CLASIFICACIÓN AUTOMÁTICA DE PERTURBACIONES DE SEÑALES DE TENSIÓN O CORRIENTE UTILIZANDO MÁQUINAS DE SOPORTE VECTORIAL (MSV)

ESTELA CAMPOS CORTÉS
ÁNGEL GABRIEL SUÁREZ DURÁN

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISICOMECÁNICAS
ESCUELA DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA Y DE TELECOMUNICACIONES
BUCARAMANGA
2007

CLASIFICACIÓN AUTOMÁTICA DE PERTURBACIONES DE SEÑALES DE TENSIÓN O CORRIENTE UTILIZANDO MÁQUINAS DE SOPORTE VECTORIAL (MSV)

ESTELA CAMPOS CORTÉS
ÁNGEL GABRIEL SUÁREZ DURÁN

Este proyecto es presentado como requisito para optar
por el título de ingeniero electrónico

Director:

Ing. Valdomiro Vega García

Codirector:

Ing. César Antonio Duarte Gualdrón

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FISICOMECÁNICAS
ESCUELA DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA Y DE TELECOMUNICACIONES
BUCARAMANGA
2007

A mis padres que desde la eternidad celebran mis logros

A mis hermanos que celebran conmigo

A Estela por su apoyo incondicional

ÁNGEL GABRIEL SUÁREZ DURÁN

A mis padres

Rosa Maria Cortés

Javier Campos

ESTELA CAMPOS CORTÉS

AGRADECIMIENTOS

Los autores expresan sus agradecimientos a:

Al MI(c) Valdomiro Vega García, director del proyecto, por su colaboración y guía, de gran importancia en la realización de este proyecto.

Al MPE. César Antonio Duarte Gualdrón codirector del proyecto, su continuo apoyo y orientación hicieron posible la culminación exitosa de este proyecto.

Al Dr. Gabriel Ordóñez Plata, por su colaboración y aportes al mejoramiento del producto final.

A los estudiantes German Morales y Álvaro Gómez pioneros en la investigación de las MSV en la Escuela E3T.

RESUMEN

TITULO
CLASIFICACIÓN AUTOMÁTICA DE PERTURBACIONES DE SEÑALES DE TENSIÓN O CORRIENTE UTILIZANDO MÁQUINAS DE SOPORTE VECTORIAL (MSV).

AUTORES
ESTELA CAMPOS CORTÉS
ÁNGEL GABRIEL SUÁREZ DURÁN

PALABRAS CLAVE
MÁQUINA DE SOPORTE VECTORIAL, HIPERPLANO CLASIFICADOR, ENTRENAMIENTO, VALIDACIÓN CRUZADA, BICLASIFICADOR, MULTICLASIFICADOR, KERNEL, PERTURBACIONES ELECTROMAGNÉTICAS, CALIDAD DE LA ENERGÍA

RESUMEN

En este trabajo se busca adaptar la herramienta de inteligencia artificial máquinas de soporte vectorial (MSV) en la solución del problema de clasificación de patrones de perturbaciones que afectan la calidad de las señales tensión o corriente en los sistemas de energía eléctrica. La clasificación se realiza mediante la herramienta computacional MSVToolbox1.0 utilizando patrones obtenidos a partir del análisis multiresolución tiempo-frecuencia de la transformada wavelet y de otras técnicas como el valor eficaz, valor medio y desviación estándar. Las pruebas fueron realizadas con el objetivo de encontrar los mejores parámetros de la MSV para este tipo de patrones en particular.

Inicialmente se hace una breve explicación de las diferentes técnicas de inteligencia artificial aplicadas en problemas de clasificación profundizando en la técnica MSV de la cual se presenta su principio de funcionamiento y características generales.

Posteriormente se presenta la herramienta computacional MSVToolbox1.0 desarrollada en este proyecto con sus respectivas especificaciones, principio de funcionamiento, generalidades, etc. De igual forma se consignan los resultados de las pruebas realizadas clasificando patrones de perturbaciones electromagnéticas, con el objetivo de encontrar los mejores parámetros de la MSV para este tipo de patrones en particular (tipo de *kernel*, método de optimización y método de descomposición). También se analizan los resultados de las simulaciones realizadas para la clasificación de diferentes tipos de datos (distribución bimodal, elipse, new_thyroid).

Trabajo de grado
Facultad de Ingenierías Físico-Mecánicas
Escuela de Ingenierías Eléctrica Electrónica y Telecomunicaciones
Ing. Valdomiro Vega García

SUMMARY

TITLE
AUTOMATIC CLASSIFICATION OF DISTURBANCES OF VOLTAGE OR CURRENT SIGNALS USING SUPPORT VECTOR MACHINES (SVM).

AUTHORS
ESTELA CAMPOS CORTÉS
ÁNGEL GABRIEL SUÁREZ DURÁN

KEY WORDS
SUPPORT VECTOR MACHINES (SVM). HYPERPLANE SEPARATING, TRAINING, CROSS VALIDATION, BINARY CLASIFIERS, MULTI CLASSIFIERS, KERNEL, ELECTROMAGNETIC DISTURBANCES, POWER QUALITY.

ABSTRACT

This project is intended to adapt support vector machines (SVM) for the solution of the classification problem of disturbance patterns that affect the quality of voltage or current signals in the electrical energy system. The classification is carried out by means of the computational tool MSVToolbox1.0 by using patterns which are obtained from the time-frequency multi-resolution analysis through Wavelet transform and other techniques such as the RMS value, average value and standard deviation. The objective of the classification tests is to find the best parameters of the SVM for this type of patterns in particular.

At first, a brief description of the different techniques of artificial intelligence is given. They are concisely described and SVM operation and characteristics are also presented.

Afterwards, the computational tool MSVToolbox1.0 developed on this project is explained. Its specifications such as operation, general characteristics, etc are presented. Also the results of the classifying patterns tests are exposed in order to find the best parameters of the SVM for this type of patterns in particular (type of kernel, method of optimization and method of decomposition). The results of classification using other data (bimodal distribution, ellipse, new-thyroid) are analyzed as well.

Trabajo de grado
Facultad de Ingenierías Físico-Mecánicas
Escuela de Ingenierías Eléctrica Electrónica y Telecomunicaciones
Ing. Valdomiro Vega García

CONTENIDO

LISTA DE TABLAS

LISTA DE FIGURAS

LISTA DE ANEXOS

INTRODUCCIÓN

El término "calidad de energía eléctrica" se emplea para describir la variación de la tensión, corriente, y frecuencia en el sistema eléctrico causado por diferentes eventos o fenómenos electromagnéticos. Estas variaciones ocurren principalmente debido a descargas atmosféricas, fallas en el sistema, conexión de cargas no lineales, maniobras (conexión y desconexión de bancos de condensadores) y otras operaciones que se realizan en el sistema. Además, debido al incremento en el número de dispositivos construidos con semiconductores, se presentan variaciones en magnitud, frecuencia y fase de la forma de onda sinusoidal y también desbalances en sistemas trifásicos. Estos fenómenos se clasifican en la norma [NTC 5000, 02] e [IEEE 1159, 95] de la siguiente manera: Transitorios electromagnéticos (tipo impulso y oscilatorio), armónicos, fluctuaciones de tensión (*Flicker*), huecos de tensión (*sags, dips*), sobretensiones (*swells*), desbalances de tensión, interrupciones, muescas de tensión (*notching*) y variaciones de frecuencia.

Los fenómenos electromagnéticos son causa de pérdidas económicas considerables, tanto para la industria eléctrica como para los usuarios, por lo cual se hace necesario la identificación, detección y clasificación de estos eventos o perturbaciones en el sistema eléctrico de manera rápida y automática. Considerando que actualmente no se cuenta con dichas estrategias (o algoritmos) para resolver este problema se están trabajando en la Escuela de ingenierías Eléctrica, Electrónica y de Telecomunicaciones de la Universidad Industrial de Santander en forma paralela diferentes proyectos que tienen como objetivo detectar, identificar y clasificar estas perturbaciones.

Este trabajo de grado busca específicamente diseñar una estrategia de clasificación automática utilizando la técnica de inteligencia artificial, máquinas de soporte vectorial (MSV) que permita clasificar patrones de fenómenos electromagnéticos, de igual forma se realizan pruebas con la MSV para determinar los mejores parámetros de la misma (tipo de *kernel*, método de optimización y método de descomposición) para la clasificación específica de este tipo de patrones. Esta tesis de grado se basa en el trabajo desarrollado por [Morales & Gómez, 05], donde se utiliza la técnica MSV, aplicada al problema de la localización de fallas en sistemas de distribución.

Como las MSV son herramienta de clasificación basada en inteligencia artificial generalmente implican un elevado costo computacional [Vapnik, 95]. Para aplicaciones de clasificación existen problemas debido a la cantidad de iteraciones. Al tener que utilizar grandes vectores de prueba en muchos casos las simulaciones

son inconclusas y se consumen todos los recursos del procesador dejándolo inoperable. Otro inconveniente es la no convergencia; en algunos casos esto puede ocurrir durante una simulación lo cual implica pérdida de tiempo de cómputo sin ningún resultado. Para resolver los problemas planteados se recomienda la metodología de entrenamiento aplicada en este trabajo que permite bajo tiempo de entrenamiento con altos resultados de exactitud. La herramienta computacional MSVToolbox1.0 permite al usuario hacer rápidos entrenamientos con cualquier tipo de datos, para posteriormente clasificar de forma fácil y efectiva.

El trabajo realizado en este proyecto de grado se encuentra estructurado en este documento de la siguiente forma:

En el capítulo 1 se presenta el estado del arte, donde se destacan algunos trabajos realizados con técnicas de inteligencia artificial para solucionar problemas de clasificación. También se presentan de forma resumida diferentes técnicas de inteligencia artificial, utilizadas en la solución de problemas de clasificación, dentro de las cuales se profundiza en la MSV, destacándola como una herramienta útil en trabajos e investigaciones anteriores. La validación cruzada en la MSV, se presenta al final de este capítulo donde se hace una explicación ejemplificada de su funcionamiento.

La descripción de las diferentes funciones utilizadas por la MSVToolbox1.0, así como su diagrama de flujo son expuestos en el capítulo 2.

En el Capítulo 3 se presenta la herramienta de clasificación automática denominada MSVToolbox1.0, en este apartado se consignan sus respectivas especificaciones, generalidades, funcionamiento y demás características.

Los resultados de entrenamiento y clasificación así como las diferentes pruebas realizadas con la MSVToolbox1.0 para determinar los parámetros de la MSV que mejor se ajustan al problema de clasificación de perturbaciones de las señales de tensión o corriente se presentan en el capítulo 4, de igual forma se pueden observar los resultados de las pruebas realizadas con diferentes tipos de datos con el fin de mostrar la utilidad de la MSVToolbox1.0.

Finalmente en el capítulo 5 se destacan las conclusiones y observaciones más referentes de este trabajo.

1. TÉCNICAS DE CLASIFICACIÓN AUTOMÁTICA

La **inteligencia artificial (IA)** se entiende como el modelado y la simulación de las actividades cognitivas complejas (percepción, memoria, solución de problemas, etc.) que caracterizan a los organismos avanzados y en particular a los seres humanos. Las técnicas de inteligencia artificial han tenido un gran auge en la solución de problemas de clasificación de datos o patrones. Las técnicas más utilizadas son: las redes neuronales artificiales, lógica *fuzzy* o lógica borrosa, los sistemas expertos basados en reglas, la técnica de decisión de *Bayes*, el análisis discriminante cuadrático (QDA), el análisis discriminante linear (LDA), los árboles de decisión y las máquinas de soporte vectorial, entre otras. A continuación se muestran las técnicas de inteligencia artificial que pueden llegar a utilizarse en la solución del problema de clasificación planteado en este proyecto [Bahamonde, 04].

1.1 ESTADO DEL ARTE

1.1.1 Trabajos desarrollados

La incorporación de agentes de decisión inteligente, redes neuronales, sistemas expertos, algoritmos genéticos, autómatas programables, MSV, etc., para optimización de sistemas de clasificación es una tendencia activa en el ambiente industrial de países con alto desarrollo tecnológico y con una gran inversión en investigación.

Actualmente la aplicación de una técnica de inteligencia artificial está ligada a la naturaleza del problema a resolver; por ello, es difícil dar una ponderación entre las diferentes técnicas, esto es posible sólo si se aplican al mismo problema. A continuación se muestran trabajos donde se emplean las técnicas de inteligencia artificial más utilizadas en problemas de clasificación.

Dentro de los métodos para la clasificación y síntesis de información sobresalen los métodos basados en *ecuaciones funcionales* (funciones que operan con otras funciones tomándolas como su argumento), entre ellos los trabajos de J. Aczél y C. Alsina (1986), A.A.Marley (1992), F.S. Roberts (1991) y los basados en *técnicas estadísticas* donde se encuentran los trabajos de R.C. Luo, M.G. Kay (1989), A.A.J. Marley (1992), K-C. Ng y B. Abramson (1992) y J.A. Benediktson y P.H. Swain (1992).

Aplicando la lógica borrosa a la clasificación [Robaina; Cruz, 00] presenta un algoritmo de clasificación para el diagnóstico de trastornos cardiovasculares capaz

de diagnosticar la posible presencia de hipertensión pulmonar y clasificarla en sus variantes: venocapilar, arteriolar, mixta e hipercinética.

Algunos estudios comparativos de técnicas de inteligencia artificial como [Muñoz, Ibargüen & López, 05] plantean el reconocimiento de 10 letras del alfabeto y 10 números manuscritos, se hace una comparación entre las redes neuronales y las máquinas de soporte. La red neuronal (perceptrón multicapa) tenía como funciones de transferencia, en cada una de sus capas (2 ó 3 capas ocultas que variaban el número de neuronas entre 10, 50 ó 100), funciones de tipo tangente sigmoidal; la red tenía 10 neuronas en la capa de salida, es decir una por cada tipo clase. En la máquina de soporte se utilizó como función núcleo (*kernel*), una función de tipo polinomial, probando polinomios de diferente orden. Los resultados obtenidos, muestran a la MSV por encima de la RNA, superándola con porcentajes del 97% contra un 93%.

Las Máquinas de Soporte Vectorial (MSV), en los últimos años, han dado buenos resultados en la clasificación y reconocimiento de patrones en general. Su potencialidad supera a otras técnicas empleadas como las redes neuronales artificiales (RNAs), los árboles de clasificación (CART), las técnicas bayesianas, etc. Por ejemplo, a diferencia del método bayesiano, las MSV no requieren ningún tipo de hipótesis sobre la densidad de probabilidad de los datos. Frente a las RNAs ofrecen la ventaja de reducir la cantidad de parámetros que es necesario especificar. Caso contrario ocurre con las redes neuronales, donde primero se escoge el tipo de red a entrenar, luego la función de activación, seguido el número de capas y el número de neuronas por capas, es decir, que la búsqueda se vuelve bastante compleja, y su implementación requiere gran esfuerzo [Burges; Schölkopf & Smola, 99]

En el proyecto de grado de la Universidad Industrial de Santander [Morales & Gómez, 05], se realiza el estudio e implementación de una herramienta basada en máquinas de soporte vectorial aplicada a la localización de fallas en sistemas de distribución eléctrica, en el cual se presentó a la MSV como la mejor técnica de inteligencia artificial aplicable, por lo tanto tal proyecto es punto de partida para este trabajo.

1.1.2 Artículos IEEE

A continuación se referencian artículos IEEE donde se muestra la aplicabilidad de técnicas de inteligencia artificial en problemas de clasificación.

En [Wang, 01] se propone un algoritmo de identificación y clasificación de perturbaciones eléctricas, este nuevo método presentado se basa en el empleo del plano de ambigüedad tiempo-frecuencia, utilizando kernel bifuncional discriminante así como las redes neuronales, mostrando muy buenos resultados en la clasificación de 6 categorías de señales de disturbio. Esta combinación de

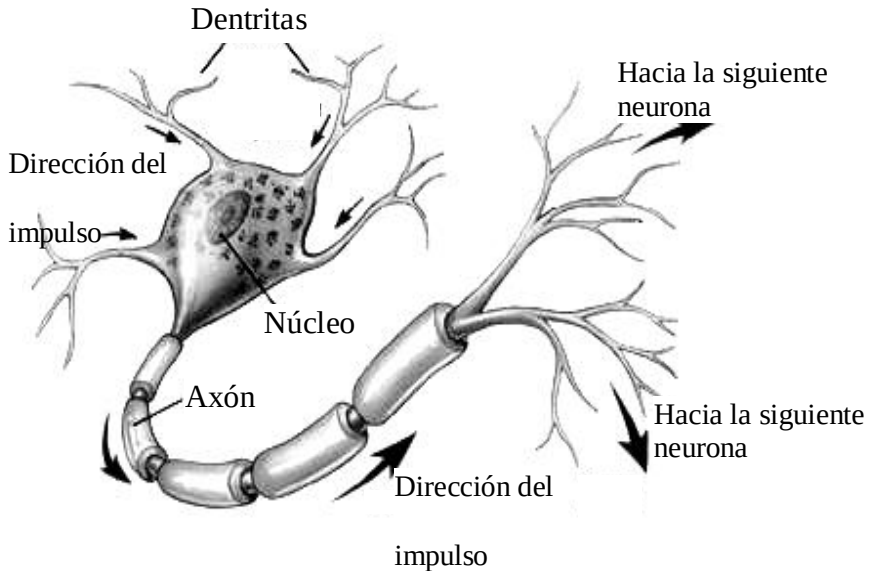
los métodos también aporta la posibilidad de discriminar automáticamente y exactamente disturbios del tipo transitorio.

[Murphey & Zhihang, 03] investiga los algoritmos eficientes y eficaces para el entrenamiento de las SVMs con grandes cantidades de datos. Descomponiendo el problema del aprendizaje en dos etapas. En la primera etapa se desarrolla un algoritmo que utiliza una secuencia de subconjuntos pequeños de datos de entrenamiento para seleccionar los parámetros del kernel. En la segunda etapa se desarrolla un algoritmo que genera un sistema de vectores de soporte confiable para luego de la clasificación, usando un subconjunto pequeño de los datos.

1.2 REDES NEURONALES

Las Redes Neuronales Artificiales (RNA o ANN del inglés *Artificial Neural Networks*) fueron originalmente una simulación abstracta de los sistemas nerviosos biológicos, formados por un conjunto de unidades llamadas "neuronas" o "nodos" conectadas unas con otras, [Hilera & Martinez, 95]. Estas conexiones tienen una gran semejanza con las dendritas y los axones en los sistemas nerviosos biológicos, véase Figura 1.

Figura 1. Representación de una neurona biológica [GNU, 03]

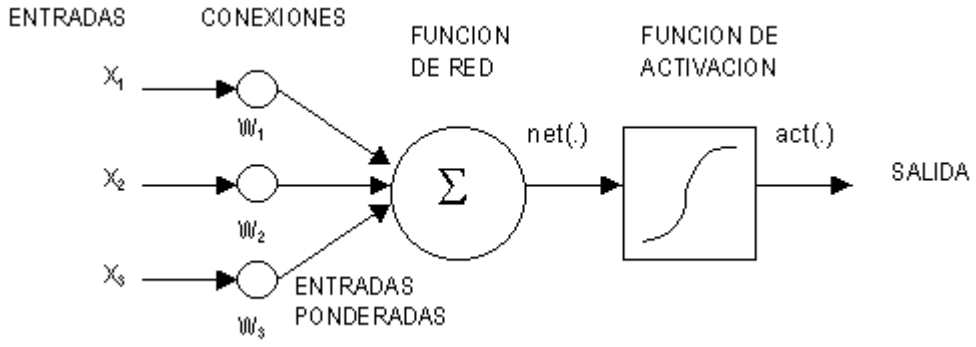


La gran diferencia del empleo de las redes neuronales en relación con otras aplicaciones de la computación radica en que no son algorítmicas, esto es no se programan haciéndoles seguir una secuencia predefinida de instrucciones. Las RNA generan ellas mismas sus propias "reglas", para asociar la respuesta a su entrada; es decir, aprende por ejemplos y de sus propios errores [Hilera & Martínez, 95].

1.2.1 Composición de las redes neuronales artificiales

Las neuronas se modelan mediante **unidades de proceso**. Cada unidad de proceso se compone de una red de conexiones de entrada**;** una función de red(propagación), encargada de computar la entrada total combinada de todas las conexiones; un núcleo central de proceso, encargado de aplicar la función de activación; y la salida, por dónde se transmite el valor de activación a otras unidades (ver Figura 2) [Wassermann, 89].

Figura 2. Unidad de proceso típica de una RNA [Estadístico, 01]



- **La función de propagación o de red** calcula el valor de base o entrada total a la unidad, generalmente como simple suma ponderada de todas las entradas recibidas, es decir, de las entradas multiplicadas por el peso o valor de las conexiones (equivale a la combinación de las señales excitatorias e inhibitorias de las neuronas biológicas).

- **La función de activación** produce un nuevo estado de activación en una neurona a partir del estado que existía y la combinación de las entradas con los pesos de las conexiones. Esta función puede ser una función escalón, función lineal, función sigmoidal o una función gausiana, éstas se suelen distinguir entre funciones lineales, en las que la salida es proporcional a la entrada; funciones de umbral, en las cuales la salida es un valor discreto (típicamente binario 0/1) que depende de si la estimulación total supera o no un determinado valor de umbral; y funciones no lineales, no proporcionales a la entrada. [Quinlan, 91], [Rumelhart & McClelland, 88].

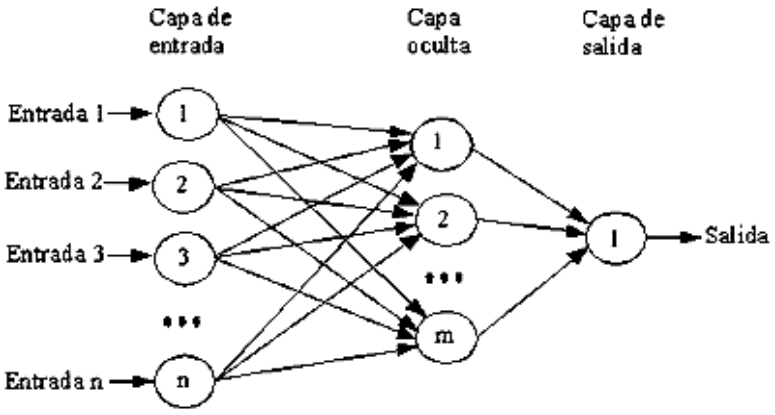
- **Las Conexiones ponderadas** hacen el papel de las conexiones sinápticas, el peso de la conexión equivale a la fuerza o efectividad de la sinápsis. La existencia de conexiones determina si es posible que una unidad influya sobre otra, el valor de los pesos y el signo de los mismos definen el tipo (excitatorio/inhibitorio) y la intensidad de la influencia.

- **La salida** obtiene respuesta de la neurona en función de la activación de la misma, aunque normalmente no se aplica más que la función identidad y se toma como salida el valor de activación. El valor de salida cumpliría la función de la tasa de disparo en las neuronas biológicas.

1.2.2 Estructuras y formas de interconexión

En el diseño de una red se establece la conexión de las unidades entre sí y se determina adecuadamente los pesos de las conexiones. Lo más usual es disponer las unidades en forma de capas, pudiéndose hablar de redes de una, de dos o de más de dos capas, las llamadas redes multicapa. Lo más usual es disponer de tres o más capas: la primera capa, **capa de entrada**, actúa como almacenador de entrada, almacenando la información bruta suministrada a la red o realizando un sencillo pre-proceso de la misma; la **capa de salida** actúa como interfaz o almacenador de salida, almacenando la respuesta de la red para que pueda ser leída; y **las capas ocultas** son las capas intermedias, encargadas de extraer, procesar y memorizar la información [Wassermann, 89].

Figura 3. Modelo de red en cascada de 3 capas



http://es.wikipedia.org/wiki/Red_neuronal_artificial

1.2.3 Modo de operación (aprendizaje)

En cualquier red neuronal cabe distinguir la fase o proceso de **aprendizaje**, opcionalmente una fase de **prueba** (prueba**)** y la fase de **aplicación** (recall).

El aprendizaje consiste en la presentación de patrones a la red, y la subsiguiente modificación de los pesos de las conexiones siguiendo alguna regla de aprendizaje que trata de optimizar su respuesta, generalmente mediante la minimización del error o la optimización de alguna "función de energía". El modo de aprendizaje más sencillo consiste en la presentación de patrones de entrada junto a los patrones de salida deseados (targets) para cada patrón de entrada, por eso se llama ***aprendizaje***

supervisado. Si no se le presentan a la red los patrones de salida deseados, se dice que se trata de ***aprendizaje no supervisado***, ya que no se le indica a la red que resultados debe dar, sino que se le deja seguir alguna regla de auto-organización. Un tercer tipo de aprendizaje es el ***aprendizaje reforzado***, en este caso el supervisor se limita a indicar si la salida ofrecida por la red es correcta o incorrecta, pero no indica que respuesta debe dar [Wassermann, 89].

Aunque la capacidad de aprendizaje es esencial y casi definitoria se han desarrollado modelos que no requieren modificar los pesos de las conexiones, sino que son precalculados y pre-establecidos antes de presentarle patrones a la red, como en las redes de Hamming y en redes de Hopfield [Fausett, 94].

Tabla 1. Redes neuronales según el tipo de aprendizaje

Tipo de aprendizaje		
Fijo	**No supervisado**	**Supervisado**
Red de Hamming	Mapa de características	Basadas en decisión
Red de Hopfield	Aprendizaje competitivo	Perceptrón
		ADALINE (LMS)
		Perceptrón multicapa
		Modelos temporales dinámicos
		Modelos ocultos de Markov

http://www.gc.ssr.upm.es/inves/neural/ann2/concepts/taxonomy.htm

Los pasos a seguir para el aprendizaje supervisado son:

- Aleatorizar los pesos de todas las conexiones.
- Seleccionar un par de entrenamiento.
- Presentar el patrón de entrada y calcular la salida de la red.
- Calcular el error o discrepancia entre la salida obtenida y la deseada.
- Aplicar la regla de aprendizaje.

1.3 LÓGICA DIFUSA

"En la lógica clásica una proposición sólo admite dos valores: puede ser verdadera o falsa, por eso se dice que la lógica usual es bivalente o binaria. La lógica multivaluada incluye sistemas lógicos que admiten varios valores de verdad posibles. La lógica difusa (borrosa o, en inglés *fuzzy logic*) es una de ellas, que se caracteriza por querer cuantificar esta incertidumbre por ejemplo, si P es una proposición, se le puede asociar un número V (P) en el intervalo [0, 1] Tal que:

Si v (P) = 0, P es falso
Si v (P) = 1, P es verdadero
La veracidad de P aumenta con v (P)

Salta a la vista la semejanza con la teoría de la probabilidad. Esta simple idea nació en un artículo titulado "*Fuzzy Sets*" (Conjuntos difusos) [Zadeh, 65]. La lógica difusa permite representar de forma matemática conceptos o conjuntos imprecisos, tales como frío, calor, alto, bajo, mucho, poco etc.

Así, por ejemplo, una persona que mida 2 metros es claramente una persona alta (es alta con grado 1) y una persona que mida 1 metro no es una persona alta en absoluto (es alta con grado 0). De forma intermedia podemos decir que una persona que mida 1,82 es alta con grado 0,75 indicando que es "bastante alta". De este ejemplo puede extraerse fácilmente que la lógica y la teoría de conjuntos son isomorfismos matemáticos [Kaufmann, 82].

En la teoría de conjuntos difusos se definen también las operaciones de unión, intersección, diferencia, negación o complemento... y otras operaciones sobre conjuntos.

Subconjunto difuso o parte difusa. Es un conjunto que puede contener elementos de forma parcial, es decir que la propiedad $x \in A$ puede ser cierta, falsa o solamente posible. Se mide esta posibilidad de pertenecer (o pertenencia) con un número $\mu_A(x)$ entre 0 y 1, llamado **grado de pertenencia** de x a A. Si es 0, x no pertenece a A, si es 1, entonces $x \in A$, totalmente, y si $0 < \mu_A(x) < 1$, x pertenece a A de una manera parcial. Un subconjunto A de B es por lo tanto caracterizado por esta función matemática de pertenencia μ_A, de B hacia [0; 1].Es preciso fijar el conjunto B para definir la función μ_A que a su vez define A. Por eso se habla de subconjunto difuso y no de conjunto difuso [Wikipedia, 06].

La lógica difusa o lógica borrosa se utiliza para la resolución de una variedad de problemas, principalmente los relacionados con control de procesos industriales complejos y sistemas de decisión en general. Los sistemas basados en lógica difusa imitan la forma en que toman decisiones los humanos, con la ventaja de que son más rápidos. Estos sistemas son generalmente robustos y tolerantes a imprecisiones y ruidos en los datos de entrada" [Wikipedia, 06].

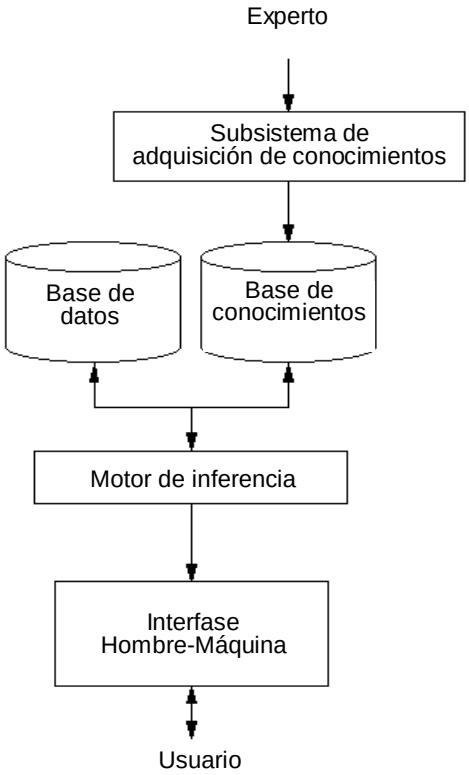
1.4 SISTEMAS EXPERTOS

"Un sistema experto (SE) es una aplicación informática que simula el comportamiento de un experto humano, en cuanto a que es capaz de decidir, aunque sea en un campo restringido. Para esto, se debe tener en cuenta que la

principal característica del experto humano viene a ser el conocimiento, por consiguiente, un sistema experto debe ser capaz de representar ese conocimiento profundo con el objetivo de utilizarlo para resolver problemas, justificar su comportamiento e incorporar nuevos conocimientos" [Ibarreta, 06].

Los componentes de un sistema experto se muestran en la Figura 4 y se definen a continuación:

Figura 4. Modelo de un sistema experto



http://www.monografias.com/trabajos23/sistemas-expertos/sistemas-expertos.shtml

- **Subsistema de adquisición de conocimientos,** permite que se puedan añadir, eliminar o modificar elementos de conocimiento (reglas).
- **Base de conocimientos,** contiene las reglas o el conocimiento especializado extraído del experto.
- **Base de datos o base de hechos,** es una parte de la memoria del ordenador que contiene conocimiento sobre el caso concreto en que se trabaja.
- **Motor de inferencia,** es el "supervisor", un programa que extrae conclusiones a partir de los datos simbólicos que están almacenados en las bases de hechos y de conocimiento.

- **Interfaz Hombre-Máquina, l**os resultados finales y la forma en que se obtienen se expresan a través de esta interfaz.
- **Componente de explicación,** justifica y explica el análisis completo del problema y las soluciones propuestas. A pesar de su importancia, aún no se ha podido cumplir de manera óptima con sus requisitos. [Canca, 06].

Un sistema experto es muy eficaz cuando tiene que analizar una gran cantidad de información, interpretándola y proporcionando una recomendación a partir de la misma [Canca, 06]. Entre las principales ventajas, se pueden mencionar que estos codifican y tratan el conocimiento de un experto, pudiendo alcanzar el nivel de prestaciones de un experto humano y también la posibilidad de trazar el razonamiento seguido por el sistema experto. Entre las desventajas se encuentra la extracción del conocimiento siendo el problema más complejo que se les plantea a los ingenieros de conocimientos, también la incapacidad de los SE de reconocer un problema para el que su propio conocimiento es inaplicable o insuficiente. Además, las interfaces con usuarios no son lo suficientemente amigables, falta personal competente para investigar y desarrollar aplicaciones y el costo es bastante alto.

