Universidad de las Ciencias Informáticas

Facultad 6

Título: “Implementación y evaluación de nuevos kernels para la predicción de actividad biológica aplicando Máquinas de Soporte Vectorial.”

Trabajo de Diploma para optar por el título de
Ingeniero Informático

Autor(es): Yixander Yero Tarancón

Yoamel Acosta Morales

Tutor(es): Dr. Ramón Carrasco Velar

Msc. Aurelio Antelo Collado

Ing. Andy Machín González

Ciudad de la Habana, Julio de 2008

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas y al Centro de Química Farmacéutica los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Yixander Yero Tarancón

_____

Firma del Autor

Yoamel Acosta Morales

_____

Firma del Autor

Ing. Andy Machín González

_____

Firma del Tutor

Msc. Aurelio Antelo Collado

_____

Firma del Tutor

Dr. Ramón Carrasco Velar

_____

Firma del Tutor

"El arte es "yo"; la ciencia es nosotros".

Claude Bernard

DATOS DE CONTACTO

Tutores:

Dr. Ramón Carrasco Velar
Centro de Química Farmacéutica, Ciudad de La Habana, Cuba.
ramon.carrasco@cqf.sld.cu
rcarrasco@uci.cu

Msc. Aurelio Antelo Collado
Universidad de las Ciencias Informáticas, Ciudad de La Habana, Cuba.
aantelo@uci.cu

Ing. Andy Machín González
Universidad de las Ciencias Informáticas, Ciudad de La Habana, Cuba.
amachin@uci.cu

AGRADECIMIENTOS

Agradecemos a Fidel y la Revolución por darnos la oportunidad de ser y hacer.
A nuestros padres por su confianza y apoyo.
A nuestros tutores el Dr. Carrasco, el Msc. Aurelio Antelo y en especial al Ing. Andy Machín, que nos dio piernas para andar.
A todos los que de una manera u otra nos brindaron su ayuda.

DEDICATORIA

De Yixander:

Esta tesis es una realidad gracias al sueño de nuestro Fidel...
A la eterna confianza de mi madre, siempre enfrentando mis defectos y cultivando mis valores, con el propósito volverme útil y capaz de encontrar mi realización personal y profesional.
A mi padre, siempre en mi pensamiento, modelo como hombre y como profesional, sin olvidarlo sin importar los años que transcurran.
A los consejos de mi hermano, mi mejor amigo, la vida no sería tan fácil si no estuvieras delante.
A mi queridísima abuela, por su cariño y su confianza, madre dos veces, siempre protectora y condescendiente.
A mi "tía" Mabel, siempre cerca para escucharme y alentarme, gran amiga de siempre.
A mis amigos de siempre Yordanis, Yadir, Amek, Yendry, Milton, Yasel, Yenny, Yadira por su preocupación, aceptación y por su apoyo.
A mis compañeros, la argamasa que formó estos años de mi vida.
A Karina, por su presencia inspiradora.
A ellos les dedico este trabajo...

Dedicatoria

De Yoamel:

Dedico especialmente mi tesis a Dios que me brindó la claridad para poder salir adelante.
A mi Comandante en Jefe Fidel Castro, por sus sueños.
A mis padres Olguita y Raúl, por tanto amor y apoyo.
A mi familia toda, por ser tan grande en número y amor. Carmen y Lucía mis abuelas favoritas.
A Daviana, por ser mi inspiración.
A mis amigos de la UCI y de siempre: Milton, Daumel, Wilcer, Yadir, Amek, Rosendo, Lisdany, Oigrés, Yasel, Yendry, Erick y a todos los que han sido mi sustento en todos estos años.
A mi hermano, que se graduó antes que yo.
Al profesor Manuel Luis, por ser mi ejemplo a seguir por 5 años.
Y a todos los cubanos por esta Revolución tan hermosa.

RESUMEN

Esta investigación surge en el marco de trabajo del Proyecto "Plataforma Inteligente para la Predicción de Actividad Biológica de Compuestos Orgánicos" (GRAph TOol), una herramienta gráfica que se desarrolla conjuntamente por el Centro de Química Farmacéutica y el Grupo de Bioinformática de la Universidad de las Ciencias Informáticas, para el cual se implementa un conjunto de módulos independientes. Se desea la funcionalidad de predecir actividad biológica en compuestos orgánicos utilizando Máquinas de Soporte Vectorial (MSV), para ello se creó el presente equipo de investigación.

El descubrimiento de nuevas moléculas de interés farmacéutico está estrechamente relacionado con la búsqueda de información en grandes bases de datos y con la determinación y estimación de estructuras, es por eso que el desarrollo de aplicaciones informáticas para extraer conocimiento de la información generada en los laboratorios se manifiesta como una de las necesidades importantes en la Bioinformática.

Dentro de los objetivos de esta investigación se encuentra implementar kernels capaces de generar modelos predictivos de actividad biológica de compuestos orgánicos, aplicando Maquinas de Soporte Vectorial. Para cumplir este objetivo se emplean además algoritmos de aprendizaje combinados, en especial los del tipo infinito. Se trabaja en diferentes kernels de procesamiento, como son el Stump, el Perceptron, el Laplacian y el Exponencial.

PALABRAS CLAVE

Predicción, actividad biológica, Máquinas de Soporte Vectorial, kernel, modelo, Stump, Perceptron, Laplacian, Exponencial.

ÍNDICE

Índice

INTRODUCCIÓN

El presente trabajo está enmarcado dentro del proyecto de investigación conjunta entre el Centro de Química Farmacéutica y la Facultad 6 de la Universidad de las Ciencias Informáticas denominado **"Una Plataforma Inteligente para la Predicción de Actividad Biológica de Compuestos Orgánicos"**.

La gran cantidad de recursos y tiempo que es necesario invertir en el proceso de investigación - desarrollo de nuevos fármacos, ha animado al sector farmacéutico a incrementar el empleo de las técnicas modernas de informática, en busca de competitividad. La predicción de la actividad biológica de compuestos químicos es un importante eslabón en la producción de nuevos fármacos. Unido esto a los adelantos de las ciencias relacionadas con este campo como son la bioorgánica, la fisiología, la bioquímica, la medicina y las técnicas de computación han acelerado increíblemente el proceso de producción farmacológica a nivel internacional. Dada la amplitud de compuestos conocidos y el alto costo de los ensayos que deben realizarse para determinar la utilidad o no de uno en particular en los últimos tiempos la industria farmacéutica viene prestado especial atención a métodos que permitan una selección racional de compuestos con propiedades deseadas [1]. Muchos de los más exitosos enfoques que se le ha dado al diseño de fármacos asistidos computacionalmente están basados en la correlación entre estructura química y propiedades de las moléculas. Describir la estructura química y procesar correctamente los datos es el obstáculo a vencer para lograr una predicción efectiva desde el punto de vista informático. Dada su complejidad se acostumbra a dividir la molécula en partes, para identificar las regiones responsables de determinada respuesta.

Para optimizar el proceso anteriormente descrito juega un papel fundamental la búsqueda racional de entidades biológicamente activas candidatas a medicamentos. Sin el empleo de las técnicas de la computación este es un proceso complejo que suele llevar alrededor de una década de investigación y desarrollo con un altísimo gasto en recursos, tras lo cual tan solo un 5 o 10% de las moléculas que llegan a fase de ensayos clínicos terminan siendo comercializadas [2]. Esta es la situación que conduce al surgimiento del proyecto, una herramienta que, en su primera versión, permitirá el escrutinio virtual de moléculas orgánicas en busca de actividad biológica. Será una plataforma en la que se aplicará, como forma de descripción de la estructura química de las moléculas el Índice del Estado Refractotopológico para la ponderación de los vértices del grafo químico; este es un descriptor atómico de reciente creación en nuestro país. Mediante la suma de los pesos de cada uno de los

átomos en un fragmento se determinará el Índice Refractotopológico Total de dicho fragmento, de la relación de estos fragmentos y su valor de índice a una actividad biológica determinada, se podrán establecer similitudes cuantitativas o cualitativas, entre los factores actividad biológica y estructura química.

Es entonces una necesidad de esta plataforma conocer cuáles son los resultados de predicción de la actividad biológica en compuestos orgánicos aplicando para el procesamiento de los datos una técnica de Inteligencia Artificial (IA) de probada eficiencia como son las Máquinas de Soporte Vectorial (MSV) a partir de la descripción de la molécula por diferentes vías: fragmentos o descriptores. Lo anteriormente planteado conduce a la formulación del siguiente *problema:*

¿Cómo mejorar la predicción de actividad biológica para la plataforma GRAphTOol aplicando Máquinas de Soporte Vectorial como técnica de Inteligencia Artificial?

Existen diferentes técnicas de IA que han sido aplicadas con éxito en la modelación de moléculas con posible actividad biológica. Las más importantes son las redes neuronales, lógica difusa, algoritmos genéticos y sus diferentes variantes. La opción propuesta en la investigación son las Máquinas de Soporte Vectorial. Por lo que se define como *objeto de estudio*:
Las Máquinas de Soporte Vectorial aplicadas a la predicción de actividad biológica.

Las Máquinas de Soporte vectorial (MSV) son un nuevo sistema de aprendizaje el cual ha tenido un gran desarrollo en los últimos años tanto en la generación de nuevos algoritmos como en las estrategias para su implementación. Máquinas de Soporte Vectorial es un sistema de aprendizaje basado en el uso de un espacio de hipótesis de funciones lineales en un espacio de mayor dimensión inducido por un Kernel, en el cual las hipótesis son entrenadas por un algoritmo tomado de la teoría de optimización el cual utiliza elementos de la teoría de generalización. Además Máquinas de Soporte Vectorial es un sistema para entrenar máquinas de aprendizaje lineal eficientemente, y tanto que para la clasificación como para la regresión se han encontrado muchas aplicaciones como clasificación de imágenes, reconocimiento de caracteres, detección de proteínas, clasificación de patrones, identificación de funciones, etc. Las Máquinas de Soporte Vectorial están basadas en el principio de minimización del riesgo estructural, principio originado de la teoría de aprendizaje estadístico desarrollada por Vapnik, el cual ha demostrado ser superior al principio de minimización del riesgo empírico, utilizado por las redes neuronales convencionales. Algunas de las razones por las que este

método ha tenido éxito es que no padece de mínimos locales y el modelo solo depende de los datos con más información llamados vectores de soporte. Las ventajas que este método posee son:

1) Una excelente capacidad de generalización debido a la minimización del riesgo estructurado.
2) Existen pocos parámetros a ajustar; el modelo solo depende de los datos con mayor información.
3) La estimación de los parámetros se realiza a través de la optimización de una función de costo convexa, lo cual evita la existencia de un mínimo local.
4) La solución de Máquinas de Soporte Vectorial es escasa (sparse), esto es, que la mayoría de las variables son cero en la solución de Máquinas de Soporte Vectorial, esto quiere decir que el modelo final puede ser escrito como una combinación de un número muy pequeño de vectores de entrada, llamados vectores de soporte. [3]

Por lo que se formula como *Campo de Acción*:
Las Máquinas de Soporte Vectorial (MSV) aplicadas al estudio de la relación estructura-actividad biológica en compuestos orgánicos utilizando fragmentos y descriptores.

Se trazó como *Objetivo general*:
Implementar kernels capaces de generar modelos predictivos de actividad biológica de compuestos orgánicos, aplicando Maquinas de Soporte Vectorial.

Para dar cumplimiento al objetivo general se definieron como *objetivos específicos*:

- Proponer kernels para la predicción de actividad biológica utilizando Máquinas de Soporte Vectorial.
- Implementar un software de prueba para comprobar los algoritmos propuestos.
- Analizar los resultados de las pruebas.

Para alcanzar dichos objetivos se planteó desarrollar las siguientes *tareas:*

- Revisión del estado del arte acerca de sistemas de predicción basados en Máquinas de Soporte Vectorial desarrollados en el mundo.
- Revisión del estado del arte acerca de los algoritmos de las funciones kernel desarrollados por otros investigadores.
- Análisis del estado del arte que permita dejar definido la posición del investigador respecto al

uso de los algoritmos encontrados.

- Implementación de las funciones kernel.
- Implementación del software de prueba para comprobar los algoritmos.
- Realización de pruebas para comprobar los kernels.
- Análisis de los resultados obtenidos con las pruebas del software.

Como *aporte práctico* se espera:
El resultado práctico que se espera del presente trabajo de diploma es desarrollar una investigación en pos del mejoramiento de la predicción de actividad biológica mediante Máquinas de Soporte Vectorial e implementar una serie de algoritmos que integrados a la plataforma GRAph-TOol, permitan la aplicación de los resultados investigativos.

Este documento está compuesto por un resumen, introducción, 3 capítulos que constituyen el cuerpo fundamental del documento, conclusiones generales, bibliografía y referencias bibliográficas. Los capítulos son:

Capítulo 1: **Fundamentación Teórica**. En este capítulo se presentará un resumen de la investigación realizada sobre las Máquinas de Soporte Vectorial. Se aborda el surgimiento de estas técnicas en el mundo. Se señalan las tendencias actuales y el estado del arte a tener en cuenta. Igualmente se describen las aplicaciones informáticas más relevantes desarrolladas hasta el momento sobre este tema.

Capítulo 2: **Materiales y Métodos**. En este capítulo se realiza la descripción de la investigación como base para la continuación del estudio por otros investigadores. Se abordan además los métodos más conocidos sobre Máquinas de Soporte Vectorial realizados por otros autores así como la caracterización de los kernels introducidos a partir de la teoría ensemble. Se realiza la justificación de los materiales y métodos seleccionados y sus alternativas.

Capítulo 3: **Resultados y Discusión**. En este capítulo se explica brevemente los aspectos esenciales que se tuvieron en cuenta para la construcción del software de prueba, así como la implementación de los algoritmos, la integración con la plataforma y el aprovechamiento de la capacidad de cómputo que brinda una plataforma de cálculo distribuido. Además se ilustran y analizan

los resultados obtenidos de la investigación y las pruebas realizadas al software. Se realiza una evaluación de las implicaciones, trascendencia y beneficios de estos resultados.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

En este capítulo se abarca la fundamentación teórica de la solución. En los últimos tiempos la acumulación y aplicación de conocimientos e información científica se han convertido en factores determinantes en cualquier tipo de organización que aspire a ser competitiva. Más allá el manejo oportuno de estos conocimientos puede resultar determinante a la hora de acelerar y simplificar el proceso de obtención de resultados a los que se aspire, de manera competitiva y en un tiempo aceptable. El desarrollo de las tecnologías de la información y las comunicaciones se ha convertido en una pieza fundamental en este sentido. Además ha adquirido una relevancia impresionante la aplicación de técnicas de IA para el procesamiento de las grandes cantidades de información científica acumulada. En este marco se sitúa el trabajo, el procesamiento y la clasificación de grandes volúmenes de conocimientos, aplicando la IA. Para ello se emplea una novedosa técnica denominada Máquinas de Soporte Vectorial. Al finalizar este capítulo aspiramos a dejar en claro una serie de conceptos que resultarán fundamentales en la solución final.

1.1.1 Inteligencia Artificial

Se denomina IA a la rama de la informática que desarrolla procesos que imitan a la inteligencia de los seres vivos. La principal aplicación de esta ciencia es la creación de máquinas para la automatización de tareas que requieran un comportamiento inteligente. Este es uno entre los más sencillos conceptos de IA enmarcado fundamentalmente al enfoque informático.

Una de las más novedosas y efectivas técnicas de IA, aplicada fundamentalmente al procesamiento de grandes cantidades de información, son las Máquinas de Soporte Vectorial. Veamos en detalles en que consiste esta técnica.

1.2 Máquinas de Soporte Vectorial.

Las Máquinas de Soporte Vectorial son una moderna y efectiva técnica de Inteligencia Artificial, que ha tenido un formidable desarrollo en los últimos años, a continuación se presentarán los fundamentos teóricos que definen estos sistemas de aprendizaje.

Uno de los conceptos fundamentales en esta técnica es el algoritmo Vector de Soporte (VS) es una generalización no-lineal del algoritmo Semblanza Generalizada, desarrollado en la Rusia en los años sesenta. Como tal, este está firmemente enlazado con la Teoría del Aprendizaje Estadístico, el cual se desarrolló a finales de las últimas tres décadas por Vapnik [1982, 1995] y Chervonenkis [1974]. Por otra parte, esta Teoría de Aprendizaje Estadístico caracteriza las propiedades de aprendizaje, habilitándole el poder de generalización de datos desconocidos. El desarrollo de los VS trae consigo el surgimiento de las Máquinas de Soporte Vectorial. Estas son sistemas de aprendizaje que usan un espacio de hipótesis de funciones lineales en un espacio de rasgos de mayor dimensión, entrenadas por un algoritmo proveniente de la teoría de optimización. Este algoritmo presenta las propiedades que debe cumplir la solución del problema de optimización las cuales llevan a su vez a la implementación de algoritmos de aprendizaje derivados de la Teoría del Aprendizaje Estadístico.

