TESIS DEFENDIDA POR

Nubia Martínez Navarro

Y APROBADA POR EL SIGUIENTE COMITÉ

Dr. Hugo Homero Hidalgo Silva
Director del Comité

Dr. Carlos Alberto Brizuela Rodríguez
Miembro del Comité

Dr. Andrei Tchernykh Graboskaya
Miembro del Comité

Dr. Axayácatl Rocha Olivares
Miembro del Comité

Dr. Alexei Fedorovish Licea Navarro
Miembro del Comité

Dr. Pedro Gilberto López Mariscal
Coordinador del programa de posgrado en Ciencias de la Computación

Dr. David Hilario Covarrubias Rosales
Director de Estudios de Posgrado

23 de febrero de 2009

CENTRO DE INVESTIGACIÓN CIENTÍFICA Y DE EDUCACIÓN SUPERIOR DE ENSENADA

PROGRAMA DE POSGRADO EN CIENCIAS
EN CIENCIAS DE LA COMPUTACIÓN

ESTUDIO DEL NÚCLEO P-SPECTRUM EN MÁQUINAS DE VECTORES DE SOPORTE APLICADA A LA PREDICCIÓN DE INTERACCIÓN PROTEÍNA-PROTEÍNA

TESIS

que para cubrir parcialmente los requisitos necesarios para obtener el grado de MAESTRO EN CIENCIAS

Presenta:

NUBIA MARTÍNEZ NAVARRO

Ensenada, Baja California, México, febrero de 2009.

RESUMEN de la tesis de **Nubia Martínez Navarro,** presentada como requisito parcial para la obtención del grado de MAESTRO EN CIENCIAS en Ciencias de la Computación. Ensenada, Baja California. Febrero de 2009.

ESTUDIO DEL NÚCLEO P-SPECTRUM EN MÁQUINAS DE VECTORES DE SOPORTE APLICADA A LA PREDICCIÓN DE INTERACCIÓN PROTEÍNA-PROTEÍNA

Resumen aprobado por:

_____
Dr. Hugo Homero Hidalgo Silva
Director de Tesis

Las proteínas son macromoléculas que llevan a cabo la mayor parte de las funciones de la célula. El poder determinar si un par de proteínas pueden interactuar, es decir, relacionarse para llevar a cabo alguna función, es uno de los problemas más importantes en el estudio de las proteínas. El problema es atacado desde el punto de vista biológico experimental y computacional. Dentro de este último enfoque existen varias propuestas que difieren en función del modelo usado para representar el problema. Un modelo ampliamente adoptado es considerar al problema como uno de aprendizaje de máquina. Bajo este modelo se utiliza la información acerca de interacciones conocidas para enseñar a una máquina de aprendizaje y, basado en ella, se predice la interacción o no para una pareja de proteínas cuya interacción es desconocida. Uno de los métodos de aprendizaje de máquina más exitosos son los denominados de núcleo o kernel, que representan una colección de datos complejos usando una función núcleo que define la similitud entre cualquier par de datos u objetos dados. La máquina de vectores de soporte (SVM), que es un algoritmo de clasificación supervisado, es el método núcleo más popular.

Un enfoque empleado para abordar el problema de clasificación basado en SVM es el núcleo de cadenas conocido como *p*-spectrum. Sin embargo, en la literatura no queda claro cuál es el valor apropiado de *p* cuando se trabaja con este núcleo bajo el esquema de núcleos de pares. El problema que abordamos consiste en determinar experimentalmente el efecto que tiene el valor de *p* sobre la exactitud para predecir la interacción entre un par de proteínas usando solamente la estructura primaria de las mismas.

Para nuestro conjunto de datos fueron seleccionadas 400 proteínas: 100 pares con interacción conocida y 100 pares sin interacción conocida, los que se utilizaron como ejemplos positivos y negativos, respectivamente, para fines de entrenamiento y prueba. Se calcularon las matrices para el núcleo de cadena *p*-spectrum con distintos valores de *p*. Posteriormente, se realizó la fase de

entrenamiento y prueba. Se realizaron tres experimentos con diferentes proporciones para los datos de entrenamiento y prueba: 50-50, 30-70 y 70-30, respectivamente. Los resultados indican que con valores de p = 4, 5, 6, 7 y 8, no se provee suficiente información a la SVM para realizar la predicción adecuadamente. Mientras que p = 2 no está proporcionando información relevante para realizar una correcta discriminación. En términos generales, podríamos decir que p = 3 aporta los resultados más confiables, aunque no satisfactorios (56.67% para la proporción 50-50, 54% para 30-70 y 61% para 70-30). El que la exactitud de los resultados no sea alta puede deberse a factores como: i) que el núcleo *p*-spectrum considera secciones contiguas para las comparaciones y ii) que al trabajar sólo con secuencias de aminoácidos no es suficiente para captar la esencia de las interacciones. En este trabajo se sugieren algunos procedimientos que ayudarían a elevar la exactitud de los resultados.

Palabras Clave: interacción proteína-proteína, métodos núcleo, máquina de vectores de soporte (SVM), núcleo de cadena *p*-spectrum.

ABSTRACT of the thesis presented by **Nubia Martínez Navarro** as a partial requirement to obtain the MASTER OF SCIENCE degree in Computer Science. Ensenada, Baja California, México. February 2009.

STUDY OF THE P-SPECTRUM KERNEL IN SUPPORT VECTOR MACHINES APPLIED TO THE PREDICTION OF PROTEIN-PROTEIN INTERACTIONS

Proteins are macromolecules that carry out most of the functions in the cell. Determining whether a pair of proteins can interact, i.e., relate to each other to perform any function, is one of the most important problems in the study of proteins. The problem is approached from an experimental-biological as well as computational points of view. Under the latter approach, there are several proposals that differ depending on the model used to represent the problem. One of them consider the problem in the machine learning perspective. Under this model, information about known interactions is used to teach the machine, and based on it, the interaction for a new couple of proteins is predicted. Among the most successful machine learning methods are the ones known as kernel methods, which represent a collection of complex data using a kernel function that defines the similarity between any given pair of data or objects. The support vector machine (SVM) is the most popular kernel method, it is a supervised classification algorithm.

One approach to address the classification problem based on SVM is the string kernel known as *p*-spectrum. However, it is not clear in the literature the appropriate value of *p* when working with this kernel, under the kernel of pairs scheme. The problem we face is to experimentally determine the impact of *p* in the accuracy at predicting interaction between a pair of proteins using only information about their primary structure.

Our data set consists of 400 proteins: 100 pairs of known interactions and 100 pairs with no interactions, which were used as positive and negative examples, respectively, for training and testing purposes. The arrays to the *p*-spectrum string kernel with p = 2, 3, 4, 5, 6, 7 and 8 were calculated. Subsequently, the training and testing phase were conducted in LibSVM 2.86. Three experiments were performed primarily with different proportions of the positive and negative samples for training and testing: 50-50, 30-70 and 70-30. The results indicate that *p* values of 4, 5, 6, 7 and 8 are not suitable for SVM to make the prediction properly. P = 2 is not providing relevant information to make a correct discrimination. Generally speaking, we could say that p = 3 provides the most reliable results, although not satisfactory (56.67% for the 50-50 ratio, 54% for 30-70 and 61% for 70-30). The fact that the accuracy of the results is low may be due to factors including: the importance the *p*-spectrum kernel gives to contiguous sections in the amino acid sequence. These results, however, allow us to suggest some ideas that would help improve the classification accuracy.