1.5 CLASIFICADOR BAYESIANO

La teoría de decisiones de Bayes constituye la base de los métodos estadísticos de reconocimiento de patrones [Cruz, 04]. Para describir sus principios básicos considérese que se tienen K clases, denominadas w_k y un vector de parámetros de entrada *X*; además se define:

- $P(w_k/X)$: probabilidad de que la clase correcta sea w_k dado un vector de parámetros como entrada. Esta probabilidad se denomina probabilidad a *posteriori*, ya que puede ser estimada únicamente después de que los datos han sido observados.
- $P(w_k)$: probabilidad de la clase w_k. Se conoce como probabilidad a *priori*, porque puede ser evaluada antes de que *X* sea observado.
- P(X/Wk): es la distribución de probabilidad de *X* condicionada a una clase particular.

Empleando la regla de Bayes, la probabilidad *a posteriori* $P(w_k/X)$ puede calcularse con base en la probabilidad *a priori* y la probabilidad condicional $P(X/w_k)$ como se presenta en la ecuación (1.1).

$$P(w_k / X) = \frac{P(X / w_k) \cdot P(w_k)}{P(X)} \tag{1.1}$$

Como regla de decisión para un clasificador, puede establecerse que la clase correcta es la que presenta la mayor probabilidad a *posteriori;* es decir, el clasificador asignará un vector *X* a la clase k, si para todo $i \neq k$:

$$g_k(X) > g_i(X) \tag{1.2}$$

Con:

$$g_k(X) = P(w_k / X) = \frac{P(X / w_k) \cdot P(w_k)}{P(X)} \tag{1.3}$$
$$g_k(X) = P(X / w_k) \cdot P(w_k)$$

Esta simplificación puede hacerse cuando *P(X)* es constante para todas las clases.

Una comparación del desempeño de esta técnica respecto a las RNA en la clasificación de perturbaciones de señales de tensión o corriente se realizó en [Vega, Duarte & Ordóñez, 06]. Las siguientes son las características de la red neuronal: red perceptrón milticapa, función de desempeño feedforward, función de activación tangente sigmoidal, 3 capas ocultas y una de salida y [8 6 4 1] neuronas por capa. El resultado se presenta en la Tabla 2.

Tabla 2. Porcentajes de acierto de las técnicas Bayes y RNA en la clasificación de perturbaciones de señales de tensión o corriente.

Perturbación	Porcentaje de acierto	
	BAYES	**RNA**
Sag	92	93
Swell	51	97
Flicker	85	86
Transitorio Oscilatorio	0	79
Armónico	0	95

[Vega , Duarte & Ordoñez, 06]

Las perturbaciones clasificadas fueron muestreadas a 128 muestras por ciclo de 60[Hz]. Los patrones utilizados son el resultado de la extracción de parámetros utilizando la transformada *wavelet* discreta y una técnica de cálculo de diferencias de energía [Vega & Duarte & Ordoñez, 06]. De esta tabla se deduce que el clasificador Bayesiano (clasificador lineal) no es adecuado para la clasificación de estos datos en particular, haciendo a la RNA como la más optima para este caso.

Es posible mejorar los resultados de la Tabla 2 utilizando una herramienta novedosa conocida como máquina de soporte vectorial (MSV), que se describe a continuación en el capítulo 2.

2. MÁQUINAS DE SOPORTE VECTORIAL (MSV)

El análisis de la técnica de inteligencia artificial máquinas de soporte vectorial que se describe en este capítulo, está basado en estudios realizados por [Morales & Gomez, 05], como también en los textos de [Burges; Schölkopf & Smola, 99], [Burges vol. 2, 98], complementados por investigaciones de los autores, incorporando una explicación ejemplificada del funcionamiento de la MSV así como una clara y didáctica explicación de la validación cruzada en la MSV.

Las Máquinas de Soporte Vectorial (MSV) son una nueva técnica de inteligencia artificial que ha mostrado buenos resultados en problemas de clasificación [Burges; Schölkopf & Smola, 99], [Burges vol. 2, 98]. Esta técnica utiliza el aprendizaje a través de ejemplos, es decir, estima una dependencia desconocida entrada-salida de un sistema mediante un número limitado de observaciones [Vapnik, 95], [Vapnik, 98], [Cherkassky & Mulier, 98].

Las MSV realizan la clasificación mediante un hiperplano que separa dos clases (biclasificación). Para clasificar más de dos clases (multiclasificación) se implementan dos o más máquinas biclasificadoras.

2.1 TÉCNICA DE APRENDIZAJE DE LAS MSV

Sea **X** el vector de entrenamiento de n datos N dimensionales $\vec{x}_i$ pertenecientes a dos clases con sus respetivas etiquetas $\vec{y}_i$,

$$\vec{x}_i \in \Re^N \qquad \text{y} \qquad y_i \in \{1, -1\} \tag{2.1}$$

Con estos datos se busca estimar una función f tal que para una entrada en $\Re^N$ produzca una salida en {±1},

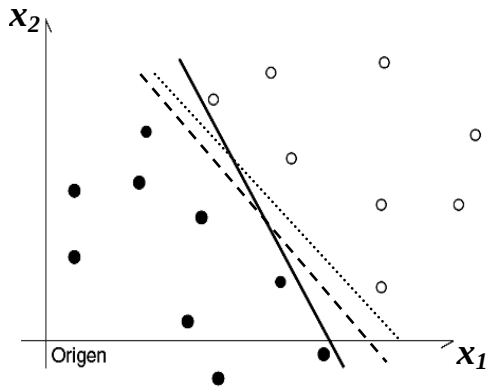
$$F : \Re^N \rightarrow \{1,-1\} \tag{2.2}$$

Para que así pueda clasificar correctamente un nuevo dato $(\vec{x}, y)$ (téngase en cuenta que $y = f(\vec{x})$ para este nuevo dato, y es generado con la misma distribución de probabilidad P($\vec{x}_i, y_i$) de los datos de entrenamiento).

2.2 EL HIPERPLANO CLASIFICADOR ÓPTIMO

Los clasificadores de soporte vectorial están basados en hiperplanos que separan los datos de entrenamiento en dos subgrupos de manera que aquellos con igual etiqueta queden al mismo lado del hiperplano. En un caso de dos dimensiones los hiperplanos se representan por líneas rectas como se puede apreciar en la Figura 5.

Figura 5. Hiperplanos que separan correctamente los datos.

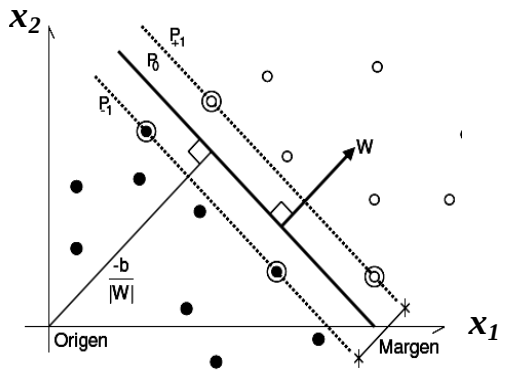


A Tutorial on Support Vector Machines for Pattern Recognition [Burges vol. 2, 98]

En medio de todos los posibles planos de separación entre las dos clases existe un único *hiperplano de separación óptimo* (OSH), de forma que la distancia entre el hiperplano óptimo y el patrón de entrenamiento más cercano sea máxima, con la intención de forzar la generalización de la máquina de aprendizaje [Burges, 98], [Schölkopf & Smola, 02]. Esto significa encontrar un w y b tal que

$$g(\vec{x}) = (\vec{w} \cdot \vec{x}) + b = 0 \qquad (2.3)$$

Figura 6. Hiperplano de separación óptimo (OSH)



A Tutorial on Support Vector Machines for Pattern Recognition [Burges vol. 2, 98].

2.3 MAXIMIZACIÓN DEL MARGEN

Para maximizar el margen (Figura 6). Se proponen dos planos paralelos al plano representado por la ecuación (2.3) que contienen los puntos más cercanos al OSH, definiendo como $1/\|\vec{w}\|$ la distancia entre el OSH y el punto más cercano al mismo. Las ecuaciones de dichos planos son:

$$p+1 \ : \ (\vec{w}\cdot\vec{x}_i)+b=+1 \tag{2.4}$$

$$p-1 \ : \ (\vec{w}\cdot\vec{x}_i)+b=-1 \tag{2.5}$$

Siendo el margen la distancia perpendicular entre (2.4) y (2.5) y recordando que la distancia perpendicular que hay de cualquier punto a un plano es $\frac{(\vec{w}\cdot\vec{x}_i)+b}{\|\vec{w}\|}$, se obtiene:

$$(\vec{w}\cdot\vec{x}_{+1})+b - (\vec{w}\cdot\vec{x}_{-1})+b = (+1) - (-1)$$

$$\vec{w}\cdot(\vec{x}_{+1}-\vec{x}_{-1}) = 2$$

$$\frac{\vec{w}}{\|\vec{w}\|}\cdot(\vec{x}_{+1}-\vec{x}_{-1}) = \frac{2}{\|\vec{w}\|} \tag{2.6}$$

Y la distancia perpendicular del OSH al origen:

$$(\vec{w}\cdot\vec{x}_i)+b=0$$

$$\vec{x}_i = 0$$

$$\frac{(\vec{w} \cdot \vec{x}_i) + b}{\|\vec{w}\|} = \frac{b}{\|\vec{w}\|} \tag{2.7}$$

Donde los planos y el margen están representados por las ecuaciones (2.4), (2.5), (2.6) y (2.7) se muestran en la Figura 6.

Debe notarse que entre (2.4) y (2.5) no existen datos de entrenamiento, todos los datos deben cumplir:

$$(\vec{w} \cdot \vec{x}_i) + b \geq +1 \; para \;\; y_i = +1 \tag{2.8}$$

$$(\vec{w} \cdot \vec{x}_i) + b \leq -1 \;\; para \;\; y_i = -1 \tag{2.9}$$

Luego la función decisión $f_{w,b}(\vec{x}_i) = yi$, puede definirse como el signo que resulta de evaluar un dato en la ecuación del OSH (2.3):

$$f_{w,b}(\vec{x}_i) = sign\,(g(\vec{x}_i)) = sign\,(\vec{w} \cdot \vec{x}_i) + b) \tag{2.10}$$

Combinando (2.8) y (2.9) se obtiene:

$$y_i\,(\vec{w} \cdot \vec{x}_i + b) \geq 1 \tag{2.11}$$

Si existe un hiperplano que satisfaga (2.11), se dice que los datos son linealmente separables. Para encontrar el OSH se debe maximizar el margen (2.6) teniendo en cuenta la restricción (2.11). Equivalente a resolver el siguiente problema:

$$\min_{w} \frac{1}{2}(\vec{w} \cdot \vec{w}) \tag{2.12}$$

$$Sujeto \;\; a \qquad y_i\,(\vec{w} \cdot \vec{x}_i + b) \geq 1, \;\; \forall i \tag{2.13}$$

La función (2.12) es llamada función objetivo, y junto con (2.13) es llamado un problema de optimización cuadrático con restricciones. Los problemas de este tipo son tratados introduciendo el método de multiplicadores de Lagrange.

Para esta clase de problemas con restricciones, se introducen los multiplicadores de Lagrange $\alpha_i \geq 0$ (uno por cada restricción). Para las restricciones de la forma $R_i \geq 0$ cada restricción es multiplicada por α_i (un multiplicador de Lagrange positivo) y se restan de la función objetivo, para así formar la función de Lagrange [Burges, 1998]:

$$L(\vec{w},b,\vec{\alpha}) = \frac{1}{2}\|\vec{w}\|^2 - \sum_{i=1}^{n}\alpha_i \left[y_i \left(\vec{w}\cdot\vec{x}_i + b \right) - 1 \right] \qquad (2.14)$$

La función de Lagrange (2.14) debe ser minimizada con respecto a las variables primarias w y b, y maximizada sobre los α_i (en otras palabras encontrar el punto de silla) [Schölkopf & Smola, 02]. Para el caso de $y_i(\vec{w}.\vec{x}_i + b) - 1 > 0$, el correspondiente α_i debe ser cero, debido a que este es el valor de α_i es el que maximiza (2.14), los $\alpha_i \neq 0$ son para el caso en que $y_i(\vec{w}.\vec{x}_i + b) - 1 = 0$ (estos son los patrones de entrenamiento que quedan sobre los planos paralelos (2.4) y (2.5) al OSH). Este último enunciado corresponde a las condiciones de *Karush-Kuhn-Tucker* (condiciones complementarias de optimización) [Kuhn & Tucker, 51]:

$$\alpha_i \left[y_i \left(\vec{w}\cdot\vec{x}_i + b \right) - 1 \right] = 0 \qquad \forall i \qquad (2.15)$$

$$\frac{\partial}{\partial w} L(\vec{w},b,\vec{\alpha}) = 0 \qquad \Rightarrow \qquad \vec{w} = \sum_{i=1}^{n}\alpha_i y_i \vec{x}_i \qquad (2.16)$$

$$\frac{\partial}{\partial b} L(\vec{w},b,\vec{\alpha}) = 0 \qquad \Rightarrow \qquad \sum_{i=1}^{n}\alpha_i . y_i = 0 \qquad (2.17)$$

La solución de $\vec{w}$ en (2.16) queda en función de un subconjunto de patrones de entrenamiento, aquellos cuyo multiplicador de Lagrange es diferente de cero, es decir, el soporte de $\vec{w}$ está en los patrones de entrenamiento más cercanos al OSH. De aquí el nombre de Máquinas de Soporte Vectorial.

Reemplazando (2.16) y (2.17) en (2.14), se eliminan las variables primarias $\vec{w}$ y b llegando así al problema de *optimización dual de Wolfe*, el cual es el problema que se resuelve en la práctica:

$$\max_{\alpha}\left[\sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j (\vec{x}_i\cdot\vec{x}_j) \right] \qquad (2.18)$$

$$\text{Sujeto a} \quad \alpha_i \geq 0, \quad \forall i \qquad \text{y} \qquad \sum_{i=1}^{n}\alpha_i y_i = 0 \qquad (2.19)$$

Este problema puede ser resuelto con métodos de programación cuadrática estándar [Burges, 98]. Una vez obtenido el vector $\vec{\alpha}$ se pueden obtener los parámetros $\vec{w}$ y b con (2.16) y (2.15) respectivamente:

$$\vec{w} = \sum_{i=1}^{n}\alpha_i y_i \vec{x}_i$$

$$b = y_i - \vec{w} \cdot \vec{x}_i$$

Ahora la ecuación del OSH y la función de decisión pueden ser escritas como:

$$g(\vec{x}) = \sum_{i=1}^{n} \alpha_i y_i (\vec{x}_i \vec{x}) + b \quad (2.20)$$

$$f(\vec{x}) = sign\left(\sum_{i=1}^{n} \alpha_i y_i \vec{x}_i \cdot \vec{x} + b \right) \quad (2.21)$$

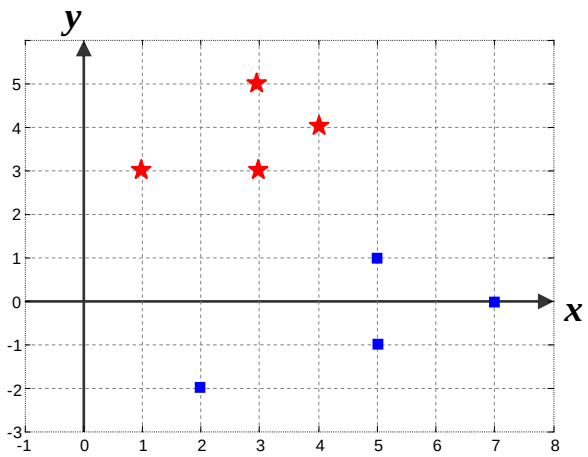
Se trabaja con la teoría de Lagrange por dos razones fundamentales [Burges, 1998]:

- Las restricciones (2.13) fueron remplazadas y quedaron en términos de α_i (2.19), lo cual es mucho más fácil de manejar.
- En la reformulación del problema, los datos de entrenamiento $\vec{x}_i$ sólo aparecen en forma de productos punto entre ellos mismos (ecuaciones (2.18), (2.20) y (2.21)), lo cual es conveniente.

2.4 EJEMPLO DE UNA MÁQUINA DE SOPORTE VECTORIAL LINEAL

A continuación se presenta un ejemplo que muestra el funcionamiento de una máquina de soporte vectorial lineal. Sea el conjunto de datos: (1,3), (3,3), (3,5), (4,4) pertenecientes a una clase y el conjunto de datos (2,-2), (5,-1), (5,1) y (7,0) pertenecientes a otra. En la Figura 7 se presenta la ubicación de estos puntos en el plano cartesiano.

Figura 7. Ejemplo de separación de dos clases



Elaborado por los autores

- El primer paso que se debe realizar, es la asignación de una etiqueta a cada clase; por ejemplo a los datos anteriores (datos de entrenamiento) se les asignará una etiqueta de la siguiente forma: para el primer conjunto de datos (estrellas rojas) la etiqueta +1 y para el segundo conjunto (cuadros azules) la etiqueta -1. Los datos quedaran organizados como se muestra en la Tabla 3.

Tabla 3. Datos etiquetados

$\vec{x}_i$		$\vec{y}_i$
1	3	1
3	3	1
3	5	1
4	4	1
2	-2	-1
5	-1	-1
5	1	-1
7	0	-1

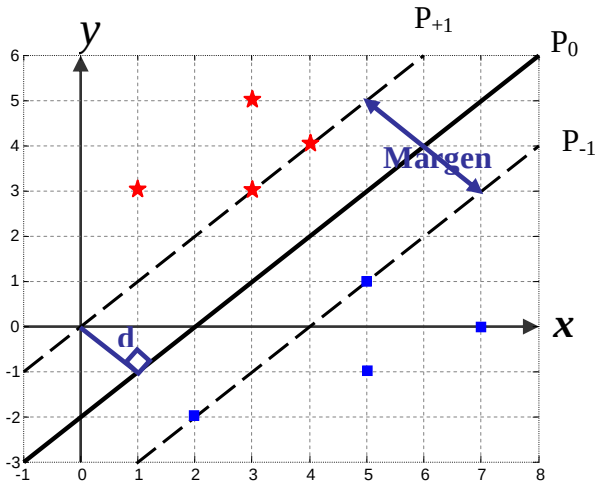
Elaborado por los autores

- El segundo paso, es entrenar la MSV. El entrenamiento consiste en hallar una función que separe las dos clases (OSH, que para este caso es una recta).

En este ejemplo, los datos están en dos dimensiones y son bien diferenciados. Es posible hallar el OSH de forma gráfica, como se muestra en la Figura 8, teniendo en cuenta que la distancia de los patrones de entrenamiento de cada clase al OSH debe ser máxima.

Después de tener la mejor opción del plano, es decir el OSH, se genera dos líneas paralelas a esté y se desplazan hasta encontrarse con los primeros parámetros de entrenamiento. Nota: si es el mejor hiperplano la distancia de los primeros patrones de entrenamiento a cada lado del plano debe ser la misma (véase Figura 8).

Figura 8. Hiperplano óptimo de separación



Elaborado por los autores

En la Figura 8, **d** es la distancia perpendicular del OSH al origen.

Ahora se hallan las ecuaciones de estos planos:

$$P_0: \; y = x - 2 \qquad \text{reescribiendo:} \qquad -x + y + 2 = 0 \tag{2.22}$$

$$P_{+1}: \; y = x \qquad \text{reescribiendo:} \qquad -x + y + 1 = +1 \tag{2.23}$$

$$P_{-1}: y = x - 4 \qquad \text{reescribiendo:} \qquad -x + y + 3 = -1 \tag{2.24}$$

La ecuación (2.22) es la ecuación del OSH pero ésta debe ser modificada para que $\vec{w}$ y b cumplan con los requisitos de magnitud requeridos en (2.6) y (2.7).

Expresando la ecuación (2.22) de la forma $P_0: \; w \cdot x_i + b = 0$:

$$\begin{bmatrix} -1 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + 2 = 0 \tag{2.25}$$

De aquí se obtiene $\vec{w} = \begin{bmatrix} -1 & 1 \end{bmatrix}$ y $b = 2$.

De la Figura 8 se halla geométricamente el valor del margen ($2\sqrt{2}$) y reemplazando en la ecuación (2.6) se obtiene:

$$2\sqrt{2} = \frac{2}{\|\vec{w}\|}$$
$$\|\vec{w}\| = \frac{1}{\sqrt{2}} \tag{2.26}$$

Tomando la distancia perpendicular del OSH al origen (d= $\sqrt{2}$), Figura 8, y utilizando la ecuación (2.7) se tiene:

$$\frac{b}{\|\vec{w}\|} = \sqrt{2}$$
$$b = 1 \tag{2.27}$$

Ahora se debe modificar la magnitud de $\vec{w}$ para que cumpla con las condiciones exigidas, normalizando $\vec{w}$ y multiplicando por la magnitud encontrada en (2.26) se tiene:

$$\vec{w} = \frac{1}{\sqrt{2}} \cdot \frac{1}{\sqrt{2}} \cdot [-1,1]$$
$$\vec{w} = \left[-\frac{1}{2}, \frac{1}{2}\right] \tag{2.28}$$

Reemplazando los nuevos valores de $\vec{w}$ y b en (2.25) se obtiene la ecuación del OSH:

$$P_0 : \left(\begin{bmatrix} -\frac{1}{2} & \frac{1}{2} \end{bmatrix} \bullet \begin{bmatrix} x \\ y \end{bmatrix} + 1\right) = 0 \tag{2.29}$$

Nótese que la ecuación (2.22) y (2.29) representan el mismo plano, solo que esta última ha sido escalada por la constante $\frac{1}{2}$.

Reescribiendo las ecuaciones (2.23) y (2.24) y multiplicándolas por la misma constante se tiene:

$$P_{+1} : \quad -\frac{1}{2}x + \frac{1}{2}y + 1 = 1 \qquad P_{+1} : \left(\begin{bmatrix} -\frac{1}{2} & \frac{1}{2} \end{bmatrix} \bullet \begin{bmatrix} x \\ y \end{bmatrix} + 1\right) = 1 \tag{2.30}$$

$$P_{-1}: \quad -\frac{1}{2}x+\frac{1}{2}y+1=-1 \qquad P_{-1}: \left(\begin{bmatrix}-\frac{1}{2} & \frac{1}{2}\end{bmatrix}\bullet\begin{bmatrix}x\\ y\end{bmatrix}+1\right)=-1 \tag{2.31}$$

Las ecuaciones (2.29), (2.30) y (2.31) representan los planos paralelos buscados en el entrenamiento, con las cuales se halla la función decisión:

$$f_{w,b}(\vec{x}_i)=signo\left(\begin{bmatrix}-\frac{1}{2} & \frac{1}{2}\end{bmatrix}\bullet\begin{bmatrix}x\\ y\end{bmatrix}+1\right) \tag{2.32}$$

Una vez hallada la función de decisión la máquina ya está entrenada, ahora se pueden clasificar nuevos datos.

Para hacer una prueba con un punto cualquiera, por ejemplo (4,5):

$$f_{w,b}(\vec{x}_i)=signo\left(\begin{bmatrix}-\frac{1}{2} & \frac{1}{2}\end{bmatrix}\bullet\begin{bmatrix}4\\ 5\end{bmatrix}+1\right)=signo(.5)=+1 \tag{2.33}$$

Este dato lo ha clasificado con etiqueta +1, perteneciente a los datos estrellas rojas.

Otro punto a clasificar podría ser (4,0):

$$f_{w,b}(\vec{x}_i)=signo\left(\begin{bmatrix}-\frac{1}{2} & \frac{1}{2}\end{bmatrix}\bullet\begin{bmatrix}4\\ 0\end{bmatrix}+1\right)=signo(-1)=-1 \tag{2.34}$$

Este dato ha clasificado como -1, perteneciente a los datos del grupo cuadros azules. Este ejemplo muestra claramente el funcionamiento simplificado de las MSV. Todo su procedimiento se basa en encontrar el OSH el cual conduce directamente a la función decisión que finalmente es la que permite realizar la clasificación.

2.5 TOLERANCIA AL RUIDO DE LAS MSV

Si existen datos erróneos, ruido o alto solapamiento de clases en los datos de entrenamiento, puede afectar el hiperplano clasificador óptimo. Por esta razón se cambia un poco la perspectiva y se busca el mejor hiperplano clasificador que pueda

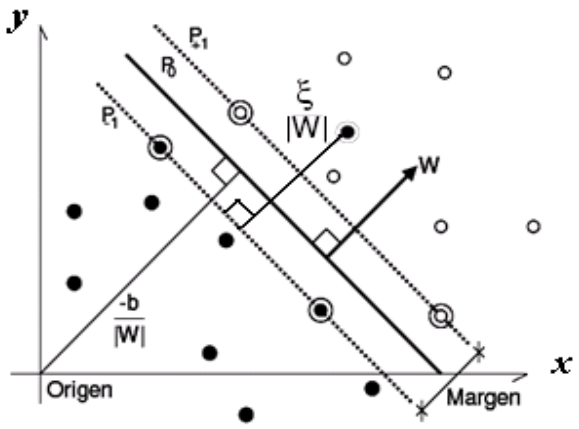
tolerar el ruido en los datos de entrenamiento, para permitir la posibilidad de ejemplos que violen la restricción (2.13), Cortes y Vapnik [Cortes & Vapnik, 95] introducen las variables de relajación (*slack*) basándose en [Bennett & Mangasarian, 92], ver la Figura 9. [Morales & Gomez, 05]

$$\xi_i \geq 0, \quad \forall_i \tag{2.35}$$

Para formular una nueva restricción:

$$y_i (\vec{w} \cdot \vec{x}_i + b) \geq 1 - \xi_i, \quad \forall_i \tag{2.36}$$

Figura 9. Hiperplano lineal clasificador para el caso no separable



[Burges vol. 2, 98]

Luego el clasificador que generaliza bien es hallado controlando tanto su capacidad de clasificación (con $\|\vec{w}\|$), como el límite superior del número de errores de entrenamiento ($\sum_{i=1}^{n} \xi_i$). La forma de obtener el hiperplano clasificador óptimo con margen débil es minimizando la función:

$$\min_{w}\left[\frac{1}{2}(\vec{w}\cdot\vec{w})+C\sum_{i=1}^{n}\xi_i\right] \tag{2.37}$$

$$Sujeto\ \ a \quad y_i\,(\vec{w}\cdot\vec{x}_i+\ b)\ \geq 1-\xi_i\,,\ \forall i \tag{2.38}$$

El parámetro C es elegido a priori por el usuario de tal manera que un valor grande es una alta penalización a lo errores. Si se usan los multiplicadores de Lagrange el problema se transforma en:

$$\max_{\alpha}\left[\sum_{i=1}^{n}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j(\vec{x}_i\cdot\vec{x}_j)\right] \tag{2.39}$$

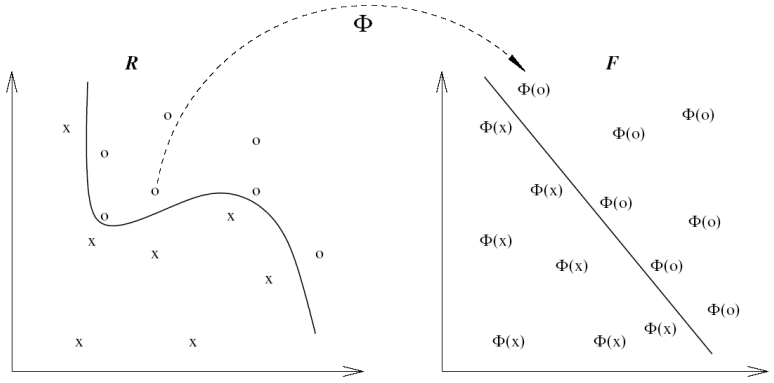
$$\text{Sujeto a}\quad 0\leq\ \alpha_i\ \leq \text{C},\quad \forall i \qquad \text{y} \qquad \sum_{i=1}^{n}\alpha_i y_i=0 \tag{2.40}$$

Cuya solución da como resultado (2.16), luego el hiperplano separador solución puede ser escrito como en (2.20), y la función de decisión (2.21).

2.6 MSV NO LINEALES

El principio de la MSV no lineal consiste en mapear el espacio de entrada a un espacio de representación de dimensión alta a través de una función no lineal, generalmente una función *kernel* [Boser; Guyon & Vapnik, 92]. Por medio de esta función se trazan los datos de entrada $\vec{x}_i \in \Re^N$, a algún espacio de mayor dimensión llamado espacio característico (F), ver Figura 10.

Figura 10. Transformación de espacio de entrada R al espacio F [Cristianini, 01]



http://www.cs.wisc.edu/~shavlik/SVM_icml01_tutorial.pdf

Los *kernels* más utilizados son:

- *Kernel* Polinomial:

$$k(\vec{x},\vec{y}) = ((\vec{x})\cdot(\vec{y}) + c)^d \quad para \;\; c > 0 \tag{2.41}$$

- Función de base radial (RBF):

$$k(\vec{x},\vec{y}) = e^{\left(-\frac{|\vec{x}-\vec{y}|^2}{2\sigma^2}\right)} \tag{2.42}$$

- Sigmoide:

$$k(\vec{x},\vec{y}) = \tanh(\vec{x}\cdot\vec{y}) + \Theta) \tag{2.43}$$

Ahora se pueden reescribir (2.39) sujeta a (2.40):

$$\max_{\alpha}\left[\sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{n}\alpha_i\alpha_j y_i y_j k(\vec{x}_i,\vec{x}_j)\right] \tag{2.44}$$

$$\text{Sujeto a} \quad 0 \le \alpha_i \le C, \quad \forall i \qquad \text{y} \qquad \sum_{i=1}^{n}\alpha_i y_i = 0 \tag{2.45}$$

y las ecuaciones del OSH (2.20) y función decisión (2.21) también cambiarán a:

$$g(\vec{x}) = \sum_{i=1}^{n}\alpha_i y_i k(\vec{x}_i,\vec{x}) + b \tag{2.46}$$

$$f(\vec{x}) = sign\left(\sum_{i=1}^{n}\alpha_i y_i k(\vec{x}_i,\vec{x}) + b\right) \tag{2.47}$$

2.7 MULTICLASIFICACIÓN CON SVM

"Hasta el momento sólo se ha tratado el problema de la biclasificación (en los cuales las clases solo pueden tomar valores ±1), pero en la práctica muchos problemas son

de más de dos clases ($yi \in 1,....l, \; l > 2$). Para resolver el problema de multiclasificación con máquinas de vectores de soporte se admiten dos tipos de arquitectura." [Morales & Gómez, 05].

2.7.1 Máquinas multiclasificadoras

Esta máquina construye una función clasificadora global directamente considerando todas las posibles clases a la vez. Este tipo de solución es lenta, y además no existe ninguna inclusión de técnicas que mejoren la robustez del sistema, ni ningún estudio teórico sobre la cota de error [Angúlo, 01]. [Morales & Gomez, 05]

2.7.2 Máquinas biclasificadoras generalizadas

Este tipo de máquina construye una función clasificadora global a partir de un conjunto de funciones biclasificadoras. Existen técnicas de descomposición y reconstrucción que permiten a las MSV biclasificadoras manejar problemas de multiclasificación con mayor simplicidad y/o menor tiempo de respuesta que una MSV generalizada a multiclasificación [Morales & Gomez, 05].

2.8 ARQUITECTURAS DE DESCOMPOSICIÓN NORMALIZADAS

"En el esquema de descomposición estándar se construyen m máquinas biclasificadoras en paralelo, que son entrenadas sobre modificaciones del conjunto de aprendizaje, creándose una matriz de descomposición. Los elementos de unas clases son asignados a salidas positivas, los de otras a salidas negativas, y si es el caso, los restantes no son tenidos en consideración en aquel clasificador en particular [Morales & Gomez, 05].

2.8.1 Uno contra el resto

Conocido como 1-v-r (del ingles *one-versus-rest*), este esquema se basa en la idea de que si existe un grupo de n datos de entrenamiento donde existen l clases (>2), se pueden tener un grupo de m clasificadores binarios (donde $m = l$), cada uno entrenado para separar una clase del resto de clases existentes (-1). La descomposición se realiza de la siguiente manera: existe un grupo de datos (n_j) que pertenecen a la j-ésima clase $(i \in \{1.....l\})$, a los cuales se les dará una etiqueta positiva (t_j = +1) y al resto de datos $(n_r = n - n_j)$ se les dará una etiqueta negativa (t_r = −1) para el entrenamiento de la i-ésima MSV. Así se crea una matriz de descomposición (D_{1-v-r}) de m filas y l columnas:

$$Di,j = \begin{cases} +1 & si \quad n_h \in n_j \\ -1 & si \quad n_h \in n_r \end{cases} \tag{2.48}$$

Por ejemplo, una máquina de clasificación multiclase (1-v-r) con l = 5, se obtiene $m = 5$. Entonces la correspondiente matriz de descomposición es la siguiente:

$$D_{1-v-r} = \begin{pmatrix} +1 & -1 & -1 & -1 & -1 \\ -1 & +1 & -1 & -1 & -1 \\ -1 & -1 & +1 & -1 & -1 \\ -1 & -1 & -1 & +1 & -1 \\ -1 & -1 & -1 & -1 & +1 \end{pmatrix} \tag{2.49}$$

En esta arquitectura propuesta por [Vapnik, 95], [Cortes & Vapnik, 95], el tiempo de entrenamiento es proporcional al número de clases, y debido a que el entrenamiento de cada biclasificador es con el conjunto de datos de entrenamiento completo, su costo computacional es alto [Morales & Gomez, 05].