A partir de la Teoría de la Generalización se considera un conjunto de entrenamiento dado $(Xi,Yi)=\{(x1,y1),(x2,y2),\ldots(xN,yN)\}$,con la entrada $xi \epsilon Rn$ y la salida $yi \epsilon R$. El problema consiste en encontrar una función f (x) que aproxime de manera óptima a una salida deseada y=f(x) y que dicha función sea capaz de clasificar correctamente los elementos de las muestras. Esta función se puede encontrar optimizando una medida de desempeño del modelo entrenado y esta medida es el Riesgo Esperado. Evaluar esta función de riesgo es un proceso complicado, pues usualmente no se conoce la función de distribución, de ahí la necesidad de emplear el riesgo sobre el conjunto de entrenamiento el cual se conoce como Riesgo Empírico. La minimización del riesgo empírico y la dimensión de Vapnik-Chervonenkis son fundamentales en las Máquinas de Soporte Vectorial. Dicho de manera más sencilla el algoritmo se enfoca en el problema general de aprender a discriminar entre miembro positivos y negativos de una clase de vectores de n-dimensional dada.

El algoritmo de Máquinas de Soporte Vectorial opera mapeando el set de entrada en un posible espacio de características dimensionalmente amplio y tiende a localizar en ese espacio un plano que separa las muestras positivas de las muestras negativas. Habiendo encontrado dicho plano, la

Máquina de Soporte Vectorial puede entonces predecir la clasificación de de una muestra mapeándola en el espacio de características y preguntando en cuál lado del plano separador se encuentra la muestra. La mayor parte del poder de las Máquinas de Soporte Vectorial viene de su criterio para seleccionar un plano separador cuando existen muchos planos candidatos: la Máquina de Soporte Vectorial escoge el plano que mantiene el margen máximo desde cualquier punto del módulo de entrenamiento. La teoría estadística de aprendizaje sugiere que, para algunas clases de datos con buen comportamiento, la selección del hiperplano de margen máximo llevará a una generalización máxima cuando se esté prediciendo la clasificación de muestras no vistas anteriormente.

Las Máquinas de Soporte Vectorial evitan la sobrecarga escogiendo el hiperplano separador de mayor margen de entre los muchos que pueden separar loas muestras positivas de las negativas en el espacio de características. También, la función de decisión para la clasificación de puntos con respecto al hiperplano sólo involucra "dot products" entre puntos en el espacio de características. Porque el algoritmo que encuentra el hiperplano separador en el espacio de características puede ser denominado en términos de vectores en el espacio de entrada y dot products en el espacio de características. La Máquina de Soporte Vectorial puede localizar el hiperplano sin representar el espacio explícitamente, simplemente definiendo una función llamada una función kernel, que juega el rol del dot product en el espacio de características. Esta técnica evita la dificultad computacional de representar explícitamente los vectores de características.

1.3 Hiperplano

Uno de los conceptos más empleados en las Máquinas de Soporte Vectorial es el de hiperplano. Un hiperplano es un concepto de geometría. Es una generalización del concepto de plano.

En un espacio de una única dimensión (como una recta), un hiperplano es un punto; divide una línea en dos líneas. En un espacio bidimensional (como el plano xy), un hiperplano es una recta; divide el plano en dos mitades. En un espacio tridimensional, un hiperplano es un plano corriente; divide el espacio en dos mitades. Este concepto también puede ser aplicado a espacios de cuatro dimensiones y más, donde estos objetos divisores se llaman simplemente hiperplanos.

Definición formal

En general, un hiperplano es un espacio afín de codimensión. En otras palabras, un hiperplano es un análogo de muchas dimensiones al plano (de dos dimensiones) en el espacio tridimensional.

Un hiperplano afín en un espacio n-dimensional puede ser descrito por una ecuación lineal no degenerada con la siguiente forma:

$$a_1x_1 + a_2x_2 + ... + a_nx_n = b.$$

Aquí no degenerada significa que no todas las ai son 0. Si b=0, se obtiene un hiperplano lineal, que pasa a través del origen.

Las dos mitades del espacio definidas por un hiperplano en espacios de n dimensiones son:

$$a_1x_1 + a_2x_2 + ... + a_nx_n \leq b$$

y

$$a_1x_1 + a_2x_2 + ... + a_nx_n \geq b.$$

1.4 Minimización del Riesgo Empírico (MRE).

Veamos ahora en que consiste la minimización del Riesgo Empírico desde un punto de vista simple, la MRE consiste en encontrar la función f(x) que minimice el riesgo promedio del conjunto de entrenamiento. En la medida en que tengamos mayor cantidad de vectores, se puede asegurar que el riesgo empírico converge asintóticamente al riesgo esperado cuando el número de valores tiende a infinito ($n \to \infty$).

1.5 Dimensión de Vapnik- Chervonenkis (VC).

Otro de los conceptos fuertemente manejados en la teoría de las Máquinas de Soporte Vectorial es la Dimensión de Vapnik Chervonenkis (VC), definida por el número máximo h de vectores z1...zn que se

pueden separar de todas las maneras posibles 2^h por los hiperplanos, a partir de funciones que miden el mayor número de muestras que pueden ser explicadas por el sistema. Donde para un conjunto de datos R, la familia de hiperplanos que se genera está dada por el término R^D en donde D es al menos D+1 para valores de D mayor que cero. Teniendo como propiedades que:

i) La dimensión de VC h_k de cada conjunto S_k de funciones es finita. Por lo tanto, $h_1 \leq h_2 ... \leq h_n ....$

ii) Cualquier elemento S_k de la estructura contiene un conjunto de funciones.

La relación existente entre los conceptos anteriormente explicados viene dada por:

$$R(f) \leq R_{emp}(f) + \underbrace{\sqrt{\frac{h(ln(2N/h)+1) - ln(\eta/4)}{N}}}_{Confianza\ VC}$$

Donde:

N = Número de muestras entrenadas.

R (f)= Riesgo Esperado.

Remp (f)= Riesgo Empírico.

h= Dimensión de Vapnik-Chervonenkis.

Si se realiza un análisis de la misma se puede asegurar que: el modelo más adecuado se encuentra cuando se logra un balance entre el Riesgo Empírico (Remp(f)) y la dimensión de VC, este problema se conoce como la minimización del riesgo estructural. Además, a medida que la relación N/h se hace mayor, la confianza VC se hace menor y el Riesgo Esperado se acerca al Riesgo Empírico.

Una Máquina de Soporte Vectorial mapea los puntos de entrada en un espacio de características de una dimensión mayor y encuentra el hiperplano óptimo que los separe y maximice el margen **m** entre las clases, en este espacio. **Ver Figura1**.

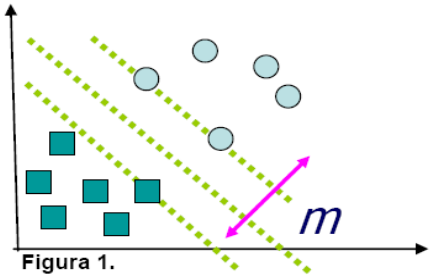


Figura 1.

A partir del conjunto de entrenamiento, donde el dominio de las imágenes se encuentran en el intervalo [-1; 1], las ecuaciones de los hiperplanos H1 y H2 viene dada por: H1= wxi+b=1, H2= wxi+b=-1 y el hiperplano óptimo por wxi+b=0, partiendo de que ||w|| es la norma del vector normal al hiperplano para las clases que se representan. Por lo tanto el margen m se calcula por la expresión m=2/||w||. **Ver Figura 2**. [4]

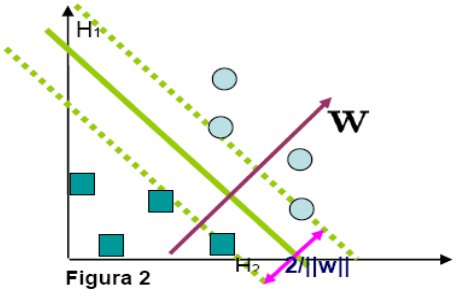


Figura 2

.

Maximizar dicho margen es un problema de programación cuadrática. Uno de los métodos para resolver el problema es a través de la teoría de Lagrange el cual fue extendido al problema de optimización con restricciones. Los conceptos principales de esta teoría son el concepto de multiplicadores de Lagrange (α) y la función del Lagrangiano (Lp (w, b, α)) relacionadas en la ecuación:

$$L_P(w,b,\alpha) = \frac{1}{2}\|w\|^2 - \sum_{i-1}^{N} \alpha_i \left[y_i \left(w^T x_i + b \right) - 1 \right]$$

Dicha teoría es una generalización del resultado de Fermat en 1629, el cual plantea el problema de optimización sin restricciones.

En 1951 Kuhn y Tucker generalizaron la teoría de Lagrange permitiéndose entonces la introducción de restricciones de desigualdad en el problema de optimización. Solo los puntos que estén situados en unos de los hiperplanos cumple que α>0 y se les denomina vectores de soporte, que son los que ayudan a definir el hiperplano óptimo, según la expresión:

$$\frac{\partial J(w,b,\alpha)}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^{N} \alpha_i y_i x_i$$

En el que se utiliza el producto del punto, con funciones en el espacio de características que son llamadas Kernels. La minimización del Riesgo Empírico y la dimensión de Vapnik-Chervonenkis son fundamentales para lograr la búsqueda de la solución óptima mediante funciones que en un espacio numérico determinado, logre el hiperplano separador. En la representación de estos conjuntos de

entrenamiento, los mismos pueden estar linealmente separados. De no ser así, se utilizan las funciones Kernels para llevar estas muestras a un plano de mayor dimensión donde puedan ser linealmente separables. Dentro de los más utilizados se encuentran: el Kernel Polinomial y el de Base Radial. Las Máquinas de Soporte Vectorial pueden trabajar con dos o más clases en dependencia del problema al que se apliquen.[5]

1.6 Condiciones Karush-Kuhn-Tucker (KKT) necesarias.

Uno de los aspectos más relevantes para la aplicación de las Máquinas de Soporte Vectorial, es el cumplimiento de las condiciones Karush-Kuhn-Tucker (KKT), estas son:

Condición del gradiente:

$$\frac{\partial L_p}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{l} \lambda_i y_i \mathbf{x_i} = 0,$$

$$\frac{\partial L_p}{\partial \mathbf{w}} = \left(\frac{\partial L_p}{\partial w_1}, \frac{\partial L_p}{\partial w_2}, \ldots, \frac{\partial L_p}{\partial w_d} \right)$$

$$\frac{\partial L_p}{\partial b} = \sum_{i=1}^{l} \lambda_i y_i = 0,$$

$$\frac{\partial L_p}{\partial \lambda_i} = g_i(\mathbf{x}) = 0.$$

Condición de ortogonalidad:

λigi = -λi(yi(w*xi-b)-1)=0 i=1,...,l

Condición de viabilidad:

-yi (w*xi-b)+1<=0 i=1,....l

Condición de no-negatividad:

λi >= 0 i=1,.....,l.

El punto de ensilladura de Lagrange determina la solución al problema de la optimización.

El problema dual es:

Maximizar

$$L_D = \sum_{i=1}^{l} \lambda_i - \frac{1}{2} \sum_{i=1}^{l} \sum_{i=1}^{l} \lambda_i \lambda_j y_i y_j \mathbf{x_i} \cdot \mathbf{x_j}$$

Donde

$$\sum_{i=1}^{l} \lambda_i y_i = 0$$

$$\lambda_i \geq 0 \qquad i = 1, \ldots, l.$$

Se pueden encontrar todos los λ' solucionando la ecuación con la Programación Cuadrática.
w se obtiene mediante la siguiente ecuación

$$\mathbf{w} = \sum_{i=1}^{l} \lambda_i y_i \mathbf{x_i}.$$

λi es > 0 en el punto donde un vector de soporte y su restricción correspondiente son activos. Es cero en el punto que no es un vector de soporte y su restricción correspondiente son inactivos. Podemos calcular b usando la ecuación

λi (yi (w*xi-b) -1) = 0 i=1,.....,l

Eligiendo el xi con λ distinto de cero.

La clase de los datos de entrada x se determina con: **class(x) = sign (w*x-b)**

En la práctica debido a la implementación numérica, el valor medio de b es encontrado calculando el promedio.

1.7 Máquinas de Soporte Vectorial para la Clasificación.

Entre las aplicaciones más relevantes de las Máquinas de Soporte Vectorial se encuentra la Clasificación, veamos a continuación como funciona el conjunto de algoritmos de aprendizaje para esta función.

El problema de la clasificación puede reducirse a examinar dos clases sin pérdida de generalidad. En este problema el objetivo es separar mediante una función inducida por los datos, las dos clases presentes.

$"D = \{(x^1, y^1), \ldots, (x^l, y^l)\}, \quad x \in R^n, y \in \{-1, 1\}.$

La tarea es encontrar un clasificador que funcione bien en datos futuros, es decir que generalice bien la clasificación.

Consideremos el siguiente ejemplo:

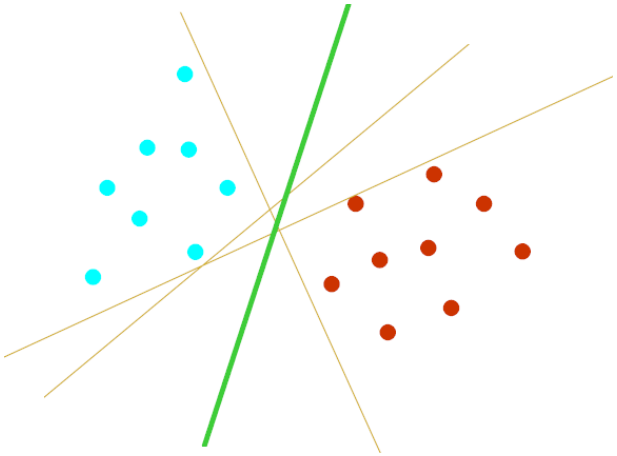


Figura 3 Hiperplano Separador Óptimo

Aquí hay muchos posibles clasificadores lineales que pueden separar los datos; sin embargo solo existe un clasificador que maximiza el margen, maximiza la distancia entre él y los puntos más cercanos de los datos de cada clase. A este clasificador lineal se le denomina hiperplano separador óptimo.

De forma intuitiva se puede esperar que este separador generalice bien la clasificación para datos futuros.

1.7.1 El hiperplano separador óptimo.

El concepto de hiperplano separador óptimo surge cuando se considera el problema de separar el conjunto de vectores de entrada en dos clases a las que pertenecen; usando un hiperplano.

$$\langle w, x \rangle + b = 0.$$

Se dice que un conjunto de vectores ha sido óptimamente separado por un hiperplano, cuando se separan sin errores y la distancia entre el vector más cercano y el hiperplano es la máxima.

Sin perder generalidad se puede considerar el hiperplano canónico donde los parámetros w, b se expresan mediante:

$$\min_i \left| \langle w, x^i \rangle + b \right| = 1$$

La norma del peso del vector debe ser igual a la inversa de la distancia del punto más cercano en el conjunto de datos al hiperplano. Como se muestra en la siguiente figura:

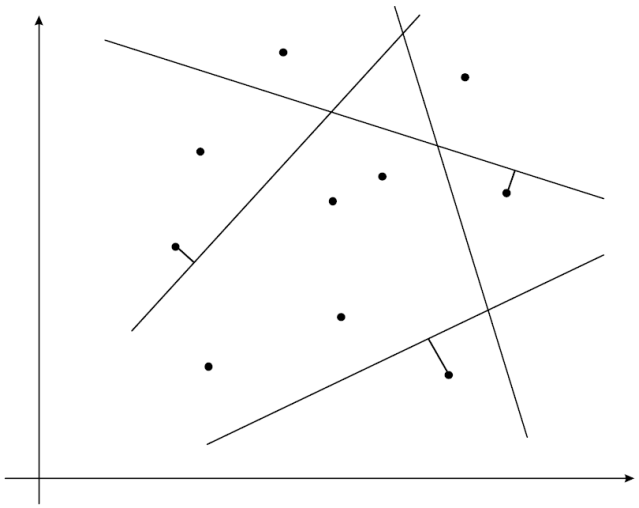


Figura 4 Hiperplanos Canónicos

Un hiperplano separador en forma canónica debe satisfacer las siguientes condiciones.

$$y^i \left[\langle w, x^i \rangle + b \right] \geq 1, \quad i = 1, \ldots, l.$$

La distancia d (w, b ; x) al punto x desde el hiperplano (w , b) es:

$$d(w, b, x) = \frac{|\langle w, x^i \rangle + b|}{\|w\|}$$

El hiperplano óptimo viene dado por maximizar el margen, *p,* sujeto a la ecuación:

$$y^i[\langle w, x^i \rangle + b] \geq 1, \quad i = 1, \ldots, l.$$

El margen viene dado por:

$$\rho(w, b) = \min_{x^i; y^i = -1} d(w, b; x^i) + \min_{x^i; y^i = 1} d(w, b; x^i)$$

$$= \min_{x^i; y^i = -1} \frac{|\langle w, x^i \rangle + b|}{\|w\|} + \min_{x^i; y^i = 1} \frac{|\langle w, x^i \rangle + b|}{\|w\|}$$

$$= \frac{1}{\|w\|} \left(\min_{x^i; y^i = -1} |\langle w, x^i \rangle + b| + \min_{x^i; y^i = 1} |\langle w, x^i \rangle + b| \right)$$

$$= \frac{2}{\|w\|}$$

De ahí que el hiperplano que separa de forma óptima los datos es el que minimiza:

$$\phi(w) = \frac{1}{2} \|w\|^2$$

Es independiente de (b) pues ya se considera que es un hiperplano separador. Se supone entonces que:

$$\|w\| < A$$

Entonces:

$$d(w, b, x) \geq \frac{1}{A}$$

De acuerdo a esto los hiperplanos no pueden acercarse más de 1/A a ningún punto de los datos. Como se aprecia en la siguiente figura:

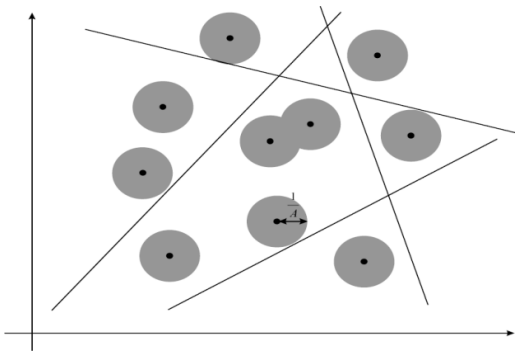


Figura 5 Restricción del Hiperplano Canónico

La dimensión Vapnik-Chervonenkis, h, del conjunto canónico de hiperplanos en el espacio n dimensional es delimitado por:

$$h \leq \min[R^2A^2, n] + 1,$$

Donde R es el radio de la hiperesfera que engloba todos los puntos del juego de datos.