Keywords: protein-protein interaction, kernel methods, support vector machines (SVM), *p*-spectrum string kernel.

Dedicatorias

A Dios…

A mi amado esposo, Enrique.

A mis padres y familia.

Agradecimientos

Dr. Hugo Hidalgo, por su paciencia y ayuda.

Dr. Carlos Brizuela, por su comprensión, apoyo y dedicación.

Dr. Tchernykh, Dr. Rocha y Dr. Licea, miembros de mi comité, por sus invaluables aportaciones.

A mis compañeros Adolfo y Argelia.

Y a CONACYT por la oportunidad y el soporte que me proporcionó para realizar este trabajo.

CONTENIDO

	Página
Resumen español……	**i**
Resumen ingles……	**iii**
Dedicatorias……	**iv**
Agradecimientos……	**v**
Contenido……	**vi**
Lista de Figuras……	**viii**
Lista de Tablas……	**ix**
Capítulo I. Introducción……	**1**
I.1 Antecedentes y motivación……	**1**
I.2 Definición del problema……	**2**
I.3 Objetivo de la investigación……	**3**
I.4 Contribución al conocimiento……	**3**
I.5 Organización de la tesis……	**4**
Capítulo II. Las proteínas e interacción proteína-proteína……	**5**
II.1 Proteínas……	**5**
II.2 Proteómica……	**12**
II.3 Interacción proteína-proteína……	**14**
II.4 Métodos computacionales para predecir la interacción proteína-proteína……	**17**
II.5 Investigación previa relevante……	**20**
II.6 Los priones, un caso especial……	**25**
Capítulo III. Métodos núcleo o kernel……	**29**
III.1 Introducción a los métodos núcleo……	**29**
III.2 Núcleos para cadenas (string kernels)……	**33**
III.3 Núcleo *p*-spectrum……	**35**
III.4 Núcleo all-subsequences……	**37**
III.5 Núcleo fixed length subsequences……	**40**
III.6 Núcleo gap-weighted subsequences……	**41**
III.7 Núcleo soft matching string……	**42**
III.8 Núcleo *p*-spectrum basado en trie……	**43**
Capítulo IV. Máquinas de vectores de soporte……	**45**
IV.1 Introducción a las máquinas de vectores de soporte……	**45**
IV.2 Máquinas de vectores de soporte……	**46**

CONTENIDO (continuación)

Página

Capítulo V. Experimentos y análisis de resultados..... 54
V.1 Experimentos……... **54**
V.2 Resultados…… **60**
V.3 Análisis de resultados…… **65**
V.4 Propuesta para mejorar los resultados…… **69**

Capítulo VI. Conclusiones y trabajo futuro…..... 72
VI.1 Sumario…….. **72**
VI.2 Conclusiones…….. **73**
VI.3 Trabajo futuro…… **74**

Referencias…….. 75
Apéndice A: LIBSVM……..... 82
Apéndice B: Conjunto de datos……... 84
Apéndice C: Extracción de secuencias…… 89
Apéndice D: Código de aminoácidos…… 97
Apéndice E: Formato en LIBSVM…… 98

LISTA DE FIGURAS

Figura		**Página**
1	Hemoglobina (modelo de cintas)…….	6
2	Código genético, aminoácidos, nucleótidos y codones……	8
3	Nucleótidos y proteínas…….	9
4	Estructura de proteínas…….	11
5	Partiendo del genoma hasta el metaboloma…….	13
6	Estructura tridimensional de proteína criónica (modelo de cintas)……..	27
7	Dos representaciones diferentes de un mismo conjunto de datos…….	30
8	Núcleos y espacio de características…….	31
9	Representación producto interior del núcleo …….	32
10	Ejemplo de estructura de datos trie…….	43
11	Hiperplano óptimo…….	48
12	Separación de clases…….	50
13	Discriminación de los hiperplanos……..	53
14	Separación de un conjunto de datos en dos dimensiones mediante un hiperplano H…….	53
15	Parámetro nu…….	60

LISTA DE TABLAS

Tabla		Página
I	Ejemplo de núcleo 2-spectrum…….……..	36
II	Matriz núcleo para 2-spectrum……..……..	36
III	Ejemplo de núcleo all-subsequences……..	38
IV	Matriz núcleo para all-subsequences……..	38
V	Matriz núcleo all-subsequences con programación dinámica……...	38
VI	Ejemplo de matriz núcleo all-subsequences con programación dinámica……..	39
VII	Extracto del conjunto de datos empleado en esta investigación…….	57
VIII	Resultado del experimento para *p*=2 con proporción 50-50	61
IX	Resultado del experimento para *p*=3 con proporción 50-50	61
X	Resultado del experimento para *p*=4, 5, 6, 7 y 8 con proporción 50-50…….	62
XI	Resultado del experimento para *p*=2 con proporción 30-70	62
XII	Resultado del experimento para *p*=3 con proporción 30-70	63
XIII	Resultado del experimento para *p*=4, 5, 6, 7 y 8 con proporción 30-70…….	63
XIV	Resultado del experimento para *p*=2 con proporción 70-30	64
XV	Resultado del experimento para *p*=3 con proporción 70-30	64
XVI	Resultado del experimento para *p*=4, 5, 6, 7 y 8 con proporción 70-30…….	65
XVII	Fragmento de la matriz Kp para *p*=3……..	67

LISTA DE TABLAS (Continuación)

Tabla		Página
XVIII	Fragmento de la matriz Kp para *p*=8……..	68
XIX	Conjunto de datos empleado en esta investigación …….	84
XX	Pares de proteínas en AC …….....	86
XXI	Extracto del archivo “yeast.txt”……	89
XXII	Fragmento del archivo “locusyacsolo.txt”…….	91
XXIII	Fragmento del archivo “cienpares_locus.txt”…….	91
XXIV	Fragmento del archivo “pares_ac.txt”……..	92
XXV	Fragmento del archivo “seq.txt”…….	92
XXVI	Código de aminoácidos ……	97
XXVII	Fragmento de la matriz Kp para *p*=3……..	98
XXVIII	Fragmento de la matriz Kp para *p*=3 en formato coherente con LibSVM 2.86, datos de entrenamiento …….	99
XXIX	Fragmento de la matriz Kp para *p*=3 en formato coherente con LibSVM 2.86, datos de prueba ……....	102

Capítulo I

Introducción

I.1 Antecedentes y Motivación

Las proteínas son macromoléculas que llevan a cabo la mayor parte de las funciones de la célula y, por ende, son de vital importancia para el funcionamiento correcto de la misma. Como ejemplos de proteínas tenemos: la insulina que regula el nivel de azúcar en la sangre; la hemoglobina, que transporta el oxígeno de los pulmones a todas las células mediante los glóbulos rojos. Como estas dos proteínas, existen miles en nuestro organismo (Bolívar Zapata, 2004).