2.8.2 Uno contra uno

Conocido como 1-v-1 (del ingles *one-versus-one*), se realiza implementando $m = \frac{l(l-1)}{2}$ clasificadores binarios. Luego, el entrenamiento de la $i-esima$ MSV se realiza con solo 2 de las l $(l > 2)$ clases existentes en el grupo de n datos de entrenamiento, otorgándole etiqueta positiva (t_j = +1) a los datos (n_j) que pertenecen al subgrupo de datos de la clase j $(j \in 1,...,l)$, y etiqueta negativa $(tp = -1)$ a los datos (n_p) que pertenecen al subgrupo de datos de la clase p $(p \in 1,..,l \text{ y } p \neq j)$. Los demás datos ($n_r$ = n − n_j − n_p) no se utilizan en el entrenamiento de la i-ésima MSV, por lo tanto, son etiquetados con cero (t_r=0), creándose la matriz de descomposición $(D_{1\text{-}v\text{-}1})$ [Morales & Gomez, 05].

$$D_{i,j} = \begin{cases} +1 & si \quad n_h \in n_j \\ -1 & si \quad n_h \in n_p \\ 0 & si \quad n_h \in n_r \end{cases} \tag{2.50}$$

Por ejemplo, para una máquina de clasificación multiclase (1-v-1) con l = 5, se obtiene m = 10. Entonces la correspondiente matriz de descomposición es la siguiente:

$$D_{1-v-1} = \begin{pmatrix} +1 & -1 & 0 & 0 & 0 \\ +1 & 0 & -1 & 0 & 0 \\ +1 & 0 & 0 & -1 & 0 \\ +1 & 0 & 0 & 0 & -1 \\ 0 & +1 & -1 & 0 & 0 \\ 0 & +1 & 0 & -1 & 0 \\ 0 & +1 & 0 & 0 & -1 \\ 0 & 0 & +1 & -1 & 0 \\ 0 & 0 & +1 & 0 & -1 \\ 0 & 0 & 0 & +1 & -1 \end{pmatrix} \quad (2.51)$$

2.8.3 Arquitectura de descomposición ECOC

La técnica ECOC (del ingles *Error Correcting Output Codes*) [Dietterich & Bakiri, 95], utiliza la codificación estándar para obtener robustez contra fallos en las máquinas biclasificadoras. Se denomina codificación estándar a cada una de las posibles particiones de todo el conjunto de clases yi $\in$ {1, . . . , l}, en problemas de biclasificación, que asignan etiquetas positivas $t_p = +1$ a los patrones de entrenamiento n_j de un cierto subconjunto de clases Y_j, y etiquetas negativas $t_p = -1$ a los patrones de entrenamiento nr representantes del resto de clases Y_r. La descomposición generada por

$$Di, j = \begin{cases} +1 & si \quad n_h \in n_j \\ -1 & si \quad n_h \in n_r \end{cases} \quad (2.52)$$

debe ser tan diferente como sea posible en términos de la distancia Hamming para añadir redundancia, en este caso $m = 2^{l-1} - 1$. Por ejemplo, para una máquina de clasificación multiclase (ECOC) con $l = 4$, se obtiene $m = 7$. [Morales & Gomez, 05]

Entonces la correspondiente matriz de descomposición es la siguiente:

$$D_{ECOC} = \begin{pmatrix} +1 & -1 & -1 & -1 \\ +1 & -1 & -1 & +1 \\ +1 & -1 & +1 & -1 \\ +1 & -1 & +1 & +1 \\ +1 & +1 & -1 & -1 \\ +1 & +1 & -1 & +1 \\ +1 & +1 & +1 & -1 \end{pmatrix} \quad (2.53)$$

2.9 MÉTODOS DE RECONSTRUCCIÓN

Cada máquina biclasificadora entrenada emite una respuesta en forma numérica $z^i = g_i(\vec{x})$ a una entrada $\vec{x}$. La información más importante en esta respuesta, en principio, se encuentra en el signo $s^i = f_i(\vec{x}) = sign(g_i(\vec{x}))$ que adopta la función de decisión. En la determinación de la respuesta final facilitada por el método de reconstrucción de la máquina de aprendizaje multiclase han de ser tomados en consideración los siguientes elementos:

- Las predicciones numéricas parciales de los nodos de dicotomía, $z^i = g_i(\vec{x})$.
- El signo de las predicciones numéricas, $s^i = fi(\vec{x}) = sign(g_i(\vec{x}))$.
- Un elemento intérprete de las predicciones numéricas y binarias, $\Theta(z^i, s^i)$, con el fin de asignar o no, una o varias clases como posible respuesta de clasificación a una entrada $\vec{x}$.
- Un elemento $\Psi(\Theta(z^1, s^1), \ldots, \Theta(z^m, s^m))$ de combinación de las predicciones, que tenga o pueda tener en consideración las predicciones numéricas, sus signos y/o la clase o clases asignadas.

El esquema de votación es la forma de reconstrucción más habitual. Se tiene en consideración sólo el signo de las predicciones de todas las máquinas biclasificadoras. Estos signos son interpretados en función de las clases implicadas en las máquinas biclasificadoras utilizado en el esquema de descomposición.

1. i-ésimo 1-v-r máquina biclasificadora

$$\Theta(z^i) = \begin{cases} y_i & si \quad s^i = +1 \\ 0 & si \quad s^i = -1 \end{cases} \quad (2.54)$$

2. i-ésimo 1-v-1 máquina biclasificador

$$\Theta(i) \equiv \begin{cases} y_i & si \quad s^i = +1 \\ y_p & si \quad s^i = -1 \end{cases} \tag{2.55}$$

3. i-ésimo ECOC máquina biclasificadora

$$\Theta(i) \equiv \begin{cases} Y_i & si \quad s^i = +1 \\ Y_r & si \quad s^i = -1 \end{cases} \tag{2.56}$$

Tras la interpretación de las predicciones, el elemento de combinación realiza un recuento del número de clases votadas, acción de la que toma el nombre de esquema de reconstrucción, que posee diferentes variantes. Se define a continuación algunas de estas posibilidades para las arquitecturas de descomposición:

4. Votación por unanimidad: se determina como respuesta aquella única clase que haya obtenido todos los votos posibles en las predicciones.

5. Votación por mayoría absoluta: se determina como respuesta final aquella única clase que haya obtenido más de la mitad de los votos posibles.

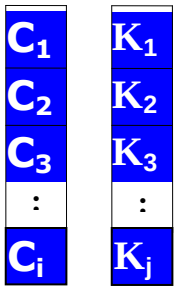
6. Votación por mayoría simple: se determina como respuesta final aquella única clase que haya obtenido más votos que el resto de clases" ,[Morales & Gómez, 05].

2.10 VALIDACIÓN CRUZADA EN LA MSV

La validación cruzada es una técnica que se utiliza para tratar problemas iterativos repartiendo la muestra en dos sistemas de datos, uno se utiliza para construir el modelo, y el otro para probarlo. Esta técnica incorporada en el entrenamiento de la MSV, se utiliza para seleccionar los mejores parámetros (parámetros del *kernel* y de penalización C) de la MSV para un grupo específico de datos. [Martínez W. & Martínez A. 02]

El procedimiento que se explica a continuación se realiza para cada pareja de parámetros (estas parejas se forman con todas las combinaciones posibles de los parámetros del *kernel* y C), (ver Figura 11).

Figura 11. Parejas de parámetros del kernel y del C

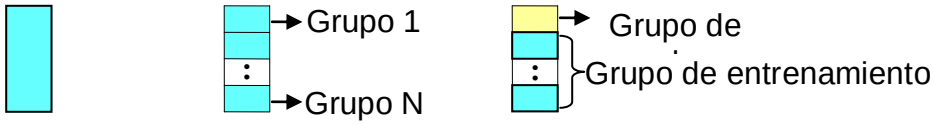


Numero de combinaciones: i*j

Elaborado por los autores

Los datos de entrenamiento se dividen en N grupos (o **partes**) de forma aleatoria, se escoge el primer grupo como grupo prueba (el término "prueba" se seguirá utilizando en adelante para indicar prueba o validación) y con las restantes partes se forma el grupo de entrenamiento, como se muestra en la Figura 12.

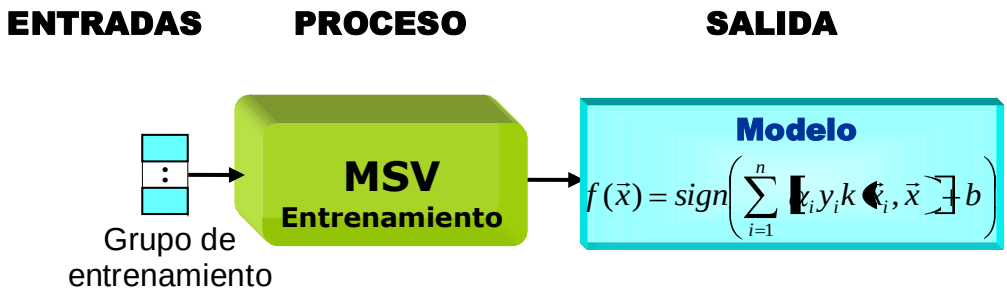
Figura 12. Formación de los grupos



Elaborado por los autores

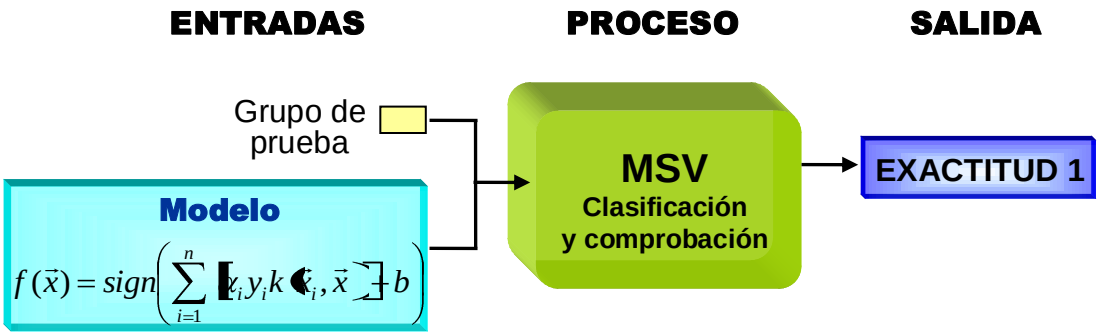
Con el grupo de entrenamiento se entrena la primera máquina obteniéndose un modelo, véase la Figura 13. Con este modelo se clasifica el grupo de prueba, esta clasificación se compara con las verdaderas etiquetas que le corresponden a estos datos, hallándose un resultado de exactitud a la que se llamará exactitud 1, (ver Figura 14).

Figura 13. Entrenamiento con el primer grupo de datos



Elaborado por los autores

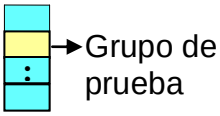
Figura 14. Clasificación y exactitud utilizando el grupo de prueba



Elaborado por los autores

Seguidamente se selecciona el segundo grupo como grupo de prueba y los restantes datos serán ahora grupo de entrenamiento (ver Figura 15) con los cuales se entrena otra máquina. Con el nuevo modelo obtenido se clasifica el grupo de prueba, hallando la exactitud la cual será llamada exactitud2. Este procedimiento se realiza con los N grupos.

Figura 15. Formación de los segundos grupos (entrenamiento y prueba)



Grupo de Entrenamiento en azul

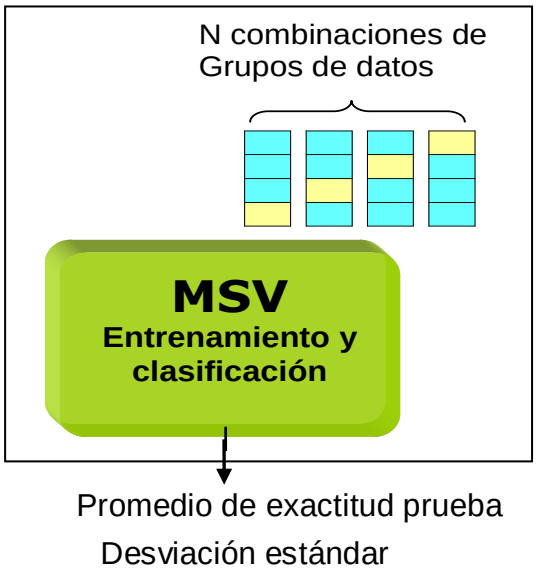
Elaborado por los autores

Finalmente se tienen N resultados de exactitud a los cuales se les calcula su promedio y su desviación estándar como se indica en la Figura 16. Estos dos valores son los que servirán posteriormente para comparar entre si los resultados obtenidos para cada pareja de parámetros.

Después de salir de la validación se tendrán tantos valores promedio de exactitud y desviación estándar como pares de parámetros que se haya entrenado, los parámetros que correspondan a la mayor exactitud serán los mejores y si hay un empate se desempata con la menor desviación estándar, esto se hace para

garantizar que la pareja de parámetros escogidos sea estable, de persistir el empate se optará por escoger la máquina con el mayor valor de C, este indica mayor penalización a los errores.

Figura 16. Resultados de la primera pareja de parámetros



Elaborado por los autores

A continuación se hace una explicación ejemplificada de la validación cruzada en la MSV.

- **Ejemplo de validación cruzada**

En este ejemplo se mostrará la forma de utilizar la validación cruzada para resolver un problema simbólico:

Se desea encontrar la mejor pareja de parámetros para una clase específica de datos, se cuenta con 500 muestras y se busca probar 3 parámetros (uno del kernel y dos de C) estos serán llamados K= (k1) y C= (c1, c2), los datos se dividirán en 5 grupos (G1, G2, G3, G4 y G5).

De esta forma se tendrán dos parejas de parámetros (k1, c1) y (k1, c2), los 5 grupos tendrán 100 datos cada uno, por lo cual quedaran 400 datos de entrenamiento y 100 de prueba.

Para la iteración 1

Para (k1, c1)		exactitud de prueba
Máquina 1	Se entrena con (G2, G3, G4 y G5) se prueba con (G1)	0,8

Máquina 2	Se entrena con (G1, G3, G4 y G5) se prueba con (G2)	0,8
Máquina 3	Se entrena con (G1, G2, G4 y G5) se prueba con (G3)	0,9
Máquina 4	Se entrena con (G1, G2, G3 y G5) se prueba con (G4)	0,5
Máquina 5	Se entrena con (G1, G2, G3 y G4) se prueba con (G5)	0,7

Exactitud promedio	**0,74**
Desviación estándar	**0,157**

Ahora para la iteración 2

Para (k1, c2)		exactitud de prueba
Máquina 6	Se entrena con (G2, G3, G4 y G5) se prueba con (G1)	0,7
Máquina 7	Se entrena con (G1, G3, G4 y G5) se prueba con (G2)	0,4
Máquina 8	Se entrena con (G1, G2, G4 y G5) se prueba con (G3)	0,9
Máquina 9	Se entrena con (G1, G2, G3 y G5) se prueba con (G4)	0,5
Máquina 10	Se entrena con (G1, G2, G3 y G4) se prueba con (G5)	0,7

Exactitud promedio	**0,64**
Desviación estándar	**0,194**

Como se puede apreciar el número de máquinas entrenadas es igual a la cantidad de parámetros del kernel (K) multiplicada por los parámetros de penalización C y por el número de partes, en este caso 1x2x5=10.

Los resultados muestran a la pareja (k1, c1) como los mejores parámetros. Si se hubiese presentado un empate, se decidiría por la menor desviación en este caso 0,157 de la iteración 1. De persistir el empate se selecciona el mayor valor de C.

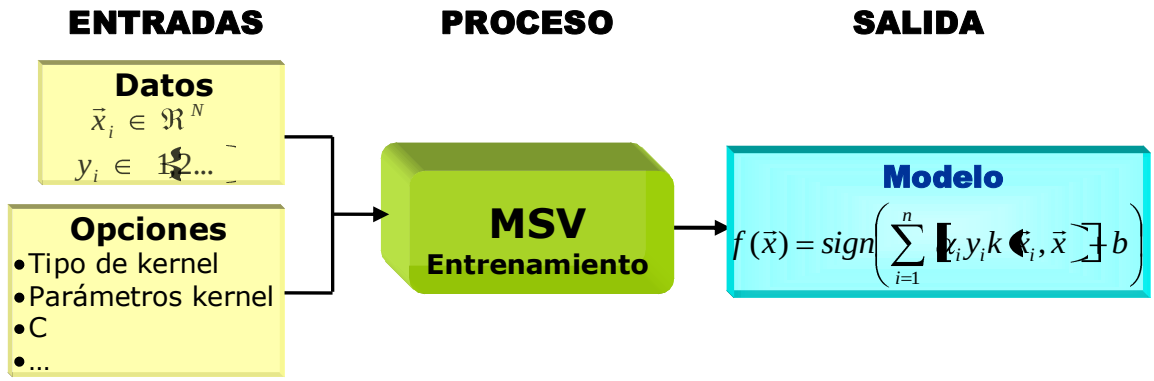
Pruebas realizadas en este trabajo (capítulo 5) muestran que el número de partes o grupos en que se dividen los datos determina en gran parte la calidad del resultado, un análisis simple determina que si se escogen demasiados grupos, el grupo de prueba quedará proporcionalmente con muy pocos datos, por lo que su validación no será suficientemente confiable. En el caso contrario, si se escogen pocos grupos los datos de entrenamiento serán insuficientes para hacer una buena predicción. Por esto se recomienda que el número de partes para realizar la validación cruzada sea 4 o 5, es decir, con un 20 a 25% de los datos para prueba y el restante 75 a 80% para entrenamiento.

3. IMPLEMENTACIÓN DE UNA INTERFAZ GRÁFICA DE CLASIFICACIÓN: MSVToolbox1.0

En este capítulo se presenta la estructura y funcionamiento de los diferentes programas o funciones utilizadas por la interfaz MSVToolbox1.0, estas funciones son el producto de la corrección y optimización de las funciones implementadas en la herramienta computacional svm_tool desarrollada por [Morales & Gomez, 05].

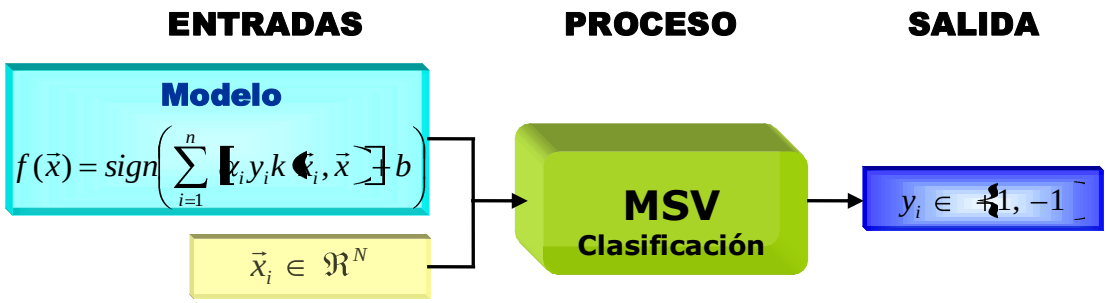
En las funciones se utiliza convencionalmente el orden de entradas-proceso-salida para determinar completamente cada una de las ejecuciones de la MSVToolbox1.0. En las Figuras 17 y 18 se aprecia el proceso de entrenamiento y clasificación respectivamente, en estas se puede observar las diferentes entradas, el proceso de entrenamiento o clasificación y sus salidas.

Figura 17. Proceso de entrenamiento



Elaborado por los autores

Figura 18. Proceso de clasificación



Elaborado por los autores

Las funciones más importantes usadas por la interfaz MSVToolbox1.0 se presentan a continuación.

Funciones usadas en biclasificación:
msv_2entrenar : función de entrenamiento
msv_2clasif : función de clasificación
msv_kernel : función usada por msv_2entrenar y msv_2clasif, para hallar la matriz kernel dependiendo del tipo de kernel RBF, polinomial o sigmoidal.

Funciones usadas en multiclasificación
entrenar_kclases : función de entrenamiento
multiclasificador : función de clasificación
mat_descomposicion : función usada por entrenar_kclases y multiclasificador para hallar la matriz de descomposición dependiendo del método usado OVO, OVR o ECOC_2C
exactitud : función de clasificación con exactitud

Búsqueda de parámetros
Validación cruzada : función de entrenamiento y clasificación que busca los mejores parámetros.

En la Tabla 4 se presentan dos estructuras; *datos* y *opciones*. La primera se utiliza para almacenar los vectores de entrenamiento y sus respectivas etiquetas y

dependiendo de qué función se esté utilizando podría almacenar los datos a clasificar.

La estructura opciones se utiliza en las funciones de entrenamiento y permite almacenar los valores de los parámetros de penalización C, tipo de *kernel* y sus parámetros, el método de descomposición y de optimización escogido y las variables eps y tol (relacionadas al método de optimización).

Tabla 4. **Variables empleadas por las funciones**

Nombre [tipo de dato]	Descripción
datos [struct]	
.X [NxP]	Vectores de entrenamiento, donde se tienen N vectores de dimensión [1xP]
.Y [Nx1]	Etiquetas de los datos de entrenamiento (1,2,...,K)
opciones [struct]	
.C [1x1] .C [1xDIMC]	Parámetro de penalización Parámetros de penalización que se utilizarán en la validación cruzada
.Kernel [string]	Tipo de kernel
'RBF'	RBF, gausiano
'poly'	Polinomial
'sigmoid'	Sigmoide
.Kernel_par [1x1] o [2x1] .Kernel_par [1xDIMpar] [2xDIMpar]	Parámetro del kernel Parámetros del kernel que se utilizarán en la validación cruzada
* .descom [string]	Tipo de descomposición para multiclasificación
'ovo'	Uno contra uno (por defecto)
'ovr'	Uno contra el resto
'ecoc_2c'	Ecoc doble capa
* .método [string]	Algoritmo de optimización cuadrática
'smo'	Optimización mínima secuencial (por defecto)
'irwls'	Algoritmo propuesto por Fernando Pérez Cruz
* .eps	Error que tolera el algoritmo de optimización (0,001 por defecto)
* .tol	Tolerancia de las condiciones KKT (0,001 por defecto)

Elaborado por [Morales & Gomes, 05] *Campos opcionales

En la Tabla 5 se muestra la estructura modelo, ésta es creada al entrenar un grupo de datos, es decir al ejecutar cualquiera de las funciones de entrenamiento

se obtendrá la estructura modelo. Esta estructura es usada como entrada a las funciones de clasificación.

En la estructura *modelo* se almacenan los datos más relevantes con los que fue entrenada la máquina MSV como son: el tipo de *kernel* y sus parámetros, el parámetro C entre otros. También en esta estructura se guarda la máquina en si, o sea los vectores de soporte, los alpha y el b. Adicionalmente en ésta se almacena la exactitud con la que fue entrenada o sea que tan buena es esta máquina entrenando los datos. (es decir, la probabilidad de que un dato sea bien clasificado).

Tabla 5. Estructura de la SVM entrenada [Morales & Gomes, 05]

Nombre [tipo de dato]	Descripción
modelo [struct]	
.Kernel [string]	Tipo de kernel
.Kernel_par [2x1]	Parámetro del kernel
.C [1x1]	Parámetro de penalización
.clases [1x1]	Numero de clases K
.exact_entr [1x1]	Exactitud de entrenamiento
.exact_prueba[1x1]	Exactitud de prueba, este campo aparece por la validación cruzada
.detalles [struct]	
.método [string]	Método que se utilizó en la optimización
.descom [string]	Tipo de descomposición
.eps [1x1]	Error de parada
.tol [1x1]	Tolerancia de las condiciones KKT
.num_SV [1x1]	Número de vectores de soporte
.SV_y [1xK]	Número de vectores de soporte por clase
.SVM [struct][1xL]	Donde L es el número de máquinas biclasificadoras que se ejecutan para la multiclasificar K clases L = size(D,1). D matriz de descomposición L = K*(K+1)/2 si se usa 'ovo' L = K si se usa 'ovr' L = K*(K+1) si se usa 'ECOC_2C'
.SV [NixP]	Vectores de soporte. Ni es el número de VS.
.Alpha [Nix1]	Multiplicadores de lagrange
.b [1x1]	Constante de la función de decisión
exact_entr [1x1]	Exactitud de entrenamiento de cada máquina biclasificadora
.clase1 [1xa1]	Clases con etiqueta 1 implicadas en cada máquina biclasificadora
.clase2 [1xa2]	Clases con etiqueta 2 implicadas en cada máquina biclasificadora
.prueba	Este campo aparece si se entrego datos de prueba
.Exact [1x k]	Exactitud de entrenamiento por clase
.acierto [1x k]	Numero de datos bien clasificados por clase
.cantidad [1x k]	Total de datos clasificados por clase
.datos_err	
.X[NxP]	Datos mal clasificados donde se tienen N vectores de dimensión [1xP]
.Y[Nx2]	Etiquetas de los datos mal clasificados La primera columna son de las etiquetas erróneas halladas con la TOOLBOX La segunda son las etiquetas dadas por el usuario

Las funciones de la toolbox se explicarán mediante un diagrama de flujo. Para este fin se utilizan las siguientes convenciones:

Inicio y entrada de datos

Encierra todo el proceso de la función

Llamada a programa

Proceso

Salida de datos

Ciclo

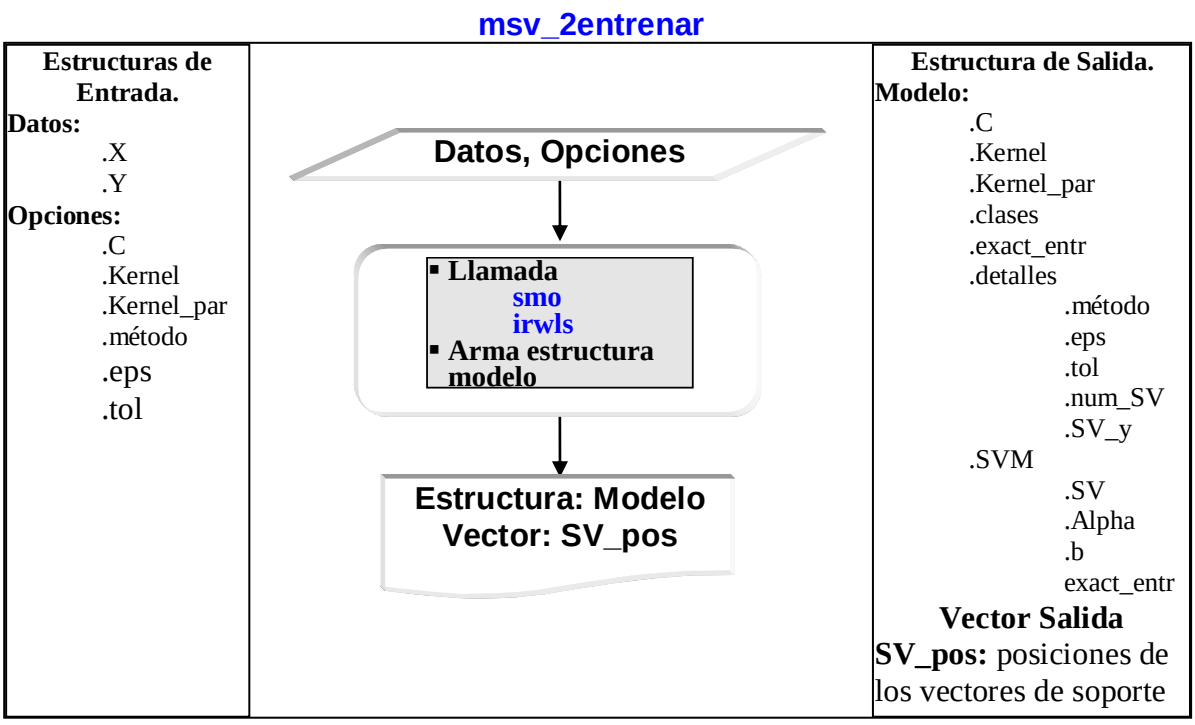
3.1 FUNCIÓN msv_2entrenar

Esta función que entrena la MSV de dos clases (etiquetas 1 y 2).

Sintaxis: modelo = **msv_2entrenar** (datos, opciones)

Esta función es tal vez el corazón de la toolbox, debido a que contiene los algoritmos de optimización ('smo','irwls'). La respuesta de esta función es la estructura modelo.

Figura 19. Diagrama de bloques de la función msv_2entrenar



Elaborado por los autores

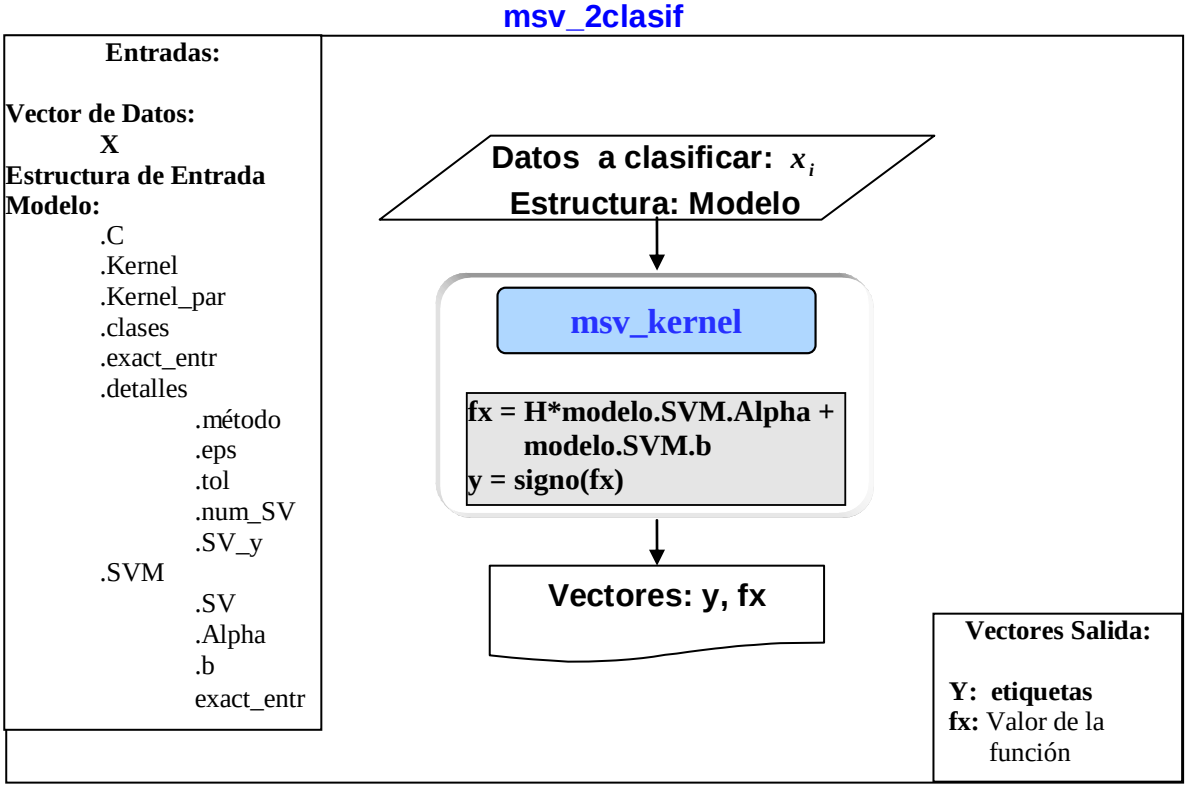
3.2 FUNCIÓN msv_2clasif

Esta función clasifica los datos, el número de clases debe ser igual a dos (2).

Sintaxis: [Yi, fx] = msv_2clasif(X, modelo)

Esta función tiene como variables de entrada los datos a clasificar (X) y la estructura generada por msv_2entrenar (modelo) y las variables de salida Yi correspondiente a la clasificación o etiqueta hallada por esta función y fx corresponde a la evaluación numérica de la función de decisión.