Evitando la pérdida del límite superior en la dimensión Vapnik Chervonenkis empleamos el punto de ensilladura dado por la función de Lagrange.

$$\phi(w, b, \propto) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{l} \propto_i \left(y^i\left[\langle w, x^i\rangle + b\right] - 1\right)$$

Donde α es el multiplicador de Lagrange, la lagraniana debe ser minimizada con respecto a w, b y maximizada con respecto a α >= 0.

La clásica dualidad lagraniana permite que el problema primario de la ecuación 2.11 sea transformado en el siguiente problema dual, que resulta de más fácil solución. El problema dual viene dado por:

$$\max_{\propto} W(\propto) = \max_{\propto}\left(\min_{w,b} \phi(w, b, \propto)\right).$$

El mínimo con respecto a w, b en la lagraniana ϕ viene dado por:

$$\frac{\partial \phi}{\partial b} = 0 \Rightarrow \sum_{i=1}^{l} \propto_i y_i = 0$$

$$\frac{\partial \phi}{\partial w} = 0 \Rightarrow w = \sum_{i=1}^{l} \propto_i y_i x_i = 0$$

Combinando las tres ecuaciones anteriores el problema dual sería:

$$\max_{\propto} W(\propto) = \max_{\propto} -\frac{1}{2} \sum_{i=1}^{l} \quad \sum_{j=1}^{l} \propto_i \propto_j y_i y_j \langle x_i, x_j \rangle + \sum_{k=1}^{l} \propto_k$$

Así la solución al problema viene dada por:

$$\propto^* = arg \min_{\propto} \frac{1}{2} \sum_{i=1}^{l} \quad \sum_{j=1}^{l} \propto_i \propto_j y_i y_j \langle x_i, x_j \rangle - \sum_{k=1}^{l} \propto_k,$$

Sujeto a:

$$\propto_i \geq 0 \quad i = 1, \ldots, l$$

$$\sum_{i=1}^{l} \propto_j y_j = 0.$$

Resolviendo la ecuación 2.15, con las sujeciones 2.16 se determina el multiplicador de Lagrange y el hiperplano separador óptimo viene dado por:

$$w^* = \quad \sum_{i=1}^{l} \propto_i y_i x_i,$$

$$w^* = -\frac{1}{2}\langle w^*, x_r + x_s \rangle.$$

Donde x_r y x_s es cualquier vector de soporte para cada clase que satisfaga:

$$\alpha_r, \alpha_s > 0, \qquad y_r = -1, y_s = 1.$$

El clasificador fuerte es entonces:

$$f(x) = sgn(\langle w^*, x \rangle + b)$$

Alternativamente, un clasificador suave puede ser usado para interpolar linealmente el margen

$$f(x) = h(\langle w^*, x \rangle + b) \quad where \quad h(z) = \begin{cases} -1 & : \quad z < -1 \\ z & : \quad -1 \leq z \leq 1 \\ +1 & : \quad z > 1 \end{cases}$$

Esta puede ser una solución más apropiada que el clasificador fuerte pues produce un valor de salida real entre -1 y 1 cuando el clasificador es solicitado dentro del margen, donde no residen datos de entrenamiento.

De las condiciones de Kuhn-Tucker *(ver Condiciones KKT en este mismo capítulo)

$$\alpha_i \left(y^i \left[\langle w, x^i \rangle + b \right] - 1 \right) = 0, i = 1, \ldots, l$$

Así solo el punto x^i puede satisfacer:

$$y^i \left[\langle w, x^i \rangle + b \right] = 1$$

De esta forma no se tendrán multiplicadores de Lagrange de valor 0.

Estos puntos reciben el nombre de vectores de soporte. Si la información es linealmente separable todos los vectores de soporte se encontrarán ajustados al margen y además el número de vectores de soporte será muy pequeño.

Como consecuencia el hiperplano es determinada por una pequeña parte de los datos de entrenamiento, el resto de los puntos podrán ser removidos del conjunto de entrenamiento y al recalcular el hiperplano dará el mismo resultado. De esta forma las Máquinas de Soporte Vectorial pueden ser usadas para resumir la información contenida en un juego de datos a sus vectores de soporte. Si las clases son linealmente separables la siguiente igualdad se mantendrá:

$$\|w\|^2 = \sum_{i=1}^{l} \alpha_i = \sum_{i \in SV_s} \alpha_i = \sum_{i \in SV_s} \sum_{j \in SV_s} \alpha_i \alpha_j y_i y_j (x_i, x_j)$$

Así de la ecuación 2.10 la dimensión Vapnik Chervonenkis del clasificador está dada por:

$$h \leq \min \left[R^2 \sum_{i \in SV_s}, n \right] + 1$$

X1	X2	y
1	1	-1
3	3	1
1	3	1
3	1	-1
2	2.5	1
3	2.5	-1
4	3	-1

Tabla de datos linealmente separables y si los datos de entrenamiento, x, son normalizados para quedar dentro de la hiperesfera.

$$h \leq 1 + \min \left[\sum_{i \in SV_s}, n \right]$$

1.8 Máquinas de Soporte Vectorial para Regresión.

Las Máquinas de Soporte Vectorial se desarrollaron inicialmente para solucionar problemas de clasificación, pero se han ampliado para problemas de regresión. Los resultados finales a los que se puede arribar luego del empleo de las SVM pueden ser cualitativos o cuantitativos, para en el análisis cuantitativo se emplean Máquinas de Soporte Vectorial para regresión. Dicho método es una extensión del anteriormente explicado. Se conoce que la regresión crea una regla para predecir rasgos numéricos desconocidos desde rasgos numéricos conocidos y entre los cuales la relación que existe es la naturaleza del vector, por lo que esta es una forma de aprendizaje supervisado. Los métodos para estimar dependencias funcionales basadas en los datos se introdujeron por los matemáticos Gauss (1777-1855) y Laplace (1749-1827) los cuales basaron su teoría en el cálculo de los mínimos cuadrados y el módulo de los mismos. Es necesario decir que aunque estos procedimientos matemáticos, que la fundamentan son antiguos, la Regresión Soportada por Vectores (RSV) fue creada en el año 1996.

El empleo de estimadores robustos para determinar los valores que tienen ruido dentro de la predicción se estiman a través de funciones de pérdida, donde los primeros pasos en este sentido se dieron por Tuckey quien demostró que en situaciones reales, se desconoce el modelo del ruido y dista de las distribuciones supuestas. A raíz de esto, Huber crea el concepto de estimadores robustos los cuales están determinados por funciones de pérdida. En la actualidad, las más utilizadas son: las funciones de pérdida cuadrática y lineal, y la de Huber, entre otras.

La función de pérdida conocida como ε-*insensitive*, le aporta una propiedad nueva a las soluciones que dan las máquinas de soporte vectorial, la llamada *esparcidad* de la solución. Ella tiene la ventaja de que no se necesitan todos los patrones de entrada para describir el vector de regresión, posibilitando una vía más eficiente, desde el punto de vista computacional, para su implementación.

Las Máquinas de Soporte Vectorial para regresión utilizada están basadas en la función de costo ε-*insensitive.* Esta función tiene la estructura que combina dos funciones, una de las cuales es $f(u)=|u|$ y la función constante $f(u) = const...$ Definida por $y; -(w .x;)-b \leq \epsilon.$

Este tipo de funciones pueden crearse además, en la medida en que se desarrolle una estimación determinada, dando paso así a nuevas definiciones y modificaciones en los datos de entrada*: (x_1 , y_1),..., (x_l , y_l) Є X x R* donde *X* denota el espacio de entrada de los patrones.

En el método de regresión por vectores de soporte, usando la función de costo ε-insensitive, el objetivo es encontrar una función $f(x)$ que tenga como máximo una desviación ε de los valores de salida y_i (Figura 6) para todos los datos de entrenamiento y al mismo tiempo sea tan lineal como sea posible. Se describe la función como: $f(x)=\langle w, x\rangle + b$ con w Є X, b Є R.

La solución de esta ecuación se convierte en un problema de optimización convexo, tal como se representa a continuación.

$$\text{Minimizar} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^*)$$

$$\text{Sujeto a:} \quad \begin{cases} y_i - \langle w, x_i\rangle - b \le \varepsilon + \xi_i \\ \langle w, x_i\rangle + b - y_i \le \varepsilon + \xi_i^* \end{cases}$$

$$\xi_i,\ \xi_i^* \ge 0$$

Donde ξ_i y ξ_i^* son variables de holgura introducidas para las situaciones en las que es necesario permitir errores y la constante C >0 determina el compromiso entre el aplanamiento de *f* y la cantidad cual las desviaciones mayores que ε, se toleran, siendo un parámetro de regularización y libre de ser ajustado a la maximización del margen m. La formulación anterior es la correspondiente a trabajar con la función de pérdida ε-insensitive que se describe como:

$$\|\xi\|_\varepsilon = \begin{cases} 0 & si\ |\xi_i| \le \varepsilon \\ |\xi_i| - \varepsilon & en\ otro\ caso \end{cases}$$

Solamente los puntos que se encuentren fuera de esta condición son los que contribuyen con las funciones de pérdida. **Ver Figura 6.**

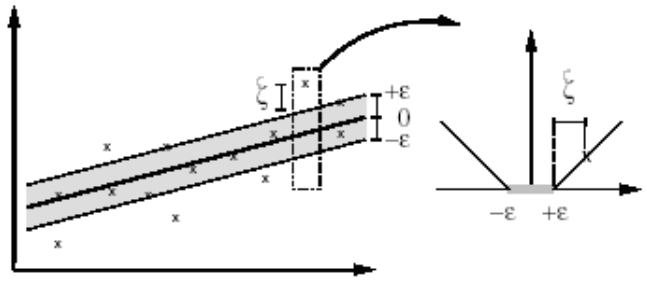


Figura 6.

La solución de este problema de optimización se realiza, al igual que con las Máquinas de Soporte Vectorial, mediante el empleo de los multiplicadores de Lagrange y la función del Lagrangiano.

Por lo tanto, se puede afirmar que en este tipo de técnicas, su estructura se determina sobre la base del conjunto de entrenamiento necesitándose pocos parámetros para el mismo. Dicho entrenamiento se reduce a la solución de un problema de optimización que se reduce a un problema de programación cuadrática. Al mismo tiempo, el uso de las funciones Kernels muestra una gran eficiencia en el resultado de la predicción.

1.9 Estudios realizados en los últimos años.

Las Máquinas de Soporte Vectorial han tenido gran aplicación en la bioinformática, principalmente en la predicción de actividad biológica en compuestos y en muchos otros campos de estudio en los últimos años, por ejemplo:

Año 2001

Se emplearon en el diseño de medicamentos para análisis de datos farmacéuticos en la Universidad de London.[6]

Año 2003

Fueron utilizadas para el diseño de fármacos y el estudio de similitud de los Medicamentos y predicción de la inhibición de enzimas. En el Departamento de Química de la Universidad de Moscú.[7]

Año 2004

Las Máquinas de Soporte Vectorial fueron utilizadas en la Universidad de París para desarrollar los modelos de QSAR que correlacionan las estructuras moleculares a su toxicidad y bioactividades. El funcionamiento y la capacidad predictiva de Máquinas de Soporte Vectorial, fueron descritas usando los parámetros fisicoquímicos o los descriptores moleculares. En ambos casos estudiados, la capacidad predictiva del modelo de Máquinas de Soporte Vectorial es comparable o superior. Los resultados indican que Máquinas de Soporte Vectorial se puede utilizar como herramienta para modelar los estudios QSAR.[8]

Se empleó en la Universidad de París en el estudio cuantitativo de la relación estructura-movilidad de los ácidos carboxílicos en la electroforesis de vasos capilares.[9]

Las Máquinas de Soporte Vectorial fueron utilizadas en el Departamento de Química de la Universidad de Lanzhou para desarrollar un modelo cuantitativo de la relación de la Estructura-Característica (QSPR) de la energía en enlace de la disociación del Oxígeno con el hidrógeno(O-H) de 78 fenoles substituidos.[10]

Las Máquinas de Soporte Vectorial, como tipo novedoso de máquina que aprendía, fueron utilizadas para desarrollar un modelo cuantitativo de la relación de la estructura-movilidad (QSMR) de 58 alifáticos y de ácidos carboxílicos aromáticos basados en los descriptores moleculares.[11]

Año 2005

La Universidad de Lanzhou utilizó Máquinas de Soporte Vectorial en el estudio QSAR de los compuestos de interrupción de la endocrina natural, sintética y ambiental para atar al receptor del andrógeno. Donde a partir de cinco descriptores el método mostró excelentes resultado.[12]

Año 2006

Fue empleada para determinar la predicción de un modelo cuantitativo de la relación de la estructura-característica (QSPR) se ha desarrollado para la degradación electroquímica de fenoles substituidos. Comparado con los modelos desarrollados con los cuadrados (PLS) y la regresión linear múltiple

(MLR), donde estaban 0.804 y 0.799 Q2 respectivamente, Máquinas de Soporte Vectorial demostraron rendimientos más altos.[13]

Se emplearon para predecir relaciones cuantitativas de la estructura característica para los pesticidas en la universidad de Lanzhou, donde se probaron con: método heurístico (HM) y Máquinas de Soporte Vectorial y los modelos lineares y no lineales, generados por ambos se compararon los resultados de estos dos métodos, resultando las Máquinas de Soporte Vectorial el mejor método.[14]

Se emplearon en relación cuantitativa de la estructura-actividad de modelos para la predicción de los irritantes sensoriales de productos químicos orgánicos volátiles. De ellos se han desarrollado: modelos de clasificación y de regresión para la predicción de los irritantes sensoriales (LOGRD50) de los productos químicos orgánicos volátiles (VOCs). Cada compuesto fue representado por los descriptores estructurales.[15]

Año 2007

En un estudio conjunto entre las venezolanas Universidad Nacional Experimental Francisco de Miranda y la Universidad Simón Bolívar y el Real Hospital Victoria y la Universidad de Ulster del Reino Unido se aplicaron las Máquinas de Soporte Vectorial para la predicción de la ocurrencia de Shocks eléctricos en la cardioversión interna de pacientes con fibrilación atrial también conocida como auricular. [16]

En la Universidad de las Ciencias Informáticas se realizó un ensayo que demostró la aplicabilidad de las Máquinas de Soporte Vectorial en la predicción de actividad biológica en compuestos orgánicos.

1.10 Conclusiones

Tras analizar el funcionamiento y la efectividad de las Máquinas de Soporte Vectorial en la solución de problemas como el que se plantea, al no tener una solución libre y fácilmente integrable con la Plataforma GRAphTOol y al identificar numerosas mejoras posibles a partir de la de la implementación de nuevos kernels en los paquetes actualmente utilizables, se decidió desarrollar una solución informática aplicando las Máquinas de Soporte Vectorial.

Con este capítulo se aclararon los más importantes conceptos relativos a las Máquinas de Soporte Vectorial y se garantizó un fundamento para la futura comprensión del trabajo. Con el desarrollo de la aplicación se logrará la creación de modelos predictivos y la aplicación de estos modelos sobre juegos de datos para realizar predicciones de la actividad biológica, empleando la Plataforma GRAphTOol.

Esto permitirá además hacer más breve el proceso de búsqueda de nuevas compuestos con posible actividad biológica en un sentido determinado, las cuales servirán para el desarrollo de nuevos fármacos y disminuirá sus costos de investigación y desarrollo.

CAPÍTULO 2: MATERIALES Y MÉTODOS.

2.1 Introducción

En este capítulo se presentarán los materiales utilizados para el desarrollo de los algoritmos y los métodos que se siguieron a la hora de utilizar dichos materiales. Se enfocarán los algoritmos de aprendizaje ensemble y ensemble infinitos; en el marco de estos se abordarán los nuevos kernels de procesamiento, añadidos a las librerías tradicionales y se dará una sencilla explicación de cada uno, como ejemplos de los métodos más relevantes para arribar a la solución.