El conocimiento de las proteínas puede llegar a tener un impacto profundo en el mejoramiento de la calidad de vida de los seres humanos; por ejemplo, en el diseño de medicamentos.

En (Bolívar Zapata, 2004b) se menciona que para el tratamiento de diversos problemas clínicos se han desarrollado, a lo largo de los años, un conjunto muy amplio de productos farmacéuticos, algunos de los cuales se dirigen a proteínas específicas. Conforme aumenta el conocimiento del genoma y del proteoma humano se podrán diseñar mejores fármacos llegando, incluso, al punto de desarrollar drogas específicas para cada individuo.

"Pero el camino, desde la identificación de los genes hasta la obtención de los tratamientos efectivos, es largo y complejo" (Cañedo y Arencibia, 2004).

Realizar estudios de farmacogenómica *in vitro* es una tarea muy costosa en tiempo y recursos materiales (ACEFFYN, 2000). Este costo podría ser reducido si podemos tener alguna orientación previa acerca de la posibilidad de interacción de un par de proteínas, es decir, limitar el espacio de búsqueda (Chung *et al*., 2006; Carugo y Franzot, 2004; Gomez *et al*, 2003; Ben-Hur y Noble, 2005). La investigación propuesta aquí está enfocada a entender mejor lo que puede lograrse con algoritmos recientes usados para la predicción de interacción proteína-proteína.

La contribución de la investigación dentro del área de las Ciencias de la Computación proveerá lineamientos fundamentados para resolver un problema de clasificación dentro del contexto del aprendizaje de máquinas y la bioinformática. Una investigación como la que se propone aquí servirá de punto de partida para el desarrollo de mejores algoritmos para la predicción de interacción proteína-proteína.

I.2 Definición del problema

Existen diferentes algoritmos de clasificación para el estudio de interacción proteína-proteína. En particular, los que emplean máquinas de soporte vectorial o SVM (Support Vector Machines) son muy utilizados debido a su buen desempeño (Lewis *et al*., 2006).

Un enfoque empleado para abordar el problema de clasificación basado en SVM es el núcleo de cadenas conocido como *p*-spectrum (Shawe-Taylor y Cristianini, 2004). Sin embargo, en la literatura no queda claro cuál es el valor apropiado del parámetro *p* cuando se trabaja con este núcleo bajo el esquema de núcleos de pares (adecuado para predecir interacción proteína-proteína). El problema que queremos resolver consiste en determinar experimentalmente el efecto que tiene el valor de *p* sobre la exactitud de la clasificación.

I.3 Objetivo de la investigación

I.3.1Objetivo General

Tomando un conjunto de pares de proteínas (estructura primaria) y un algoritmo de clasificación de SVM basado en el núcleo conocido como *p*-spectrum, el objetivo es: evaluar el efecto del valor del parámetro *p* sobre la exactitud[1] en la predicción de la interacción entre un par de proteínas.

I.3.2 Objetivos específicos

- Aplicar un algoritmo de clasificación y la forma de representación de los datos de estructura primaria (secuencia de aminoácidos) de proteínas a ser usados por la SVM en la determinación de la posible interacción proteína-proteína.
- Determinar el conjunto de interacciones proteína-proteína a ser usado como conjunto de prueba.
- Determinar el valor del parámetro *p* que arroja los mejores resultados en cuanto a exactitud en la determinación de la posible interacción entre pares de proteínas.

I.4 Contribución al conocimiento

Un análisis del núcleo *p*-spectrum y del efecto de *p* para la predicción de interacción proteína-proteína permite generar información relevante y comparativa, quedando dicha información disponible para la comunidad científica.

Así mismo, con base en los resultados de la investigación propuesta es posible que se pudieran llegar a desarrollar nuevos algoritmos para la predicción de interacción entre proteínas que tengan un mejor desempeño que los disponibles actualmente.

[1] Exactitud: referente al éxito de clasificación para un conjunto predefinido de proteínas.

I.5 Organización de la tesis

La presente tesis consta de seis capítulos y cinco apéndices que complementan la información. Su organización es la siguiente:

En el primer capítulo se presenta una breve introducción al tema con antecedentes, la definición del problema, los objetivos de la investigación, entre otros, para darnos una idea de sus alcances y limitaciones.

En el Capítulo II se habla sobre los conceptos básicos de las proteínas y la interacción proteína-proteína, así como algunos métodos computacionales para predecir la interacción y se mencionan ciertas investigaciones previas con características relacionadas con las de esta investigación.

El tercer capítulo versa sobre los métodos núcleo o kernel, se profundiza en los núcleos para cadena, presentando algunos ejemplos, en especial el núcleo *p*-spectrum.

Las máquinas de soporte vectorial son el tema correspondiente al Capítulo IV, concentrando la información más relevante al respecto y haciendo uso de ejemplos para mayor comprensión.

En el Capítulo V se presentan los experimentos realizados en la investigación, sus resultados, se realiza un análisis de los mismos y una propuesta con recomendaciones que pueden incrementar la exactitud de los resultados obtenidos.

Por último, en el Capítulo VI se exponen las conclusiones y algunos elementos que podrían quedar para trabajos futuros.

Capítulo II

Las proteínas e interacción proteína-proteína

II.1 Proteínas

La palabra “proteína” viene del griego *proteios*, que significa “primero”, aludiendo a la importancia que tienen para la vida. Pero, también podrían deber su nombre al dios griego *Proteo*, que según la Odisea de Homero, tiene la capacidad de adoptar miles de formas diferentes (López Munguía, 2005).

En las células de todo organismo vivo existen dos tipos de macromoléculas biológicas en las que reside la información genética y funcional, mediante la cual la célula viva lleva a cabo sus funciones. El primero son los ácidos nucleicos: el ADN (ácido desoxiribonucleico) y el ARN (ácido ribonucleico); el otro tipo de macromolécula son las proteínas (Bolívar Zapata, 2004).

Hay proteínas nocivas para el ser humano: como la que inyecta el alacrán cuando pica, que bloquea los canales de sodio, calcio o potasio; asimismo está la toxina botulínica producida por la bacteria *Clostridium botulinum*, que paraliza los músculos, incluso los que utilizamos para respirar, aunque esta proteína tiene uso también en la industria cosmética para estirar la piel y combatir las arrugas. Existen proteínas llamadas bactericidas producidas por algunas bacterias y funcionan como antibióticos, impidiendo que otras bacterias se reproduzcan causando enfermedades. Las proteínas BT (de la bacteria *Bacillus thuringensis*) son tóxicas para los insectos, por lo que están ayudando a sustituir a los