Figura 20 Diagrama de bloques de la función msv_2clasif



Elaborado por los autores

3.3 FUNCIÓN msv_kernel

Evalúa la matriz *kernel*.

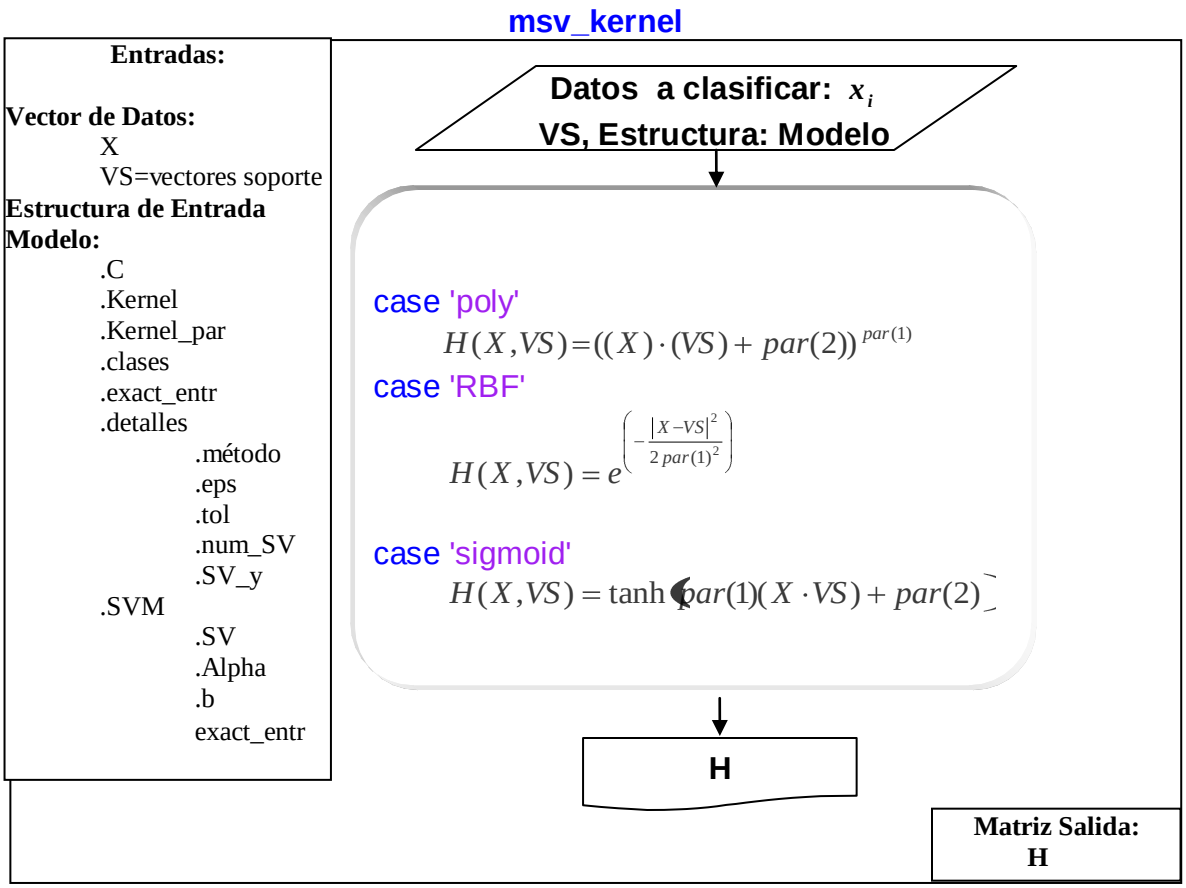
Sintaxis: H = **msv_kernel**(X1, X2, opciones)

Halla la matriz *kernel*, donde: $H(i,j)=k(X_1(i),X_2(j))$. Esta matriz es necesaria para el entrenamiento y la clasificación (en el entrenamiento esta función no se usa porque esta incluida en los programas de optimización('smo' e 'irwls')).

Los tipos de *kernel* disponibles son:

- RBF (gausiano): 'RBF' ,
- Polinomial: 'poly' ,
- Sigmoide: 'sigmoid' ,

Figura 21. Diagrama de bloques de la función msv_kernel



Elaborado por los autores

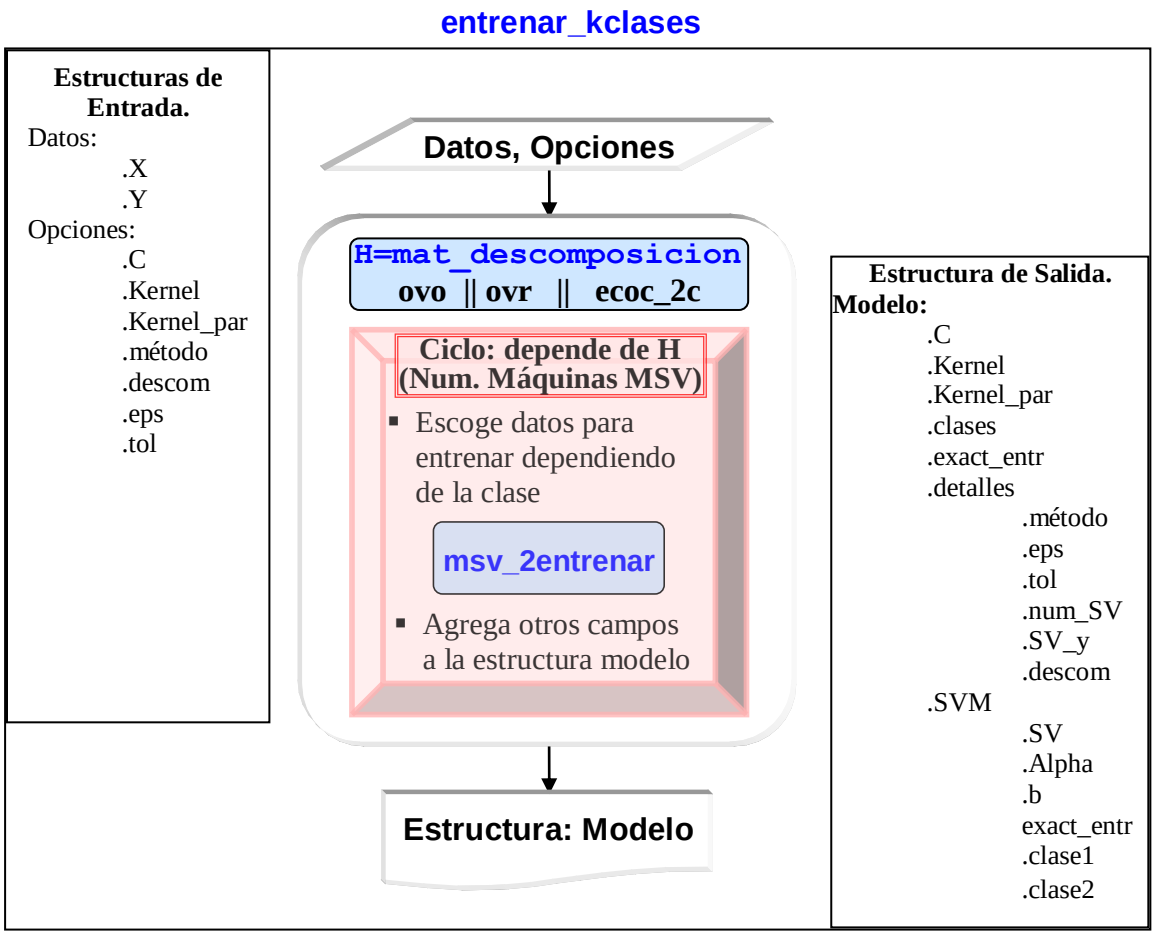
3.4 FUNCIÓN entrenar_kclases

Entrena la MSV para múltiples clases (etiquetas 1, 2, 3, …).

Sintaxis: modelo = entrenar_kclases(datos, opciones)

Entrena la MSV para multiclasificación, donde el problema de K clases es descompuesto en problemas de clasificación binaria.

Figura 22. Diagrama de bloques de la función entrenar_kclases



Elaborado por los autores

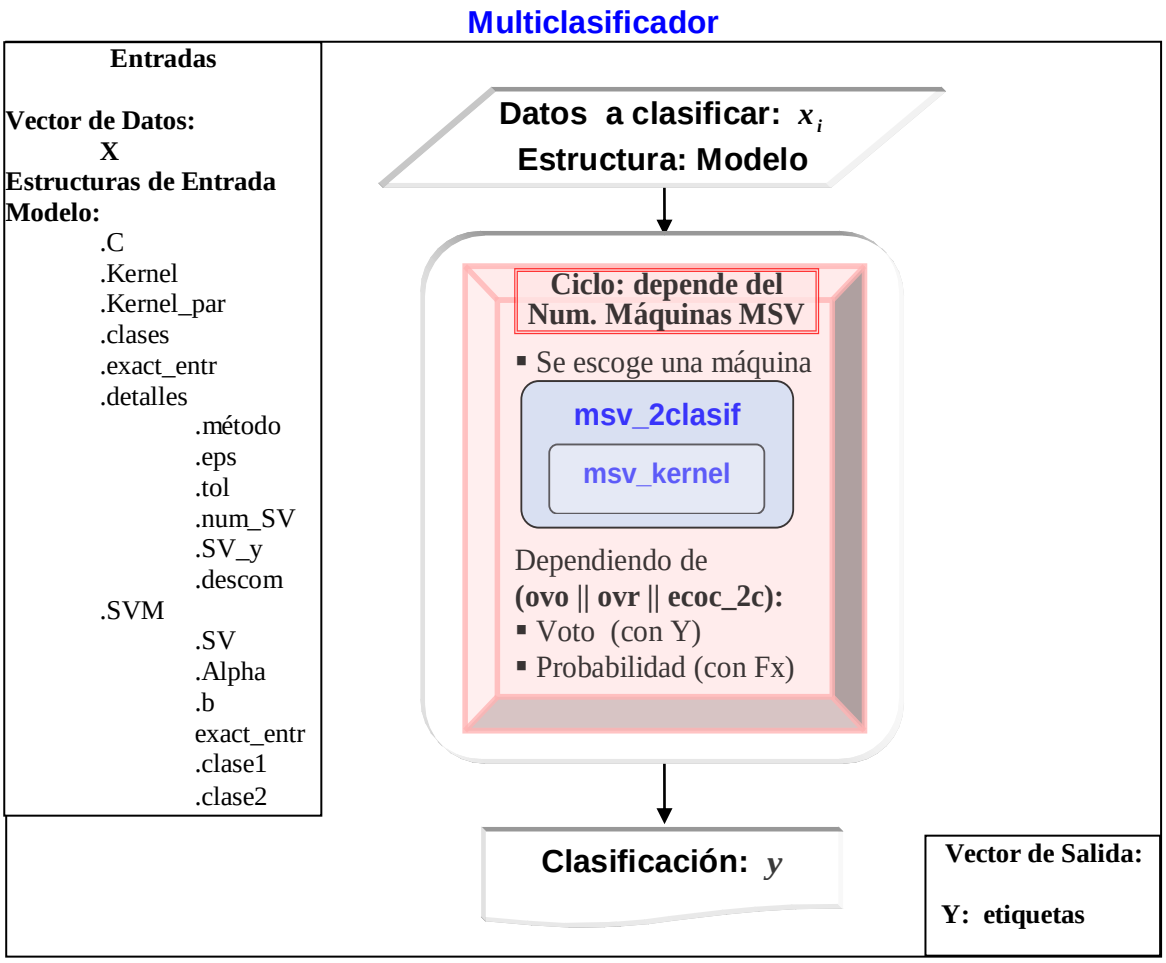
3.5 FUNCIÓN multiclasificador

Esta función clasifica datos de dos o más clases (etiquetas 1, 2, 3, …).

Sintaxis: Y = multiclasificador (X, modelo)

El vector de salida Y corresponde a la clasificación o etiqueta hallada por esta función.

Figura 23. Diagrama de bloques de la función multiclasificador



Elaborado por los autores

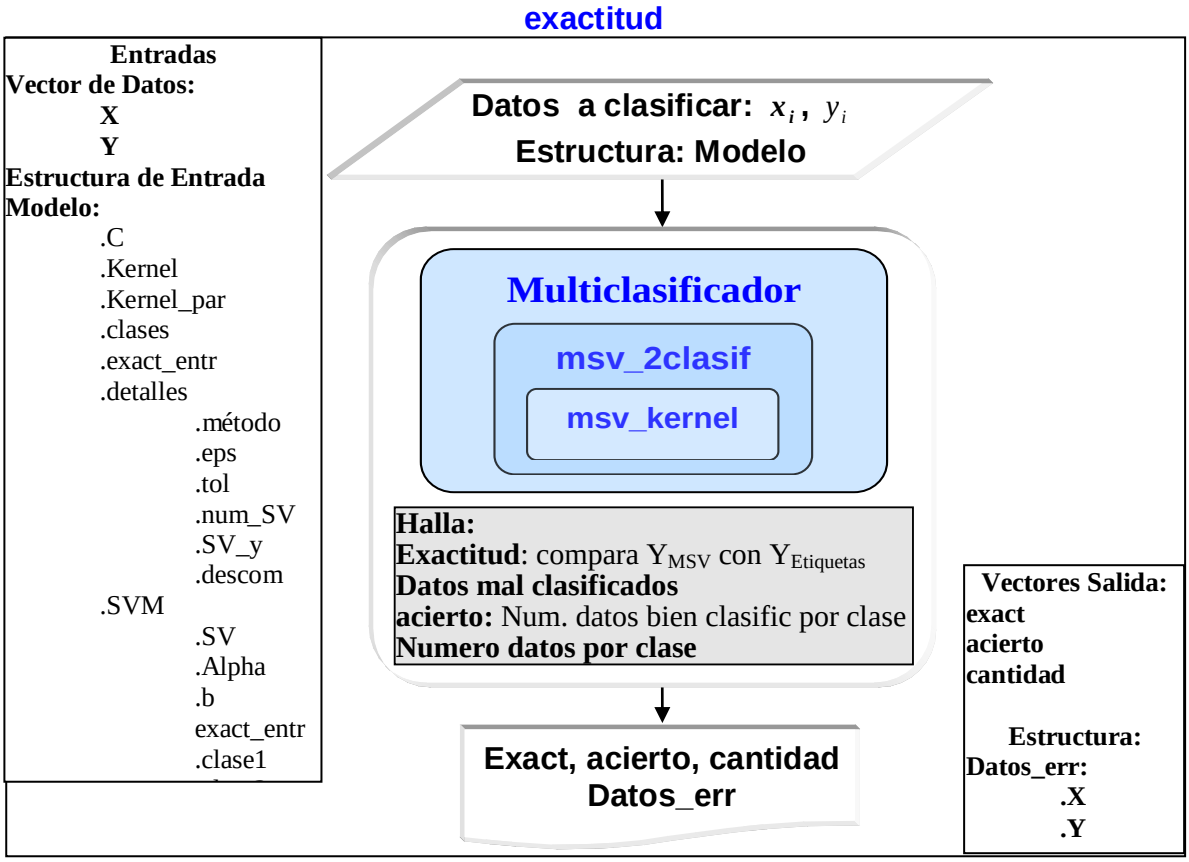
3.6 FUNCIÓN exactitud

Evalúa exactitudes de las etiquetas halladas por la MSV.

Sintaxis: [exact, acierto, cantidad, Datos_err] = exactitud(X, modelo,Y)

Esta función halla la etiqueta de la entrada X y la compara con la etiqueta verdadera (Y), sacando así los vectores de exactitud, datos mal clasificados, la cantidad y el acierto.

Figura 24. Diagrama de bloques de la función exactitud



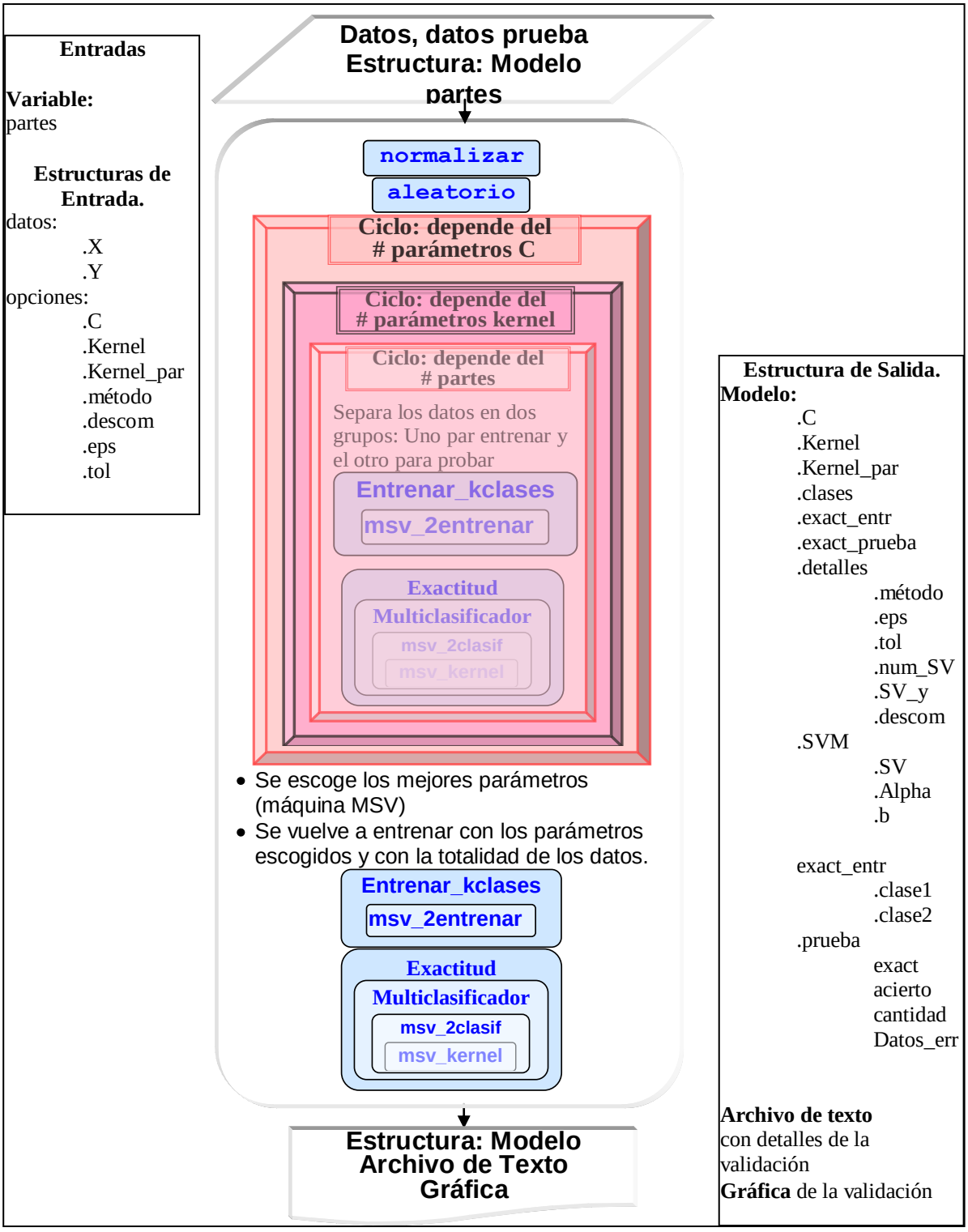
Elaborado por los autores

3.7 FUNCIÓN Validación Cruzada

Sintaxis: modelo = valcros(datos, opciones, datos_tst, partes)

Esta función entrena las MSV con el método de búsqueda en malla y validación cruzada para escoger los mejores parámetros (C y *kernel*). Si se ingresan datos de prueba, al final del entrenamiento se clasifican estos datos obteniéndose el porcentaje de acierto. Se puede especificar el número de particiones que se van a tomar para la validación cruzada como se presenta en la Figura 25.

Figura 25. Diagrama de bloques de la función validación cruzada



Elaborado por los autores

En este capítulo se presentó la estructura y funcionamiento de los programas o funciones más importantes utilizadas por la interfaz MSVToolbox1.0 por medio de diagramas.

Se puede apreciar en los diagramas de cada programa cómo una función hace uso de otras, por ejemplo se puede observar en el diagrama de Validación cruzada, Figura 25, cómo esta función llama a todas las demás funciones. Este hecho muestra la complejidad del programa de Validación cruzada y así mismo brinda la posibilidad de un mejor entendimiento de esta técnica.

4. RESULTADOS DE SIMULACIONES

En este capítulo se presentan los resultados de las simulaciones realizadas con la herramienta computacional MSVToolbox1.0. Estas pruebas se separan en dos partes: inicialmente se presentan las pruebas necesarias para encontrar los mejores componentes de la MSV empleada en la clasificación de las perturbaciones en las señales de tensión o corriente (tipo Kernel, método de optimización, método de descomposición y reconstrucción), seguidamente se muestran las pruebas realizadas para observar el comportamiento de la MSVToolbox1.0 con diferentes tipos de datos.

Las simulaciones se realizan previendo una de las deficiencias de las MSV mencionadas en el estado del arte; el elevado tiempo de cómputo. Este problema se debe a la gran cantidad de iteraciones necesarias para encontrar los parámetros del kernel adecuados para un tipo de datos en particular. Un aporte de este trabajo de grado que resuelve este problema es el procedimiento Prueba-análisis con el cual se consigue excelentes resultados con un bajo costo computacional.

4.1 PROCEDIMIENTO PRUEBA-ANÁLISIS

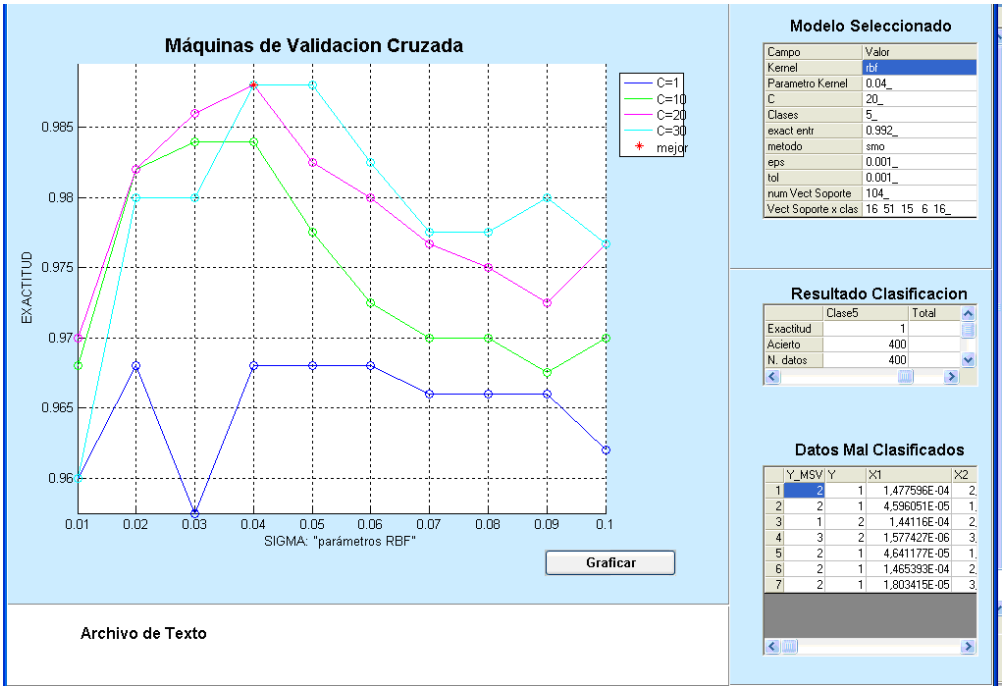
- **Primer paso**: inicialmente se realiza una búsqueda en malla utilizando pocos parámetros con un tamaño de paso grande para determinar la mejor zona donde se debe realizar la siguiente búsqueda, esto se hace con el fin de disminuir el número de iteraciones ya que la cantidad de parámetros afecta directamente el tiempo de cómputo. Se recomienda no utilizar la totalidad de los datos para este primer paso ya que el tiempo de entrenamiento aumenta de forma crítica al utilizar grandes cantidades de datos.

- **Segundo paso**: teniendo los resultados de la prueba 1 se analiza la gráfica parámetros *kernel* contra exactitud de entrenamiento de los diferentes parámetros **C** (véase Figura 26) para así elegir el siguiente intervalo de búsqueda.

  En la Figura 26 se muestran los diferentes valores de exactitud encontrados por las diferentes máquinas entrenadas en la validación cruzada, para este caso se puede ver claramente dónde se debe realizar la siguiente validación (0.03-0.06), así mismo se pueden descartar valores de **C** que muestran resultados no adecuados, en este caso para C=1 y C=10. Se puede presentar

más de una zona con buenos resultados (origen del problema de convergencia para el método propuesto en [Henao, 04]), por lo tanto el seguimiento visual de las pruebas permite llegar mucho más rápido a los mejores parámetros.

Figura 26. Parámetros kernel Vs exactitud de cada una de las máquinas entrenadas



Elaborado por los autores

- **Tercer paso**: Se realiza el mismo análisis y procedimiento del paso 2 continuando hasta encontrar la exactitud deseada.

  Según la experiencia y los resultados de las diferentes pruebas realizadas durante el desarrollo de la MSVToolbox1.0 se recomienda el procedimiento **prueba–análisis** como el más efectivo para la búsqueda de parámetros.

Trabajos realizados por [Henao, 04] muestran un método alternativo automatizado que disminuye el tiempo de cómputo realizando una predicción del tamaño del paso de la búsqueda en malla, pero presenta el grave problema de no convergencia. Con el método propuesto, en este trabajo, **prueba–análisis** se ataca directamente el problema de la no convergencia y de alto costo computacional inherente a las MSV, el método es sencillo y requiere pocas pruebas para encontrar buenos parámetros. La búsqueda de parámetros se hace mediante la validación cruzada y de forma interactiva revisando los resultados para tomar la siguiente decisión de búsqueda.

4.2 MSV PARA LA CLASIFICACIÓN DE LOS PATRONES DE LAS PERTURBACIONES DE LA ENERGÍA ELÉCTRICA

Para determinar cada uno de los componentes de la MSV utilizada en la clasificación de los patrones de las perturbaciones de las señales de tensión o corriente es necesario entrenar y clasificar los mismos datos determinando en cada prueba el comportamiento de la toolbox en cuanto a tiempo, exactitud y NVS (número de vectores de soporte). Las simulaciones se realizan con una muestra de los patrones a clasificar buscando exactitudes de clasificación superiores al 90%.

4.2.1 Características de los datos

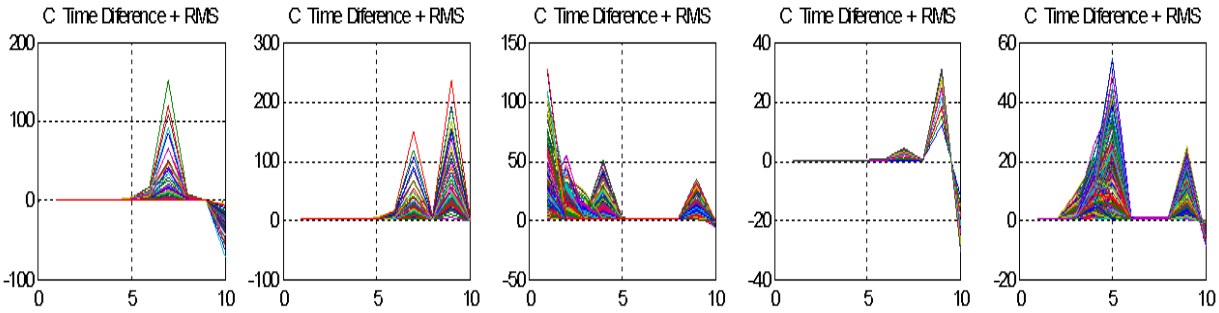
Las pruebas se realizan con datos (patrones) reales, estos patrones se utilizan para la identificación y caracterización de huecos de tensión, elevaciones de tensión, transitorios electromagnéticos de alta y baja frecuencia, transitorios tipo impulso, muescas de tensión, armónicos y *flicker*, estos patrones se obtienen a partir de la descomposición de las perturbaciones en diferentes niveles de energía utilizando la transformada *wavelet* y el valor eficaz del evento. En la Figura 27 se muestran los patrones obtenidos con las estrategias de identificación descritas. Estos patrones (ver Tabla 6) fueron obtenidos el proyecto de maestría del ing. Valdomiro Vega uno de los proyectos paralelos en que se está trabajando en la Escuela de ingenierías Eléctrica, Electrónica y de Telecomunicaciones de la Universidad Industrial, proyectos que tienen como objetivo detectar, identificar y clasificar estas perturbaciones.

Tabla 6. Características de los datos

Nombre de los datos	# datos	Dimensión	Clases
Patr C	2500	10	5

Elaborado por los autores

Figura 27. Patrones de identificación y caracterización de perturbaciones en las señales de tensión o corriente



Grafica elaborada en actual trabajo de maestría realizado por el Ing. Valdomiro vega

4.2.2 Selección del tipo de kernel

Las pruebas del *kernel* se realizan entrenando y clasificando los mismos datos, modificando en cada prueba el tipo de *kernel* que utiliza la máquina. Cada tipo de *kernel* trabaja con sus propios parámetros *Kernel* (σ1 y σ2 para polinomial y sigmoidal ,σ para RBF) según la clase de datos, por ello antes de mostrar resultados comparativos entre los diferentes tipos de *kernel* se deben encontrar los mejores parámetros de dicho *kernel*, en las Tablas 9, 11 y 13 se muestran las resultados de las diferentes pruebas que se realizaron para encontrar los mejores parámetros del *kernel* y C y en las Tablas 8, 10 y 12 se encuentran los parámetros del *kernel* y del C que se probaron utilizando los diferentes tipos de *kernel*.

Las siguientes pruebas siguen el procedimiento prueba-análisis, y las características de éstas aparecen en la Tabla 7.

Tabla 7. Características de la pruebas del tipo de kernel

Nombre de los datos	# datos Entrenamiento	Grupo prueba	Partes	Clases	Método optimización	Método descomp.
Patr C	500	2000	5	5	SMO	OVO

Elaborado por los autores *SMO optimización mínima secuencial *OVO uno contra uno

- **Pruebas utilizando Kernel Polinomial**

Inicialmente se utiliza un rango amplio, teniendo en cuenta que los parámetros son positivos, se toma desde 0.01 hasta 100, con un rango de parámetros de penalización C igualmente amplio de 1 a 1000, ver la Tabla 8.

Tabla 8. Parámetros del kernel y del C para la prueba Polinomial

	Parámetro de Penalización C				Parámetro Kernel 1			Parámetro Kernel 2		
					Inicio	Fin	paso	Inicio	Fin	paso
Prueba 1	1	10	100	1000	0,01	100	10	0.01	100	10
Prueba 2	1	10	100	1000	0,001	0,10	0,01	0,001	0,10	0,01
Prueba 3	1000				0,1	0,55	0,05	0,1	0,55	0,05

Elaborado por los autores

Los resultados de las pruebas con el *kernel* polinomial se muestran en la Tabla 9, la primera prueba presenta un alto número de vectores de soporte con un valor de exactitud bajo, lo cual indica que los parámetros utilizados no son los más adecuados. Para la prueba 2 se cambia el intervalo y se observa mejoría en el mejor valor de exactitud, probando claramente la efectividad y sencillez del método prueba-análisis. En el prueba 3 se encuentran exactitudes superiores al 90%. Si se suma el tiempo total resulta menos de 8 minutos de trabajo con una exactitud estimada de clasificación futura de 97,05% un resultado bastante bueno. Claramente se observa que continuando el procedimiento se pueden encontrar parámetros que brinden mayor exactitud si los requerimientos así lo exigen.

Tabla 9. Resultados utilizando el kernel polinomial

Nº de prueba	MSV					
	Exactitud Prueba (%)	NVS/datos	Tiempo Entrenam (seg.)	C	σ1	σ2
1	0,78	441/500	104	1000	0,01	0,01
2	0,882	343/500	321,78	1000	0,051	0,051
3	0,9705	204/500	20,235	1000	0,2	0,2

Elaborado por los autores *σ parámetro del kernel

- **Pruebas utilizando Kernel RBF**

Nuevamente se toma un rango amplio de valores para el primer intervalo de búsqueda (ver Tabla 10).

