657721

Cuadro 1: Ejemplo de un clasificador obtenido mediante boosting, para un conjunto de datos con 3 clases. Por cada literal, se asocia un peso a cada clase.

		Longitud			Ejemplos	
Conjuntos de Datos	Variables	Mínimo	Máximo	Clases	Total	Test
Trace	4	268	394	16	1600	800
Vocales Japonesas	12	7	29	9	640	370
Auslan	22	45	136	95	2565	Valid. Cruz.

Cuadro 2: Características de los conjuntos de datos.

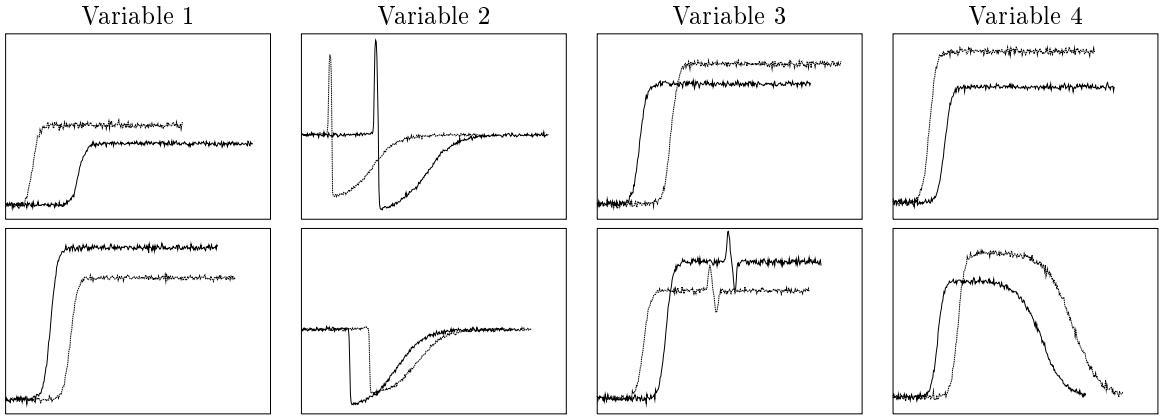

Figura 1: Conjunto de datos *Trace*. Cada ejemplo esta formado por 4 variables, cada variable tiene dos posibles comportamientos. Las gráficas de la fila superior muestran el primer comportamiento, las de la inferior el segundo. En las gráficas se muestran dos ejemplos de cada uno de los comportamientos.

3.1.2. Vocales Japonesas

Este conjunto de datos se introduce en [6]. A pesar de su nombre, no se trata de reconocer vocales, sino locutores. Hay 9 locutores diferentes, que pronunciaron las 2 vocales japonesas /ae/ sucesivamente. Para cada pronunciación se aplicó un análisis de predicción lineal de 12 grados para obtener una serie temporal discreta con 12 coeficientes cepstrum LPC. Cada pronunciación de cada locutor forma una serie cuya longitud está entre 7 y 29, y cada punto de la serie está formado por 12 valores.

3.1.3. Auslan

Auslan es el lenguaje australiano de signos para sordos. Se recogieron ejemplos de cada signo utilizando un guante de realidad virtual [4]. Hay dos versiones de este conjunto de datos, obtenidas utilizando distintos equipos. Según [4], en términos de la calidad de los datos, el segundo equipo es claramente superior al primero. En consecuencia en este trabajo se utiliza la segunda versión.

3.2. Resultados

Para aquellos conjuntos de datos con una partición especificada, los experimentos se repitieron 5 veces, debido a que el método no es determinista. Para el conjunto de datos *Auslan*, se uso validación cruzada con 5 grupos, debido a que es lo habitual para este conjunto de datos [4].

3.2.1. Boosting

Los resultados obtenidos con boosting para los conjuntos de datos se muestran en la tabla 3. El error de test se compara con el número de iteraciones del método boosting, esto es, el número de literales. El máximo número de iteraciones consideradas es 100. En algunos casos es posible obtener mejores resultados utilizando más literales, pero el objetivo no es obtener el mejor resultado posible mediante boosting. Si se consideran los diferentes resultados para el número máximo de iteraciones, el aspecto más sorprendente es que el mejor resultado para el conjunto de datos *Auslan* se obtiene usando el predicado basado en valores puntuales. Por otro lado, para el conjunto de datos *Trace*, utilizando dicho predicado el sistema no es capaz de aprender. Estos resultados son un tanto desalentadores, porque el único caso en que queda claramente demostrado la conveniencia de usar predicados basados en intervalos es el primero, que es un conjunto de datos artificial. Estos resultados muestran que los dos problemas reales podrían abordarse utilizando técnicas de aprendizaje convencionales, esto es, que no manejan el concepto del tiempo, obteniendo resultados razonables.