insecticidas químicos usados en los cultivos para protegerlos de las plagas. Las enzimas, por otro lado, son proteínas responsables de catalizar las reacciones bioquímicas que la célula requiere para vivir, como hidrolizar, unir o modificar sustancias. Por ejemplo: la tripsina, que es una proteína que ayuda en la digestión de otras proteínas provenientes de organismos que forman parte de nuestra dieta alimenticia; o la amilasa, que hidroliza las cadenas de azúcares, como el almidón que se encuentra en todos los cereales. Las inmunoglobulinas son proteínas que se sintetizan principalmente en las células blancas o linfocitos, cuando se presenta un invasor o antígeno, los linfocitos producen proteínas guerreras o anticuerpos para atacar y expulsar al agente externo. Otro ejemplo es la queratina, que es una proteína fibrosa y resistente que forma parte de las uñas, el pelo o los cuernos de los animales; la miosina que es la proteína más importante de nuestros músculos ó el colágeno, que es la proteína más abundante en nuestro cuerpo y es el componente fibroso más importante de la piel, huesos, tendones, cartílagos y dientes (López Munguía, 2005). Otros ejemplos de proteínas son: la insulina (proteína producida por el páncreas) que regula el nivel de azúcar en la sangre; la hemoglobina, que transporta el oxígeno de los pulmones a todas las células mediante los glóbulos rojos, ver Figura 1. Como estas proteínas, existen miles en nuestro organismo. Gracias a ellas y a la información funcional específica contenida en cada una, el organismo y sus diferentes órganos, tejidos y células, llevan a cabo sus tareas (Bolívar Zapata, 2004).

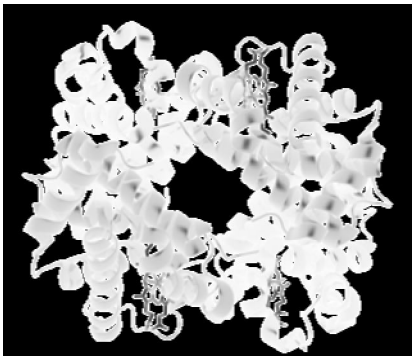

Figura 1: Hemoglobina (modelo de cintas). Obtenida de http://www.answers.com/topic/hemoglobin?cat=technology

Así como el ADN está constituido por cadenas de los cuatro nucleótidos (adenina, guanina, citosina y timina) en una doble hélice, las proteínas son el resultado de la unión de aminoácidos (en este caso, 20 diferentes) (Bolívar Zapata, 2004). Estamos formados y rodeados por cientos de miles de proteínas, constituidas por cadenas de varios tamaños en las que se combinan sus 20 aminoácidos (López Munguía, 2005). Un aminoácido es un compuesto, que como su nombre lo dice, tiene un grupo amino [$N(H_2)$] y un ácido (COOH). Lo que hace diferente a cada aminoácido del resto es la naturaleza del grupo unido al átomo de carbón central (Gavilanes, 2001).

Un dato curioso es que nuestro organismo no puede por sí mismo producir los 20 aminoácidos necesarios para obtener las proteínas, sólo produce 12 y, por ende, los 8 restantes son suministrados al organismo mediante nuestra dieta alimenticia (López Munguía, 2005).

Conforme al antiguo dogma central de la biología molecular, a partir de la información contenida en el ADN, la célula sintetiza casi todas sus proteínas. Esto se lleva a cabo mediante un proceso de dos fases: la transcripción, que es la síntesis de moléculas de ARN usando regiones específicas o genes del ADN como molde; y la traducción, que es la síntesis de proteínas a través de la “lectura” de las moléculas del ARN mensajero (ARNm) en los ribosomas. Dicha lectura, se realiza codificando un aminoácido por cada tres nucleótidos (al que se le llama triplete o codón) (Bolívar Zapata, 2004), ver Figura 2.

Aminoácidos:

- Alanina	Ala	A	- Leucina	Leu	L
- Arginina	Arg	R	- Lisina	Lys	K
- Asparagina	Asn	N	- Metionina	Met	M
- Ac. aspártico	Asp	C	- Prolina	Pro	P
- Cisteína	Cys	D	- Serina	Ser	S
- Fenilalanina	Phe	F	- Tirosina	Tyr	Y
- Glicina	Gly	G	- Treonina	Thr	T
- Ac. Glutámico	Gln	Q	- Triptófano	Trp	W
- Glutamina	Glu	E	- Valina	Val	V
- Histidina	His	H	- Terminación de		
- Isoleucina	Ile	I	la traducción	fin	

Nucleótidos:

- Guanina	G
- Adenina	A
- Timina	T
- Citosina	C

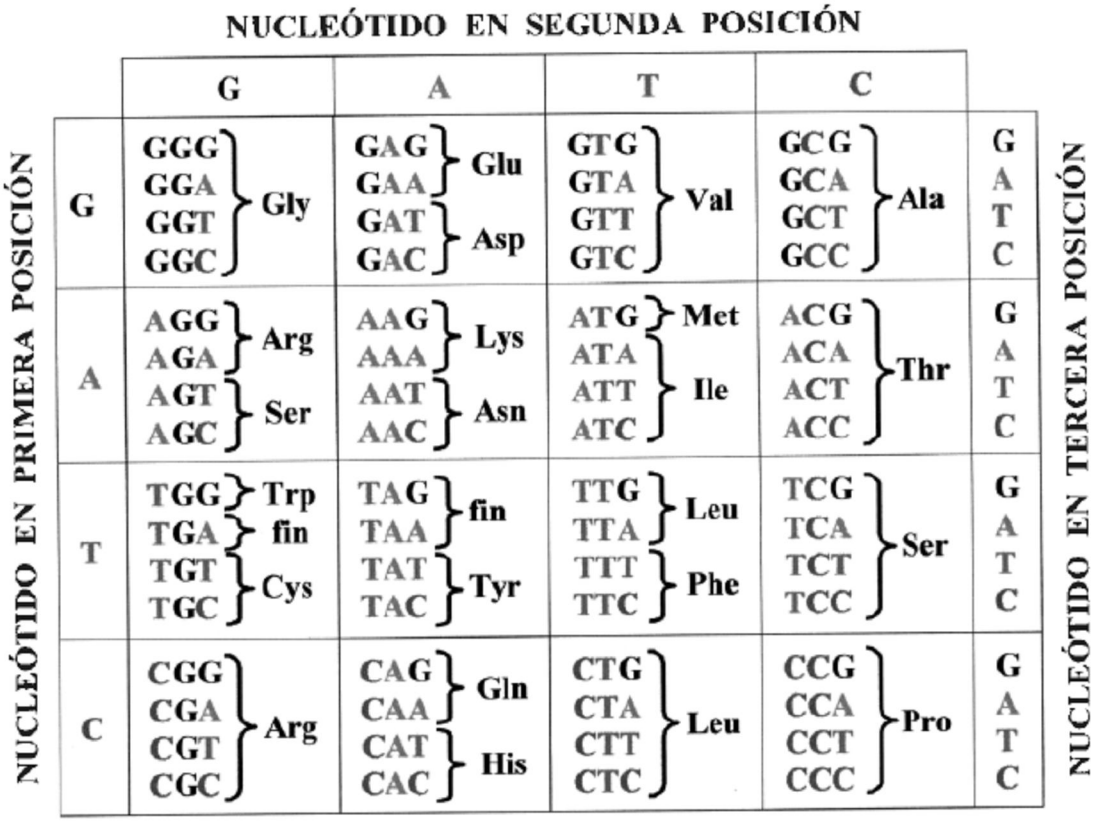

Figura 2: Código genético, aminoácidos, nucleótidos y codones.
Obtenida de (Soberón Mainero y Montero Morán, 2004).