Tabla 10. Parámetros del kernel y del C para la prueba RBF

	Parámetro de Penalización C				Parámetro Kernel 1		
					Inicio	Fin	paso
Prueba 1	1	10	100	1000	0,01	100	10
Prueba 2	1	10	100	1000	0,001	0,1	0,01
Prueba 3	1000				0,01	0,1	0,01

Elaborado por los autores
Los resultados utilizando el *kernel* RBF se muestran en la Tabla 11.

Tabla 11. Resultados utilizando el kernel RBF

Nº de prueba	MSV				
	Exactitud Prueba (%)	NVS/datos	Tiempo Entrenam (seg.)	C	σ
1	0,9815	219/500	215,94	100	0,01
2	0,982	216/500	318,59	10	0,01
3	0,9925	134/500	36,37	10	0,03

Elaborado por los autores

- **Pruebas utilizando Kernel sigmoidal**

En la Tabla 12 aparecen los vectores de los parámetros del *kernel* y del C de las pruebas del *kernel* sigmoidal.

Tabla 12. Parámetros del kernel y del C para la prueba sigmoidal

	Parámetro de Penalización C				Parámetro Kernel 1			Parámetro Kernel 2		
					Inicio	Fin	paso	Inicio	Fin	paso
Prueba 1	1	10	100	1000	0,01	100	10	0,01	100	10
Prueba 2	1	10	100	1000	0,01	0,1	0,01	0,01	0,1	0,01
Prueba 3	1000				0,3	0,4	0,01	0,3	0,4	0,01

Elaborado por los autores

Los resultados utilizando el *kernel* sigmoidal muestran en la Tabla 13.

Tabla 13. Resultados utilizando el kernel sigmoidal

Nº de prueba	MSV					
	Exactitud Prueba (%)	NVS/datos	Tiempo Entrenam (seg.)	C	σ1	σ2
1	0,814	436/500	234,67	1000	0,01	0,01
2	0,962	238/500	235,14	1000	0,1	0,1
3	0,983	157/500	134,11	1000	0,4	0,4

Elaborado por los autores

4.2.3 Análisis de resultados de la prueba del *kernel*

En la Tabla 14 se muestran los valores de exactitud, tiempo y NVS para cada uno de los tipos de *kernel*, así como sus respectivos parámetros. Es importante tener en cuenta, que según pruebas preliminares, que si se continua con el procedimiento prueba-análisis pueden variar los resultados pero no de forma significativa como para cambiar la selección del tipo de *kernel* esta característica se puede apreciar en la Tabla 15, donde se puede observar el resultado después de 10 iteraciones.

Tabla 14. Comparación de resultados entre lo diferentes tipos de kernel

Kernel	MSV				
	Exactitud Prueba (%)	NVS/datos	Tiempo Entrenam (seg.)	C	σ
Polinomial	0,9705	204/500	**20,235**	1000	0,2
RBF	**0,9925**	134/500	36,370	10	0,03
Sigmoidal	0,9830	157/500	**134,110**	1000	0,4

Elaborado por los autores

Los valores registrados en la Tabla 15 muestran al *kernel* **RBF** como el más adecuado en tiempo de cómputo y exactitud, así mismo muestra como el emplear el *kernel* sigmoidal aumenta el tiempo de entrenamiento. Es necesario aclarar que el hacer una comparación entre los diferentes tipos de *kernel* es poco viable si no se especifica inicialmente lo que se busca, rapidez o exactitud. Según los resultados un *kernel* puede brindar mayor exactitud empleando un tiempo de entrenamiento mayor para un mismo intervalo de búsqueda lo cual se observa en este caso con el *kernel* sigmoidal el cual supera al polinomial.

Tabla 15. Comparación de resultados después de 10 iteraciones

KERNEL	MSV				
	Exactitud Prueba (%)	NVS/datos	Tiempo Entrenam (seg.)	C	σ
Polinomial	0,9735	191/500	37,114	1000	0,235
RBF	**0,9945**	126/500	**32,58**	17	0,031
Sigmoidal	0,9922	56/500	**47,23**	10000	0,41

Elaborado por los autores *σ1 = σ2 = σ *mejor *valor crítico

En la bibliografía revisada no existe una forma clara de seleccionar las mejores variables para una MSV para determinada clase de patrones o datos a clasificar. En este proyecto de grado se propone utilizar una figura de mérito donde se tiene en cuenta la exactitud y el tiempo como los factores más influyentes para determinar los mejores parámetros de la MSV (Ecuación 4.1).

$$\mathbf{FM} = EXACT + {}^{1}\!/_{t} \qquad (4.1)$$

Donde:
FM : figura de merito
$EXACT$: exactitud de prueba (exactitud de clasificación de los datos de prueba)
t : tiempo de entrenamiento en segundos

En la FM el factor dominante es como debe ser, la exactitud de clasificación de los datos de prueba (datos de prueba en la MSVToolbox1.0), cuyo valor máximo sería 1 es decir un acierto del 100%. El tiempo se toma como un factor que afecta, si es necesario realizar entrenamientos con grandes cantidades de datos.

Tabla 16. Figura de mérito de los diferentes tipos de kernel

Kernel	FM
Polinomial	1,0199
RBF	1,0200
Sigmoidal	0,9904

Elaborado por los autores

Analizando los valores de la FM de las pruebas realizadas con los diferentes tipos de *kernels* (ver Tabla 16), muestra como el *kernel* RBF y el polinomial registran los mejores valores confirmando la selección de estos como los más adecuados.

4.2.4 Selección del método de optimización

En estas pruebas se comparan los métodos de optimización SMO e IRWLS, otros métodos de optimización (loqo, qpsolver) fueron descartados por su altísimo costo computacional generando en muchos casos entrenamientos inconclusos debido a la falta de convergencia.

Las pruebas de selección del método de optimización se realizan con diferentes cantidades de datos (500, 400, 300 y 200) empleando el *kernel* RBF con el cual se

obtuvieron buenos resultados de exactitud (ver Tabla 14), de igual forma se utilizan los parámetros del *kernel* hallados en la sección 5.2.3.

4.2.5 Análisis de resultados de las pruebas del método de optimización

Los resultados de la Tabla 17 muestran poca diferencia en exactitud y tiempo de entrenamiento entre los métodos de optimización, por lo que funcionaria cualquiera de los dos indistintamente, pero según pruebas preliminares el método de optimización IRWLS presento para este tipo de datos; entrenamientos inconclusos (5%) por problemas de convergencia por lo cual se recomienda la utilización de SMO.

Tabla 17. Comparación de resultados de entrenamiento entre los métodos de optimización SMO e IRWLS

Nº de prueba	Método	MSV				
		Exactitud Prueba (%)	NVS	Tiempo Entrenam (seg.)	C	σ
1	SMO	0,9925	131/500	6,422	10	0,03
	IRWLS	0,9935	125/500	5,015	10	0,03
2	SMO	0,9922	87/400	4,32	10	0,03
	IRWLS	0,9911	111/400	4,11	10	0,03
3	SMO	0,9833	97/300	4,04	10	0,03
	IRWLS	0,9899	124/300	3,98	10	0,03
4	SMO	0,9846	117/200	3,27	10	0,03
	IRWLS	0,9905	47/200	3,33	10	0,03
Total	**SMO**	**0,9881**	**432**	**18,052**	**10**	**0,03**
	IRWLS	**0,9912**	**407**	**16,43**	**10**	**0,03**

Elaborado por los autores

La FM hallada para la selección del método de optimización se muestra en la Tabla 18.

Tabla 18. Figura de merito de los diferentes métodos de optimización

Método de optimización	FM
SMO	1,0435
IRWLS	1,0506

Elaborado por los autores

En este caso la FM muestra al método IRWLS como el más adecuado.

4.2.6 Selección del método de descomposición y reconstrucción

En esta sección se muestran los resultados de las pruebas realizadas con los diferentes métodos de descomposición OVO (uno contra uno), OVR (uno contra el resto), Ecoc_2c (corrección de errores de salida), así mismo se prueba con los diferentes tipos de *kernel* para determinar el comportamiento con cada uno de ellos. La prueba se realiza con los parámetros encontrados por validación cruzada en la sección 5.2.3, el tiempo de computo es por tanto el de entrenamiento de una sola máquina, por lo que es necesario tener en cuenta que cualquier diferencia de tiempo será multiplicada por el número de máquinas a entrenar, cuando se ejecute la validación cruzada.

4.2.7 Análisis de las pruebas del método de descomposición

Los valores de exactitud usando *kernel* polinomial y sigmoidal muestran claramente a OVO como el mejor método por encima de OVR y ECOC_2C. Al utilizar el *kernel* RBF los resultados en exactitud son muy similares, pero debido a que el uso de ECOC_2C u OVR aumenta el tiempo de entrenamiento, no los hace elegibles (ver Tabla 19).

Tabla 19. Resultados de las pruebas del método de descomposición

Kernel	Método de descomposición	MSV				
		exactitud Prueba	NVS/datos	Tiempo Entren(s)	C	σ
Polinomial	**OVO**	**0,97**	205/500	**3,625**	1000	0,2
	OVR	0,889	313/500	67,5	1000	0,2
	ecoc_2c	0,902	439/500	**209,06**	1000	0,2
RBF	OVO	0,994	133/500	8,797	10	0,03
	OVR	0,9905	151/500	95,156	10	0,03
	ecoc_2c	0,9945	172/500	12,36	10	0,03
Sigmoidal	OVO	0,9795	156/500	3,594	1000	0,4
	OVR	0,896	241/500	30,296	1000	0,04
	ecoc_2c	**0,8685**	423/500	103,36	1000	0,4

Elaborado por los autores

Utilizando los *kernels* polinomial y sigmoidal el tiempo de entrenamiento aumentó en forma crítica con los métodos ECOC_2C y OVR respecto al registrado con el método OVO, por lo cual se descartan para esta clase de datos. Para el *kernel*

RBF el método ECOC_2C la proporción de tiempo es de 1.4 veces la de OVO y al no ofrecer significativas ventajas en exactitud se descarta.

La FM (ver Tabla 20) confirma los análisis anteriores y muestra al método OVO como la forma de descomposición y reconstrucción más adecuada.

Tabla 20. Figura de mérito de los diferentes métodos de optimización

Kernel	Método de descomposición	FM
POLINOMIAL	OVO	1,246
	OVR	0,904
	ecoc_2c	0,907
RBF	OVO	1,029
	OVR	1,001
	ecoc_2c	1,075
SIGMOIDAL	OVO	1,258
	OVR	0,929
	ecoc_2c	0,878

Elaborado por los autores

Analizando los resultados de clasificación para cada clase y teniendo en cuenta que:

- La clase 1, corresponde a los huecos de tensión (*sags*)
- La clase 2, corresponde a las elevaciones de tensión (*swells*)
- La clase 3, corresponde a los transitorios oscilatorios
- La clase 4, corresponde a las fluctuaciones de tensión (*flicker*)
- La clase 5, corresponde a los armónicos

Se concluye que los huecos y elevaciones de tensión representados en los patrones por las clase 1 y 2 presentan los mayores errores 97 y 99,5% respectivamente. Estos resultados sirven como guía para revisar la calidad de los patrones obtenidos por diferentes métodos.

Después de analizar los resultados de las diferentes pruebas realizadas con una muestra de los patrones utilizados en la identificación y caracterización de perturbaciones electromagnéticas, se concluye que las mejores variables del la MSV para la clasificación específica de esta muestra de patrones son los siguientes:

Tipo de kernel
RBF
Parámetro del kernel y parámetro de penalización C
0.031 y 17 respectivamente
Método de optimización
SMO o IRWLS
Método de descomposición
OVO

En las simulaciones realizadas aplicando la metodología prueba análisis clasificando patrones de perturbaciones de las señales de tensión y corriente se encontró al *kernel* RBF y polinomial como los más adecuados para entrenamiento y clasificación de la muestra de patrones de dichas perturbaciones. Se comprobó además con los resultados de las pruebas del método de optimización que los dos métodos SMO e IRWLS son las mejores opciones aclarando algunos inconvenientes de convergencia del método IRWLS. Para el método de descomposición se muestra claramente los mejores resultados del OVO sobre los demás métodos.

4.3 PRUEBAS REALIZADAS A LA MSVTolbox1.0

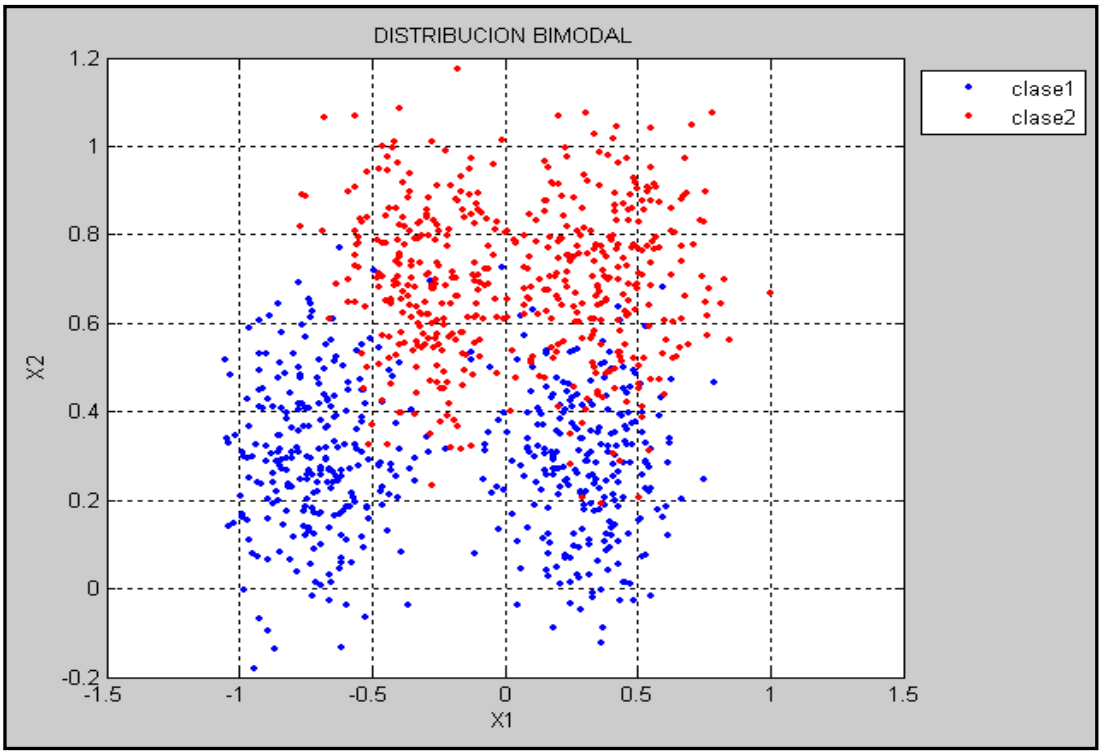
Para mostrar el funcionamiento y aplicación en la clasificación de la MSVtoolbox 1.0 se realiza entrenamiento y clasificación con tres tipos de datos reales (Bimodal, Elipse, new_thyroid). Estos datos son utilizados en diferentes tipos de investigaciones [INAOE, 06], la cantidad de datos utilizados para el entrenamiento es de un 10 a 30 % del total de los datos esto con el fin de observar la capacidad de generalización de la MSV. Seguidamente se muestran los resultados utilizando una mayor cantidad de datos.

Se inicia buscando los mejores parámetros para cada tipo de *kernel*, utilizando el método de optimización SMO y método descomposición OVO.

4.3.1 Datos Bimodal

El grupo de datos bimodal es bidimensional y esta compuesto por 1000 datos balanceados 500 por clase. Su distribución gráfica se observa en la Figura 28.

Figura 28. Gráfica de los datos Distribución Bimodal



Elaborado por los autores

❖ **Primer entrenamiento**

El primer entrenamiento se realiza con 1000 datos, 200 para entrenamiento y 800 datos de prueba, los vectores de prueba utilizando *kernel* polinomial, RBF y sigmoidal se observan en las Tablas 21,22 y 23 respectivamente.

- **Kernel Polinomial**

En la Tabla 21 aparecen los vectores de los parámetros del *kernel* y del C de las pruebas del *kernel* polinomial.

Tabla 21. Parámetros del kernel y del C para la prueba Polinomial

	Parámetro de Penalización C				Parámetro Kernel 1			Parámetro Kernel 2		
					Inicio	Fin	paso	Inicio	Fin	paso
Prueba 1	1	10	100	1000	1	10	1	1	10	1
Prueba 2	-	-	100	1000	2	4	0,2	2	4	0,2
Prueba 3	1	2	3	-	2,2	2,4	0,4	2,2	2,4	0,4

Elaborado por los autores

- **Kernel RBF**

En la Tabla 22 aparecen los vectores de los parámetros del *kernel* y del C de las pruebas del *kernel* RBF.

Tabla 22. Parámetros del kernel y del C para la prueba RBF

	Parámetro de Penalización C				Parámetro Kernel 1		
					Inicio	Fin	paso
Prueba 1	1	10	100	1000	1	2	0,2
Prueba 2	-	-	-	1000	1	3	0,5
Prueba 3	-	-	-	1000	0,8	1,1	0,05

Elaborado por los autores

- **Kernel Sigmoidal**

En la Tabla 23 aparecen los vectores de los parámetros del *kernel* y del C de las pruebas del *kernel* sigmoidal.

Tabla 23. Parámetros del kernel y del C para la prueba Sigmoidal

	Parámetro de Penalización C				Parámetro Kernel 1			Parámetro Kernel 2		
					Inicio	Fin	paso	Inicio	Fin	paso
Prueba 1	1	10	100	1000	10	15	1	10	15	1
Prueba 2	1	10	100	1000	0,1	1	0,1	0,1	1	0,1
Prueba 3	0,01	0,1	1	-	0,1	1	0,1	0,1	1	0,1

Elaborado por los autores

Los resultados de la Tabla 24 muestran el comportamiento de la MSV utilizando los tres tipos de *kernel*, para el *kernel* polinomial y RBF como se ha visto en pruebas anteriores los resultados son muy similares con porcentajes de exactitud superiores al 89% y tiempos de entrenamiento bajos. Los resultados par el *kernel* sigmoidal no son buenos y si se observa en al misma tabla el número de vectores de soporte es alto, lo cual como se ha mencionado, genera sobreentrenamiento y por ende malos resultados de clasificación.

Tabla 24. Resultados del primer entrenamiento con los datos bimodal

Kernel	Nº de prueba	MSV				
		Exactitud Prueba (%)	NVS/datos	Tiempo Entren(s)	C	σ
Polinomial	1	0,885	58/200	436,42	1000	1
	2	0,89	58/200	85,35	100	1
	3	0,898	57/200	37,9	100	1,05
RBF	1	0,895	65/200	57,62	100	3,01
	2	0,9	153/200	62,39	1	2,4
	3	0,9	151/200	37,9	1	2,28
Sigmoidal	1	0,34	126/200	19,133	1	14
	2	0,89	122/200	13,35	1	0,4
	3	0,887	150/200	165,62	0,3	0,6

Elaborado por los autores

En la Figura 28 se puede observar que la región de frontera entre los datos rojos y azules es más difícil de diferenciar por la presencia de las dos clases de datos y es precisamente en estas zonas donde se presenta el mayor error de clasificación de la MSV.

❖ **Segundo entrenamiento**

Ya que inicialmente se utilizaron pocos datos de entrenamiento para mostrar la capacidad de generalización de la MSV, se realiza un segundo entrenamiento con 332 datos y se valida con 668 datos.

Los resultados del segundo entrenamiento se muestran en la Tabla 25 y se puede observar que la exactitud de clasificación aumenta hasta alcanzar un 92%.

Siendo los datos elipse patrones no adecuados, al poseer solo dos características (bidimensional), que son las mínimas que posee un patrón, fueron clasificados de buena forma.

Tabla 25. Resultados del segundo entrenamiento con los datos bimodal

Kernel	ITERACIÓN	MSV				
		exactitud	NVS/datos	Tiempo Entrenam	C	σ
Polinomial	1	0,916	90/332	30,5	100	1,05
RBF	1	0,920	90/332	157,5	10	2,15
Sigmoidal	1	0,89	193/332	48,32	1	0,35

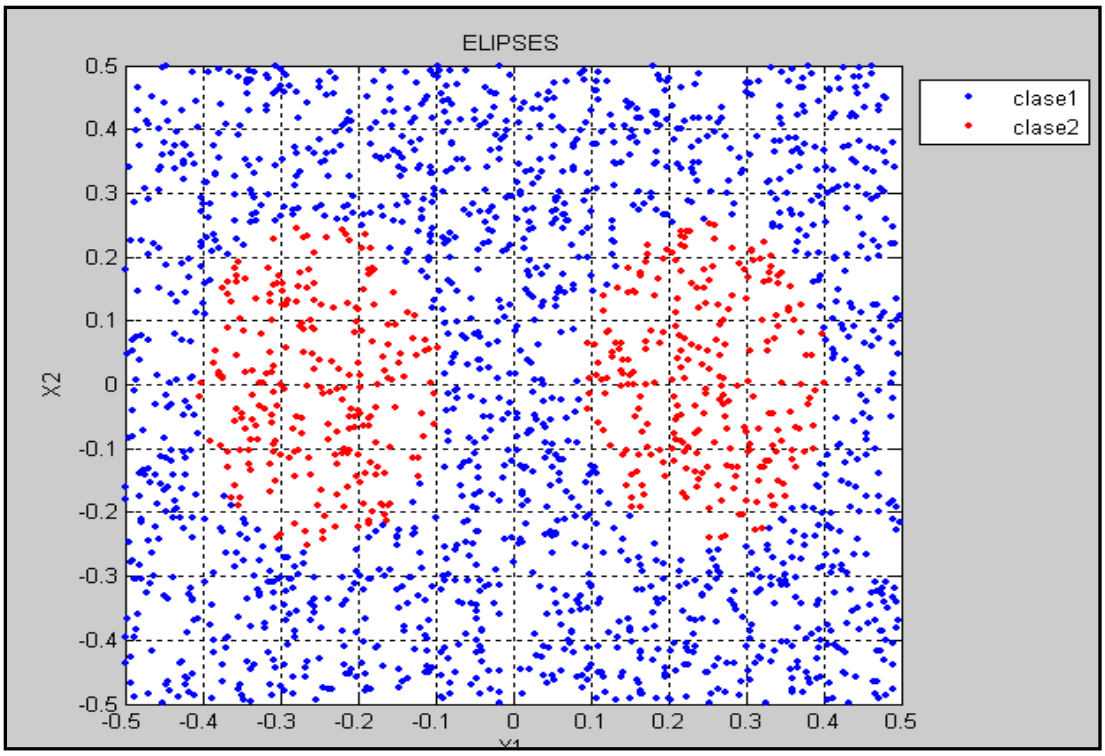
Elaborado por los autores

Teniendo en cuenta lo anterior se puede concluir que la MSV generaliza y muestra buenos resultados con poca información resultando adecuada para la clasificación de este tipo de patrones.

4.3.2 Datos elipse

El archivo de datos elipses es bidimensional y está compuesto por 2000 datos no balanceados de [1513 ,487] datos por clase, su distribución se puede observar en la Figura 29.

Figura 29. Gráfica de los datos Elipse



Elaborado por los autores

❖ **Primer entrenamiento**

La prueba se realiza con 2000 datos, 200 para entrenamiento y 1800 datos de prueba

Los vectores de prueba utilizando *kernel* polinomial, RBF y sigmoidal se observan en las Tablas 26, 27 y 28 respectivamente.

- **Kernel polinomial**

En la Tabla 26 se presentan los vectores de los parámetros del *kernel* y del C de las pruebas del *kernel* polinomial.

Tabla 26. Parámetros del kernel y del C para la prueba Polinomial

	Parámetro de Penalización C				Parámetro Kernel 1			Parámetro Kernel 2		
					Inicio	Fin	paso	Inicio	Fin	paso
Prueba 1	1	10	100	1000	10	15	1	10	15	1
Prueba 2	-	-	100	1000	15	20	1	15	20	1
Prueba 3	-	-	100	1000	10	100	5	10	100	5
Prueba 4	-	-	100	1000	1	100	5	1	1000	5

Elaborado por los autores

La prueba número 4 no se registra en la Tabla 29 por ser un entrenamiento inconcluso. El valor de C=1000 es muy alto para este intervalo lo cual genera un entrenamiento extremadamente demorado (horas).

- **Kernel RBF**

En la Tabla 27 se presentan los vectores de los parámetros del *kernel* y del C de las pruebas del *kernel* RBF.

Tabla 27. Parámetros del kernel y del C para la prueba Kernel

	Parámetro de Penalización C				Parámetro Kernel 1		
					Inicio	Fin	paso
Prueba 1	1	10	100	1000	1	10	1
Prueba 2	-	-	-	1000	0,1	1	0,1
Prueba 3	-	-	-	1000	0,01	0,1	0,01

Elaborado por los autores

- **Kernel Sigmoidal**

En la Tabla 28 se presentan los vectores de los parámetros del *kernel* y del C de las pruebas del *kernel* sigmoidal.

Tabla 28. Parámetros del kernel y del C para la prueba Sigmoidal

	Parámetro de Penalización C				Parámetro Kernel 1			Parámetro Kernel 2		
					Inicio	Fin	paso	Inicio	Fin	paso
Prueba 1	1	10	100	1000	10	15	1	10	15	1
Prueba 2	1	10	100	1000	1	5	0,5	1	5	0,5
Prueba 3	0,01	0,1	1	-	0,1	1	0,1	0,1	1	0,1

Elaborado por los autores

La Tabla 29 muestra mejores resultados para el *kernel* RBF con alta exactitud pero con mayor tiempo de entrenamiento, los resultados superiores al 95% hacen de la MSV una excelente herramienta para la clasificación de este tipo de datos. El *kernel* sigmoidal presenta resultados no adecuados por lo que no se utiliza para estos patrones en particular.

Tabla 29. Resultados del primer entrenamiento con los datos elipse

Kernel	**Nº de prueba**	**MSV**				
		exactitud	**NVS/datos**	**Tiempo Entrenam**	**C**	**σ**
Polinomial	1	0,936	19/200	13,99	100	15
	2	0,923	18/200	21,05	1000	20
	3	0,89	18/200	17,48	1000	100
RBF	1	0,84	88/200	93,7	1000	1
	2	0,89	79/200	113,9	1000	0,8
	3	0,96	18/200	68,52	1000	0,25
Sigmoidal	1	0,59	81/200	6,3	1000	15
	2	0,60	74/200	7,56	10	5
	3	0,76	101/200	21,5	10	0,3

Elaborado por los autores

❖ **Segundo entrenamiento**

Se realiza un segundo entrenamiento con 400 datos y se valida con 1600 datos, los resultados se muestran en la Tabla 30.

Tabla 30. Resultados del segundo entrenamiento con los datos elipse

Kernel	Nº de prueba	MSV				
		exactitud	NVS/datos	Tiempo Entrenamiento	C	σ
Polinomial	1	0,983	18/400	134,66	1000	20
RBF	1	0,983	37/400	57,32	100	0,25
Sigmoidal	1	0,756	202/400	51,32	100	0,05

Elaborado por los autores

Como se observa en al tabla 30 al utilizar mayor cantidad de datos de entrenamiento los resultados para el *kernel* RBF y polinomial mejoran hasta alcanzar 98%, el resultado con sigmoidal sigue siendo malo ya que presenta el problema de exactitud 100% en una clase y 0% en la otra por lo cual se descarta para este tipo de datos.

4.3.3 Datos New_thyroid

New_thyroid son datos reales no balanceados es decir tiene cantidades diferentes de datos por clase. Este archivo tiene 215 datos de dimensión 5 y tres clases, el número de datos por clase es: [150 35 30]

Se realizan dos entrenamientos con cantidades de datos diferentes Inicialmente se utilizan 215 datos, 86 para entrenamiento y 129 para prueba. Los vectores de prueba utilizando *kernel* polinomial, RBF y sigmoidal se observan en las Tablas 31, 32 y 33 respectivamente.

- **Kernel polinomial**

En la Tabla 31 se presentan los vectores de los parámetros del *kernel* y del C de las pruebas del *kernel* polinomial.

Tabla 31. Parámetros del kernel y del C para la prueba polinomial

	Parámetro de Penalización C				Parámetro Kernel 1			Parámetro Kernel 2		
					Inicio	Fin	paso	Inicio	Fin	paso
Prueba 1	1	10	100	1000	1	10	1	1	10	1
Prueba 2	-	-	-	1000	0,7	1,3	0,1	0,7	1,3	0.1
Prueba 3	1	10	100	1000	1	1,7	0,5	1	1,7	0,5
Prueba 4	1	10	100	1000	1	1,5	0,03	1	1,5	0,03

Elaborado por los autores

- **Kernel RBF**

En la Tabla 32 se presentan los vectores de los parámetros del *kernel* y del C de las pruebas del *kernel* RBF.

Tabla 32. Parámetros del kernel y del C para la prueba RBF

	Parámetro de Penalización C				Parámetro Kernel 1		
					Inicio	Fin	paso
Prueba 1	1	10	100	1000	1	10	1
Prueba 2	-	-	-	1000	0,1	1	0,1
Prueba 3	-	-	-	1000	0,02	0,5	0,01

Elaborado por los autores

- **Kernel Sigmoidal**

En la Tabla 33 se presentan los vectores de los parámetros del *kernel* y del C de las pruebas del *kernel* sigmoidal.

Tabla 33. Parámetros del kernel y del C para la prueba Sigmoidal

	Parámetro de Penalización C					Parámetro Kernel 1			Parámetro Kernel 2		
						Inicio	Fin	paso	Inicio	Fin	paso
Prueba 1	1	10	100	1000	-	1	10	1	1	10	1
Prueba 2	-	-	100	1000	10000	0,1	2	0,01	0,1	2	0,01
Prueba 3				1000	10000	0,2	0,3	0,01	0,2	0,3	0,01

Elaborado por los autores

La Tabla 34 muestra los resultados de las pruebas realizadas a los datos New_thyroid empleando los diferentes tipos de *kernel*. Dados que los datos New_thyroid poseen más información que los bimodal y elipse (5 dimensiones) los resultados son excelentes para cualquier tipo de *kernel.*

Tabla 34. Resultados del primer entrenamiento con los datos new_thyroid

Kernel	Nº de prueba	MSV				
		exactitud	NVS/datos	Tiempo Entrenam	C	σ
Polinomial	1	0,96	21/86	15,71	1000	1
	2	0,96	21/86	2,23	1000	1
	3	0,89	42/86	17,45	1	5
	4	0,97	18/86	14,40	1000	1,3
	5	0,97	17/86	10,82	1000	1,3
RBF	1	0,96	20/86	4,42	1000	1
	2	0,95	24/86	33,59	100	0,4
	3	0,96	30/86	109,89	100	0,27
Sigmoidal	1	0,90	19/86	3,96	1000	1
	2	0,96	16/86	53,78	10000	0,25
	3	0,96	17/86	7	10000	0,2

Elaborado por los autores

❖ **Segundo entrenamiento**

Se efectúa un segundo entrenamiento esta vez con 129 datos de entrenamiento y 86 de prueba los resultados se muestran en al Tabla 35

Tabla 35. Resultados del segundo entrenamiento con los datos new_thyroid

Kernel	Nº de prueba	MSV				
		exactitud	NVS/datos	Tiempo Entrenamiento	C	σ
Polinomial	1	1,00	23/129	4,39	1000	1,5
RBF	1	1,00	26/129	29,06	1000	0,99
Sigmoidal	1	0,97	32/129	5,89	1000	0,3

Elaborado por los autores

Se comprueba la efectividad de la MSV con buenos patrones para este caso se logran resultados de clasificación perfectos para los *kernel* polinomial y RBF en el caso del polinomial en menos de 5 segundos se realiza el entrenamiento y clasificación con 100% de exactitud.

Revisando los resultados de las diferentes pruebas de clasificación de datos realizadas con la herramienta computacional MSVToolbox1.0 se concluye que es aplicable en la clasificación de patrones de diferentes calidades. Muestra gran capacidad de generalización al obtenerse resultados de clasificación de 100% de exactitud, es importante observar que el resultado final de clasificación depende

en gran medida de que tan diferenciados son los patrones de entrenamiento. Se comprobó estos al clasificar datos de 2 dimensiones (bimodal, elipse) y 5 dimensiones (New_thyroid), obteniéndose mejores resultados con los últimos.

5. OBSERVACIONES Y CONCLUSIONES

5.1 CONCLUSIONES

- Se desarrolló la herramienta computacional **MSVToolbox1.0** interfaz gráfica que permite clasificar automáticamente patrones almacenados en una base de datos en formatos o extensiones estándar (.mat, .txt y csv).