Aunque el conjunto de datos *Trace* es artificial, entendemos que refleja alguna de las situaciones que se presentan en un sistema dinámico. Por tanto, creemos que los resultados previos sí que alientan el uso de predicados basados en intervalos. Por otro lado, en este trabajo no se están considerando todos los predicados definidos. Utilizando algunos de ellos sí que es posible mejorar los resultados obtenidos utilizando valores puntuales. El objetivo del presente trabajo no es demostrar la utilidad de estos predicados, sino estudiar la posibilidad de mejorar los resultados obtenidos con boosting mediante SVM. En consecuencia, no se van a incluir más detalles sobre resultados experimentales obtenidos mediante boosting y predicados. Dichos resultados se pueden consultar en [9].

3.2.2. SVM

La implementación utilizada del método SVM es la disponible en la biblioteca WEKA [16]. Se basa en el algoritmo de optimización secuencial minimal (SMO) [7, 5]. El único kernel utilizado fue el polinomial, con exponentes máximos 1 y 2, esto es, lineal y cuadrático. Los parámetros no se ajustaron de ningún modo, se usaron los valores por defecto. Para problemas multiclase, la herramienta seleccionada construye un clasificador por cada par de clases.

Los resultados se muestran en la tabla 4. También se incluyen los resultados obtenidos a partir de los conjuntos de datos originales. Las series son de longitud variable, pero es necesario expresarlas utilizando un número fijo de atributos. La solución utilizada es tener tantos atributos como la longitud de la serie más larga (multiplicada por el número de variables). Para los atributos correspondientes a posiciones posteriores al final de la serie, el valor asignado es el de *desconocido*, ya que la herramienta utilizada permite la asignación de dicho valor. La misma aproximación se

Conjunto de datos *Trace*

	10	20	30	40	50	60	70	80	90	100
valor punto	73.28	73.15	72.67	73.22	73.20	73.05	73.03	73.30	72.92	73.92
media y desviación	34.52	20.32	12.57	10.32	9.72	10.82	10.12	9.93	10.22	10.90

Conjunto de datos *Vocales Japonesas*

	10	20	30	40	50	60	70	80	90	100
valor punto	31.95	18.54	12.16	10.11	9.51	8.59	7.62	7.46	6.81	6.32
media y desviación	26.49	15.19	10.38	9.30	7.19	5.95	5.78	5.35	4.97	4.86

Conjunto de datos *Auslan*

	10	20	30	40	50	60	70	80	90	100
valor punto	78.61	53.19	38.35	28.02	22.41	17.26	14.25	12.28	11.21	9.49
media y desviación	84.44	63.45	45.11	31.71	26.65	20.05	16.16	12.79	11.34	10.17

Cuadro 3: Resultados obtenidos con boosting. Se muestra el error de test frente al número de iteraciones.

utiliza para los predicados basados en intervalos, si el intervalo en cuestión es posterior al final de la serie.

El uso de boostig con valores puntuales es de hecho un método de selección de atributos, ya que los atributos seleccionados estaban disponibles en el conjunto de datos original. La reducción en el número de atributos es bastante importante, por ejemplo para el conjunto de datos *Auslan* el número de atributos de los datos originales era de $136 \times 25 = 2992$. El número de atributos seleccionados es sólo 100.