2.2 Materiales

Dado el gran número de herramientas disponibles hoy para el desarrollo de software se hace imprescindible un trabajo de selección de las que resulten óptimas para el trabajo.

La efectividad de las predicciones realizadas con Máquinas de Soporte Vectorial alrededor del mundo ha dado lugar a un auge en la creación de software utilizando esta técnica de Inteligencia Artificial, como se puede apreciar en el capítulo anterior.

A continuación se enumerarán las tecnologías seleccionadas para el trabajo y se expondrán las razones que llevaron a esta selección.

2.2.1 Lenguaje de Modelado (UML)

Según su definición, UML (Unified Modeling Language – Lenguaje Unificado de Modelado) es un lenguaje de modelado visual que se usa para especificar, visualizar, construir y documentar artefactos de un sistema de software desde una perspectiva orientada a objetos.
UML es un lenguaje que:

- ✓ Sus modelos pueden conectarse de forma directa con una gran variedad de lenguajes de programación.

- ✓ Proporciona un vocabulario y unas reglas que se centran en la representación conceptual y

física de un sistema y que indican cómo crear y leer modelos bien formados.

- ✓ Mezcla gráficos y texto, de hecho, detrás de cada símbolo en la notación UML hay una semántica bien definida, de manera que un desarrollador puede escribir un modelo en UML, y otro desarrollador, o incluso otra herramienta, puede interpretar ese modelo sin ambigüedad.

- ✓ Cubre toda la documentación de la arquitectura de un sistema y todos sus detalles. También proporciona un lenguaje para expresar requisitos y pruebas del software.

- ✓ Especifica todas las decisiones de análisis, diseño e implementación que deben realizarse al desarrollar y desplegar un sistema.

- ✓ Modela las actividades de planificación de proyectos y gestión de versiones.

¿Por qué UML?

Siendo UML el estándar internacional para el modelado, se puede decir que es necesario y obligatorio el mejorar la calidad del desarrollo de software y para esto se deben adoptar procedimientos, metodologías y herramientas que permitan una estandarización en la ingeniería de software, esto es precisamente lo que ofrecen los lenguajes de modelado de software, un lenguaje común que permite el crear una disciplina con estándares que tiene las siguientes características:

- ✓ Modela sistemas utilizando técnicas orientadas a objetos (OO).

- ✓ Permite especificar todas las decisiones de análisis, diseño e implementación, construyéndose así modelos precisos, no ambiguos y completos.

- ✓ Puede conectarse con lenguajes de programación (Ingeniería directa e inversa).

- ✓ Permite documentar todos los artefactos de un proceso de desarrollo (requisitos, arquitectura, pruebas, versiones, etc.).

- ✓ Cubre las cuestiones relacionadas con el tamaño, propias de los sistemas complejos y críticos.

- ✓ Es un lenguaje muy expresivo que cubre todas las vistas necesarias para desarrollar y luego desplegar los sistemas.

- ✓ Existe un equilibrio entre expresividad y simplicidad, pues no es difícil de aprender ni de utilizar.

- ✓ UML es independiente del proceso, aunque para utilizarlo óptimamente se debería usar en un proceso que fuese dirigido por los casos de uso, centrado en la arquitectura, iterativo e incremental.

2.2.2 Herramienta CASE.

La herramienta *CASE* utilizada para modelar el programa es el *Visual Paradigm* que es un producto de *Visual Paradigm UML Community*. Tiene disponible distintas versiones: *Enterprise*, *Professional*, *Standard*, *Modeler*, *Personal* y *Community* la cual es gratuita. Unas de las principales diferencias entre el *Visual Paradigm* y otros sistemas basados en *Windows,* está en la fuerza de la interfaz de la aplicación presentado a los usuarios y la base de todas las aplicaciones de base de datos. *Visual Paradigm* provee a los usuarios de una presentación con *Multiple Document Interface* (MDI). Todas las vistas tienen menús *pull-down*, barra de botones, de teclado y de navegación de los datos. Las vistas proporcionan menús *pop-up* del contexto, ayuda del estado en todos los campos, listas de la selección y las tablas de la validación, así como confirmaciones y advertencias en todas las operaciones de los datos. *Visual Paradigm* es una herramienta que sirve para realizar modelado UML siguiendo el estándar UML. Esta herramienta tiene unas características graficas muy cómodas que facilitan la realización de los diagramas de modelado. Entre otras características importantes que tiene es la integración con algunos IDE de programación como Eclipse desarrollado por IBM, *Netbeans de Sun* o *JBuilder de Borland*.

2.2.3 Plataforma de Desarrollo.

En cuanto a plataforma escogida se optó por jdk versión 1.5.0_10 y como lenguaje de programación Java debido a que es un lenguaje de propósito general y al hecho de ser independiente de la plataforma, buscando la portabilidad en los diferentes sistemas operativos y plataformas de hardware.

2.2.4 Entorno de Desarrollo.

Se utilizó Eclipse como entorno de desarrollo, es una plataforma de software de código abierto, extensible. Esta plataforma, típicamente ha sido usada para desarrollar entornos integrados de desarrollo, como el IDE de Java llamado *Java Development Toolkit* (JDT) y el compilador (ECJ) que se embarca como parte de Eclipse y que son usados también para desarrollar este entorno. Sin embargo, también se puede usar para otros tipos de aplicaciones cliente además de ser un entorno de desarrollo integrado que ofrece el control del editor de códigos, del compilador y del depurador desde una única interfaz de usuario. Este entorno de desarrollo integrado ofrece, el control del editor de código, del compilador y del depurador desde una única interfaz de usuario. Además Eclipse es una plataforma universal para integrar herramientas de desarrollo, basada en plug-ins.

2.2.5 Gestor de Bases de Datos.

Finalmente como sistema gestor de bases de datos se ha utilizado MySQL en su versión 5.0 debido a ser software libre, es uno de los gestores más rápidos que se encuentran en el mercado, presenta versiones en varios sistemas operativos y compatibilidad entre sus versiones y es muy fácil de usar.

MySQL es un sistema de administración de bases de datos relacional (RDBMS). Se trata de un programa capaz de almacenar gran cantidad de datos y distribuirlos para cubrir las necesidades de cualquier tipo de organización, desde pequeños establecimientos comerciales a grandes empresas y organismos administrativos. Existe la seguridad de contar con una importante cuota de mercado y de saber que es una solución estable, mantenida por un buen equipo de desarrolladores y e incluso con soporte de pago. Compite con sistemas RDBMS como Oracle, *SQL Server* entre otros. Incluye todos los elementos necesarios para instalar el programa, preparar diferentes niveles de acceso de usuarios, administrar el sistema y proteger los datos. Utiliza el lenguaje de consulta estructurado.

Facilidades.

MySQL es un sistema fácil de instalar y configurar en servidores Windows, Linux entre otros. Además ofrece facilidades para la realización de consultas para el manejo de los datos. Permite centralizar la base de datos y brinda mayor seguridad para estas.

Funcionalidad.

MySQL dispone de muchas funciones vitales para el desarrollo profesional cómo puede ser el volcado online, la duplicación etc.

Portabilidad.

MySQL puede correr en la inmensa mayoría de sistemas operativos, por lo que junto a otro lenguaje de programación de lado de servidor de alta portabilidad como Java y PHP permite el desarrollo de aplicaciones, el acceso y copia de los datos desde cualquier Sistema Operativo.

2.3 MÉTODOS.

2.3.1 Algoritmos de aprendizaje ensemble.

Entre los métodos más importantes aplicados en la investigación, se encuentran los algoritmos de aprendizaje ensemble, que representan toda una corriente de pensamiento en la creación de kernels de procesamiento.

El paradigma de aprendizaje ensemble denota una amplia colección de algoritmos de aprendizaje. En lugar de considerar un poderoso modelo de aprendizaje G, un algoritmo de aprendizaje conjunto A, maneja modelos de aprendizaje base H, que son generalmente simples.

El clasificador $h \in H$ es también llamado algoritmo de aprendizaje básico. El algoritmo entonces construye una función de decisión g^* by 2

$$g^*(x) = \operatorname{sign}\left(\sum_{t=1}^{T} w_t h_t(x)\right),$$
$$w_t \geq 0, t = 1, 2, \cdots, T.$$

Cualquier clasificador g que pueda ser expresado con esta función es considerado un clasificador ensemble y la w_t en la función es llamada el *peso de la hipótesis.*

Aunque H podría ser de tamaño infinito, en teoría, el conjunto tradicional de los algoritmos de aprendizaje suele ser finito y predefinido en T. Denominados *algoritmos finitos de aprendizaje ensemble.* Estos algoritmos suelen compartir otra característica común: eligen cada hipótesis h_t llamando a otro algoritmo de aprendizaje A_H, conocido como algoritmo de aprendizaje base.

Los algoritmos ensemble de aprendizaje son útiles por contar con algunas o todas las siguientes propiedades: estabilidad, precisión y eficiencia.

Estabilidad: Cuando un algoritmo de aprendizaje devuelve una muy diferente función de decisión g ante un pequeño cambio en los datos, llamamos a este algoritmo inestable. Los algoritmos inestables no son deseables pues suelen ser susceptibles al ruido, medidas imprecisas o incluso pequeños errores en la entrada de los datos. Los algoritmos de aprendizaje ensemble son un acercamiento para brindarle estabilidad a los algoritmos base, dado que la combinación de varios de estos algoritmos base da como resultado una mayor generalización estabilizando los resultados.

Precisión: La precisión de los algoritmos de aprendizaje ensemble viene dada por la eliminación o reducción del error individualmente acumulado por cada algoritmo base, logrando que el posible error en el algoritmo ensemble sea eliminado o reducido.

Eficiencia: La eficiencia de los algoritmos de aprendizaje radica en que en vez de trabajar con un modelo G complejo que generalmente complica el procesamiento y reduce la eficiencia, el tipo de algoritmos ensemble divide el problema en varios problemas pequeños, es decir trabaja con varios algoritmos base más simples, con esa técnica de divide y vencerás el problema antes complejo queda reducido a un buen número de modelos sencillos que pueden ser rápidamente procesados, brindándole eficiencia al algoritmo resultante.

2.3.2 Algoritmos de aprendizaje ensemble infinitos.

Ya se han introducido los principios básicos de los algoritmos ensemble de aprendizaje. Ahora se hará un acercamiento a las motivaciones y posibles dificultades al trabajar con algoritmos de aprendizaje ensemble infinitos.

La más importante razón para pasar de algoritmos de aprendizaje ensemble finitos a infinitos es para aumentar aún más la capacidad de los modelos de aprendizaje. La capacidad de los algoritmos ensemble de aprendizaje está limitada en poder.

Hasta hoy los algoritmos de aprendizaje ensemble, que son finitos, han sido vistos como una simulación del comportamiento cuando T tiende a infinito, es decir son una simulación de un algoritmo de aprendizaje ensemble infinito. Hasta ahora unos algoritmos tratan de aplicar el paradigma de aprendizaje infinito, mientras otros hacen crecer la cantidad de algoritmos base para que T se acerque cada vez más a infinito. Eso nos sugiere que un trabajo con algoritmos de aprendizaje ensemble infinitos resulta imprescindible.

2.3.3 Kernels.

Las funciones kernel son funciones matemáticas que se emplean en las Máquinas de Soporte Vectorial. Estas funciones son las que le permiten convertir lo que sería un problema de clasificación no lineal en el espacio dimensional original, a un sencillo problema de clasificación lineal en un espacio dimensional mayor.

Para poder ser consideradas candidatas a kernels, las funciones deben cumplir tres condiciones iniciales fundamentales; deben ser continuas, simétricas y positivas. Estos son los requerimientos básicos para poder ser expresadas como un producto escalar en un espacio dimensional alto.

El espacio dimensional simulado mediante las funciones kernel se define tomando a cada característica de los datos como una dimensión. Esto convierte a las entradas en un conjunto de puntos en un espacio euclidiano n-dimensional. Es mucho más fácil establecer relaciones entre los datos expresados en esta forma.

Con el objetivo de identificar los kernels óptimos a incorporar se analizaron numerosas opciones disponibles, entre las posibilidades estudiadas destacan los siguientes kernels: Gaussiano, Fisher, String, Hubert y Gaussiano de anchura modificada. Algunos de los anteriores kernels llegaron hasta la fase de implementación y prueba. No obstante se decidió finalmente trabajar sobre una familia de kernels muy vinculada a la teoría de algoritmos ensemble infinitos.

A continuación se caracterizarán cada uno de los nuevos kernels incorporados a la solución:

2.3.3.1 Stump kernel.

Es un kernel que encarna un número infinito de decisiones stump. La decisión stump es uno de los modelos de aprendizaje base más simples que se aplican, se trata de una decisión simple, pero potente. El conjunto de decisiones stump es un simplísimo modelo de aprendizaje. Algoritmos con esta base pueden obtener usualmente un rendimiento razonable. Además suelen ser eficientes y fáciles de implementar; lo que lo convierte en un algoritmo de aprendizaje base muy popular para aprendizaje ensemble.

2.3.3.2 Perceptron kernel.

Podemos extender la solución stump para obtener el kernel Perceptron. El Perceptron es un muy importante modelo de aprendizaje relacionado con el aprendizaje en redes neuronales. Se ha demostrado que el trabajo con el kernel Perceptron equivale a construir una red neuronal con un número infinito de neuronas. Al igual que el kernel stump, el Perceptron trabaja con una dimensión infinita.

2.3.3.3 Laplacian kernel.

Aplicando sobre el kernel stump, un grupo de transformaciones matemáticas, obtenemos una expresión del kernel stump como árbol infinito, dada la combinación de varias funciones stump, utilizando la expansión de la serie de Taylor, obtenemos un equivalente en la teoría ensemble infinita, del tradicional kernel de base radial: Laplace.
expresado mediante la fórmula:

$k(x,x')=\exp(-\gamma||x-x'||_1), \gamma>0.$

2.3.3.4 Exponencial kernel.

Igualmente aplicando un grupo de transformaciones similares, a partir del kernel Perceptron obtenemos un equivalente del popular kernel exponencial, expresado como árbol de decisiones Perceptron infinitas.

expresado mediante la fórmula:

$k(x,x')=\exp(-\gamma||x-x'||_2),\gamma>0.$

2.4 Conclusiones

Muchos son los factores a tener en cuenta a la hora de elaborar una solución informática, si nuestro objetivo es desarrollar una aplicación de calidad, competitiva y que cumpla con todos los objetivos para los que fue creada.

No prestar la debida atención a detalles como la selección del lenguaje de implementación o la manera en que documentamos el desarrollo del proyecto puede dar origen a riesgos potenciales o al fracaso del proceso al no cumplir con los objetivos iniciales trazados por el equipo de desarrollo.

La selección y aplicación correcta de los materiales y los métodos de desarrollo a emplear es por tanto una cuestión de gran importancia. En este capítulo se expuso la forma en que se hizo frente a estos retos y las decisiones que a la postre se tomaron.

CAPÍTULO 3: RESULTADOS Y DISCUSIÓN

3.1 Introducción

El capítulo que sigue es de especial relevancia para la comprensión de la solución informática. Se explicarán los algoritmos fundamentales que se implementaron, el diseño de clases que se utilizó. Se expondrá además una vista global del sistema y su integración con la plataforma GRAphTOol. Se partirá del modelo conceptual para lograr la comprensión del sistema. Se discutirán también los resultados fundamentales obtenidos a través de las distintas pruebas realizadas con la aplicación.

3.2 Modelo conceptual

Dada la importancia del modelo conceptual como herramienta para el análisis orientado a objetos a continuación se expone el que identifica la solución informática. El modelo conceptual define los más relevantes conceptos en el dominio del problema, permite identificar atributos y asociaciones dentro del mismo. Un modelo conceptual es una representación del mundo real, no componentes del software, de ahí que sea de especial importancia para clientes y desarrolladores. Los modelos conceptuales se componen de conceptos, sus relaciones y atributos.

La siguiente figura muestra el modelo conceptual en el marco de una herramienta que crea modelos predictivos basados en la estructura-actividad biológica de compuestos, y emplea estos modelos para la predicción o clasificación de otros compuestos en cuanto a sus niveles de estructura-actividad biológica.