- Las pruebas realizadas durante este trabajo muestran que las máquinas de soporte vectorial MSV son una gran herramienta de clasificación, probando su eficiencia al mostrar resultados por encima del 99% de exactitud en la clasificación de patrones característicos de las perturbaciones de señales de tensión o corriente.

- A partir de las pruebas realizadas con los patrones utilizados en la identificación y caracterización de perturbaciones electromagnéticas, se concluye que el *kernel* RBF es el más adecuado para trabajar este tipo de patrones, con parámetro sigma 0.031 y parámetro de penalización C de 17. Se llegó a la conclusión que los dos métodos de optimización SMO e IRWLS son las mejores opciones salvo algunos inconvenientes de convergencia del método IRWLS. El método de descomposición que mostró mejores resultados fue OVO (one versus one).

- Inicialmente las pruebas realizadas para la búsqueda de parámetros de *kernel* y parámetro de penalización C se realizan con grandes vectores con lo cual resultaban simulaciones de larga duración (días), por lo que se decidió probar otro tipo de metodología de trabajo. Con la ayuda de la gráfica entregada por la toolbox se realizan simulaciones cortas con pocos parámetros de prueba pero con amplio rango, buscando acercarse en cada prueba a los mejores parámetros, se probó (con los diferentes datos descritos en el capítulo 5) que en solo tres pruebas se encuentran exactitudes superiores al 90% con los patrones menos adecuados y del 99% con patrones bien diferenciados.

5.2 OBSERVACIONES

- Al ser la teoría de clasificación con MSV una herramienta relativamente innovadora en nuestro entorno, se muestra en la sección 1.5 un ejemplo que

ilustra el principio básico de funcionamiento de la MSV con el cual se demostró su aplicabilidad en aplicaciones de clasificación de datos.

- La toolbox trabaja con diferentes tipos de datos, pensando en esto se diseñó para realizar la validación cruzada y la búsqueda en malla de forma fácil, además se implementó una gráfica que muestra los resultados de esta búsqueda facilitando el análisis de los parámetros *kernel* y de penalización C. La gráfica ha sido un aporte original a la teoría de búsqueda en malla mediante la validación cruzada ya que las que se encuentran en la bibliografía son confusas, están en tres dimensiones, por lo que los resultados mostrados no son fáciles de interpretar.

- La MSV toolbox fue diseñada (o dividida) en tres módulos; Validación cruzada y Búsqueda en malla, Entrenamiento y Clasificación, con el fin de facilitar su uso. Cuenta con ayudas que le facilitan al usuario disipar cualquier duda, también se incluyó información teórica de las MSV para facilitar la comprensión de esta técnica ayudando al usuario a realizar mejores búsquedas.

- En la toolbox se incluyeron tres tipos de *kernel*; polinomial, RBF y sigmoidal. Ya que según las pruebas el tener diferentes tipos de *kernels* posibilita trabajar con una gran variedad de tipo de datos. Por la misma razón se han incluido dos métodos de optimización SMO e IRWLS, cada uno con características diferentes para brindarle al usuario diferentes opciones de trabajo según su elección. También se implementó con tres métodos de descomposición dando la posibilidad de escoger entre las características tiempo y/o exactitud que según las pruebas realizadas en el capítulo 5 son las más importantes.

- Existen muchos métodos de descomposición, aunque en la MSVToolbox1.0 solo se implementaron dos; SMO e IRWLS, otros tales como LOGO y QPSOLVER se descartaron por ofrecer una exactitud menor a los escogidos, aunque se pudieron implementar todos se dejaron solo dos para brindarle la seguridad al usuario de que elegirá solo entre las mejores opciones y no se arriesga a escoger un método no probado. La principal diferencia entre estos dos métodos escogidos es el tiempo, IRWLS es mucho más rápido que smo, pero presentando el inconveniente de la no convergencia para algunos tipos de datos. Por el contrario, con SMO se tiene la certeza que sin importar la naturaleza de los datos este algoritmo ofrece una respuesta.

- En la toolbox se implementaron tres métodos de descomposición; OVR, OVO Y ECOC_2C, estos son los métodos de descomposición más usados en MSV. Según la bibliografía ECOC_2C es el mejor método por incluir todo el conjunto

de entrenamiento en todas las máquinas biclasificadoras que la conforman y por ser más robusto contra fallos. En OVO los biclasificadores son entrenados con datos extraídos de solo dos clases del conjunto de entrenamiento por lo que la varianza es mayor, aunque en las pruebas realizadas se obtuvo una muy buena exactitud similar al ECOC_2C, ofreciendo la ventaja de ser mucho más rápido comparado con los otros dos métodos. OVR entrena con todo el conjunto de entrada pero su exactitud esta por debajo de OVO y ECOC_2C.

- El tipo de *kernel* que se debe utilizar depende de la naturaleza de los datos y solo se sabrá cuál es el apropiado realizando pruebas, no obstante se recomienda iniciar con RBF, basándose en la bibliografía y según pruebas realizadas, ya que este ofrece buenos resultados. De la misma forma se recomienda iniciar el estudio de los datos usando el método de descomposición ovo, por ser el más rápido y según las pruebas ofrece una gran exactitud.

- En la bibliografía no se encontró un método recomendado para elegir los mejores parámetros después de haber realizado la validación cruzada, estos se escogen con la máxima exactitud y si hay empate se escoge el de menor desviación estándar, pero si sigue existiendo empate ¿qué decisión se debe tomar?. El método de elección de los parámetros escogido para resolver este problema es el siguiente:

- Se busca cuáles parámetros del *kernel* tuvieron más veces esa mayor exactitud, si existe un parámetro predominante sobre los demás, se escoge este parámetros con sus respectivas parejas C y se vuelve a entrenar con la totalidad de los datos, luego se halla la exactitud de clasificación con los datos de prueba, y si vuelve a haber empate se escoge dentro de este grupo de parámetros el de mayor C. Si los datos de prueba no existen, se entrena solo una MSV; escogiendo la pareja de parámetros que correspondan al C más alto dentro de este nuevo grupo de parámetros. El resultado que se entrega es el modelo generado con el par de parámetros escogido. Este procedimiento se hace de esta forma pensando en que si ese parámetro tiene el mejor historial en máxima exactitud, debe ser el mejor parámetro dentro de esta búsqueda para ese tipo de datos. Se escoge el más alto C con el objetivo de aumentar la penalización al ruido.

  Si no llega a existir un parámetro predominante o si son demasiados y por cuestión de tiempo se hace prohibitivo su entrenamiento se entrena solo una máquina con la totalidad de los datos con los parámetros más altos del C y

kernel, si hay datos de prueba se halla la exactitud de prueba y este es el resultado que se ofrece.

5.2.1 Análisis del Algoritmo Validación cruzada y búsqueda en malla

- En el algoritmo de validación cruzada el tiempo de cómputo se encuentra concentrado en la **función de optimización**; ocupando más del 95% del tiempo total. Entonces de forma aproximada se puede decir que el tiempo de ejecución de validación cruzada depende del número de veces que se ejecuta esta función y de lo que se tarde o demore cada ejecución. El tiempo de cómputo de la función de optimización depende de la cantidad de datos de entrenamiento, a su vez el número de veces que se ejecute depende del método de descomposición, del número de clases que contengan los datos y del número de máquinas multiclasificadoras; la cantidad de máquinas depende del número de parámetros C, del número de parámetros *kernel* y del número de partes en las que se divida los datos.

$$\begin{pmatrix} Número \\ de\,máquinas \\ Multiclasificadoras \end{pmatrix} = número\ de\ parametros\ \ C * número\ de\ parametros\,del\,kernel * partes$$

  Por lo tanto es muy importante tener en cuenta estas variables para definir el entrenamiento.

- Otro factor importante de la validación cruzada es su efectividad (que la prueba ofrezca buenos resultados), en este aspecto se debe tener en cuenta el número de partes en las que se divide el grupo de entrenamiento, este no debe ser muy pequeño por que diminuye la cantidad de datos para entrenar afectando la función de decisión de cada MSV biclasificadora, perdiendo la capacidad de generalización. En caso contrario, si el número de partes es muy alto se eleva el tiempo de cómputo, además el grupo de prueba quedará con muy pocos datos, por lo que su validación no será suficientemente confiable.

6.2.2 Pruebas MSVToolbox1.0

- En el capítulo 4 se probó la MSVtoolbox1.0 con diferentes tipos de datos obteniendo muy buenos resultados en promedio por encima del 93%. las pruebas realizadas con patrones poco diferenciados son las que brindan peores resultados en 90%, de igual forma se obtuvieron resultados perfectos 100% con patrones de alta calidad (datos bien diferenciados).

- En el entrenamiento de la MSV se pueden llegar a utilizar gran cantidad de datos por lo que se sugiere realizar una búsqueda en malla previa con una parte de los datos, aplicando el método descrito en la sección 4.1, de esta forma se estaría atacando el problema de tiempo de cómputo inherente a las MSV. Teniendo ya los mejores parámetros ajustados a los datos, el entrenamiento final será mucho más rápido, es necesario aclarar que el hecho de utilizar menos datos no repercute significativamente en la calidad de modelo obtenido ya que el entrenamiento final se hace con la totalidad de los datos.

- Como un apoyo al evaluar variables de la MSV como lo son el tipo de *kernel* método de optimización y demás, se propone una figura de mérito FM que muestra claramente las mejores variables para una aplicación específica.

- Se deja como una posible investigación futura la automatización del método prueba- análisis implementado en este trabajo para la obtención de parámetros para una configuración especifica de patrones

- Finalmente es interesante rescatar la gran formación como investigadores íntegros que ha dejado la realización de este proyecto ya que muestra que usar investigaciones anteriores para hacer aportes y mejoras es un forma de contribuir al desarrollo tecnológico de un país en este caso se extracto información de grades y pequeños investigadores, complementándolas con nuevas las ideas de los autores lo cual permitió obtener un producto final de alta calidad y con grandes posibilidades de aplicabilidad actual.

REFERENCIAS BIBLIOGRÁFICAS

[Wang, 01] *Classification of power quality disturbances using time-frequency ambiguity plane and neural networks,* Power Engineering Society Summer Meeting, 2001. IEEE

[NTC 5000, 02] Norma Técnica Colombiana 5000: "*Calidad de la potencia eléctrica* (*CPE). Definiciones y términos fundamentales",* Instituto Colombiano de Normas Técnicas (ICONTEC), 2002.

[IEEE 1159, 95] IEEE Std 1159-1995, CEI 61000-4-7 "*Compatibilidad Electromagnética (CEM)*".

[Bahamonde, 04], Bahamonde, A. "Proyecto SIBAOC", Centro de Inteligencia Artificial, Universidad de Oviedo, http://www.aic.uniovi.es/, 2004.

[Murphey & Zhihang, 03], Murphey, Y.L. Zhihang Chen Putrus, M. Feldkamp, L. "SVM learning from large training data set,", Dept. of Electr. & Comput. Eng., Michigan Univ., Dearborn, MI, USA; articulo IEEE, Neural Networks, 2003. Proceedings of the International Joint Conference on.

[Hilera & Martinez, 95] Hilera, J. y Martinez, V. "*Redes neuronales artificiales*: *fundamentos modelos y aplicaciones*". Ra-ma. Madrid, 1995.

[GNU, 03] Herramientas en GNU/Linux para estudiantes universitarios 2003, http://es.tldp.org/Presentaciones/200304curso-glisa/redes_neuronales/curso-glisa-redes_neuronales-html/x24.html.

[Wassermann, 89] Wassermann, P.D."*Neural computing: Theory and Practice."* VNR. New York, 1989.

[Estadístico, 01] Construcción de Modelos de RN, 2001 http://www.estadistico.com/arts.html?20010129.

[Wikipedia, 06] red neurona artificial, 2006 http://es.wikipedia.org/wiki/Red_neuronal_artificial.

[Esi, 04] Sistemas expertos, 2004 http://www.esi2.us.es/~dco/sistemas.htm

[Gc, 02] Taxonomía de las Redes Neuronales www.gc.ssr.upm.es/inves/neural/ann2/concepts/taxonomy.htm.

[Quinlan, 91] Quinlan. "*Connectionism and Psychology*" Harvester Wheateaf. N.Y., 1991.

[Rumelhart & McClelland, 88] Rumelhart, D.E. y McClelland, JL. "*Explorations in Parallel Distributed Processing. A Handbook of Models. Programs and Exercices"*. Cambridge, MA: MIT Press., 1988.

[Fausett, 94] Fausett, L. "*Fundamentals of Neural Networks*", Prentice-Hall, 1994. ISBN 0 13 042250 9.

[Zadeh, 65] Zadeh, L. "Fuzzy Sets, *Information and Control",* Vol. 8, pp. 338-353, 1965

[Kaufmann, 82] Kaufmann, A. "*Introducción a la teoría de los subconjuntos borrosos"*, Cía. Editorial Continental, 1982.

[Wikipedia, 06]. http://www.wikipedia.com

[Ibarreta, 06] Ibarreña, J. "*Sistemas Expertos: Áreas de aplicación*". Geocities. URL: http://www.geocities.com/javierml.geo/doc/SistemasExpertos.html

[Canca, 06] Canca, J. "*Sistemas Expertos*". Esi2.us.es. URL: http://www.esi2.us.es/~dco/sistemas.htm

[Cruz, N], Cruz N, Method Based on Genetic Algorithms and Fuzzy Logic to Induce Bayesian Networks, International Conference on Computer Science ENC2004, Septiembre 2004.

[Vega & Duarte & Ordoñez, 06] Vega, V; Duarte, C; Ordoñez, G. "Automatic Power Quality Disturbances Detection and Classification Based on Discrete Wavelet Transform and Artificial Intelligence In: Transmission and Distribution Conference and exposition", PES IEEE 2006, Caracas.

[Burges; Schölkopf & Smola, 99] Burges, C; Schölkopf, B.; Smola, A. "*Advances in kernel methods: Support vector machines*". Cambridge, MA: MIT Press, 1999.

[Burges vol. 2, 98] Burges, C. "*A tutorial on support vector machines for pattern recognition*". Data Mining and Knowledge Discovery, vol. 2, no. 2, 1998.

[Vapnik, 95] Vapnik, V. "*The nature of statical learning theory*". Springer Verlag, New York, 1995.

[Vapnik, 98] Vapnik, V. "*Statistical learning treory*". Wiley, New York, 1998.

[Cherkassky & Mulier, 98] Cherkassky, V. y Mulier, F. "*Learning from data*: *concepts, theory, and methods*". John Wiley and Sons, New York, (1998).

[Burges, 98] Burges, Christopher. "*A tutorial on support vector machines for pattern recognition*". Data Mining And Knowledge Discovery, 2:121_167, 1998.

[Schölkopf & Smola, 02] Schölkopf, Bernhard y Smola, Alex. "*Learning with Kernels Support Vector Machines, Regularization, Optimization and Beyond*". The MIT Press, Cambridge, 2002.

[Kuhn & Tucker,51] Kuhn, H. W. y Tucker, A. W. "*Nonlinear programming. Proc. 2nd Berkeley Symposium on Mathematical Statistics and Probabilistics*", pages:481_492, 1951. University of California Press.

[Cortes & Vapnik, 95] Cortes, C. y Vapnik, V. "*Support vector networks. Machine Learning*", 20:273_297, 1995.

[Bennett & Mangasarian, 92] Bennett, K. P. y Mangasarian, O. L. "*Robust linear programming discrimination of two linearly inseparable set. Optimization Methods and Software*", 1:23_34, 1992.

[Boser; Guyon & Vapnik, 92] Boser, B. ; Guyon, I. ; Vapnik, V. "*A training algorithm for optimal margin classi_er. Proc. 5th ACM Workshop on Computational Learning Theory*", pages:144_152, 1992.

[Cristianini, 01] Nello Cristianini, Support Vector and Kernel Machines, Biowulf Technologies, 2001

[Weston & Weston, 99] Weston, J. y Weston, C. "*Multi-Class Support Vector Machines*". In M. Verleysen, editor, Proceedings ESANN, Brussels, 1999.

[*Blanz; Vapnik & Burges, 95] Blanz, V. ; Vapnik, V. ; Burges, C."Multiclass Discrimination with an Extended Support Vector Machine*". Talk given at AT&T Bell Labs, 1995.

[Angúlo, 01] Angúlo, Cecilo. "*Aprendizaje con máquinas núcleo en entornos de multiclasificación*" Tesis Doctoral, Universidad Politécnica de Catalunya, 2001.

[Dietterich & Bakiri, 95] Dietterich, T. G. y Bakiri, G. "*Solving multiclass learning problems via error-correcting output codes*". Journal of Arti_cial Intelligence Research, 2:263_286, 1995.

[Robaina; Cruz, 00] Robaina, E. ; Diaz, R. ; Morales, R. ; Cruz, C. *"aplicación de la lógica borrosa a la clasificación diagnóstica en Cuidados Intensivos*",cuba 2000.

[Muñoz; Ibargüen & López, 05] Muñoz, P. ; Ibargüen, F. ; López, J. "*Reconocimiento de Caracteres Manuscritos Usando Máquinas de Vectores de Soporte", X Simposio de tratamiento de señales, imágenes y visión artificial"*. Universidad del Valle, Santiago de Cali, Septiembre 14-16 de 2005.

[Morales & Gomes, 05] Morales, German y Gómez, Alvaro. "*estudio e implementación de una herramienta basada en máquinas de soporte vectorial aplicada a lalocalización de fallas en sistemas de distribución*", proyecto de grado dirigida por Hermann Vargas Torres. Universidad Industrial de Santander, 2005.

[Martínez W & Martínez A, 02], Martínez, W. ; Martínez, A. "computational statistics handbook with MATLAB", Chapman & hall / CRC, 231-239, 2002.

[INAOE, 06], "coordinación de ciencias computacionales". Instituto Nacional de Astrofísica, Óptica y Electrónica. Electrónica. http://ccc.inaoep.mx

BIBLIOGRAFÍA

Boser, B; Guyon I; Vapnik V. "*A training algorithm for optimal margin classifier*". Proc. 5th ACM Workshop on Computational Learning Theory, pages:144_152, 1992.

Burges, C. ; Schölkopf, B. ; Smola A. " *Advances in kernel methods: Support vector machines*". Cambridge, MA: MIT Press, 1999.

Burges, C. "*A tutorial on support vector machines forpattern recognition*". Data Mining and Knowledge Discovery, vol. 2, no. 2, 1998.

Comisión de Regulación de Energía y Gas – CREG, "*Resolución CREG 070 de Mayo de 1998: Reglamento de distribución de energía eléctrica*". Bogotá, 1998.

Norma Técnica Colombiana – IEC 61000-1-1: "*Compatibilidad electromagnética. Parte I: Generalidades. Sección I: Aplicación e interpretación de definiciones y términos fundamentales*", Instituto Colombiano de Normas Técnicas (ICONTEC), 2006.

Electrotek Concepts, Inc. [online]. Knoxville, Tennessee, USA. Actualizado 10 Nov

Castro, G. "*Tendencias de los Sistemas Expertos*". Netmedia.info URL: http://www.netmedia.info/netmedia/articulos.php?id_sec=32&id_art=2255

Criado, J. "*Sistemas Expertos*" Worldonline.es URL: http://home.worldonline.es/jmariocr/

Velarde, J. "*Sistemas Expertos"*. URL: http://www.ucm.es/info/eurotheo/diccionario/S/sistemas_expertos.htm

Chapa, S. "*Arquitectura de sistemas expertos*".URL http://delta.cs.cinvestav.mx/~schapa/red/intro_lm/node46.html

Samper, J. "*Sistemas Expertos: El conocimiento al poder*". Psicología.com URL: http://www.psycologia.com/articulos/ar-jsamper01_2.htm

ANEXOS

Anexo A. MANUAL DE LA MSVToolbox1.0

En este capítulo se hace la presentación de la herramienta computacional MSVToolbox1.0. Es la aplicación desarrollada en este proyecto para la clasificación automática de patrones. En este manual se explica el funcionamiento y características principales de cada una de las funciones incorporadas en ella.

La MSVTolbox1.0 es una herramienta que permite clasificar datos automáticamente, utilizando la técnica de inteligencia artificial Máquinas de Soporte Vectorial. Para realizar la clasificación es necesario entrenar previamente la MSVToolbox1.0 con un grupo representativo de los datos a clasificar. Después del entrenamiento, la máquina entrega un modelo, con el cual se clasifican nuevos datos.

Los datos de entrenamiento pueden ser n dimensionales, cada uno de ellos debe tener su respectiva etiqueta (número entero que representa la clase a la cual pertenece) y la estructura que se muestra en la Figura A.1.

La forma de separar las componentes de cada dato dentro de esta estructura es por medio de espacios de igual forma se debe separar la etiqueta del dato. Estos datos pueden estar en los formatos .mat, .txt, o csv

Figura A. 1. Matriz de los datos de entrenamiento

$$\left[\begin{array}{c|ccc} etiqueta & \multicolumn{3}{c}{dato} & \\ 1 & x_{1a} & \dots & x_{na} & \rightarrow dato\ 1 \\ 3 & x_{1b} & \dots & x_{nb} & \rightarrow dato\ 2 \\ : & : & \dots & : & \\ 1 & x_{1d} & \dots & x_{nd} & \rightarrow dato\ n \end{array}\right]$$

Elaborado por los autores

INSTALACIÓN

Los archivos necesarios para el correcto funcionamiento se encuentran en el CD de instalación. Para instalar los archivos es necesario seguir los siguientes pasos:

Introducir el CD que contiene la aplicación en la unidad correspondiente.
Se debe copiar la carpeta **MSVToolbox1.0** en un directorio determinado.

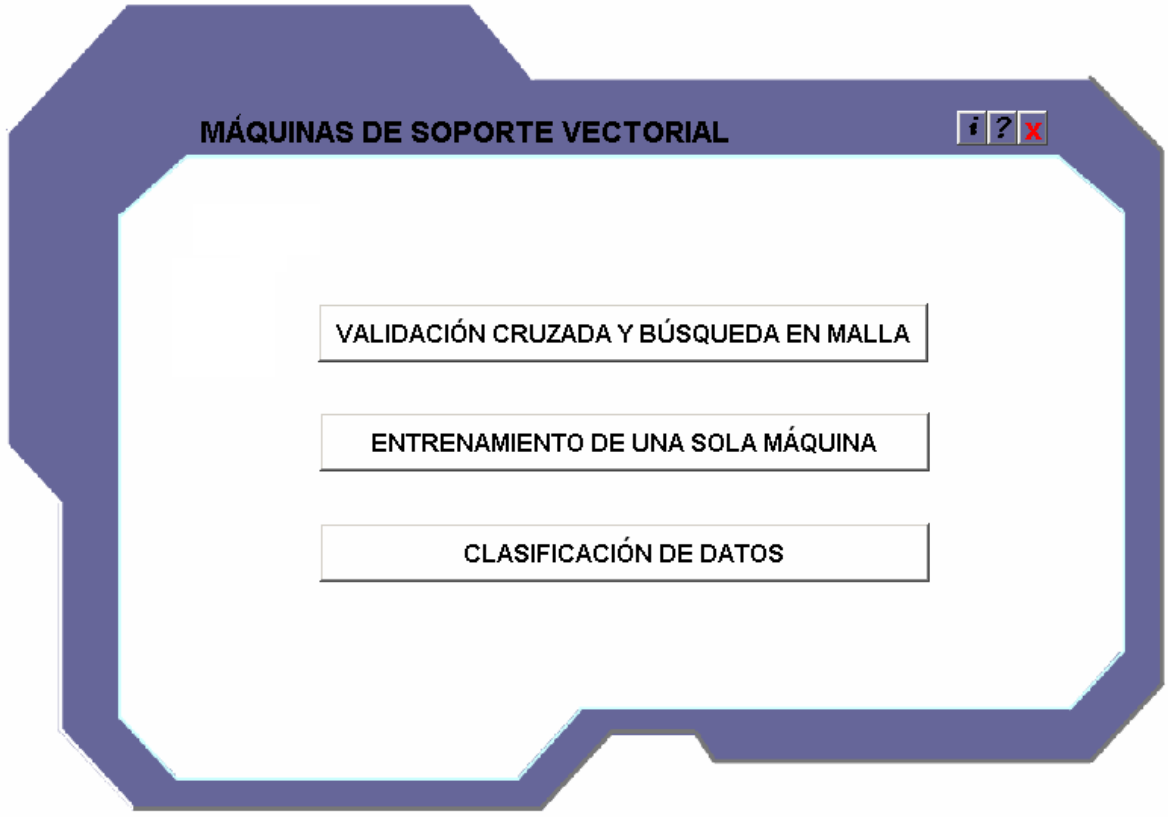
Adicionar la ruta del directorio que contiene los archivo en el programa MatLab® de la siguiente forma: addphath('C:\MATLAB\work\MSVToolbox1.0').

INICIO DE LA APLICACIÓN MSVToolbox1.0

Para iniciar la toolbox se escribe el nombre **MSVToolbox1.0** en la ventana de trabajo y se pulsa enter. Inmediatamente se abrirá la ventana de inicio (ver Figura A.2) donde se muestran las diferentes opciones de trabajo: validación cruzada y búsqueda en malla, entrenamiento y clasificación.

VENTANA DE INICIO

Figura A. 2. Ventana de inicio de la MSVToolbox1.0



Elaborado por los autores

Al seleccionar cualquiera de los botones de la Figura A.2, se despliega una ventana nueva, cada una de ellas con opciones y funciones diferentes que se explican a continuación.

VALIDACIÓN CRUZADA Y BÚSQUEDA EN MALLA

Figura A. 3. Ventana de validación cruzada de la MSVToolbox1.0

Elaborado por los autores

1. Datos de entrenamiento
2. Datos prueba*
3. No de Grupos*
4. Tipo de Kernel*
5. Cargar parámetros Kernel
6. Cargar parámetros C
7. Método de optimización*
8. Método de Descomposición*
9. ENTRENAR

Los campos marcados con asteriscos (*) son opcionales, para éstos la toolbox tiene unos valores predeterminados.
Esta ventana de trabajo, (Figura A.3), se utiliza para buscar los mejores parámetros de clasificación (parámetro del *kernel* y parámetro de penalización C) de un grupo de datos específico. Este proceso se realiza mediante la búsqueda en malla. El usuario introduce los valores de los parámetros que desea probar, terminado el proceso de validación cruzada esta aplicación entrega como resultado el mejor par de parámetros, este modelo se almacena para ser utilizado posteriormente en la clasificación de nuevos datos de este tipo.

1. Datos de entrenamiento: en este campo se cargan los datos de entrenamiento para la validación cruzada, se pueden cargar archivos con extensiones .mat, .txt y csv. Estos archivos deben contener una matriz conformada por los datos y sus respectivas etiquetas. Si el archivo no pertenece a estos formatos o si los datos no tienen la estructura mostrada en la Figura A.1, aparecerá un mensaje de error.

Después de cargados los datos se despliega un cuadro con la información más importante de estos datos (Figura A.4).

Figura A. 4. Información de los datos

Características de los Datos	
Nombre Archivo	dat_riply.mat
Tamaño de X	250x2
Tamaño de Y	250x1

Elaborado por los autores

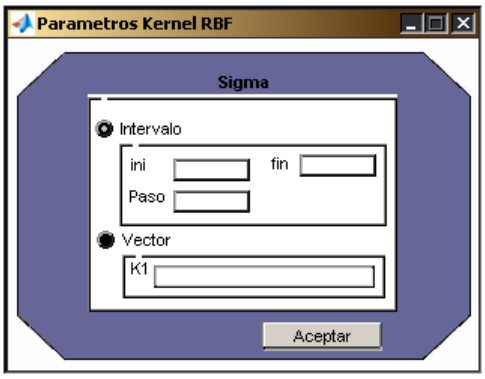
2. Datos de prueba: si se quiere valorar el modelo final, en este campo se deben cargan los datos de prueba, de igual forma que se cargan los datos de entrenamiento. Los datos deben tener la estructura de la Figura A.1. La inclusión de datos de prueba es opcional, sin embargo si se incluye, se obtendrán los resultados de la clasificación de estos datos (exactitud de prueba).

3. No de Grupos*: En este campo el usuario debe digitar el número de grupos que se desea formar con los datos de entrenamiento para la validación cruzada, esta cantidad debe ser un número entero e igual o superior a dos. Se recomienda que no sea muy alto (no superior a 5), para que el costo computacional no aumente demasiado, ni muy bajo para mayor exactitud.

4. Tipo de Kernel*: En este campo se selecciona uno de los 'radio button' para escoger el tipo de *kernel*, por defecto esta seleccionado 'RBF', la otras posibilidades son polinomial y sigmoidal.

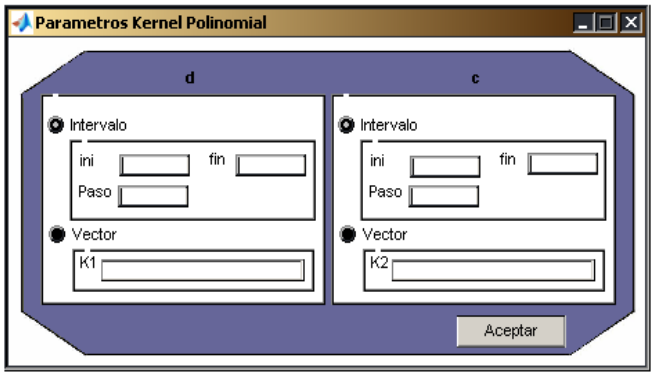
5. Cargar parámetros Kernel: dando clic sobre este botón se abre una ventana, la forma de esta depende del tipo de *kernel* que esté seleccionado, si es RBF se abre la ventana mostrada en la Figura A.5, si es polinomial o sigmoidal se abre la Figura A.6.

Figura A. 5. Ventana parámetros del kernel RBF



Elaborado por los autores

Figura A. 6. Ventana parámetros del kernel Sigmoidal y polinomial

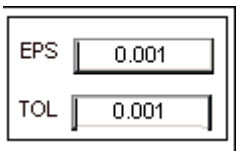


Elaborado por los autores

En estas ventanas se debe digitar los parámetros del *kernel* escogido para así poder realizar la búsqueda en malla, para esto existe dos posibilidades, la primera entrarlo con tres datos; *ini, paso, fin*, con estos datos el software construye el vector de parámetros. La segunda posibilidad es entrar directamente el vector en el campo k1 este se debe ingresar de entre corchetes [K1 K2…Kn], dejando espacio entre los diferentes parámetros.

6. Cargar parámetros C: dando clic este botón se abre una ventana similar a la que se utiliza para cargar los parámetros del *kernel*, la forma de entrar el vector C es la misma. Con los valores ingresados en los numerales 5 y 6 se realiza la búsqueda en malla para escoger los mejores parámetros.

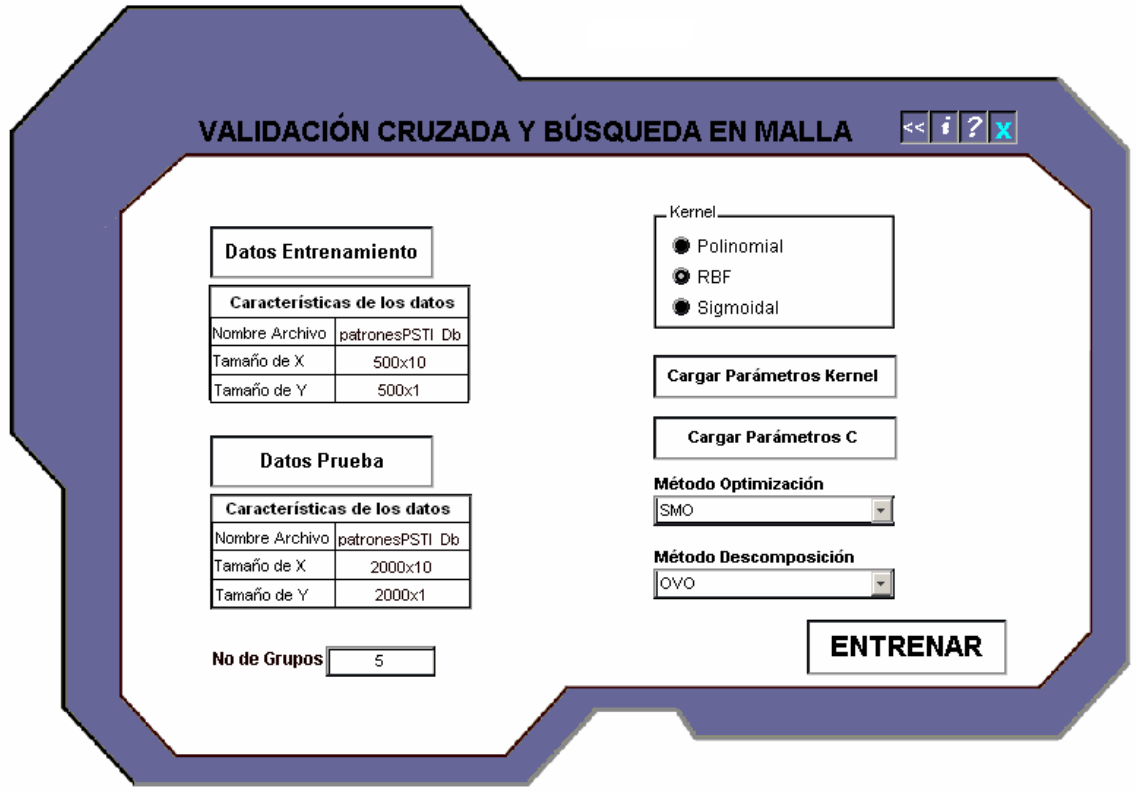
7. Método de optimización*: este es un menú despegable que tiene dos opciones 'SMO' e 'IRWLS', por defecto aparece 'SMO'. Si se da clic en este menú aparecerá dos campos más que son opcionales; EPS y TOL. EPS es el error que tolera el algoritmo de optimización, por defecto 0.001 y TOL es la tolerancia de las condiciones KKT por defecto 0.001.