La transcripción del ADN en ARN genera una molécula de ARN que es copia fiel del ADN transcrito. En el caso de las bacterias (organismo procarionte o

unicelular sin núcleo), esta molécula de ARN es directamente traducida en proteínas. Pero, en el caso de los organismos eucariontes, la mayor parte de los transcritos de ARN son procesados, para ser transformados en los ARNm que se traducen en proteínas a nivel de los ribosomas (Bolívar Zapata, 2004b), ver Figura 3.

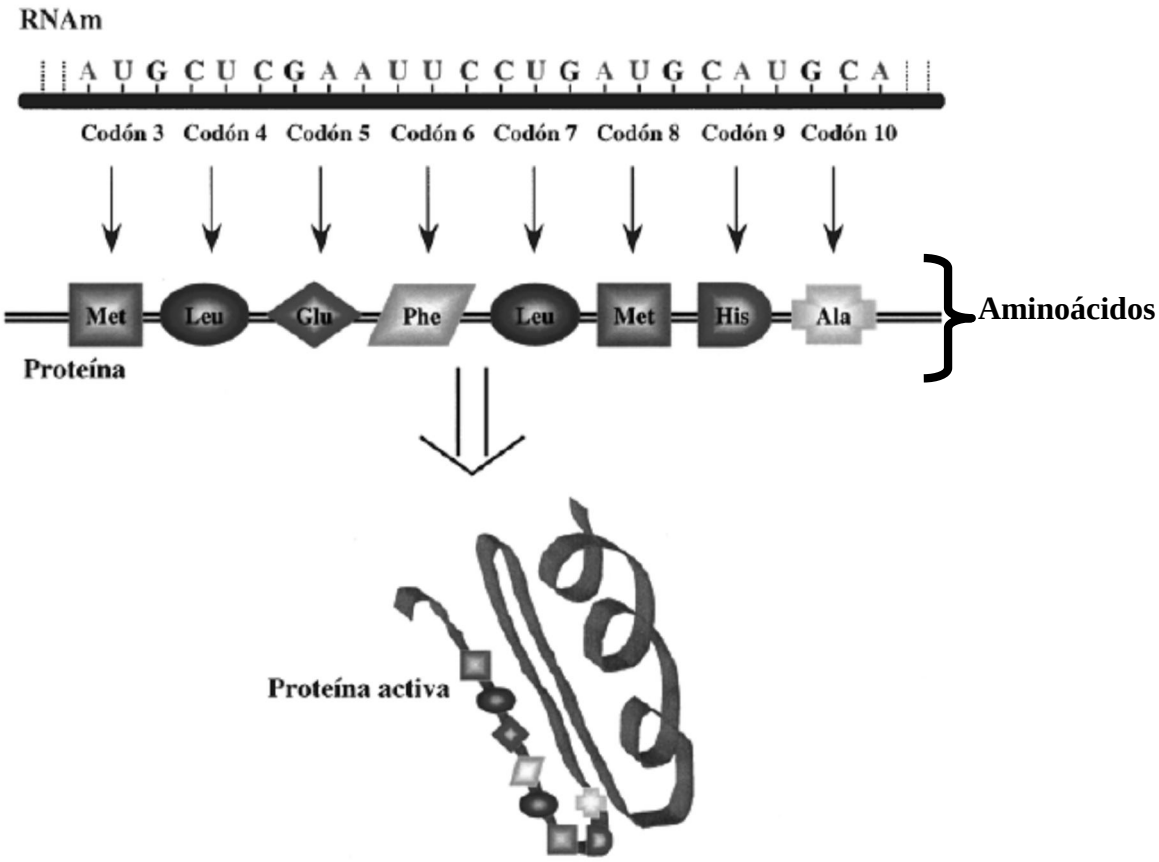

Figura 3: Nucleótidos y proteínas. La secuencia de nucleótidos en el gen es responsable de la secuencia de nucleótidos del ARN mensajero, y este último (en forma de tripletes) de la secuencia de aminoácidos en la proteína codificada. Obtenida de (Soberón Mainero y Montero Morán, 2004).

En las bacterias, organismos unicelulares, el genoma contiene de 3,000 a 5,000 genes, variando según la especie que se trate. En el caso de los humanos, existen alrededor de 40,000 genes localizados en los 46 cromosomas que forman nuestro genoma, que es diploide porque en las células del cuerpo humano hay 23 pares de cromosomas, es decir, tenemos la información genética por duplicado

(una parte proviene del padre y otra de la madre). Cada cromosoma humano está formado por una sola molécula de ADN (Bolívar Zapata, 2004b).

En cuanto a la estructura de las proteínas, ésta se puede ver desde cuatro niveles. La cadena de aminoácidos constituye la secuencia o estructura primaria. Pero, la proteína puede adquirir una estructura secundaria, α hélice o β plegada, básicamente (Soberón Mainero y Montero Morán, 2004). Los aminoácidos se acomodan en formas como hélice, como en una serpentina, pero también como hojas o láminas. Otra parte de la cadena simplemente no se acomoda o sirve para unir las hélices con las láminas (López Munguía, 2005). Las estructuras secundarias, mediante su doblamiento producen las estructuras terciarias. Las estructuras terciarias permiten la asociación de varias moléculas de proteínas y surgen las estructuras cuaternarias, ver Figura 4. La estructura de la proteína está íntimamente ligada con la función biológica de la misma (Soberón Mainero y Montero Morán, 2004).

Cabe destacar la supremacía numérica de las proteínas frente a los genes. El genoma humano, con sólo 35 ó 40 mil genes codifica para un proteoma con más de 40 mil proteínas, no sería extraño que finalmente el proteoma humano estuviera constituido por un conjunto, incluso, mayor a 100,000 proteínas (Bolívar Zapata, 2004b).