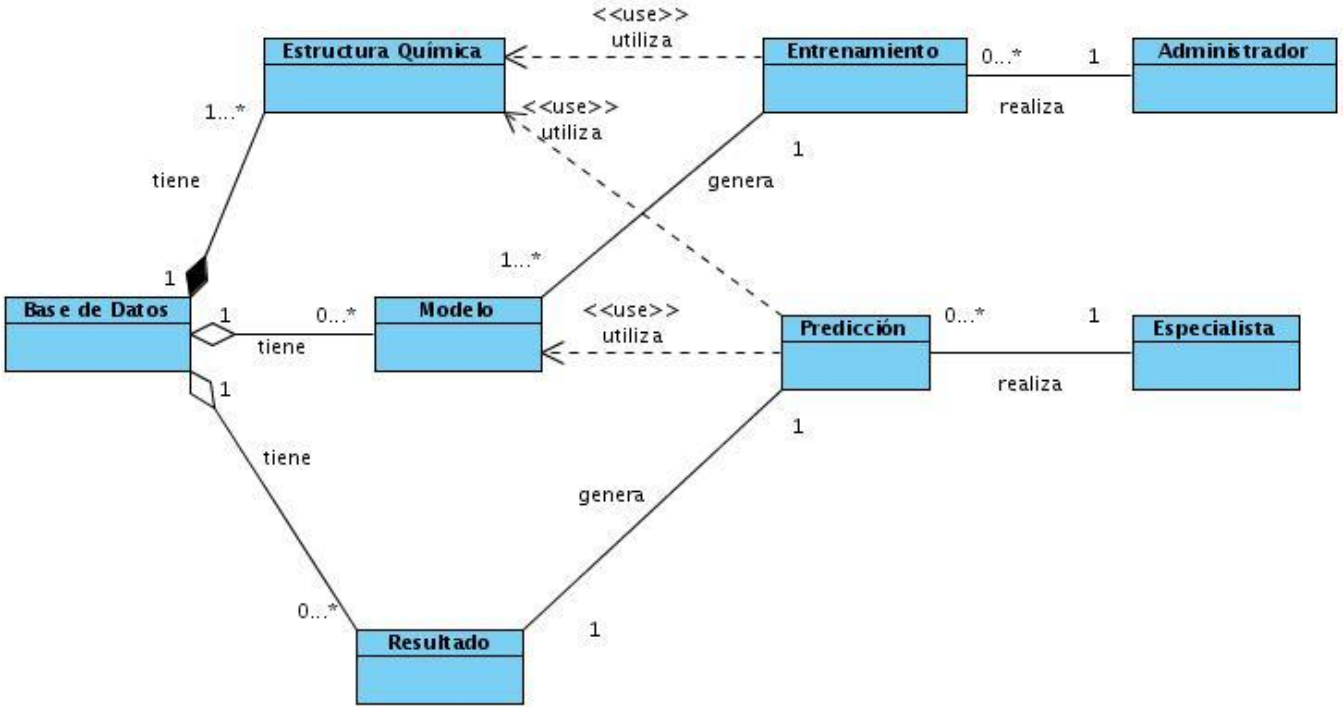


Figura 7: Modelo conceptual

3.3 Diagrama de clases

En el siguiente diagrama de clases incluimos las clases más importantes en la arquitectura de la aplicación. Destacando que la mayor parte del trabajo científico se desarrolló en modificaciones a la librería o paquete LibSVM, que en el diagrama aparece, pero que dada la complejidad de sus componentes estimamos que resultaría poco ilustrativa su representación en un diagrama de clases.

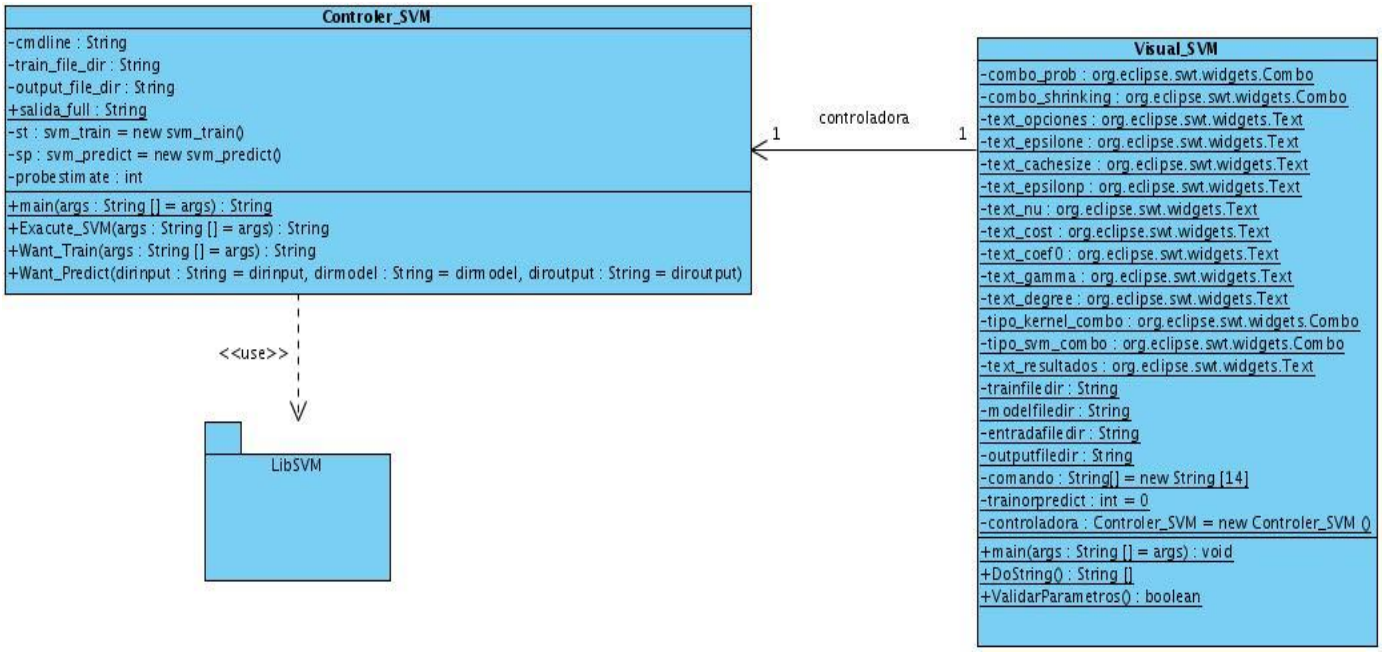


Figura 8: Diagrama de clases.

3.4 Implementación de los algoritmos

3.4.1 Revisión del pseudocódigo

El término pseudocódigo viene de la composición entre pseudo (falso) y código, lo que significa que se trata de un código que no es tal, sino una representación cercana al mismo. Un pseudocódigo es una serie de normas léxicas y gramaticales parecidas a la mayoría de los lenguajes de programación, pero sin llegar a la rigidez de sintaxis de estos ni a la fluidez del lenguaje coloquial. Se puede considerar como un lenguaje universal, pues su conversión a cualquiera de los lenguajes de programación estándar es un proceso sencillo. Esto permite, entre otras cosas codificar un programa con mayor agilidad que en cualquier lenguaje de programación, a la vez que conserva validez semántica. Resulta muy útil en el estudio algoritmos.

El pseudocódigo por tanto describe un algoritmo utilizando una mezcla de frases en lenguaje común, instrucciones de programación y palabras clave que definen las estructuras básicas. Su objetivo es permitir que el programador se centre en los aspectos lógicos de la solución a un problema.

No siendo el pseudocódigo un lenguaje formal, varían de un programador a otro, es decir, no hay una estructura semántica ni arquitectura estándar, cada cual propone su propio estilo. Por tanto se hace necesario generalmente aclarar las reglas y códigos a emplear a la hora de presentar un algoritmo en pseudocódigo.

A continuación pondremos a su disposición los pseudocódigos de algunos de los principales algoritmos empleados. Para su entendimiento se necesitan conocimientos básicos de programación. Se utilizó un lenguaje guiado por la sintaxis de java pero que no se aleja mucho del lenguaje natural, haciéndolo comprensible para cualquier programador.

Se verá ahora qué nos permitirá esta representación de los algoritmos en pseudocódigo, y se analizarán las ventajas del empleo de estas representaciones.

Ventajas de utilizar un pseudocódigo.

1. Permite representar de forma fácil operaciones repetitivas complejas.
2. Es más sencilla la tarea de pasar de pseudocódigo a un lenguaje de programación formal.
3. Si se siguen las reglas de identación se puede observar claramente los niveles en la estructura del programa.
4. En los procesos de aprendizaje de los alumnos de programación, estos están más cerca del paso siguiente (codificación en un lenguaje determinado).

Ahora se verá al algoritmo dist_1 en pseudocódigo. Este algoritmo es creado para el tratamiento de los datos a procesar empleando kernels como el Stump y el Laplace que requieren de un manejo más lineal de los datos de entrada que sus homólogos.

```
Función dist_1(svm_node[] x, svm_node[] x1)

        real sum <- 0
        entero xlen <- longitud de x
        entero x1len <- longitud de x1
        entero i <- 0
        entero j <- 0
```

```
Mientras (i < xlen-1 y j < x1len-1) hacer

        Si (x[i]índice = x1[j]índice)
        entonces
            Si ((x[i]valor - x1[j]valor)>0)
            entonces
                sum += x[i]valor - x1[j]valor

            Si no

                sum += x1[j]valor - x[i]valor

          fin Si
    incrementar i
            incrementar j


        Si no

            Si(x[i]índice > x1[j]índice)
            entonces
                Si (x1[j]valor > 0)
                entonces
                    sum += x1[j]valor
                    incrementar j

                Si no

                    sum = sum - x1[j]valor
                    incrementar j

            fin Si
            Si no
```

```
                                    Si(x[i]valor > 0)
                                    entonces
                                            sum += x[i]valor
                                            incrementar i

                                    Si no
                                     sum = sum - x[i]valor
                                            incrementar i
                                    fin Si
                            fin Si
                    fin Si
            fin Mientras

            Mientras (i < xlen-1) hacer

                    Si(x[i]valor > 0)
                    entonces
                            sum += x[i]valor
                            incrementar i

                    Si no

                            sum = sum - x[i]valor
                            incrementar i

            fin Si
fin Mientras

    Mientras (j < x1len-1) hacer

                    Si (x1[j]valor > 0)
                    entonces
                            sum += x1[j]valor
```

```
                    incrementar j

                Si no

                    sum = sum - x1[j]valor
                    incrementar j
                fin Si

        fin Mientras

Devolver sum

Fin función
```

Se verá ahora el algoritmo dot (dot product o producto escalar), perteneciente al método homónimo que se encarga al igual que el dist_1 del manejo de los datos en el procesamiento de los kernels, pero de la forma tradicional aplicando producto escalar.

```
Función dot(svm_node[] x, svm_node[] x1)

        real sum <- 0
        entero xlen <- longitud de x
        entero x1len <- longitud de x1
        entero i <- 0
        entero j <- 0
        Mientras (i < xlen y j < x1len)
        hacer
            Si (x[i]índice == x1[j]índice)
    entonces
                    sum += x[incrementar i]valor * x1[incrementar j]valor
                Si no

                    Si (x[i]índice > x1[j]índice)
```

```
                    entonces
            incrementar j
                    Si no
                        incrementar i
                Fin Si
        Fin Si
            Fin Mientras
            Devolver sum
    Fin Función
```

Se analizará ahora el algoritmo Predict Perceptron, que representa el flujo de la función k_funcion, para la predicción aplicando los kernels de procesamiento cuando es seleccionado el kernel Perceptron.

```
Función Predict Perceptron (svm_node[] x, svm_node[] x1,
                                coef0)

    real sum <- 0
    entero xlen <- longitud de x
    entero x1len <- longitud de x1
    entero i <- 0
    entero j <- 0
    Mientras (i < xlen y j < x1len) hacer

    Si (x[i]índice == x1[j]índice)
      entonces
        real d = x[incrementar i]valor - x1[incrementar j]valor
        sum += d*d

    Si no
  Si (x[i]índice > x1[j]índice)
          entonces
              sum += x1[j]valor * x1[j]valor
              incrementar j
```

```
            Si no
                        sum += x[i]valor * x[i]valor
                        incrementar i
                    fin Si

             fin Si

        fin Mientras

        Mientras(i < xlen) hacer

                    sum += x[i]valor * x[i]valor
                    incrementar i

            fin Mientras

        Mientras(j < x1len) hacer

                    sum += x1[j]valor * x1[j]valor
                    incrementar j

            fin Mientras

    Devolver sqrt(sum)+ coef0

fin Función
```

3.4.2 Análisis de la complejidad de los algoritmos.

A continuación se hace referencia a la complejidad de los algoritmos más importantes implementados en nuestra solución.

Generalmente impera el falso concepto de que mientras más complejo el algoritmo sea más robusta será la solución y más eficiente, sin embargo esto se encuentra muy lejos de la realidad. Mientras cumpla con las características y requerimientos para los que fue creado, la sencillez de un algoritmo es una condición muy deseable.

Mientras más sencillo sea un algoritmo más fácil será su verificación y su mantenimiento posterior. En caso de se trate de una solución de código abierto sobre todo la sencillez de los algoritmos representa en gran medida cuán reutilizable sean.

Adquiere relevancia en este sentido el empleo eficiente de los recursos, que suele medirse a partir de dos factores fundamentales, espacio y tiempo; es decir memoria que utiliza y tiempo que tarda en la ejecución. En función de ambos se representan el costo de la solución obtenida mediante un algoritmo determinado al problema planteado.

Junto a la complejidad de los algoritmos en si, el tiempo de ejecución de un programa estará determinado por varios factores, entre los que destacan: el tamaño de los datos de entrada suministrados, las características de las instrucciones máquina del procesador concreto que ejecute el programa y la calidad del código generado por el compilador para crear el programa objeto.

Se le llama tamaño de los datos de entrada al número de componentes sobre los que se ejecutará el algoritmo. Digamos la dimensión de un tipo de dato estructurado específico, por ejemplo una matriz o vector, o sencillamente el tamaño de una lista de datos determinada.

La no existencia de un ordenador estándar al que puedan referenciarse todas las mediciones y teniendo en cuenta como se dijo anteriormente que un importante número de factores influirá sobre el tiempo de un programa ha hecho que a nivel internacional se determine a la hora de hacer el análisis de tiempo de un algoritmo como T(n) al tiempo de ejecución de un algoritmo para una entrada de tamaño n. Esto ha permitido establecer una forma estándar independiente del resto de los factores una forma de determinar el tiempo de ejecución estimado de un algoritmo basándonos únicamente en su complejidad.

Se mide el tiempo además en función del número de operaciones elementales que el algoritmo realiza, denotadas por (OE) las operaciones elementales son aquellas que el ordenador realiza en un tiempo

acotado por una constante, es decir que no dependerá de ninguno de los datos de entrada. Por ejemplo son OE :

-Las operaciones aritméticas básicas (suma, resta, multiplicación...)

-El acceso a estructuras indexadas básicas (arreglos, listas, vectores, matrices...)

-Los saltos.

-Asignaciones a variables de tipo básico o predefinido por el compilador.

-Comparaciones lógicas.

-Devolución del resultado del algoritmo.

Conociendo esto diremos que el tiempo de ejecución de un algoritmo dado será una función T(n) que medirá el número de operaciones elementales (OE) que se realizan para una entrada de tamaño n.

Existen además un conjunto de reglas establecidas para medir el tiempo de ejecución de los algoritmos, es decir a la hora de calcular la cantidad de OE que deben realizarse.

- ✓ El tiempo de ejecución (TE) de una OE es 1.
- ✓ El tiempo de ejecución (TE) de una secuencia consecutiva de instrucciones se calcula sumando los tiempos de ejecución de cada una de las instrucciones.
- ✓ El tiempo de ejecución (TE) de una sentencia CASE se calcula sumando el tiempo de comparación de la condición lógica más el mayor de los tiempos entre los casos.
- ✓ El tiempo de ejecución (TE) de una sentencia IF C THEN S1 ELSE S2 es T = T(c) + max{T(S1),T(S2)}.
- ✓ El tiempo de ejecución (TE) de una sentencia WHILE C do S es T = T(C) + n iteraciones* (T(S) + T(C)).
- ✓ Para calcular el tiempo de ejecución (TE) del resto de las sentencias iterativas basta con expresarlas como una sentencia WHILE.
- ✓ El tiempo de ejecución (TE) de una llamada a un procedimiento o función es el tiempo que demoren en evaluarse los parámetros más el tiempo que demore el procedimiento.

Teniendo en cuenta estas reglas se determinó el orden de 3 de los más relevantes algoritmos de nuestra solución informática. La versión en pseudocódigo de dos de estos algoritmos (dist_1 y dot) se expresó anteriormente.

Orden de las funciones pertenecientes a la clase kernel:

k_function T(n) = 20 + 51n
dot T(n) = 9 + 21 n
dist_1 T(n) = 15 + 51 n

Cota Superior

Definamos la cota superior de una función:
Dada una función f , se desea estudiar aquellas funciones g que a lo sumo crecen tan deprisa como f . Al conjunto de tales funciones se le llama cota superior de f y se denota por $O(f)$. Conociendo la cota superior de un algoritmo se puede asegurar que, en ningún caso, el tiempo empleado será de un orden superior al de la cota.

Veamos algunas de las propiedades fundamentales:

1. Para cualquier función f se tiene que $f \in O(f)$.
2. Si  .
3.  .
4. Si  .
5. Si 
6. Regla de la suma: Si 
7. Regla del producto: Si 

Para simplificar, por lo general se asume que dado un algoritmo su orden de complejidad es $O(f)$ si su tiempo de ejecución para el peor caso es de orden O de f , es decir, T(n) es de orden $O(f)$.

Teniendo en cuenta las propiedades de la cota superior y los resultados obtenidos al analizar el tiempo de ejecución de los algoritmos se determinó que la cota superior de la complejidad de ambos algoritmos es del orden $O(n)$.

3.5 Resultados experimentales.

3.5.1 Descripción de los experimentos.

Se describirán a continuación los más importantes experimentos realizados sobre la solución que el presente trabajo aportó.