8. Método de Descomposición*: este es otro menú despegable que tiene tres opciones, 'OVR', 'OVO' y 'ECOC_2C', por defecto aparece 'OVO'. Estos son los métodos de descomposición que se utilizaran en el entrenamiento si los datos tienen más de dos clases. En el caso de que sean sólo dos este menú aparecerá deshabilitado.

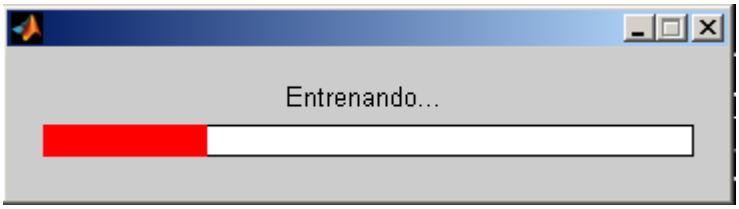
9. Entrenar: por último, después de haber introducido todos los datos se puede dar clic en este botón, encargado de ejecutar la validación cruzada y la búsqueda en malla. Antes de ejecutarse este botón la ventana se debe apreciar como se muestra en la Figura A.7.
Después de dar clic en 'ENTRENAR' (si todos los datos fueron entrados correctamente), aparecerá una ventana que le permite guardar en el destino que se desee los archivos que se generaran al ejecutarse esta aplicación. A Continuación se despliega una barra de progreso que le indicará el avance del entrenamiento, véase la Figura A.8.

Figura A. 7. Ventana de validación cruzada lista para realizar la validación



Elaborado por los autores

Figura A. 8. Barra de progreso del proceso



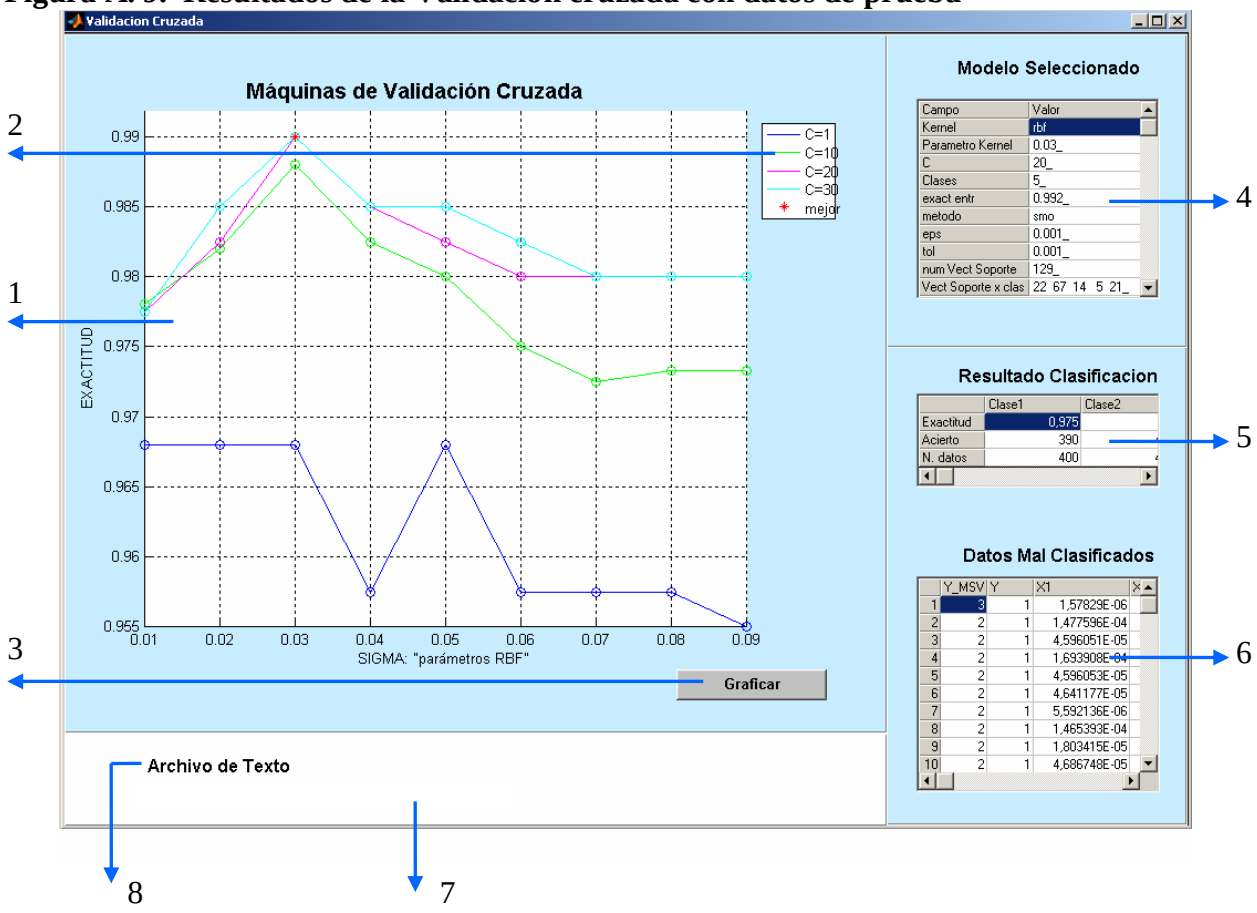
Elaborado por los autores

Al terminar la ejecución, se despliegan los resultados de la Validación cruzada en una nueva ventana como se muestra en la Figura A.9.

RESPUESTA DE LA VALIDACIÓN CRUZADA

La respuesta de la Validación cruzada se muestra en la Figura A.9, a continuación se explica esta ventana.

Figura A. 9. Resultados de la Validación cruzada con datos de prueba



Elaborado por los autores

1. Gráfica de la validación cruzada
2. Convenciones
3. Graficar
4. Modelo seleccionado
5. Resultados de la clasificación
6. Datos mal clasificados
7. Archivo de la estructura modelo .mat
8. Archivo de Texto

1. Gráfica de la validación cruzada: En esta gráfica se muestran los resultados de la búsqueda en malla que se obtuvo al entrenar la MSV con cada parámetro *kernel* y cada parámetro de penalización 'C', cada punto (círculo pequeño) que aparece representa una MSV entrenada, estos puntos se han interpolado de forma lineal sin basarse en algún concepto teórico solo con la intención de una mejor comprensión.

La exactitud de validación (de prueba o prueba) de estas máquinas entrenadas aparece en el eje de las ordenadas, el o los parámetros del *kernel* (el número de parámetros varia según el tipo de *kernel* usado RBF, polinomial o sigmoidal) en el eje de las abscisas, y cada parámetro 'C' aparece como una curva de diferente color y su valor esta especificado en el '*cuadro de convenciones'*.

Esta gráfica se implementó para facilitar la comprensión de los resultados de la búsqueda en malla, ofreciendo visualmente la variación de la exactitud con cada parámetro del *kernel* y con cada parámetro de penalización 'C'. La gráfica ha sido un aporte original a la teoría de búsqueda en malla mediante la validación cruzada ya que las que se encuentran en la bibliografía son confusas, están en tres dimensiones, por lo que los resultados mostrados no son fáciles de interpretar. Este aporte se convierte en una herramienta útil a la hora de escoger el siguiente intervalo de búsqueda de los mejores parámetros.

2. Convenciones: relaciona el valor del 'C' para cada parámetro del *kernel*.

3. Graficar: este botón permite trazar nuevamente la gráfica, en caso de que el usuario la haya alterado.

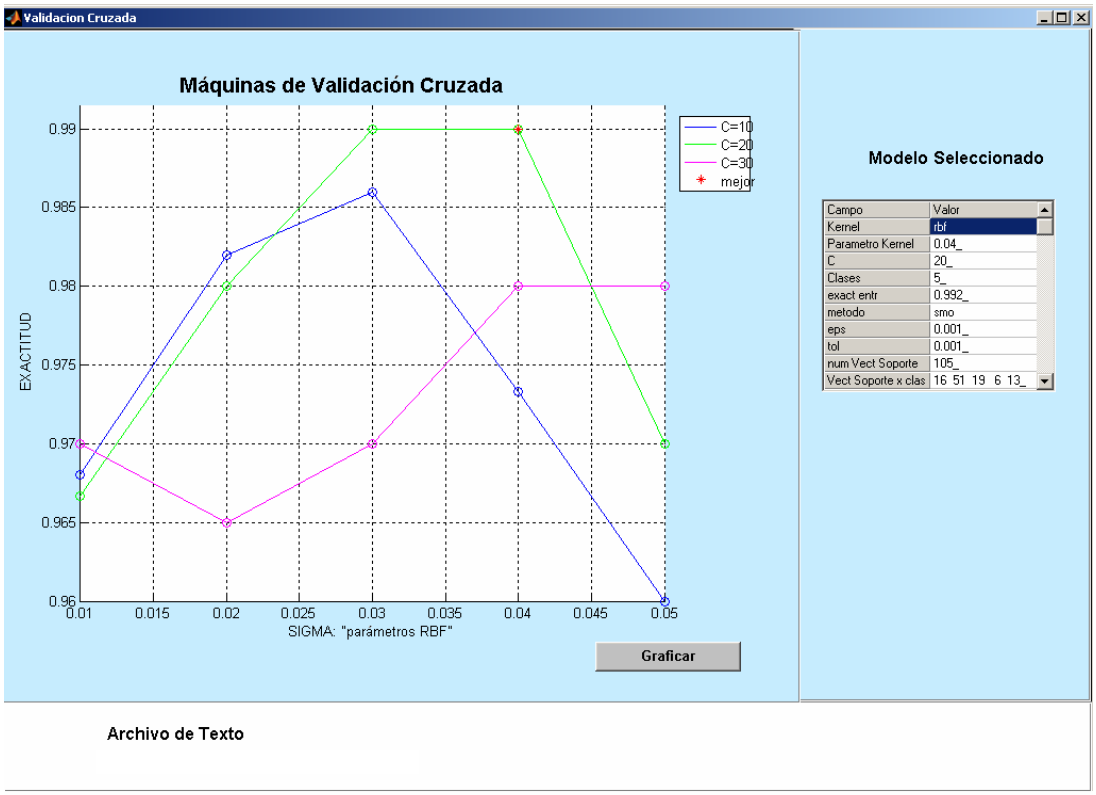
4. Modelo seleccionado: con la búsqueda en malla se encuentran los mejores parámetros, después de ser escogidos el software vuelve a entrenar con la totalidad de los datos y los parámetros escogidos, este resultado es el que se estaba buscando, 'la mejor máquina', y es la que permitirá clasificar datos nuevos en el futuro, los campos más importantes de esta máquina se muestran en esta tabla, además el archivo es guardado por el software en la ubicación que le ha indicado el usuario para uso posterior.

5. Resultados de la clasificación: esta tabla muestra la información por clases de la exactitud de clasificación de los '*Datos de prueba*', también el acierto o número de datos clasificados correctamente y el total de datos clasificados en cada una de las clases. Deslizando la barra hasta el final se encuentra los totales de estas mismas cantidades. Si no se entro '*Datos de prueba*' esta tabla no aparece.

Estos datos de prueba no deben confundirse con los que utiliza internamente la validación cruzada ya que estos datos son opcionales y solo los usa la MSV cuando clasifica para dar el valor de exactitud final del modelo escogido. Cuando

los '*Datos de prueba*' no se introducen después de la validación cruzada, la apariencia de esta ventana cambia, véase la Figura A.10.

Figura A. 10. Resultados de la Validación cruzada sin datos de prueba



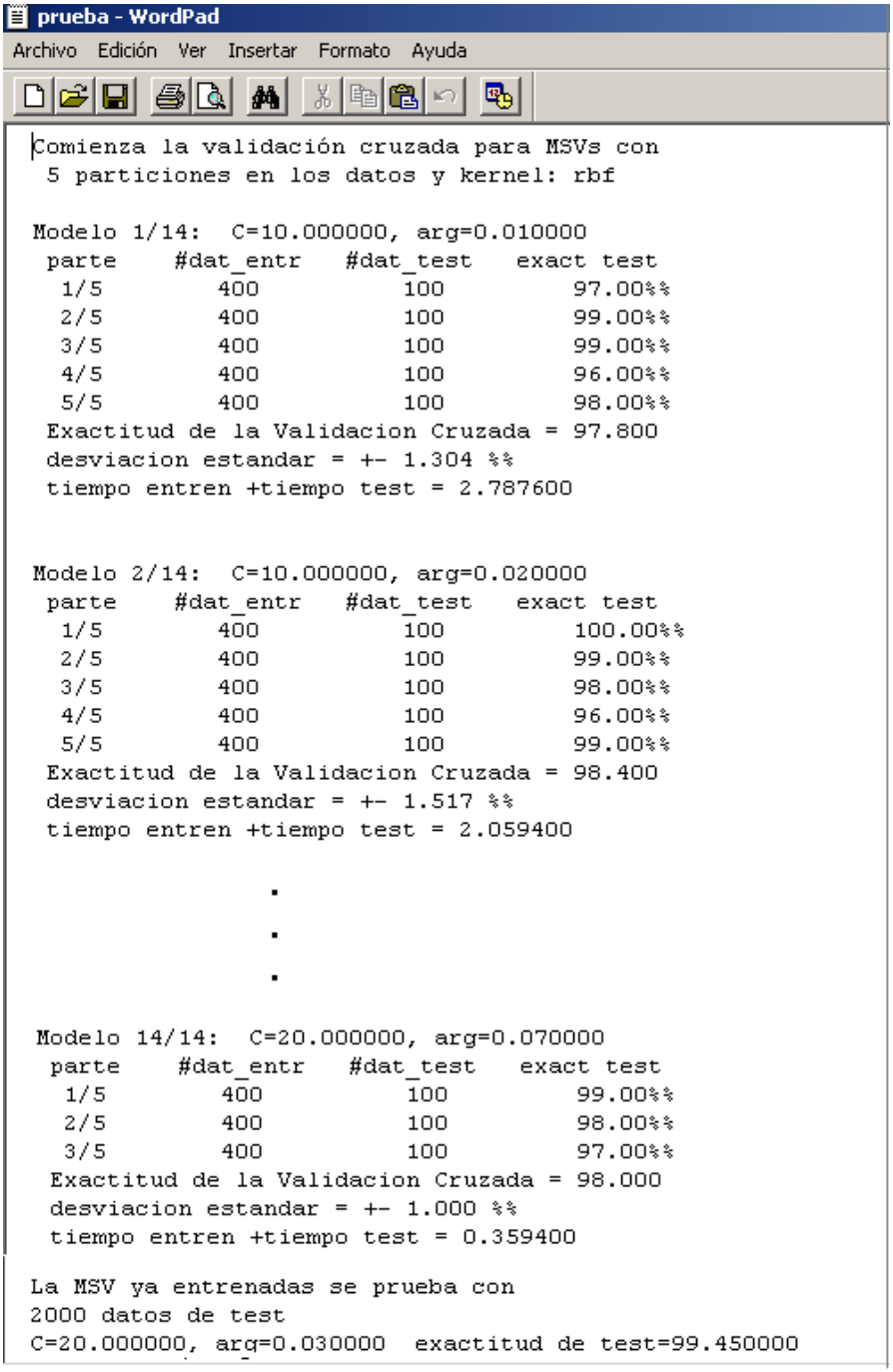
Elaborado por los autores

6. Datos mal clasificados: en esta tabla se despliegan los datos mal clasificados, la primera columna muestra las etiquetas asignadas por la MSV, la segunda las etiquetas reales dadas por el usuario, las demás columnas muestran el dato.

7. Archivo de la estructura modelo.mat: al dar clic en esta opción se carga en MatLab® el archivo .mat que contiene toda la información de la MSV escogida, este archivo es el que se usa para clasificar nuevos datos en el futuro. Este archivo se encuentra guardado en la extensión que escogió el usuario al ejecutar el botón 'ENTRENAR'.

8. Archivo de Texto: la toolbox genera un archivo de texto que contiene toda la información pertinente a la validación cruzada, este archivo es almacenado junto a *modelo.mat* y su contenido se puede apreciar en la Figura A.11.

Figura A. 11. Archivo de texto generado por MSVToolbox1.0 durante la validación



prueba - WordPad

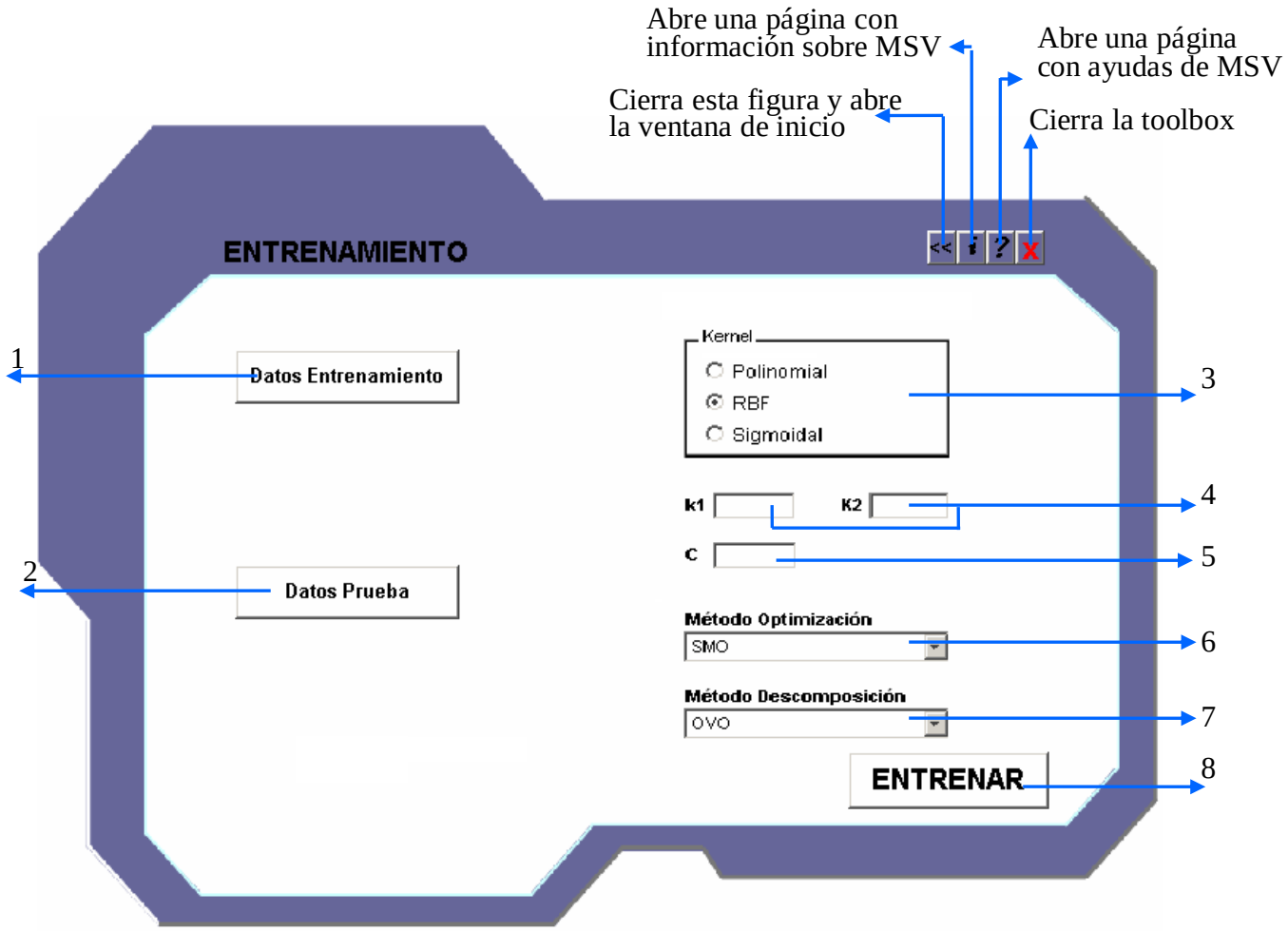
Archivo Edición Ver Insertar Formato Ayuda

```
Comienza la validación cruzada para MSVs con
 5 particiones en los datos y kernel: rbf

Modelo 1/14:  C=10.000000, arg=0.010000
 parte    #dat_entr   #dat_test   exact test
  1/5         400          100          97.00%%
  2/5         400          100          99.00%%
  3/5         400          100          99.00%%
  4/5         400          100          96.00%%
  5/5         400          100          98.00%%
 Exactitud de la Validacion Cruzada = 97.800
 desviacion estandar = +- 1.304 %%
 tiempo entren +tiempo test = 2.787600


Modelo 2/14:  C=10.000000, arg=0.020000
 parte    #dat_entr   #dat_test   exact test
  1/5         400          100          100.00%%
  2/5         400          100          99.00%%
  3/5         400          100          98.00%%
  4/5         400          100          96.00%%
  5/5         400          100          99.00%%
 Exactitud de la Validacion Cruzada = 98.400
 desviacion estandar = +- 1.517 %%
 tiempo entren +tiempo test = 2.059400

                .
                .
                .

Modelo 14/14:  C=20.000000, arg=0.070000
 parte    #dat_entr   #dat_test   exact test
  1/5         400          100          99.00%%
  2/5         400          100          98.00%%
  3/5         400          100          97.00%%
 Exactitud de la Validacion Cruzada = 98.000
 desviacion estandar = +- 1.000 %%
 tiempo entren +tiempo test = 0.359400

La MSV ya entrenadas se prueba con
2000 datos de test
C=20.000000, arg=0.030000  exactitud de test=99.450000
```

Elaborado por los autores

ENTRENAMIENTO

Figura A. 12. Ventana de Entrenamiento de la MSVToolbox1.0



Elaborado por los autores

1. Datos de entrenamiento
2. Datos prueba*
3. Tipo de Kernel*
4. Parámetros del Kernel
5. Parámetro C
6. Método de optimización*
7. Método de Descomposición*
8. ENTRENAR

Esta opción se usa para entrenar una sola MSV, con solo un parámetro de penalización y un parámetro del *kernel* (dependiendo del tipo de *kernel*), esta opción es usada cuando el usuario conoce el valor de estos parámetros para ese tipo de datos. El modelo generado es guardado para uso posterior en clasificación de este tipo de datos.

1. Datos de entrenamiento: con este botón se cargan los datos de entrenamiento para la validación cruzada, su forma de 'operar' es igual a la de validación cruzada.

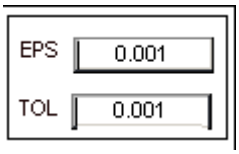
2. Datos de prueba: en este campo se cargan los datos de prueba o prueba para la validación cruzada, su forma de 'operar' es similar a su homologo en validación cruzada. Este campo es opcional, sin embargo si se incluye, se entregará resultados adicionales como la exactitud de clasificación del modelo y datos mal clasificados.

3. Tipo de Kernel*: se selecciona uno de los 'radio button' para escoger el tipo de *kernel*, por defecto esta seleccionado 'RBF', la otras posibilidades son polinomial y sigmoidal.

4. Parámetros del Kernel: en este campo se ingresa el valor o parámetro del *kernel*. Según el tipo de *kernel* seleccionado se habilitaran estos campos para RBF se habilitará solo el primero y para polinomial y sigmoidal estarán habilitados los dos.

5. Parámetro C: en este campo se ingresa el parámetro de penalización C.

6. Método de optimización*: este es un menú despegable que permite escoger entre dos opciones de optimización 'SMO' e 'IRWLS', por defecto aparece 'SMO'.



Si se da clic en este menú aparecerá dos campos más que son opcionales; EPS y TOL. EPS es el error que tolera el algoritmo de optimización, por defecto 0.001 y TOL es la tolerancia de las condiciones KKT por defecto 0.001.

7. Método de Descomposición*: este es otro menú despegable que tiene tres opciones, 'OVR', 'OVO' y 'ECOC_2C', por defecto aparece 'OVO'. Estos son los métodos de descomposición que se utilizaran en el entrenamiento si los datos

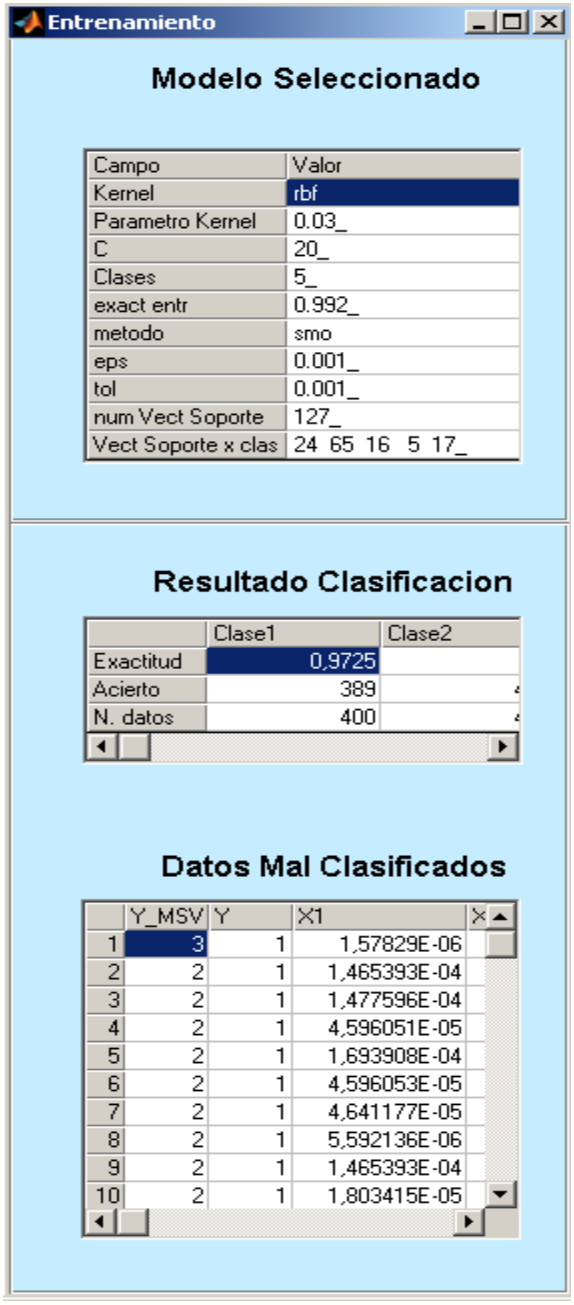
tienen más de dos clases, en el caso de que sean solo dos este menú aparecerá deshabilitado.

8. ENTRENAR: por ultimo, después de haber introducido todos los datos se puede dar clic en este botón, encargado de ejecutar el entrenamiento. Si todos los datos fueron entrados correctamente aparecerá una ventana que le permite guardar el archivo que se generará en el destino que se escoja.
Al finalizar la ejecución, los resultados del entrenamiento, aparecen en una ventana nueva como se aprecia en las Figuras A.13 y A.14.

RESPUESTA DEL ENTRENAMIENTO

Las Figuras A.13 y A.14 presentan los resultados del entrenamiento. La diferencia entre estas dos ventanas radica en que en la primera se han incluido datos de prueba, mostrando los resultados de la clasificación de estos. La tabla **Modelo** muestra la estructura modelo, esta estructura es el resultado del entrenamiento y es la que permite clasificar datos nuevos en el futuro. La tabla **Resultado Clasificación** muestra la información por clases de la exactitud de clasificación de los 'Datos de prueba', también el acierto o número de datos clasificados correctamente y el total de datos clasificados en cada una de las clases. Y en la última tabla **Datos Mal Clasificados** se despliegan los datos con una mala clasificación, la primera columna muestra las etiquetas asignadas por la MSV, la segunda las etiquetas reales dadas por el usuario, las demás columnas muestran el dato.

Figura A. 13. Resultados entrenamiento



Elaborado por los autores

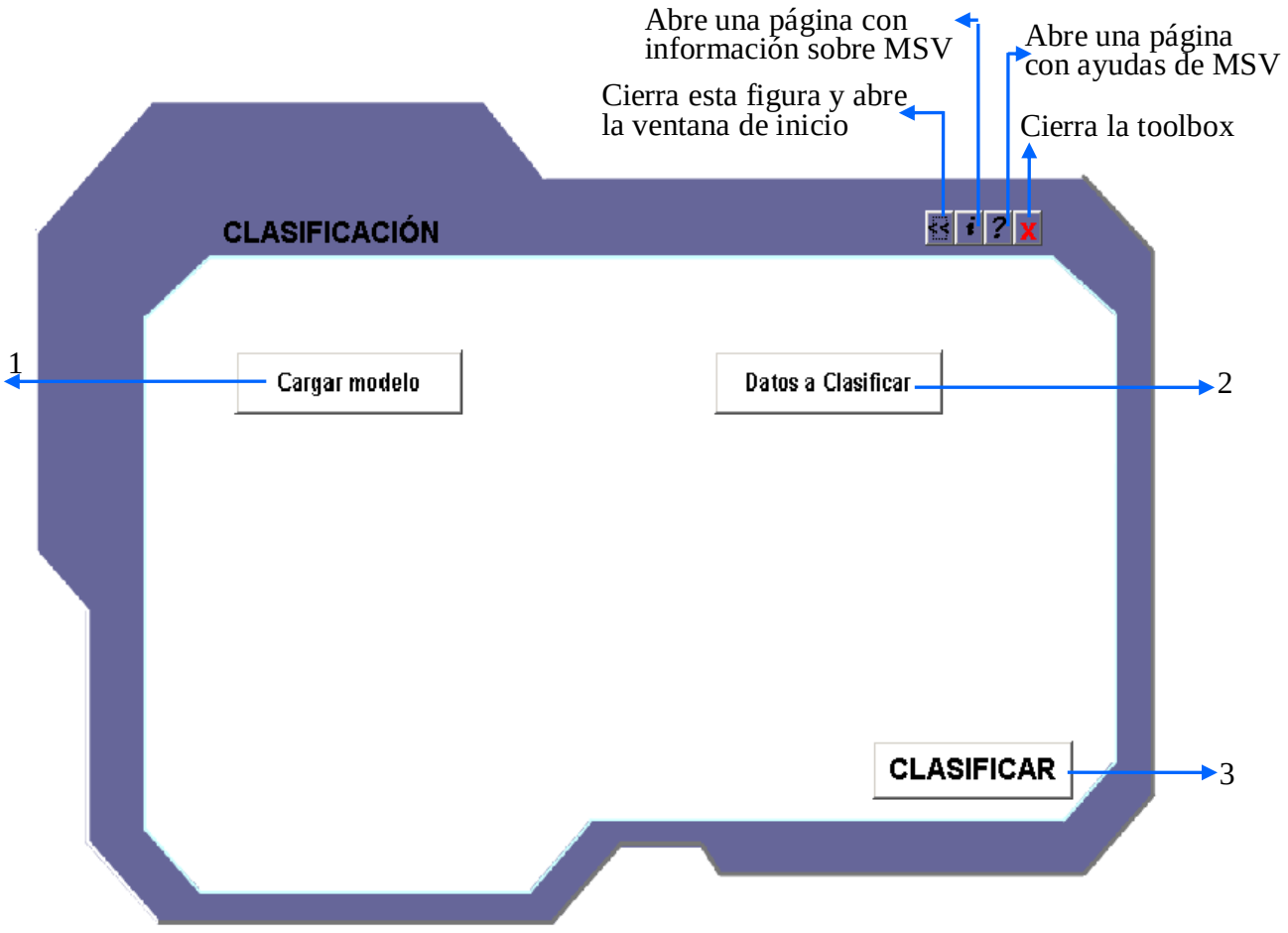
Figura A. 14. Modelo obtenido

Elaborado por los autores

CLASIFICACIÓN

En la Figura A.15 se presenta la ventana de clasificación, esta se utiliza para clasificar datos. Para esto, se debe haber entrenado una MSV con la misma clase de datos que se desea clasificar y generado un modelo (pudo haberse realizado utilizando 'Validación cruzada y búsqueda en malla' o 'Entrenamiento de una sola máquina'). El archivo modelo se carga desde esta misma ventana.