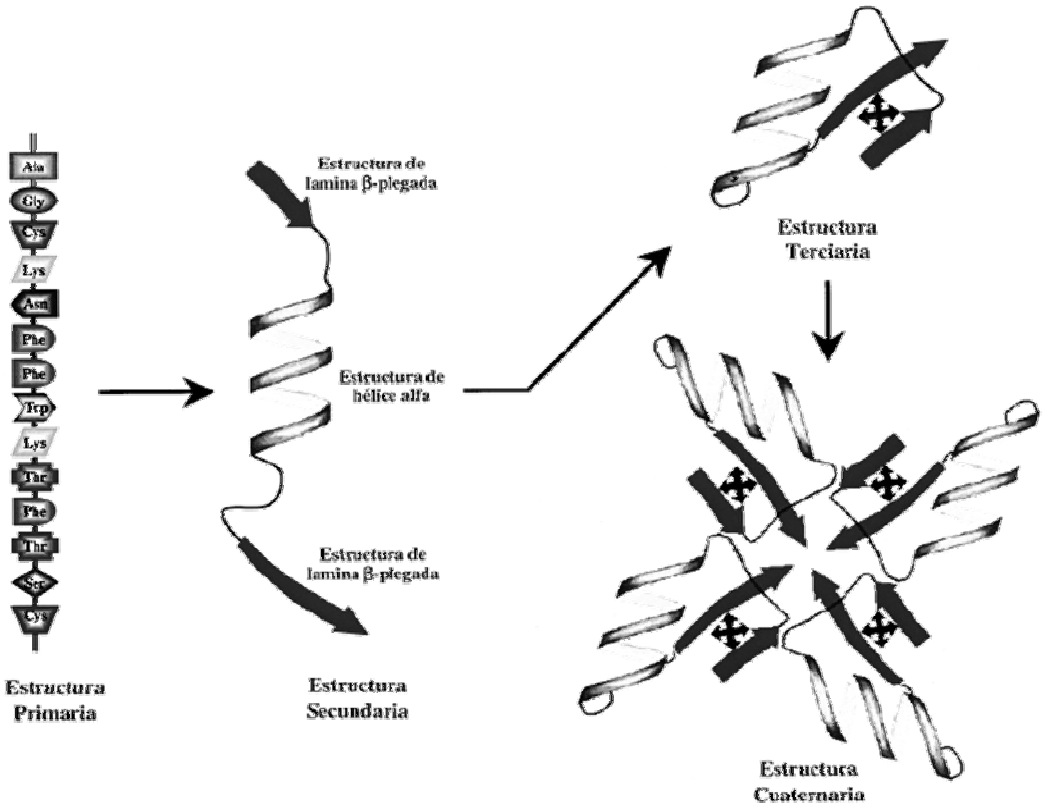

Figura 4: Estructura de proteínas. Obtenida de (Soberón Mainero y Montero Morán, 2004).

Las proteínas pueden ser agrupadas en familias. Por familias nos referimos a grupos de secuencias procedentes de varias especies y que realizan una función similar y tienen un mismo origen evolutivo. La estructura y función de proteínas de una misma familia son muy similares (Eidhammer *et al*., 2004).

Es conveniente tener presente ciertos conceptos básicos: como el de genoma, conjunto de todos los pares de nucleótidos que incluyen el total de nuestros genes y otras secuencias de ADN que se encuentran en los cromosomas; o como el proteoma, que es el conjunto de todas las proteínas que puede sintetizar un organismo (Bolívar Zapata, 2004b). En la siguiente sección se hablará más a fondo de estos conceptos.

II.2 Proteómica

La proteómica, término designado por Marc Wilkins en 1994, es un nuevo enfoque en el estudio funcional de las proteínas (Chagolla López y Barba de la Rosa, 2006). La proteómica es la disciplina encargada del análisis y caracterización del proteoma (conjunto de las proteínas expresadas por un genoma) (BorNet, 2000).

Debemos tomar en cuenta que la vida en todos sus niveles es dinámica, en constante interacción con el medio. Al conocer el proteoma de un organismo tenemos una imagen dinámica de todas las proteínas que expresa ese ser vivo, en un momento determinado y bajo condiciones concretas de tiempo y entorno. Las células expresan varios miles de proteínas diferentes y cada una de ellas puede verse modificada de acuerdo a las condiciones del ambiente. Por ejemplo, cuando un virus ataca una célula, esta producirá anticuerpos (proteínas) como reacción al ataque, pero no en otras circunstancias. Este es el caso de ciertas proteínas que se expresan ante determinados estímulos del ambiente (Cañedo Andalia y Arencibia Jorge, 2004).

El proteoma, entendido como el conjunto completo de proteínas expresadas por una célula durante su tiempo de vida, resulta mucho más complejo que el estudio del genoma. En el ADN sólo están repetidas cuatro bases nucleotídicas (adenina, guanina, citosina y timina) en forma de conjuntos de tripletes (tres bases) obteniéndose un aminoácido de la expresión de cada triplete. Las proteínas, en cambio, están constituidas por combinaciones de 20 aminoácidos (Bolívar Zapata, 2004).

El conocimiento de la secuencia de todos los genes fue la base para el desarrollo de metodologías que permiten determinar simultáneamente los niveles de todos los ARNm en la célula en un momento y condición particular. Esta

información es conocida como transcriptoma, y es sumamente valiosa para establecer la función de los productos proteicos de estos genes. La ciencia proteómica busca conocer la función e interrelación de todas las proteínas de un organismo, y de ahí conocer como funciona toda la maquinaria celular (metaboloma) (Gosset Lagarda y Bolívar Zapata, 2004), ver Figura 5.

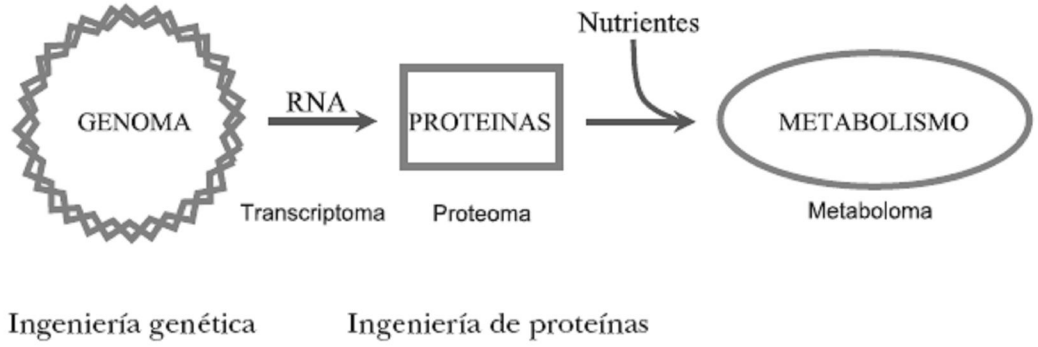

Figura 5. Partiendo del genoma hasta el metaboloma. Obtenida de (Gosset Lagarda y Bolívar Zapata, 2004).

Uno de los aspectos principales de la proteómica es detectar cuáles componentes proteicos pueden usarse como bioindicadores de errores que llevan a la aparición de desórdenes o enfermedades, buscando además medios para su detección y corrección (BorNet, 2000).

Se vislumbra un gran beneficio al poder desarrollar medicamentos sobre la base de regular la producción de proteínas específicas en distintas afecciones como cáncer, Parkinson, Alzheimer, enfermedades infecciosas, preparación de vacunas, entre otras (Maino, 2001).

Una de las etapas de la era genómica ha culminado, la proteómica ya comenzó. Sin embargo, existen otras etapas que también son de interés en la bioinformática como el interactoma (Parrish *et al*., 2006), que se refiere al estudio e identificación sistemática de interacciones de proteínas dentro de un organismo.

Debe tomarse en cuenta que la nomenclatura puede ser empleada de forma un poco general y varía desde el grado de especificación que se considere y la formalidad de expresión.

II.3 Interacción proteína-proteína

Como su nombre lo indica, la interacción proteína-proteína se refiere a las asociaciones entre proteínas y el estudio de las mismas. Ya sean éstas breves o más prolongadas en términos temporales. Existe una gran diversidad de interacciones y la función racional de algunos de esos complejos es aún desconocida (Nooren y Thornton, 2003).