Para probar la efectividad de la solución se empleó una muestra de 81 Cefalosporinas determinadas por 11 descriptores cada una. La variable dependiente que empleamos en las pruebas fue el nivel de actividad biológica de cada una de las Cefalosporinas sobre la Escherichia coli (POT_EC) y la Staphylococcus aureus (POT_SA).

También se realizó una modelación y predicción de actividad biológica empleando datos reales obtenidos del ensayo NCI Yeast Anticancer Drug Screen. Esta muestra de trabajo tiene una relación de 1x1 entre moléculas activas e inactivas. Los datos están dados por 4000 compuestos definidos por 25 descriptores cada uno. Para comprobar la calidad de las predicciones se tomaron 3500 datos para crear los modelos y los 500 restantes para predecir sobre ellos.

Durante las pruebas que se realizaron sobre los 4 nuevos kernels incorporados (Stump, Perceptron, Laplace y Exponencial) se movieron los valores de las variables gamma, nu y costo dependiendo en cada caso del kernel empleado y buscando los extremos de la efectividad de los kernels.

A continuación se mostrará el flujo desde la creación de un modelo, hasta la predicción empleando el mismo, a través de imágenes de la aplicación.

Figura 9: Imagen inicial de la aplicación desktop para crear modelos.

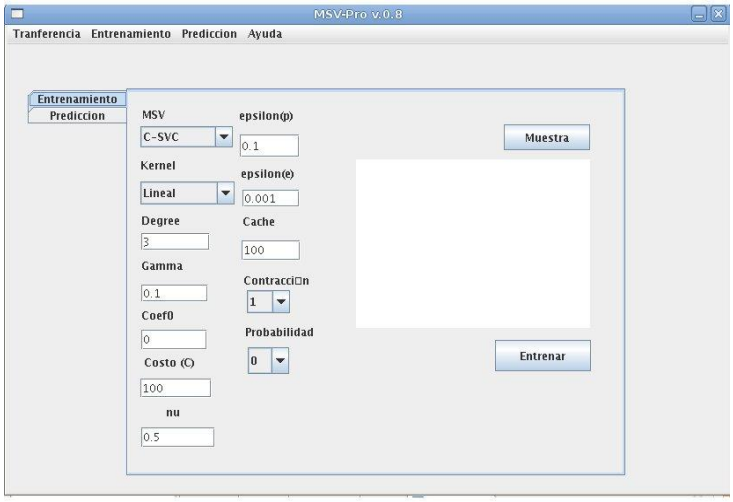


Figura 10: Selección del kernel y fijación de las variables del modelo.

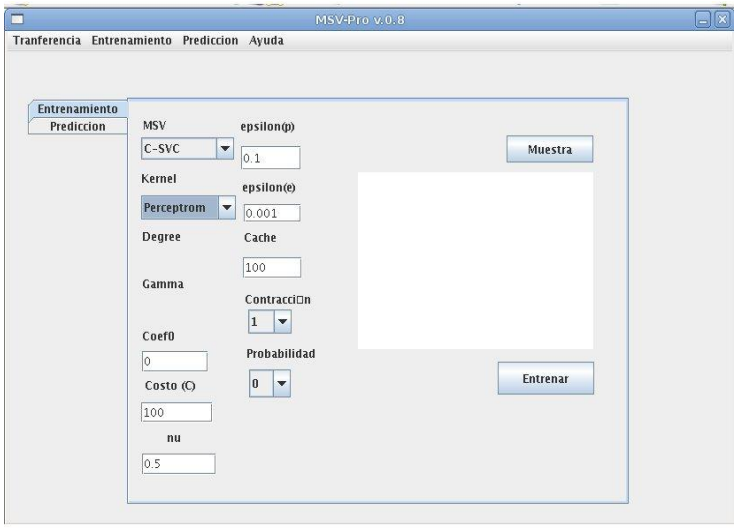


Figura 11: Cargado de los datos para crear el modelo.

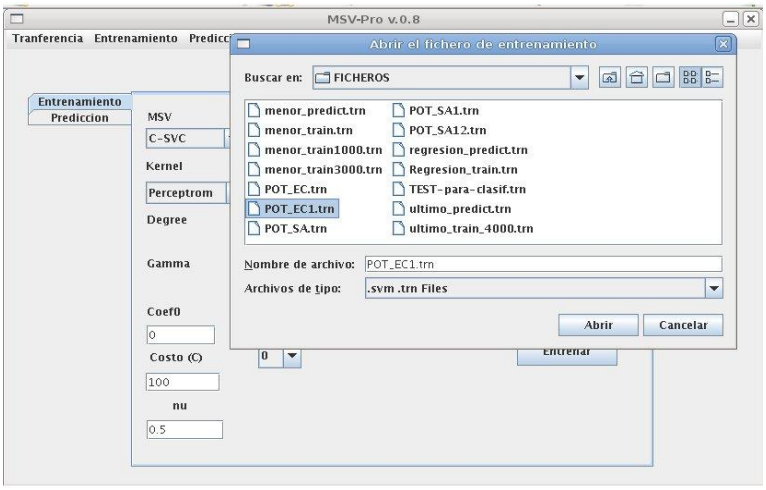


Figura 12: Creación del modelo.

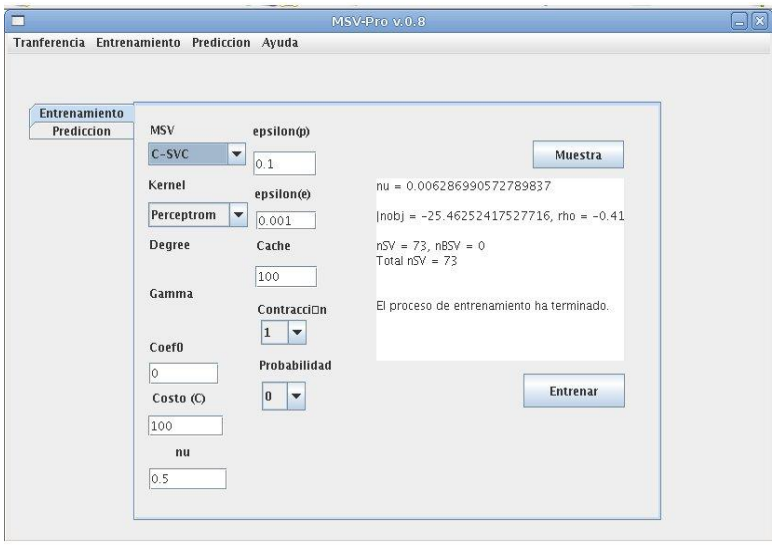


Figura 13: Imagen inicial de la aplicación desktop para realizar predicciones.

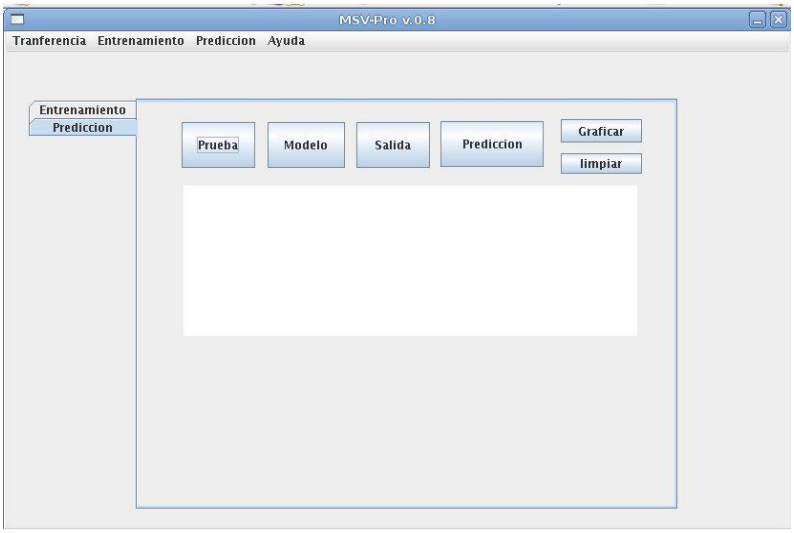


Figura 14: Cargado de los datos a clasificar o predecir.

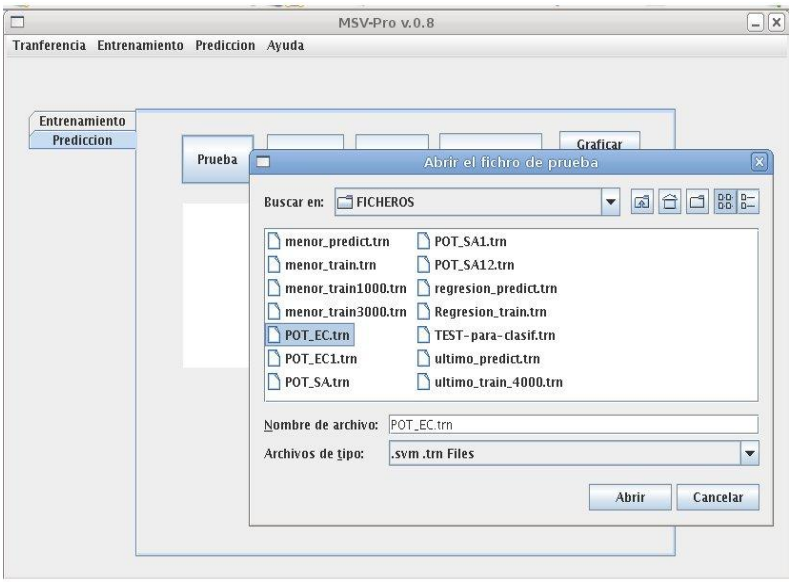


Figura 15: Cargado del modelo.

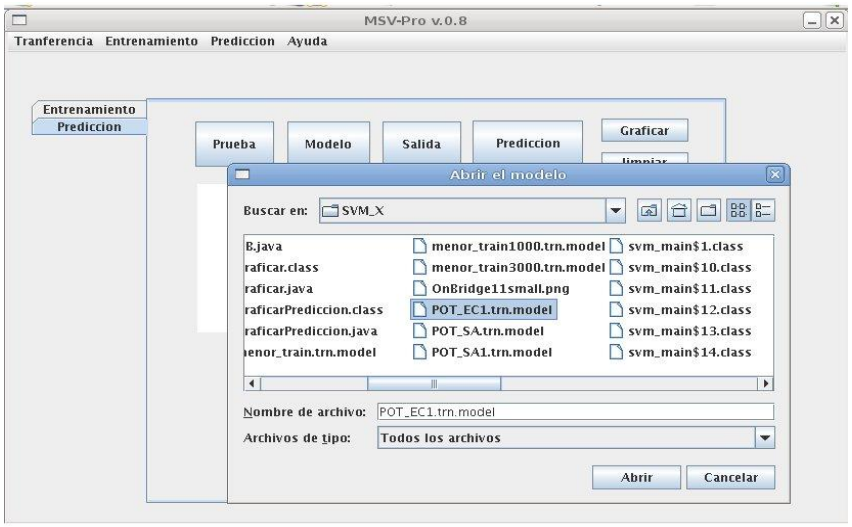


Figura 16: Selección del destino del resultado de la predicción.

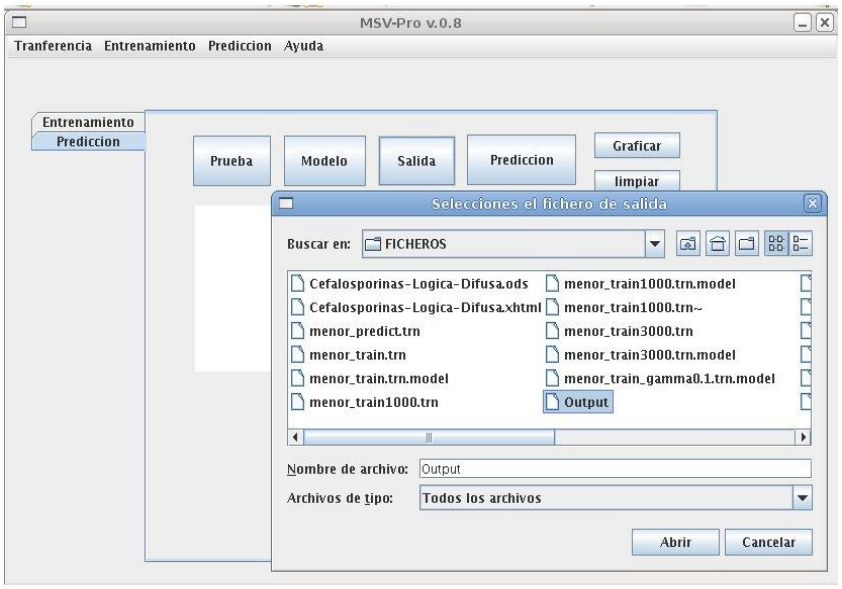
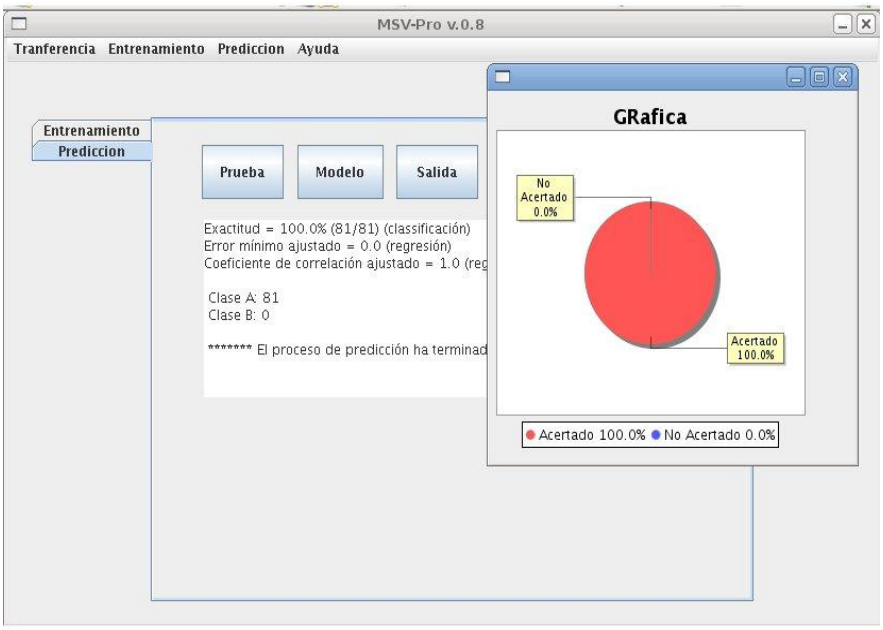


Figura 17: Predicción y evaluación de resultados.

Figura 18: Graficación de la exactitud de los resultados.



Este procedimiento se repitió varias veces modificando los kernels de procesamiento de la información y las variables de entrada.

3.6 Discusión de los resultados experimentales.

Al realizar las pruebas se obtuvieron los resultados que muestran las siguientes tablas:

		POT_EC		POT_SA	
Variables, costo=100		Kernel % exactitud		Kernel % exactitud	
ηu	gamma	Laplace	Exponencial	Laplace	Exponencial
0,5	0,1	100	100	100	100
0,1	0,1	100	100	100	100
0,5	0,5	100	100	100	100
0,1	0,5	100	100	100	100
0,01	0,1	100	100	100	100
0,01	0,5	100	100	100	100
0,01	0,99	100	100	100	100
0,01	0,01	97,53	98,76	98,07	98,07
0,5	0,01	97,53	98,76	98,07	98,07
0,99	0,01	97,53	98,76	98,07	98,07
0,8	0,01	97,53	98,76	98,07	98,07
0,9	0,01	97,53	98,76	98,07	98,07
0,95	0,01	97,53	98,76	98,07	98,07
0,88	0,01	97,53	98,76	98,07	98,07
0,1	0,9	100	100	100	100
0,5	0,9	100	100	100	100
0,3	0,9	100	100	100	100

Figura 19

Resultados obtenidos con los kernels de procesamiento Laplace y Exponencial, fijando la variable de entrada costo en 100 y haciendo oscilar los valores de nu y gamma entre 0,01 y 0,99 en ambos casos. POT_EC y POT_SA representan las pruebas hechas sobre los juegos de datos de la actividad biológica ante la Escherichia coli (POT_EC) y la Staphylococcus aureus (POT_SA), respectivamente.

		POT_EC		POT_SA	
Variables, nu=0.5		Kernel % exactitud		Kernel % exactitud	
Costo	gamma	Laplace	Exponencial	Laplace	Exponencial
10000	0,1	100	100	100	100
1000	0,1	100	100	100	100
1000	0,5	100	100	100	100
1000	0,9	100	100	100	100
10000	0,5	100	100	100	100
10000	0,9	100	100	100	100
10000	0,01	100	100	100	100
1000	0,01	100	100	100	100
10000	0,99	100	100	100	100
100	0,1	100	100	100	100
10	0,1	98,76	98,76	100	99,03
1	0,1	86,41	80,24	81,73	81,73
100	0,01	97,53	98,76	98,07	98,07
10	0,01	81,48	80,24	81,73	78,84
100	0,5	100	100	100	100
10	0,5	100	100	100	100
1	0,5	98,76	93,82	99,03	91,34
100	0,9	100	100	100	100
1	0,9	100	98,76	100	99,03

Figura 20

Resultados obtenidos con los kernels de procesamiento Laplace y Exponencial, fijando la variable de entrada un en 0.5 y haciendo oscilar los valores de costo entre 1 y 10000 y los de gamma entre 0,01 y 0,99. POT_EC y POT_SA representan las pruebas hechas sobre los juegos de datos de la actividad biológica ante la Escherichia coli (POT_EC) y la Staphylococcus aureus (POT_SA), respectivamente.