Figura A. 15. Ventana de Clasificación de la MSVToolbox1.0

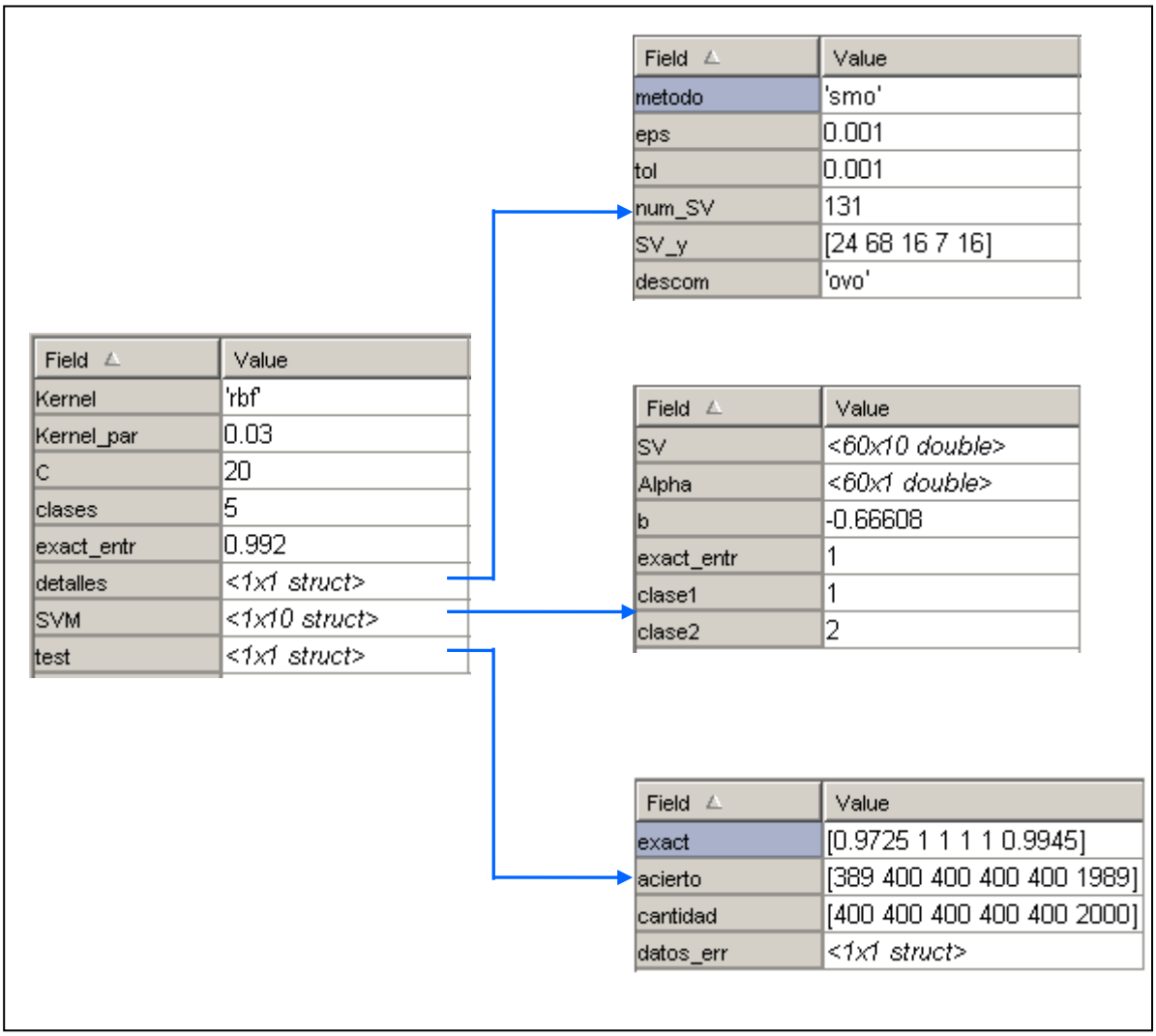


Elaborado por los autores

1. Cargar modelo
2. Datos a clasificar
3. Clasificar

1. Cargar modelo: con este botón se carga un archivo .mat que contiene la estructura generada por *validación cruzada y búsqueda en malla* o por *Entrenamiento*, esta estructura contiene la MSV o sea la función decisión que esta compuesta por los alphas, los vectores de soporte, las etiquetas correspondientes a esos vectores de soporte y otra información necesaria para la clasificación. En la Figura A.16 se presenta esta estructura.

Figura A. 16. Archivo modelo.mat generado en validación cruzada y búsqueda en malla para datos de cinco clases



Elaborado por los autores

Después de cargada esta estructura, se despliega en esta ventana la tabla que se muestra en la Figura A.17. Si el archivo cargado no corresponde a este tipo de estructura se despliega un mensaje de error.

Figura A. 17. Ventana de Clasificación después de cargar el modelo

Campo	Valor
Kernel	rbf
Kernel par	4.8
C	30
Clases	2
exact entr	0.86
metodo Optimizacion	smo
Vect soporte x	65 65
Numero MSV	1

Elaborado por los autores

2. Datos a clasificar: con este botón se carga el archivo que contiene los datos a clasificar, este puede estar en los formato .mat, .txt, y .csv. Este archivo deben contener una matriz conformada por datos en forma de vector columna (ver Figura A.18). Si el archivo no pertenece a este formato o los datos (o información) no esta puesta en columnas se despliega un mensaje de error.

Figura A. 18. Matriz de m datos de dimensión n para clasificación

$$\begin{bmatrix} x_{1a} & x_{1b} & \cdots & x_{1n} \\ x_{2a} & x_{2b} & \cdots & x_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ x_{ma} & x_{mb} & \cdots & x_{mn} \end{bmatrix} \begin{matrix} \rightarrow \text{Dato 1} \\ \rightarrow \text{Dato 2} \\ \\ \rightarrow \text{Dato m} \end{matrix}$$

Elaborado por los autores

Caracteristicas de los Datos	
Nombre Archivo	dat_riply.mat
Tamaño de X	250x2

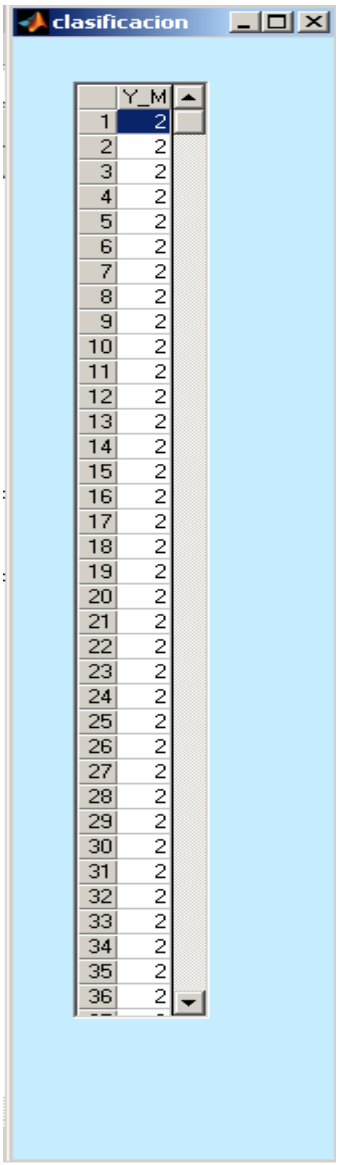
Después de cargados los datos se despliega un cuadro con el nombre del archivo y el tamaño de la matriz entrada.

3. Clasificar: después de haber introducido todos los datos se puede dar clic en este botón para ejecutar la clasificación. Si todos los datos fueron entrados correctamente aparecerá una ventana que le permite guardar el archivo .mat donde se guardaran los resultados. Al terminar de clasificar se mostrará la ventana de la Figura A.19.

RESPUESTA DE LA CLASIFICACIÓN

Esta ventana muestra la clasificación de los datos, esta clasificación aparece en el mismo orden de los datos ingresados.

Figura A. 19. Resultados de clasificación



Elaborado por los autores

AYUDAS

En este botón se despliega un archivo de texto con todas las ayudas disponibles para el manejo de la toolbox. Además cada ventana contiene un botón con las ayudas correspondientes a esa página y otro con información correspondiente a la acción a realizar en la ventana.

En este capítulo se hizo la presentación de la herramienta computacional MSVToolbox1.0. Se explicó la instalación, ejecución y las diferentes opciones de trabajo: validación cruzada y búsqueda en malla, entrenamiento, clasificación y ayudas de la toolbox.

Anexo B. FUNCIONES DE LA MSVToolbox1.0

Los códigos implementados están basados en el trabajo realizado por [Morales & Gómez, 05]. Estos códigos fueron revisados y optimizados con el fin de implementar una herramienta computacional eficiente y aplicable al problema de clasificación automática de perturbaciones electromagnéticas.
Los códigos o funciones más importantes utilizadas en la implementación de la MSVToolbox1.0

B. 1. VALIDACIÓN CRUZADA

```
function [modelo] = vvvalidacion(datos,opciones,datos_tst,partes)
% Autores:    German Morales  UIS
%                Alvaro Gomez    UIS

%Modificaciones:   Estela Campos Cortés
%                        Ángel Gabriel Suárez Durán
%              27/febrero/2007

%datos: matriz [Y,X], donde  Yson las etiquetas (1, 2, ...)

%   MODELO = VALCROS(DATOS,OPCIONES)
%   Se utiliza el metodo de validacion cruzada para evaluar la SVM con
%   parametros de kernel y las constantes de penalizacion C
%
%   MODELO = VALCROS(DATOS,OPCIONES,DATOS_TST)
%   se prueba al final la SVM entrenada con los datos DATOS_TST
%
%   MODELO = VALCROS(DATOS,OPCIONES,DATOS_TST,PARTES)
%   Se especifica el numero de particiones que se van a tomar en la
%   validacion cruzada

%   DATOS_TST: matriz [Y,X]
%   X : vectores con los cuales se va a probar la
%               exactitud de la SVM entrenada
%   Y : etiquetas de los vectores de prueba
%   si DATOS_TST = [] no se hace la prueba final de la SVM entrenada
%
%   PARTES [1x1]: numero de particiones para la Validacion Cruzada
%               5 por defecto
%
%   D [LxK]: matriz de descomposicion, metodo utilizado para SVMs de
%           multiclasificacion
%
% salidas:
```

```
%   MODELO [struct] datos del clasificador SVM
%         (vea la ayuda de K_CLASES para mas detalles)
%
tic;
%-------------------------------------------------------
%Normalización de los datos
N = length(datos.Y);
if  isempty(datos_tst)
   [datos.X,maxim] = normalizar(datos.X);
else
   [datos.X,maxim,datos_tst.X]= normalizar(datos.X,datos_tst.X);
end
%particionamiento de datos
%se cambian las posiciones de los datos de forma randomica
[X,Y] = deal(datos.X,datos.Y);
if isfield(datos,'prueba_i')
   prueba_i = datos.prueba_i;
else
   [X,Y,prueba_i] = aleatorio(X,Y,partes);
end

[exact,desvi] = deal(zeros(size(opciones.Kernel_par,2),length(opciones.C)));
opciones.clases=max(datos.Y);
D = mat_descomposicion(opciones.clases,opciones.descom);
opciones_i = opciones;
%clc;
 imprimir(fid,1,partes,opciones.Kernel);
i = 0;
maximo = 0;
h = waitbar(0,'Entrenando...');
for C = opciones.C
      opciones_i.C = C;
   for par = opciones.Kernel_par
      opciones_i.Kernel_par = par;
      imprimir(fid,2,i,length(exact(:)),C,par);
      exact_prueba = zeros(1,partes);
      waitbar(i/length(exact(:)));
      %se ejecuta la SVM para cada pareja de parametros par y C
      %para la Validacion Cruzada
      t=0;
      for j=1:partes
         ini = prueba_i(j);
         fin = prueba_i(j+1)-1;
         %se saca el grupo j del entrenamiento y se utiliza
         %para el prueba
```

```
            X_train = X([1:ini-1,fin+1:end],:);
            Y_train = Y([1:ini-1,fin+1:end]);
            X_prueba = X(ini:fin,:);
            Y_prueba = Y(ini:fin);
            imprimir(fid,3,j,partes,length(X_train),length(X_prueba));
            datos_i = struct('X',X_train,'Y',Y_train);
            tiempo=clock;
            modelo_i = AAAentrenar_kclases(datos_i,opciones_i,D);
            %clasificacion de los datos de prueba
            exact_prueba(j) = exactitud(X_prueba,modelo_i,Y_prueba);
            tiempo=etime(clock, tiempo);
            t=t+tiempo;
            imprimir(fid,4,exact_prueba(j));
           [salta,exact_prueba] = saltar(maximo,exact_prueba,j,partes);
            if salta, break; end;
        end
        t=t/partes;
        i = i+1;
        exact(i) = mean(exact_prueba);
        desvi(i) = std(exact_prueba);
           save exact exact desvi
        if exact(i)<maximo
           imprimir(fid,5,exact(i),desvi(i),t,[]);
        else
            imprimir(fid,5,exact(i),desvi(i),t,'(mejor hasta ahora)');
            maximo = exact(i);
        end
    end
end
close(h)

[desvi_i,C,par] = busca(exact,desvi,maximo,opciones.C,opciones.Kernel_par,datos_tst);
maximo2=0;
if isempty(datos_tst)
    opciones_i.C = C;
    opciones_i.Kernel_par = par;
    imprimir(fid,6,opciones_i.C,opciones_i.Kernel_par,maximo,desvi_i);
    modelo = entrenar_kclases(datos,opciones_i,D);
    'uno'
elseif length(par)==1
    imprimir(fid,13,length(datos_tst.Y));
    opciones_i.C = C;
    opciones_i.Kernel_par = par;
    modelo= entrenar_kclases(datos,opciones_i,D);
```

```
   [EXACT,ACIERTO,CANTIDAD,datos_err,posic_err] =
exactitud(datos_tst.X,modelo,datos_tst.Y);
   prueba.exact=EXACT; prueba.acierto=ACIERTO; prueba.cantidad=CANTIDAD;
   if maxim>1
      datos_err.X=datos_err.X*maxim;
   end
   prueba.datos_err=datos_err;
   prueba.posic_err=posic_err;
   modelo.prueba=prueba;
   imprimir(fid,14,C,par,EXACT(end));
   'dos'
else
   imprimir(fid,11,maximo,desvi_i);
   UE=length(C);
   for rr=1:UE
      imprimir(fid,12,C(rr),par(rr));
   end
   imprimir(fid,11,maximo,desvi_i);
   imprimir(fid,12,C,par);
   imprimir(fid,13,length(datos_tst.Y));
   %entrenando los parametros escogidos con el grupo completo de datos.
   for j=1:UE,
      opciones_i.C = C(j);
      opciones_i.Kernel_par = par(j);
      multiple = entrenar_kclases(datos,opciones_i,D);
      %clasificando datos de prueba
      [EXACT,ACIERTO,CANTIDAD,datos_err,posic_err] =
exactitud(datos_tst.X,multiple,datos_tst.Y);
      prueba.exact=EXACT; prueba.acierto=ACIERTO; prueba.cantidad=CANTIDAD;
      if maxim>1
         datos_err.X=datos_err.X*maxim;
      end
      prueba.datos_err=datos_err;
      prueba.posic_err=posic_err;
      multiple.prueba=prueba;
      MSV.multiple(j)=multiple;
      imprimir(fid,14,C(j),par(j),EXACT(end));
      if EXACT(end)>=maximo2
         maximo2 = EXACT(end);
         modelo=MSV.multiple(j);
      end
   end
 modelo.mejores_MSV=MSV.multiple;
 imprimir(fid,10,modelo.C,modelo.Kernel_par,length(datos_tst.Y),maximo2)
 'tres'
```

```
end

save grafResultados1 opciones exact maximo
tiempo_entrenamiento=toc;
modelo.norma=maxim;
modelo.tiempo_entrenamiento=tiempo_entrenamiento;

%-----------------------------------------------------------------------
%-----------------------------------------------------------------------
function [salta,exact_prueba] = saltar(maximo,exact_prueba,j,partes)
%evita iteraciones no necesarias
% Autores:    German Morales  UIS
%                Alvaro Gomez    UIS

%Modificaciones:   Estela Campos Cortés
%                        Ángel Gabriel Suárez Durán
%              27/febrero/2007
salta = logical(0);
if j<=floor(partes/3) || j==partes || maximo==0
   return;
end
temp = min(max(exact_prueba)+.1,1);
temp = [exact_prueba(1:j),repmat(temp,1,partes-j)];

if mean(temp)<maximo
   salta = logical(1);
   exact_prueba = exact_prueba(1:j);
   return;
end

%-----------------------------------------------------------------------
function [desvi_i,C,par] = busca(exact,desvi,maximo,C,par,datos_tst)
%parametros para los cuales se obtuvo la mejor exactitud
pos = (exact==maximo);
desvi_min=min(desvi(pos));
pos_desvi=(desvi==desvi_min).*pos;
num_maq=sum(sum(pos_desvi));
[pos_par,pos_C] = find(pos_desvi);
if num_maq==1||(num_maq<5 && ~isempty(datos_tst))
   desvi_i = repmat(desvi_min,[1,num_maq]);
   C = C(pos_C);
   par = par(:,pos_par);
   return;
else
   %se escoge los mejores parametros (los de mejor exactitud)
```

```
   exactX_par=max(sum(pos_desvi,2));
   filas_mejpar=find(sum(pos_desvi,2)==exactX_par);
   nueva_pos=zeros(size(exact));
   nueva_pos(filas_mejpar,:)=pos_desvi(filas_mejpar,:);
   num_maq=sum(sum(nueva_pos));
   [pos_par,pos_C] = find(nueva_pos);
   if isempty(datos_tst)||num_maq>5
      mat = sortrows([-C(pos_C)',-par(:,pos_par)']);
      desvi_i =desvi_min;
      C = -mat(1,1);
      par = -mat(1,2:end);
   else
      desvi_i = repmat(desvi_min,[1,num_maq]);
      C = C(pos_C);
      par = par(:,pos_par);
   end
end

%---------------------------------------------------------------------
function imprimir(fid,opcion,var1,var2,var3,var4)
%funcion encargada de imprimir en el documento
switch opcion
   case 1
      texto = fprintf(fid,['Comienza la validación cruzada para MSVs con\n ',...
                  '%d particiones en los datos y kernel: %s \n\n'],...
                   var1,var2);
   case 2
      texto = fprintf(fid,'Modelo %d/%d:  C=%4f, arg=%4f \n parte    #dat_entr
#dat_prueba   exact prueba\n ',...
                  var1+1,var2,var3,var4);
   case 3
      texto = fprintf(fid,'%2d/%d        %d          %d    ',...
                  var1,var2,var3,var4);
      opcion = 0;
   case 4
      texto = fprintf(fid,'     %5.2f%%%%        \n ',...
                  var1*100);
      opcion = 0;
   case 5
      texto = fprintf(fid,'Exactitud de la Validacion Cruzada = %.3f \n desviacion estandar =
+- %.3f %%%% \n tiempo entren +tiempo prueba = %4f \n\n\n',...
                  var1*100,var2*100,var3);
      texto = [texto,var4,'\n\n'];
   case 6
      texto = fprintf(fid,['Ahora se entrena el grupo completo con los ',...
```

```
                    'parametros que ofrecieron\nmejor exactitud: ',...
                    'C=%4f, arg=%4f, con exactitud= %.3f +- %.3f %%%%\n ...'],...
                     var1,var2,var3*100,var4*100);
   case 7
      texto = fprintf(fid,'\b\b\b\b\b');
      opcion = 0;
   case 8
      texto = fprintf(fid,['\n\nLa SVM ya entrenada con Validacion Cruzada se prueba
con',...
                     '\n%d datos y se obtiene una exactitud del %.3f%%%% \n'],...
                      var1,var2*100);
   case 9
      texto = fprintf(fid,['-------------------------------------',...
                     '-------------------------------------\n\n']);
   case 10
      texto = fprintf(fid,['\n\nLa mejor MSV con C=%4f, arg=%4f se prueba con %d
datos',...
                     '\n y se obtiene una exactitud del %.3f%%%% \n'],...
                      var1,var2,var3,var4);
   case 11
       texto = sprintf(['Ahora se entrena el grupo completo con los ',...
                    'parametros que ofrecieron\n',...
                    'mejor exactitud= %.3f y desviacion estandar=+- %.3f %%%%\n Estos son:
...'],...
                    var1*100,var2*100);
   case 12
      texto = sprintf(['C=%4f, arg=%4f //'],var1,var2);
   case 13
      texto = fprintf(fid,['\n\nLa MSV ya entrenadas se prueba con',...
                     '\n%d datos de prueba \n'],...
                      var1);
   case 14
      texto = fprintf(fid,'C=%4f, arg=%4f  exactitud de prueba=%4f ',var1,var2,var3*100);
end
if opcion==0
   texto = '';
end
```

B. 2. ENTRENAMIENTO EN MULTICLASIFICACIÓN

```
function modelo = entrenar_kclases(datos,opciones,D)
% Autores:    German Morales  UIS
%                 Alvaro Gomez    UIS
```

```
if ~isfield(opciones,'descom')
   descom = 'ovo';
else
   descom = opciones.descom;
   opciones = rmfield(opciones,'descom');
end

K = max(datos.Y);
if K==2;
   modelo = msv_2entrenar(datos,opciones);
   return;
else nargin==2 || isempty(D);
   D = mat_descomposicion(K,descom);
end
X = datos.X;
Y = datos.Y;
iter = size(D,1);
SV_pos = [];
%se ejecuta cada SVM biclasificadora
for i = 1:iter
   clase1 = uint8(find(D(i,:)==1));
   clase2 = uint8(find(D(i,:)==2));
   pos_1 = clases(clase1,Y);
   pos_2 = clases(clase2,Y);
   pos = [pos_1;pos_2];

   datos_i.X = X(pos,:);
   datos_i.Y = [ones(size(pos_1)); repmat(2,size(pos_2))];
   [modelo_i,SV_pos_i] = msv_2entrenar(datos_i,opciones);
   modelo_i.SVM.clase1 = clase1;
   modelo_i.SVM.clase2 = clase2;
   SVM(i)= modelo_i.SVM;
   SV_pos = [SV_pos;pos(SV_pos_i)];
end
SV_pos = unique(SV_pos);%posicion de los vectores de soporte
SV_y = Y(SV_pos);%clase a la q pertenece cada vector de soporte
modelo = modelo_i;
modelo.clases = K;
detalles = modelo_i.detalles;
detalles.num_SV = length(SV_pos);
detalles.descom = descom;
detalles.SV_y = zeros(1,K);
for i=1:K
   detalles.SV_y(i) = sum(ismember(SV_y,i));
end
```

```
modelo.SVM = SVM;
modelo.detalles = detalles;
modelo.exact_entr = exactitud(X,modelo,datos.Y);

%--------------------------------------------------------------------
function [pos_i]=clases(clase,Y)
%se retornan las posiciones de Y donde se encuentran las etiquetas
%entradas:
%   clase : vector fila de etiquetas
%   Y: etiquetas de todos los datos de entrenamiento [K-por-1]
%salidas:
%   pos_i: posiciones de Y donde se encuentran las etiquetas de entrada
pos_i = [];
for i = 1:length(clase)
   pos_i = [pos_i;find(Y==clase(i))];
end
```

B. 3. MULTICLASIFICACIÓN

```
function [y,zona] = multiclasificador(X,modelo)
% Autores:    German Morales  UIS
%                Alvaro Gomez    UIS

%Modificaciones:   Estela Campos Cortés
%                        Ángel Gabriel Suárez Durán
%              27/febrero/2007
N = size(X,1);

K = modelo.clases;
if nargout == 2
   zona = logical(zeros(N,K));
   grafica = logical(1);
else
   [zona,grafica] = deal(logical(0));
end
if K == 2
   [y,fx] = msv_2clasif(X,modelo);
   if grafica
      zona = [fx<=1,fx>=-1];
   end
   return;
end
iter = length(modelo.SVM);
%sistema de votación, en caso de empate
%se trabaja con probabilidades
```

```
[votos,prob] = deal(zeros(N,K));
modelo_i = modelo;
ecoc_2c = logical(0);
switch modelo.detalles.descom
   case 'ecoc_2c'
      ecoc_2c = logical(1);
      iter = iter/2;
end
for i=1:iter
   modelo_i.SVM = modelo.SVM(i);
   [voto1,prob1,zona] = desicion(X,modelo_i,iter,N,K,zona,grafica);
   if ecoc_2c
      modelo_i.SVM = modelo.SVM(iter+i);
      [voto2,prob2,zona] = desicion(X,modelo_i,iter,N,K,zona,grafica);
      voto1 = voto1 + voto2;
      prob1 = prob1.*prob2;
   end
   votos = votos + voto1;
   prob = prob + prob1;
end
[temp,y] = max((votos+prob/iter),[],2);
function [voto,prob,zona] = desicion(X,modelo,iter,N,K,zona,grafica)
x] = msv_2clasif(X,modelo);
clase_pos = modelo.SVM.clase1;
clase_neg = modelo.SVM.clase2;
voto = votar(fx,clase_pos,clase_neg,N,K);
prob = probabilidad(fx,clase_pos,clase_neg,N,K);

if grafica
   zona(:,clase_pos) = repmat(fx<=1,1,length(clase_pos)) | zona(:,clase_pos);
   zona(:,clase_neg) = repmat(fx>=-1,1,length(clase_neg)) | zona(:,clase_neg);
end

%-------------------------------------------------------------------
function voto = votar(fx,clase_pos,clase_neg,N,K)
%sistema de votacion
%votacion por parejas, se reciben votos a favor y en contra,
%este sistema de votacion produce mejores resultados
%que el sistema tradicional de votacion sencilla
voto = zeros(1,K);
voto(clase_pos) = 1;
voto(clase_neg) = -1;
voto = repmat(sign(fx),1,K).*repmat(voto,N,1);

%-------------------------------------------------------------------
```

```
function prob = probabilidad(fx,clase_pos,clase_neg,N,K)
%probabilidad
%Ayuda en caso de empates
prob = zeros(N,K);
pos = 1./(1+exp(-fx));
neg = 1-pos;
dimP = length(clase_pos);
dimN = length(clase_neg);
prob(:,clase_pos) = repmat(pos/dimP,1,dimP);
prob(:,clase_neg) = repmat(neg/dimN,1,dimN);
```

B. 4. ENTRENAMIENTO EN BICLASIFICACIÓN

```
function [modelo,SV_pos]=msv_2entrenar(datos,opciones)
% Autores:    German Morales  UIS
%                 Alvaro Gomez    UIS

%Modificaciones:   Estela Campos Cortés
%                        Ángel Gabriel Suárez Durán
%              27/febrero/2007
N = size(datos.X,1);
Alpha0=zeros(N,1);
X = datos.X;
Y = datos.Y;
Y(datos.Y==2)=-1;
[C,opciones.C] = deal(opciones.C(1));
%-----------------------------------------------------

switch detalles.metodo
   case 'smo'
      b0 = 0;
      par = opciones.Kernel_par;
      switch opciones.Kernel
         case 'poly'
            if length(par)==1, par(2) = 1; end
         case 'sigmoid'
            if length(par)==1, par(2) = 0; end
      end
      [Alpha, b, nsv, kercnt, trnerr, margin]...
         = smo_mex(datos.X', datos.Y', opciones.Kernel, par, C,...
         detalles.eps, detalles.tol, Alpha0, b0 );
      % organizando la salida
      SV_pos = find(Alpha);
      modelo = struct('SV',X(SV_pos,:),'Alpha',Alpha(SV_pos).*Y(SV_pos),'b',b);
```

```
        modelo = finalizar(modelo,opciones,detalles);
        [modelo.SVM.exact_entr,modelo.exact_entr] = deal(1-trnerr);
        return;

    case 'irwls'
        [Alpha, b] = irwls(X,Y,opciones.Kernel,C,opciones.Kernel_par);
        SV_pos = find(Alpha);
        modelo = struct('SV',X(SV_pos,:),...
                'Alpha',Alpha(SV_pos),...
                'b',b);
        modelo = finalizar(modelo,opciones,detalles);
        [modelo.SVM.exact_entr,modelo.exact_entr] =...
           deal(exactitud(X,modelo,datos.Y));
        return;
    case 'loqo'
        H = msv_kernel(X,X,opciones);
        H = H + diag(repmat(eps^(2/3),N,1));
        H = H.*(Y*Y');
        ub = repmat(C,N,1);
        f = -ones(N,1);
        Aeq = Y';
        beq = 0;
        lb = zeros(N,1);
        if isempty(ub) | C>1e7 %para una rapida convergencia
           ub = repmat(1e7,N,1);
        end
        Alpha = loqo(H, f, Aeq, beq, lb, ub, Alpha0, 1);
    otherwise
        opciones.metodo = 'smo';
        modelo=msv_2class(datos,opciones,Alpha0);
        return;
end
%b
SV_pos = Alpha > detalles.tol;
SV_pos1 = (SV_pos) & (Alpha < (C - detalles.eps));
b = mean(Y(SV_pos1) - H(SV_pos1,SV_pos) * Alpha(SV_pos).*Y(SV_pos1));
SV_pos = find(SV_pos);
%modelo de la MSV
modelo = struct('SV',X(SV_pos,:),...
              'Alpha',Alpha(SV_pos).*Y(SV_pos),...
              'b',b);
modelo = finalizar(modelo,opciones,detalles);
[modelo.SVM.exact_entr,modelo.exact_entr] =...
   deal(exactitud(X,modelo,datos.Y));
%-------------------------------------------------------------------
```

```
function modelo = finalizar(SVM,opciones,detalles)
detalles.num_SV = size(SVM.SV,1);
detalles.SV_y = sum(SVM.Alpha>0);
detalles.SV_y(1,2) = detalles.num_SV - detalles.SV_y;
modelo = opciones;
modelo.clases = 2;
modelo.exact_entr = [];
modelo.detalles = detalles;
modelo.SVM = SVM;
```

B. 5. BICLASIFICACIÓN

```
% Autores:    German Morales  UIS
%                 Alvaro Gomez    UIS

function [y,fx] = msv_2clasif(X,modelo)
H = msv_kernel(X,modelo.SVM.SV,modelo);
%funcion decision
fx = H*modelo.SVM.Alpha + modelo.SVM.b;
y = ones(length(fx),1);
y2 = fx<0;
y(y2) = 2;
```

B. 6. MATRIZ KERNEL

```
function H = msv_kernel(X1,X2,varargin)
%Autores:   German Morales  UIS
%          Alvaro Gomez    UIS

%Modificaciones:  Estela Campos Cortés
%                 Ángel Gabriel Suárez Durán
%                 27/febrero/2007
if isstruct(varargin{1})
   tipo = varargin{1}.Kernel;
   par = varargin{1}.Kernel_par;
elseif nargin==4
   tipo = varargin{1};
   par = varargin{2};
else
   error('Argumentos de entrada mal definidos.');
end

%matriz Kernel H dependiendo del tipo de kernel
switch tipo
```

```
    case 'poly'
        if length(par)==1, par(2) = 1; end
        H = (X1*X2'+par(2)).^par(1);
    case 'RBF'
        N1 = size(X1,1);
        N2 = size(X2,1);
        temp = repmat(sum((X1.^2),2), [1 N2])+ ...
               repmat(sum((X2.^2)',1), [N1 1]) - ...
               2*X1*(X2');
        H = exp(-.5*temp/par(1)^2);
        %Otra alternativa de trabajar el parametro es
        %H = exp(-temp*par(1));
    case 'sigmoid'
        if length(par)==1, par(2) = 0; end
        H = tanh(par(1).*(X1*X2')+par(2));
    otherwise
        error('el tipo de kernel esta mal definido.');
end
```