La mayoría de las funciones biológicas son mediadas por interacciones entre proteínas. He ahí su importancia, puesto que la vida de los organismos depende de la habilidad de sus moléculas de reconocer a otras para trasmitir señales, transferir electrones, controlar su forma, reaccionar a estímulos, etc. (Carugo y Franzot, 2004). Para entender la maquinaria molecular de la célula, necesitamos conocer las múltiples interacciones proteína-proteína que permiten a la célula funcionar (Gomez *et al*., 2003). Estas interacciones pueden ser de dos tipos: físicas o lógicas. Ejemplos de interacciones físicas son: cuando dos proteínas forman un complejo estable, en el cual la unidad funcional es formada por más de una cadena de proteínas; y asociaciones transitorias, en las cuales las cadenas de proteínas son estables pero pueden interactuar para transmitir una señal o como respuesta a condiciones externas. Las interacciones lógicas se dan cuando una o más proteínas controlan el comportamiento de otras proteínas sin interacción física. Es decir, una proteína afecta a otra mediante, por ejemplo, la regulación de su expresión o cambiando la concentración de un factor. Los dos modos de interacción (física y lógica) no son exclusivos. Una misma proteína puede tener tanto interacciones físicas como lógicas (Tramontano, 2005).

La interacción proteína-proteína puede ocurrir entre cadenas idénticas o diferentes. También las interacciones pueden ser distinguidas por el tiempo de vida de los complejos. Las interacciones permanentes son muy estables y existen en su forma compleja. Las temporales asocian y separan *in vivo*. Es importante señalar que muchas interacciones proteína-proteína no caen en una u otra categoría porque mucho depende de las condiciones fisiológicas y ambientales; por ejemplo, una interacción temporal puede volverse permanente bajo ciertas circunstancias celulares. Aunque, la localización de subunidades y la función de la proteína sugieren el tipo de interacción biológicamente relevante (Nooren y Thornton, 2003).

Una proteína, generalmente, se encuentra en un ambiente con múltiples elementos potenciales para interactuar, éstos con diferentes propiedades de superficie. La mayoría de las proteínas son muy específicas al elegir con que elemento interactuar. Sin embargo, puede darse el caso de que varios elementos coincidan con la misma proteína o compitan para interactuar con ella. Las interacciones entre proteínas son diversas y complejas (Nooren y Thornton, 2003).

El rápido incremento en el número de estructuras tridimensionales obtenidas de experimentos, proporciona la oportunidad de entender mejor y, posteriormente, predecir interacciones entre proteínas (Chung *et al*., 2006).

La interacción proteína-proteína es muy importante pues si conocemos la posibilidad de interacción entre una proteína y otra, además de sus funciones, es posible diseñar medicamentos que bloqueen la acción de una proteína deficiente. O bien, propiciar la interacción beneficiosa entre un par de proteínas. “Estos datos (de interacciones entre proteínas) constituyen la base del posible desarrollo de nuevos medicamentos que tengan como objetivo interrumpir interacciones específicas en lugar de, como hacen los medicamentos actuales, anular

completamente todas las funciones de una o varias proteínas" (Martínez Hormazábal, 2006).

Cabe señalar que del análisis de las interacciones entre proteínas, se ha llegado a la conclusión que la complejidad de los organismos no depende del número de proteínas codificadas por sus genomas, sino del número de interacciones entre ellas; en el sentido de que existen organismos con un número "similar" de genes como: la planta *Arabidopsis thaliana* (25,498), el nematodo *C. elegans* (19,099), la mosca *Drosophila melanogaster* (14,100) y el hombre (30,000-35,000). Sin embargo, sus niveles de complejidad son diferentes (De la Torre Russis *et al*., 2003).

Existen varias bases de datos sobre interacción entre proteínas para documentarlas y describirlas. Estas bases de datos incluyen toda la información obtenida por diversos métodos experimentales e información publicada en la literatura científica (De la Torre Russis *et al*., 2003). Destacan:

- BIND (Biomolecular Interaction Network Database), que es, ahora, un componente de BOND (Biomolecular Object Network Databank). Http://www.unleashedinformatics.com/index.php?pg=homev
  (De la Torre Russis *et al*., 2003), (Xenarios y Eisenberg, 2001), (Alfarano *et al*., 2005), (Bader *et al*., 2001).

- DIP (Database of Interacting Proteins). Http://dip.doe-mbi.ucla.edu/
  (De la Torre Russis *et al*., 2003), (Xenarios y Eisenberg, 2001).

- MIPS (Munich Information Center for Protein Sequences). Http://mips.gsf.de
  (Xenarios y Eisenberg, 2001).

Como puede verse, estas bases de datos y otras más pueden ser fácilmente consultadas vía Internet.

En la presente investigación se pretende realizar un análisis experimental del núcleo de cadena o string kernel *p*-spectrum (Shawe-Taylor y Cristianini, 2004), dado que tratamos sólo con la secuencia de aminoácidos de las proteínas, que finalmente, son cadenas de letras. Obtuvimos de (Ben-Hur y Noble, 2005) la forma de construir el "kernel" para los pares de proteínas. Y utilizamos una SVM, que es un método de aprendizaje de máquinas basado en funciones núcleo o kernels. Variando *p*, analizamos cual de ellos aporta la mayor precisión en la predicción de interacciones proteína-proteína basados sólo en estructura primaria. Nuestro conjunto de datos es un subconjunto del utilizado en (Ben-Hur y Noble, 2005), que es extraído de la base de datos BIND y se refiere a interacciones físicas en levadura.

Existen diferentes algoritmos para el estudio de interacción proteína-proteína. En particular, los que usan máquinas de vectores de soportes o SVM (Support Vector Machines) son muy utilizados debido a su buen desempeño (Lewis *et al*., 2006). A continuación, se ahondará sobre los métodos computacionales que ayudan en la predicción de las interacciones entre proteínas.

II.4 Métodos computacionales para predecir la interacción proteína-proteína

Además de los métodos experimentales para identificar las interacciones entre proteínas, existen métodos teóricos; los cuales son de gran importancia dada la enorme cantidad de datos biológicos existentes (Salwinski y Eisenberg, 2003). Una clasificación general de los mismos, podría ser la siguiente (Shoemaker y Panchenko, 2007):

II.4.1 Métodos de genes vecinos y agrupación de genes (gene cluster).

Genes con funciones estrechamente relacionadas codifican proteínas que interactúan potencialmente y son transcritas, con frecuencia, como una unidad u operón en bacterias y son co-reguladas en eucariontes. Operones encontrados por métodos de vecinos pueden proveer evidencia adicional acerca de vínculos funcionales entre sus genes constituyentes (Shoemaker y Panchenko, 2007). Las relaciones de vecindad son más relevantes cuando han sido conservadas en especies diferentes. La adyacencia de genes en varios genomas de bacterias se ha usado para predecir relaciones funcionales entre las proteínas correspondientes (Valencia y Pazos, 2002).

II.4.2 Métodos de perfil filogenético.