Variables		POT_EC Kernel % exactitud		POT_SA Kernel % exactitud	
ηu	costo	Perceptrom	Stump	Perceptrom	Stump
0,5	10000	100	100	100	100
0,1	1000	100	100	100	100
0,5	1000	100	100	100	100
0,1	10000	100	100	100	100
0,01	10000	100	100	100	100
0,01	1000	100	100	100	100
0,01	100	100	100	100	100
0,01	10	100	100	100	100
0,5	100	100	100	100	100
0,99	1000	100	100	100	100
0,8	100	100	100	100	100
0,9	10000	100	100	100	100
0,95	10	100	100	100	100
0,88	1	98,76	96,29	98,07	96,15
0,1	100	100	100	100	100
0,5	1	98,76	96,29	98,07	96,15
0,3	100	100	100	100	100
0,01	1	98,76	96,29	98,07	96,15
0,9	1	98,76	96,29	98,07	96,15
0,99	1	98,76	96,29	98,07	96,15

Figura 21

Resultados obtenidos con los kernels de procesamiento Perceptron y Stump, haciendo oscilar los valores de costo entre 1 y 10000 y los de nu entre 0,01 y 0,99. POT_EC y POT_SA representan las pruebas hechas sobre los juegos de datos de la actividad biológica ante la Escherichia coli (POT_EC) y la Staphylococcus aureus (POT_SA), respectivamente.

		POT_SA	
Variables, costo=100		Kernel % exactitud	
ηu	gamma	Laplace	Exponencial
0,5	0,1	92.85	71.42
0,1	0,1	92.85	71.42
0,5	0,5	71.42	71.42
0,1	0,5	71.42	71.42
0,01	0,1	92.85	71.42
0,01	0,5	71.42	71.42
0,01	0,99	71.42	71.42
0,01	0,01	92.85	71.42
0,5	0,01	92.85	71.42
0,99	0,01	92.85	71.42
0,8	0,01	92.85	71.42
0,9	0,01	92.85	71.42
0,95	0,01	92.85	71.42
0,88	0,01	92.85	71.42
0,1	0,9	71.42	71.42
0,5	0,9	71.42	71.42
0,3	0,9	71.42	71.42

Figura 22

Resultados obtenidos con los kernels de procesamiento Laplace y Exponencial, separando la muestra original de Cefalosporinas relacionadas con su actividad biológica sobre la Staphylococcus aureus, en una muestra de 90 para entrenamiento y otra muestra de 14 para realizar sobre ella la predicción. Los valores de la variable nu oscilaron entre 0.01 y 0.95; mientras que los de gamma se movieron entre 0.01 y 0.99, fijándose en 100 el valor del costo.

		POT_SA	
Variables, nu=0.5		Kernel % exactitud	
Costo	gamma	Laplace	Exponencial
10000	0,1	92,85	71,42
1000	0,1	92,85	71,42
1000	0,5	71,42	71,42
1000	0,9	71,42	71,42
10000	0,5	71,42	71,42
10000	0,9	71,42	71,42
10000	0,01	100	71,42
1000	0,01	100	71,42
10000	0,99	71,42	71,42
100	0,1	92,85	71,42
10	0,1	92,85	71,42
1	0,1	71,42	71,42
100	0,01	92,85	71,42
10	0,01	78,57	71,42
100	0,5	71,42	71,42
10	0,5	71,42	71,42
1	0,5	71,42	71,42
100	0,9	71,42	71,42
1	0,9	71,42	71,42

Figura 23

Resultados obtenidos con los kernels de procesamiento Laplace y Exponencial, separando la muestra original de Cefalosporinas relacionadas con su actividad biológica sobre la Staphylococcus aureus, en una muestra de 90 para entrenamiento y otra muestra de 14 para realizar sobre ella la predicción. Los valores de la variable costo oscilaron entre 1 y 10000; mientras que los de gamma se movieron entre 0.01 y 0.99, fijándose en 0,5 el valor de nu.

		POT_SA	
Variables		Kernel % exactitud	
ηu	costo	Perceptrom	Stump
0,5	10000	71,42	100
0,1	1000	71,42	100
0,5	1000	71,42	100
0,1	10000	71,42	100
0,01	10000	71,42	100
0,01	1000	71,42	100
0,01	100	71,42	100
0,01	10	71,42	100
0,5	100	71,42	100
0,99	1000	71,42	100
0,8	100	71,42	100
0,9	10000	71,42	100
0,95	10	71,42	100
0,88	1	71,42	92,85
0,1	100	71,42	100
0,5	1	71,42	92,85
0,3	100	71,42	100
0,01	1	71,42	92,85
0,9	1	71,42	92,85
0,99	1	71,42	92,85

Figura 24

Resultados obtenidos con los kernels de procesamiento Perceptron y Stump, separando la muestra original de Cefalosporinas relacionadas con su actividad biológica sobre la Staphylococcus aureus, en una muestra de 90 para entrenamiento y otra muestra de 14 para realizar sobre ella la predicción. Los valores de la variable costo oscilaron entre 1 y 10000; mientras que los de nu se movieron entre 0.01 y 0.99.

Variables, costo=100		NCI Yeast Anticancer Drug Screen Kernel % exactitud	
ηu	gamma	Laplace	Exponencial
0,5	0,1	83.39	87.00
0,1	0,1	83.39	87.00
0,5	0,5	85.60	86.80
0,1	0,5	85.60	86.80
0,01	0,1	83.39	87.00
0,01	0,5	85.60	86.80
0,01	0,99	87.00	86.40
0,01	0,01	84.80	88.20
0,5	0,01	84.80	88.20
0,99	0,01	84.80	88.20
0,8	0,01	84.80	88.20
0,9	0,01	84.80	88.20
0,95	0,01	84.80	88.20
0,88	0,01	84.80	88.20
0,1	0,9	87.20	86.60
0,5	0,9	87.20	86.60
0,3	0,9	87.20	86.60
0,1	0,99	87.00	86.40
0,5	0,001	87.80	88.80
0,99	0,95	87.20	86.60

Figura 25

Resultados obtenidos con los kernels de procesamiento Laplace y Exponencial, separando la muestra original de datos, en una muestra de 3500 para entrenamiento y otra muestra de 500 para realizar sobre ella la predicción. Los valores de la variable costo se fijaron en 100, mientras los de gamma oscilaron entre 0,01 y 0.99; así mismo los de nu se movieron entre 0.01 y 0.99.

		NCI Yeast Anticancer Drug Screen	
Variables, nu=0.5		Kernel % exactitud	
Costo	gamma	Laplace	Exponencial
10000	0,1	83.60	86,80
1000	0,1	83.60	86,80
1000	0,5	85.60	86,80
1000	0,9	87.20	86,60
10000	0,5	85.60	86,80
10000	0,9	87.20	86,60
10000	0,01	82.19	86,60
1000	0,01	82.00	87,00
10000	0,99	87.00	86,40
100	0,1	83.39	87,00
10	0,1	86,20	88,20
1	0,1	87,80	88,80
100	0,01	84,80	88,20
10	0,01	87,80	88,80
100	0,5	85,60	86,80
10	0,5	85,80	86,60
1	0,5	88,40	88,00
100	0,99	87,00	86,40
1	0,99	88,00	88,40
1000	0,99	87,00	86,40
1	0,01	87,80	88,80
1	0,9	88,00	88,40

Figura 26

Resultados obtenidos con los kernels de procesamiento Laplace y Exponencial, separando la muestra original de datos, en una muestra de 3500 para entrenamiento y otra muestra de 500 para realizar sobre ella la predicción. Los valores de la variable costo se movieron entre 1 y 10000, mientras los de gamma oscilaron entre 0,01 y 0.99; así mismo los de nu se fijaron en 0,5.

		NCI Yeast Anticancer Drug Screen	
Variables		Kernel % exactitud	
ηu	costo	Perceptrom	Stump
0,5	10000	86.80	82,60
0,1	1000	86.80	82,80
0,5	1000	86.80	82,80
0,1	10000	86.80	82,60
0,01	10000	86.80	82,60
0,01	1000	86.80	82,80
0,01	100	86.80	82,60
0,01	10	87,00	83,00
0,5	100	86.80	82,60
0,99	1000	86.80	82,80
0,8	100	86.80	82,60
0,9	10000	86.80	82,60
0,95	10	87,00	83,00
0,88	1	88,20	85,39
0,1	100	86.80	82,60
0,5	1	88,20	85,39
0,3	100	86.80	82,60
0,01	1	88,20	85,39
0,9	1	88,20	85,39
0,99	1	88,20	85,39

Figura 27

Resultados obtenidos con los kernels de procesamiento Perceptron y Stump, separando la muestra original de datos, en una muestra de 3500 para entrenamiento y otra muestra de 500 para realizar sobre ella la predicción. Los valores de la variable costo se movieron entre 1 y 10000, mientras los de nu oscilaron entre 0.01 y 0.99.

Como se puede apreciar en las tablas los resultados son muy satisfactorios aunque se trate de un juego de datos bastante pequeño, lo que fortalece las predicciones, no se puede perder de vista que la fortaleza de las Máquinas de Soporte Vectorial se encuentra en su capacidad de procesar grandes cantidades de datos. Por estos motivos el alcanzar resultados superiores al 78 % en la exactitud de todas las predicciones y en la mayor parte de estas lograr un 100% se trata de resultados muy alentadores.

Algo muy importante es además la estabilidad de todos los kernels, que al realizarse importantes variaciones en los parámetros de entrada, mientras se mantiene el mismo juego de datos, el sistema es capaz de lograr resultados aceptables de manera constante.

Se han realizado además pruebas sobre juegos de datos de hasta 4000 compuestos definidos por más de una veintena de descriptores y, con una correcta selección de variables, se han alcanzado predicciones entre un 96 y 99% de exactitud.

3.7 Conclusiones

En este tercer capítulo se arribó a un modelo conceptual que permitió la clara comprensión de los procesos que se llevan a cabo en la solución informática planteada. Se analizaron los algoritmos diseñados desde varios puntos de vista. Se llevó a cabo un análisis de la complejidad de los algoritmos y se comprobó su eficiencia.

Se discutieron los resultados de los experimentos arribando a conclusiones positivas, al sostener predicciones sobre el 70% de exactitud en todos los casos. Se llevaron a cabo pruebas sobre datos reales y se comprobó la no existencia de falsos positivos durante las predicciones. Esto permitió comprobar la calidad de los algoritmos desarrollados.

CONCLUSIONES GENERALES

Tras el desarrollo de este trabajo se puede arribar a un grupo de conclusiones consideradas como relevantes por los autores.

Durante un período de 8 meses entre octubre del 2007 y junio del 2008 fue desarrollado el presente trabajo. En los primeros meses se llevó a cabo una importante revisión de la bibliografía disponible en pos de definir con claridad la estrategia a seguir en el desarrollo de la solución. Fundamentalmente obtenida desde Internet, debido a lo novedoso del tema tratado, la bibliografía consultada brindó una panorámica clara de las soluciones existentes hasta el momento y de los aspectos que de estas soluciones se debían conservar o superar. Se identificó que la vía más factible de incrementar la calidad de las predicciones de las herramientas disponibles era la inclusión de nuevos kernels de procesamiento.

Finalmente se lograron cumplir los objetivos que se plantearon al inicio, veámoslo de forma desglosada.

- Se implementaron cuatro nuevos kernels de procesamiento (Stump, Perceptron, Laplace y Exponencial) que fueron incorporados a los 4 ya disponibles.
- Se desarrolló una aplicación informática utilizando los materiales y métodos seleccionados para comprobar los nuevos kernels.
- Se logró comprobar la eficacia de los nuevos kernels de procesamiento a través de pruebas que se realizaron con la ayuda de la aplicación implementada. Arrojando resultados superiores a las soluciones previamente disponibles en Cuba.

Como se puede apreciar los objetivos iniciales del trabajo se cumplieron cabalmente. Logrando hacer un aporte al desarrollo de la química farmacéutica y la inteligencia artificial en Cuba.

RECOMENDACIONES

Incrementar el número de kernels disponibles con vistas a diversificar el rango de acción la solución.

Profundizar a mayor escala desde el punto de vista matemático en el tema con el objetivo de optimizar y generar un mayor grado de vinculación entre el programador y la aplicación.

Se sugiere que para investigaciones futuras en pos de ampliar el número de kernels de procesamiento, se incorpore un matemático al equipo de desarrollo, lo que agilizaría el proceso ostensiblemente.

REFERENCIAS BIBLIOGRÁFICAS

1. **ESCALONA, J. C.; CARRASCO, R. Introducción al diseño de Fármacos [Consultado el: 12 de abril de 2008]. Disponible en: http://www.fq.uh.cu/investig/lqct/imagenes2/diseno.pdf.**

2. **GUTIÉRREZ, H. Modelización molecular de los receptores de adenosina y sus ligandos en el marco de diseño de fármacos asistido por ordenador Doctorado, Ciencias Experimentales. Universidad de Pompeu Fabra, 2004. [Consultado el: 19 de enero de 2008]**

3. **RESENDIZ, J. A. Las Máquinas de Soporte Vectorial para identificación en Línea. Maestría, Control Automático. Instituto Politécnico Nacional, 2006. [Consultado el: 9 de febrero de 2008]**

4. **CHEN, N.; YANG, J., et al. Support Vector Machine in Chemistry. World Scientific, 2004.[Consultado el: 12 de abril de 2008]**

5. **YANG, H. Margin Variations in Support Vector Regression for the Stock Market Prediction. Maestría, Ingeniería y Ciencias de la Computación. Universidad de Hong Kong, 2003. [Consultado el: 19 de diciembre de 2007]**

6. **R, B.; M, T., Drug design by machine learning: support vector machines for pharmaceutical data analysis [Consultado el: 3 de mayo de 2008]. Disponible en: http://citeseer.ist.psu.edu/528480.html.**

7. **VV, Z.; KV, B., Drug discovery using support vector machines. The case studies of drug-likeness, agrochemical-likeness, and enzyme inhibition predictions [Consultado el: 3 de febrero de 2008]. Disponible en: http://pubs.acs.org/cgi-bin/abstract.cgi/jcisd8/2003/43/i06/abs/ci0340916.html.**

8. **HX, L.; CX, X., Quantitative prediction of logk of peptides in high-performance liquid**

chromatography based on molecular descriptors by using the heuristic method and support vector machine

[Consultado el: 12 de enero de 2008]. Disponible en: http://pubs.acs.org/cgi-bin/abstract.cgi/jcisd8/2004/44/i06/abs/ci049891a.html

9. **CX, X.; RS, Z., Study of the quantitative structure-mobility relationship of carboxylic acids in**

capillary electrophoresis based on support vector machines. [Consultado el: 10 de febrero de 2008].

Disponible en: http://pubs.acs.org/cgi-bin/abstract.cgi/jcisd8/2004/44/i03/abs/ci034280o.html.

10. **XUE, C. X.; ZHANG, R. S., An Accurate QSPR Study of O-H Bond Dissociation Energy in Substituted Phenols Based on Support Vector Machines**

[Consultado el: 22 de enero de 2008]. Disponible en: http://pubs.acs.org/cgi-bin/abstract.cgi/jcisd8/2004/44/i02/abs/ci034248u.html.

11. **S, W.; C, X., Study on the quantitative relationship between the structures and electrophoretic**

mobilities of flavonoids in micellar electrokinetic capillary chromatography [Consultado el: 22 de diciembre de

2007]. Disponible en:

http://www.cababstractsplus.org/google/abstract.asp?AcNo=20043056516.

12. **CY, Z.; RS, Z., QSAR study of natural, synthetic and environmental endocrine disrupting compounds for binding to the androgen receptor [Consultado el: 10 de febrero de 2008].**

Disponible en:

http://www.informaworld.com/smpp/content~content=a727214684~db=all.

13. **S, Y.; M, X., Quantitative structure-property relationship studies on electrochemical degradation of substituted phenols using a support vector machine. [Consultado el: 23 de abril de 2008].**

Disponible en:

http://www.informaworld.com/smpp/content~content=a757824679~db=all~jumptype=rss.

14. **W, M.; F, L., Quantitative structure-property relationships for pesticides in biopartitioning micellar chromatography [Consultado el: 23 de enero de 2008]. Disponible en: http://www.aapspharmaceutica.com/search/view.asp?ID=74403.**

15. **F, L.; W, M., Quantitative structure-activity relationship models for prediction of sensory irritants (logRD50) of volatile organic chemicals. [Consultado el: 23 de enero de 2008]. Disponible en:**

http://www.ncbi.nlm.nih.gov/sites/entrez?cmd=Retrieve&db=PubMed&list_uids=16307788&dopt=Abstract.