Los resultados obtenidos utilizando la media y la desviación son siempre mejores que los resultados obtenidos con valores puntuales. A continuación se comparan los resultados con los obtenidos por otros autores para estos conjuntos de datos:

- Para el conjunto de datos *Trace* el resultado obtenido en [11], utilizando redes neuronales y wavelets es un error de 1.4 %, pero el 4.5 % de los ejemplos no se asignan a ninguna clase. Estos valores son claramente peores que los resultados con predicados sobre intervalos.
- Para el conjunto de datos *Vocales Japonesas* los resultados de [6] son un error del 5.9 % utilzando el método introducido en ese trabajo, y 3.8 % utilizando un Modelo Oculto de Markov continuo con 5 estados. En este caso, todos nuestros resultados son mejores, incluso los obtenidos con el conjunto de datos original.
- Para el conjunto de datos *Auslan*, el mejor error obtenido en [4] parece ser ligeramente inferior al 2 % (se muestra en una gráfica, no en una tabla). Este resultado se obtuvo mediante votación de varios clasificadores (9 para el mejor valor). Cada uno de estos clasificadores se obtuvo mediante construcción y selección no supervisada de atributos, y boosting de árboles de decisión sobre dichos atributos. Nuestros resultados con la media y la desviación están cercanos a 1.5 %.

4. Trabajos Relacionados

El principal defensor del uso de metaatributos para la clasificación de series temporales es Kadous [4]. Aunque algunos de sus metaatributos son similares a nuestros literales basados en intervalos, el mismo indica que su método y el nuestro [8] son muy diferentes. La principal diferencia es que en su caso existe un primer paso en el que se extraen, de un modo no supervisado, características de cada serie. Entonces, esas características se agrupan, y cada grupo se considera como un nuevo atributo.

En cuanto a los SVM, uno de los trabajos más relacionados es [1]. Usan como kernel el método *alineamiento dinámico temporal* (DTW). Afirman que aunque la distancia proporcionada por éste método no sea una métrica, porque no se cumple la desigualdad triangular, y aunque este kernel no sea definido positivo, obtienen buenas tasas de reconocimiento. Por otro lado, en [15] se usa un kernel basado en DTW.

conjunto de datos	kernel	datos originales	literales	
			valor punto	media y desviación
Trace	Lineal	64.88	66.95	0.15
	Cuadrático	63.50	65.63	0.20
Vocales Japonesas	Lineal	3.24	2.81	1.51
	Cuadrático	3.24	2.70	1.57
Auslan	Lineal	7.27	3.35	1.42
	Cuadrático	7.74	3.27	1.52

Cuadro 4: Resultados obtenidos con SVM.

5. Conclusiones y Trabajo Futuro

Se ha presentado una aproximación novedosa para la clasificación de series temporales. Soporta series de longitud variable y series multivariable. Está basada en el uso de SVM sobre atributos obtenidos previamente. Dicha obtención de los atributos podría considerarse como un preprocesamiento. Sin embargo, estos atributos se obtienen utilizando otro método de clasificación, boosting. Dicho método se aplica utilizando como clasificadores base métodos que seleccionan el mejor metaatributo. Aunque es posible utilizar directamente los clasificadores obtenidos mediante boosting, los resultados experimentales demuestran que la precisión puede mejorarse sustancialmente si se aplica SVM sobre los atributos obtenidos. La validación experimental demuestra que el método es una alternativa clara a los métodos de clasificación de series habituales.

Dentro de los objetivos y las conclusiones de este trabajo no se incluye ninguna afirmación sobre ningún tipo de superioridad de SVM frente a boosting. En primer lugar, la complejidad de los clasificadores obtenidos mediante estos dos métodos no son comparables. En ambos casos se obtienen combinaciones lineales. Pero mientras en el caso de boosting existe una combinación lineal para cada clase, en SVM hay una combinación lineal para cada *par* de clases. Esto se debe a que el método utilizado para tratar con problemas multiclase utilizado en SVM es el denominado "uno contra uno".

Los resultados obtenidos mediante boosting son, posiblemente, mejorables. Usar sólo 100 clasificadores base, bastante simples, en un problema con 95 clases está muy lejos de las configuraciones que habitualmente se utilizan en boosting, donde no es raro encontrarse con combinaciones de 100 árboles de decisión. La cuestión es que la obtención y el uso de los clasificadores base tiene un coste, y por lo tanto el mejorar la precisión a base de añadir más clasificadores no siempre es una opción válida. Por ejemplo, también se pueden tratar los problemas multiclase en boosting utilizando la estrategia "uno contra uno", pero en ese caso se obtendría una combinación de literales para cada par de clases.

Por otro lado, sería posible aplicar el mismo esquema (esto es, obtención de metaatributos mediante boosting, su combinación mediante otro método de clasificación) con otros métodos en vez de SVM. En particular, sería posible utilizar de nuevo boosting con, por ejemplo, árboles de decisión de los metaatributos previamente obtenidos. De este modo se podrían obtener resultados más en línea con los obtenidos con SVM.