16. **IV LATIN AMERICAN CONGRESS ON BIOMEDICAL ENGINEERING [Consultado el: 27 de abril de 2008]. Disponible en: http://www.claib2007.eventos.usb.ve/dmdocuments/Resumenes%20CLAIB%202007.pdf**

BIBLIOGRAFÍA

1. XUE, C. X.; ZHANG, R. S., et al. An Accurate QSPR Study of O-H Bond Dissociation Energy in Substituted Phenols Based on Support Vector Machines . Disponible en: http://pubs.acs.org/cgi-bin/abstract.cgi/jcisd8/2004/44/i02/abs/ci034248u.html.

2. CY, Z.; HX, Z., et al. Application of support vector machine (SVM) for prediction toxic activity of different data sets. Disponible en: http://www.sciencedirect.com/science?ob=ArticleURL&_udi=B6TCN-4H877RX-1&_user=2342189&_coverDate=01%2F16%2F2006&_rdoc=1&_fmt=&_orig=search&_sort=d&view=c&_acct=C000056883&_version=1&_urlVersion=0&_userid=2342189&md5=30f145309eb9dd80ed8adba10558bd03.

3. WEI, C. BAYESIAN APPROACH TO SUPPORT VECTOR MACHINES. Doctorado, Ingeniería Mecánica. Universidad Nacional de Singapore, 2003.

4. SALUD, O. M. D. L. Cáncer. Disponible en:

http://www.who.int/mediacentre/factsheets/fs297/es/print.html.

5. F, L.; R, Z., et al. Classification of the carcinogenicity of N-nitroso compounds based on support vector machines and linear discriminant analysis. Disponible en: http://pubs.acs.org/cgi-bin/abstract.cgi/crtoec/2005/18/i02/abs/tx049782q.html.

6. XJ, Y.; A, P., et al. Comparative study of QSAR/QSPR correlations using support vector machines,
radial basis function neural networks, and multiple linear regression Paris: Disponible en: http://cat.inist.fr/?aModele=afficheN&cpsidt=15967815.

7. R, B.; M, T., et al. Drug design by machine learning: support vector machines for pharmaceutical
data analysis. Disponible en:
http://citeseer.ist.psu.edu/528480.html.

8. VV, Z.; KV, B., et al. Drug discovery using support vector machines. The case studies of drug-
likeness, agrochemical-likeness, and enzyme inhibition predictions. Disponible en: http://pubs.acs.org/cgi-bin/abstract.cgi/jcisd8/2003/43/i06/abs/ci0340916.html.

9. LÓPEZ, C. A. El futuro de los medicamentos Disponible en:

http://www.ranf.com/pdf/arti/futuro.pdf.

10. JACOBSON, I.; BOOCH, G., et al. El Proceso Unificado de Desarrollo de Software. La Habana:
Felix Varela, 2004. vol. I, 105-179 p.

11. VALERO, A. T. Extracción de Información con Algoritmos de Clasificación. Instituto Nacional de
Astrofísica, Óptica y Electrónica., 2005.

12. DESHPANDE, M.; KURAMOCHI, M., et al. Frequent Sub-Structure-Based Approaches for Classifying Chemical Compounds. 2-13. Disponible en:

http://doi.ieeecomputersociety.org/10.1109/ICDM.2003.1250900.

13. MAK, G. THE IMPLEMENTATION OF SUPPORT VECTOR MACHINES USING THE

SEQUENTIAL MINIMAL OPTIMIZATION ALGORITHM. Master en Ciencias, School of Computer Science McGill University, Montreal, Canada, 2000.

14. PRESMAN, R. S. Ingenieria de Software. Un Enfoque Práctico. . La Habana: Felix Varela, 2005.
vol. 2, 165-323 p.

15. ESCALONA, J. C.; CARRASCO, R., et al. Introducción al diseño de Fármacos. 17. Disponible en: http://www.fq.uh.cu/investig/lqct/imagenes2/diseno.pdf.

16. BETANCOURT, G. A. Las Máquinas de Soporte Vectorial 67-72. Disponible en: http://www.fet2005.cs.buap.mx/DM/AUDIOVISUAL-PRESENTACIONES/svm-bueno-util.pdf.

17. RESENDIZ, J. A. Las Máquinas de Soporte Vectorial para identificación en Linea. Maestria, Control Automático. Instituto Politecnico Nacional, 2006.

18. SMOLA, A. J. Learning With Kernels. 2002, ISBN 0-262-19475-9.

19. CHANG, C.-C. y LIN, C.-J. LIBSVM: a Library for Support Vector Machines. 1-22. Disponible en: http://www.csie.ntu.edu.tw/~cjlin/libsvm/.

20. YANG, H. Margin Variations in Support Vector Regression for the Stock Market Prediction. Maestría, Ingeniería y Ciencias de la Computación. Universidad de Hong Kong, 2003.

21. GUTIÉRREZ, H. Modelización molecular de los receptores de adenosina y sus ligandos en el marco de diseño de fármacos asistido por ordenador Doctorado, Ciencias Experimentales. Universidad de Pompeu Fabra, 2004.

22. ESPINOSA, G. Modelos QSPR/QSAR/QSTR basados en sistemas neuronales cognitivos Doctorado,
URV, 2002.

23. GARDNER, A. B. A Novelty Detection Approach to Seizure Analysis from Intracranial EEG. Doctorado, Instituto Tecnológico de Georgia, 2004.

24. HSU, C.-W.; CHANG, C.-C., et al. A Practical Guide to Support Vector Classification . 1-12. Disponible en:
http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf.

25. NEELANJAN MUKHERJEE, S. M. Predicting Signal Peptides with Support Vector Machines . 3-6. Disponible en:
http://www.springerlink.com/index/MYCRNXQQFKFGVY1R.pdf.

26. F, L.; W, M., et al. Prediction of pK(a) for neutral and basic drugs based on radial basis function

Neural networks and the heuristic method. Disponible en:

http://www.ncbi.nlm.nih.gov/sites/entrez?cmd=Retrieve&db=PubMed&list_uids=16132357&dopt=
Abstract.

27. HX, L.; XJ, Y., et al. Prediction of the tissue/blood partition coefficients of organic compounds
based on the molecular structure using least-squares support vector machines. Disponible en:

http://www.ncbi.nlm.nih.gov/sites/entrez?cmd=Retrieve&db=PubMed&list_uids=16317501&dopt=
Citation.

28. DEVROYE, L.; GYORFI, L., et al. A Probabilistic Theory of Pattarn Recognition. Editado por: Springer. 1996, ISBN 0-387-94618-7.

29. CX, X.; RS, Z., et al. QSAR models for the prediction of binding affinities to human serum albumin
using the heuristic method and a support vector machine.

Disponible en: http://pubs.acs.org/cgi-bin/abstract.cgi/jcisd8/2004/44/i05/abs/ci049820b.html.

30. CY, Z.; RS, Z., et al. QSAR study of natural, synthetic and environmental endocrine disrupting
compounds for binding to the androgen receptor. Disponible

en: http://www.informaworld.com/smpp/content~content=a727214684~db=all.

31. HX, L.; CX, X., et al. Quantitative prediction of logk of peptides in high-performance liquid chromatography based on molecular descriptors by using the heuristic method and support vector machine. Disponible en: http://pubs.acs.org/cgi-bin/abstract.cgi/jcisd8/2004/44/i06/abs/ci049891a.html.

32. F, L.; W, M., et al. Quantitative structure-activity relationship models for prediction of sensory irritants (logRD50) of volatile organic chemicals. Disponible en:

http://www.ncbi.nlm.nih.gov/sites/entrez?cmd=Retrieve&db=PubMed&list_uids=16307788&dopt=Abstract.

33. S, Y.; M, X., et al. Quantitative structure-property relationship studies on electrochemical degradation of substituted phenols using a support vector machine. . Disponible en:

http://www.informaworld.com/smpp/content~content=a757824679~db=all~jumptype=rss.

34. Quantitative structure-property relationship studies on electrochemical degradation of substituted phenols using a support vector machine. Disponible en:

http://www.informaworld.com/smpp/content~content=a757824679~db=all~jumptype=rss.

35. W, M.; F, L., et al. Quantitative structure-property relationships for pesticides in biopartitioning micellar chromatography. Disponible en:

http://lib.bioinfo.pl/pmid:16490199.

36. Quantitative structure-property relationships for pesticides in biopartitioning micellar chromatography. Disponible en: http://www.aapspharmaceutica.com/search/view.asp?ID=74403.

37. RÍO, B. M. D. y MOLINA, A. S. Redes Neuronales y Sistemas Difusos. Disponible en: http://bibliodoc.uci.cu/pdf/reg00050.pdf.

38. SCHOLKOPF, B. Statistical Learning and Kernel Methods 29. Disponible en: http://www.iro.umontreal.ca/~pift6266/A06/refs/scholkopf-kernel-tutorial-2000.pdf.

39. CX, X.; RS, Z., et al. Study of the quantitative structure-mobility relationship of carboxylic acids in capillary electrophoresis based on support vector machines. Disponible en: http://pubs.acs.org/cgi-bin/abstract.cgi/jcisd8/2004/44/i03/abs/ci034280o.html.

40. S, W.; C, X., et al. Study on the quantitative relationship between the structures and electrophoretic mobilities of flavonoids in micellar electrokinetic capillary chromatography . Disponible en: http://www.cababstractsplus.org/google/abstract.asp?AcNo=20043056516.

41. CHEN, N.; YANG, J., et al. Support Vector Machine in Chemistry. World Scientific, 2004.

42. GUNN, S. R. Support Vector Machines for Classification and Regression. 1998.

43. FANG, J. Support Vector Machines in HTS Data Mining: Type I MetAPs Inhibition Study . Disponible en:

http://www.ncbi.nlm.nih.gov/sites/entrez?cmd=Retrieve&db=PubMed&list_uids=16418315&dopt=Abstract.

44. FUMERA, G. y ROLI, F. Support Vector Machines with Embedded Reject Option. 2002, 3-6 p. Disponible en: http://citeseer.ist.psu.edu/552636.html.

45. WELLING, M. Support Vector Regression 1-3.

46. Kernel Machines Blackboard. Disponible en: http://agbs.kyb.tuebingen.mpg.de

47. Hsuan-Tien Lin. Site. Disponible en: http://www.work.caltech.edu/~htlin/program/libsvm/

48. Sitio oficial de la LIBSVM. Disponible en: http://www.csie.ntu.edu.tw/~cjlin/libsvm/

49. (Vapnik, Statistical Learning Theory, 1998). The SVM algorithm can also be extended to cope with noise in the training set and with multiple classes (Cristianini and Shawe-Taylor, An Introduction to Support Vector Machines, 2000).

Bibliografía

ANEXOS

Anexo 1

SUBSISTEMA LibSVM

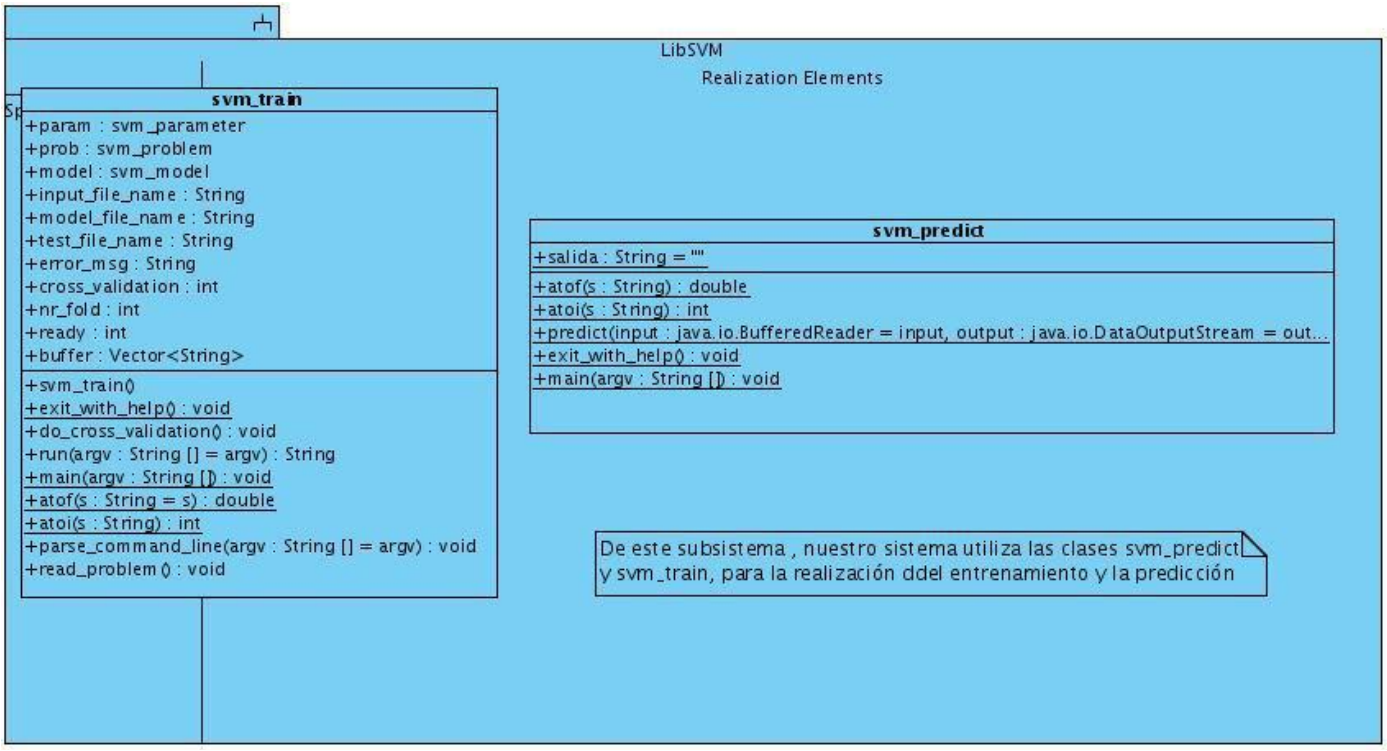


Figura 28

Anexo 2

Descripción de las Clases

Nombre: Visual_SVM	
Tipo de clase: Vista	
Modificador: -	
Padre: -	
Implementa: Serializable	
Principales Responsabilidad	
Nombre	**Tipo**
main	public static void
DoString	public static String
ValidarParametros	public static boolean
Descripción:	
Visual_SVM es la clase de interfaz visual, de donde se van a entrar los parámetros para la realización del entrenamiento y se carga un fichero con los datos para dicha acción. También soporta esta interfaz la entrada de ficheros y modelos para la realización de la predicción.	

Nombre: Controler_SVM	
Tipo de clase: Entidad	
Modificador: -	
Padre:-	
Implementa: Serializable	
Principales Responsabilidad	
Nombre	**Tipo**
Execute_SVM	public void
Want_Train	public String
want_Predict	public String
Descripción:	
La clase controladora Controler_SVM es la que manda a ejecutar las acciones para la realización del entrenamiento y de la predicción.	

GLOSARIO DE TÉRMINOS

Actividad Biológica: Actividad que caracteriza el comportamiento biológico en compuestos químicos.

Administrador: Persona que se encarga de generar los nuevos modelos y realizar el entrenamiento de los mismos.

Bioinformática: Es la aplicación de métodos informáticos sobre sistemas de cómputo y tratamiento de la información para el análisis de datos experimentales (de nivel molecular, principalmente) de sistemas biológicos, así como la simulación de los mismos.

Compuestos orgánicos: Son sustancias químicas basadas en cadenas de carbono e hidrógeno. En muchos casos contienen oxígeno, y también nitrógeno, azufre, fósforo, boro y halógenos.

CQF: Centro de Química Farmacéutica (CQF) es una institución para el desarrollo de investigaciones científico-tecnológicas dirigidas hacia la obtención de sustancias bioactivas para uso humano.

CASE: Computer Aided Software Engineering (Herramientas de ingeniería de software asistida por computadora).

Descriptor: Número que caracteriza estructuralmente la molécula.

Descriptores topológicos: Número que se calcula a partir de algoritmos que se aplican a la matriz de adyacencia o de conectividad del grafo molecular desprovisto de hidrógeno.

Fragmento: Una pequeña parte de la molécula a la cual se llega a través de un método que es el encargado de fragmentarla.

Entrenamiento: Acción que se realiza para el aprendizaje de las neuronas.

Especialista: Cliente que utilizará el sistema.

Estadística: Es una rama de la matemática que se refiere a la recolección, estudio e interpretación de los datos obtenidos en un estudio.

Metodología: Define quién hace qué, cómo y cuando.

Modelo: Representación abstracta de la realidad.

Molécula: La partícula más pequeña de una sustancia, que mantiene las propiedades químicas específicas de esa sustancia, formada por un conjunto de átomos ligados por enlaces covalentes.

Multiplataforma: Término usado para referirse a los programas, sistemas operativos, lenguajes de programación u otra clase de software, que puedan funcionar en diversas plataformas.

Predicción: Acción que el sistema realiza para emitir un resultado.

Plataforma: Es el principio, ya sea de hardware o software, sobre el cual un programa puede ejecutarse.

Plug-in: Es una aplicación informática que interactúa con otra aplicación para aportarle una función o utilidad específica.

Software: Término genérico que designa al conjunto de programas que posibilitan realizar una tarea específica en un ordenador.

Open Source: Cualidad de algunos software de incluir el código fuente en la distribución del programa. En general se usa para referirse al software libre.