Hasta el momento, el uso de SVM ha sido bastante simple. Por un lado se han utilizado kernels muy simples, por otro no ha habido ningún intento de optimizar los diferentes parámetros del método, se usaron los valores por defecto. Es bastante prometedor obtener resultados tan buenos usando SVM de un modo tan simple, pero también indica que posiblemente ser podrían obtener incluso mejores resultados.

Agradecimientos.

A los encargados del mantenimiento del repositorio de KDD de la UCI [3]. A Davide Roverso, por la donación del conjunto de datos *Trace*. A Mineichi Kudo, Jun Toyama y Masaru Shimbo, por la donación del conjunto de datos *Vocales Japonesas*. A Mohammed Waleed Kadous, por la donación del conjunto de datos *Auslan*. A los desarrolladores de la biblioteca WEKA [16].

Referencias

[1] Claus Bahlmann, Bernard Haasdonk, and Hans Burkhardt. On-line handwriting recognition with support vector machines: A kernel approach. In *8th Int. Workshop on Frontiers in Handwriting Recognition (IWFHR)*, pages 49–54, 2002. ftp://ftp.informatik.uni-freiburg.de/papers/lmb/ba_ha_bu_iwfhr02.pdf.

[2] Nello Cristianini and John Shawe-Taylor. *An Introductio to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, 2000.

[3] S. Hettich and S. D. Bay. The UCI KDD archive [http://kdd.ics.uci.edu], 1999. Irvine, CA: University of California, Department of Information and Computer Science.

[4] Mohammed Waleed Kadous. *Temporal Classification: Extending the Classification Paradigm to Multivariate Time Series*. PhD thesis, The University of New South Wales, School of Computer Science and Engineering, 2002. http://www.cse.unsw.edu.au/~waleed/phd/.

[5] S.S. Keerthi, S.K. Shevade, C. Bhattacharyya, and K.R.K. Murthy. Improvements to platt's smo algorithm for svm classifier design. Technical Report CD-99-14, Control Division, Dept. of Mechanical and Production Engineering, National University of Singapore, 1999.

[6] Mineichi Kudo, Jun Toyama, and Masaru Shimbo. Multidimensional curve classification using passing-through regions. *Pattern Recognition Letters*, 20(11–13):1103–1111, 1999. http://ips6.main.eng.hokudai.ac.jp/research/pattern/paper/mine_prpVI.ps.

[7] J. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods – Support Vector Learning*. MIT Press, 1998.

[8] Juan J. Rodríguez, Carlos J. Alonso, and Henrik Boström. Learning first order logic time series classifiers: Rules and boosting. In *Principles of Data Mining and Knowledge Discovery: 4^{th} European Conference; PKDD 2000*, 2000.

[9] Juan J. Rodríguez, Carlos J. Alonso, and Henrik Boström. Boosting interval based literals. *Intelligent Data Analysis*, 5(3):245–262, 2001.

[10] Juan J. Rodríguez Diez and Carlos J. Alonso González. Applying boosting to similarity literals for time series classification. In *Proceedings of First International Workshop on Multiple Classifier Systems, MCS2000*, 2000.

[11] Davide Roverso. Multivariate temporal classification by windowed wavelet decomposition and recurrent neural networks. In *3^{rd} ANS International Topical Meeting on Nuclear Plant Instrumentation, Control and Human-Machine Interface*, 2000. http://www.ife.no/media/383_NPIC049.pdf.

[12] Gunnar Rätsch, Takashi Onoda, and Klaus R. Müller. Regularizing adaboost, 1999. http://ida.first.gmd.de/~raetsch/ps/RaeOnoMue98d.pdf.

[13] Robert E. Schapire. A brief introduction to boosting. In *16^{th} International Joint Conference on Artificial Intelligence (IJCAI-99)*, pages 1401–1406. Morgan Kaufmann, 1999.

[14] Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. In *11^{th} Annual Conference on Computational Learning Theory (COLT 1998)*, pages 80–91. ACM, 1998.

[15] Hiroshi Shimodaira, Ken ichi Noma, Mitsuru Nakai, and Shigeki Sagayama. Dynamic time-alignment kernel in support vector machine. In *NIPS 2001*, 2001.

[16] Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 1999.