Estos métodos se fundamentan en la co-evolución de proteínas que interactúan, es decir, en detectar la presencia-ausencia de genes ortólogos[2] en genomas diferentes. “Si el patrón de proteínas ortólogas (presencia-ausencia) se conserva en organismos de la misma especie, se debe probablemente a que una de las proteínas no puede ejercer su función sin la otra” (De la Torre Russis *et al*., 2003).

II.4.3 Método Rosetta Stone.

En este método se infiere la interacción entre proteínas con base en la secuencia de proteínas en diferentes genomas. Se fundamenta en la observación de que algunas proteínas que interactúan tienen homólogos en otros genomas que son fusionados en una cadena de proteínas, llamada “Rosetta Stone protein”. La fusión de los genes ocurre, aparentemente, para optimizar la co-expresión de los genes codificados por interacción entre proteínas (Shoemaker y Panchenko, 2007).

---

[2] Genes ortólogos: genes que descienden verticalmente de un gen ancestral común y tienden a tener la misma función. Fuente: Ecolegio Glosario. URL: http://www.ecolegio.cl/glosario-o-p.htm

II.4.4 Métodos de co-evolución basados en secuencia.

Proteínas que interactúan, con frecuencia, co-evolucionan, así que cambios en una proteína que conlleva pérdida de función o interacción pueden ser compensados con cambios correlacionados en otra proteína. La co-evolución se puede reflejar en términos de similitud entre árboles filogenéticos de familias de proteínas que interactúan (Shoemaker y Panchenko, 2007). Se puede combinar información de secuencias de aminoácidos y genomas para calcular distancias evolutivas entre proteínas de familias relacionadas (De la Torre Russis *et al*., 2003).

II.4.5 Métodos de clasificación

También conocidos como métodos de aprendizaje de máquina, aprendizaje automático o “Machine Learning”. Usan varias fuentes de datos para predecir la interacción entre proteínas (Shoemaker y Panchenko, 2007). Pueden dividirse en dos grandes grupos: enfoque supervisado y no supervisado. El aprendizaje supervisado es aquel en el que se conoce la clase cierta de un conjunto de patrones. A éste se le conoce como conjunto de entrenamiento (Cortijo Bon, 2001). Es decir, el aprendizaje supervisado es cuando a un algoritmo de clasificación se le proporciona un conjunto de ejemplos de datos con la clase a la que pertenecen y se prueba en un conjunto de datos cuyas clases desconocemos (Bobadilla *et al*., 2003).

Un ejemplo de un algoritmo de aprendizaje no supervisado para determinar la interacción entre proteínas, es el propuesto por Mamitsuka. Dicho enfoque se basa en un algoritmo EM (Expectation Maximization) y en un modelo probabilístico que considera el conocimiento latente de proteínas (evidencias de interacción) (Mamitsuka, 2005).

Ahora bien, en cuanto a clasificadores supervisados, podemos mencionar los árboles de decisión (Decision Trees), el clasificador Bayesiano simple, los

bosques aleatorios de decisión (Random Forest Decision) y los métodos de kernel o núcleo como las máquinas de vectores de soporte (Support Vector Machines) (Kohutyuk y Honavar, 2006). Los métodos de núcleo son particularmente útiles en la predicción de interacción entre proteínas, ya que proporcionan una representación vectorial de los datos en el espacio de características mediante grupos de comparaciones de pares (Shoemaker y Panchenko, 2007).

Un método núcleo representa una colección de datos complejos arbitrarios usando una función núcleo que define la similitud entre cualquier par de datos u objetos dados. Las máquinas de vectores de soporte son el método de núcleo más popular, dado que tienen un desempeño robusto y permiten el tratamiento de datos no heterogéneos como los existentes en las bases de datos biológicas (Lewis *et al*., 2006).

Como mencionábamos en el apartado anterior, el método empleado en nuestro estudio cae en los últimos mencionados, es decir, en los métodos de clasificación, dentro del aprendizaje supervisado y los métodos kernel, en particular, se hace uso de las máquinas de vectores de soporte (SVM, por sus siglas en inglés) con núcleos de cadena o string kernels. Profundizaremos en las máquinas de vectores de soporte y los métodos núcleo en los siguientes capítulos de la tesis.

II.5 Investigación previa relevante

A continuación se presentan algunos de los trabajos considerados relevantes para la predicción de posibles interacciones proteína-proteína usando el método de máquinas de vectores de soporte.

II.5.1 Uso de secuencia y estructura para identificar sitios de unión proteína-proteína.

Haciendo uso de datos del PDB[3] (Protein Data Bank), Chung *et al*. realizaron un estudio que incorpora residuos de la superficie conservados estructuralmente con información derivada de alineamiento de secuencias y estructuras individuales para identificar sitios de unión proteína-proteína (protein-protein binding sites). Debido a que las funciones de las proteínas se llevan a cabo por medio de interacciones con otras moléculas, la identificación de los sitios de unión es muy importante para el diseño de medicamentos, anotaciones funcionales, y entender los mecanismos de reconocimiento molecular (Chung *et al*., 2006).

El puntaje de conservación de estructura se deriva de alineaciones de múltiples estructuras. Este puntaje se combina con el perfil de secuencia e información del área de superficie accesible para identificar sitios de unión usando SVM. Cabe destacar que el uso de conservación de estructura mejora significativamente el desempeño de la SVM. Cerca del 52% de los sitios de unión fueron detectados con precisión (más del 70% de los residuos en el sitio fueron identificados), 77% de los sitios fueron correctamente predichos (poco más del 50% de los residuos en el sitio fueron identificados). Para combinar información evolutiva derivada de alineación de secuencias e información físico-química de las estructuras, la SVM fue entrenada para identificar residuos de sitios de unión usando conservación de estructuras, áreas de superficie accesibles y perfil de secuencias de residuos vecinos en estructuras (Chung *et al*., 2006).

II.5.2 Máquinas de vectores de soporte para análisis de proteínas.

Se han llevado a cabo investigaciones para determinar el desempeño de SVM en la tarea de inferir anotaciones funcionales de genes a partir de una combinación de secuencias de aminoácidos y estructura de proteínas. Con sólo

[3] URL: http://www.pdb.org/

dos tipos de datos, una SVM entrenada por una combinación de grupos de datos no ponderados, funciona igual o mejor que algoritmos más sofisticados que asignan pesos a tipos de datos individuales. Aunque, cuando los datos contienen mucho ruido, el enfoque de pesos tiene un desempeño mejor. Además, se hace hincapié en que los métodos núcleo o kernel tienen la útil habilidad de manejar grandes cantidades de datos heterogéneos, tan comunes en aplicaciones biológicas. La combinación de los núcleos puede ser llevada a cabo mediante operaciones algebraicas como la suma (Lewis *et al*., 2006). Una suma sin pesos de núcleos fue usada con éxito en la predicción de interacciones proteína-proteína en un estudio anterior (Ben-Hur y Noble, 2005).

II.5.3 Aprendiendo a predecir interacciones proteína-proteína

El modelo de atracción y repulsión se basa en que la interacción entre un par de proteínas es representada por la suma de las fuerzas de atracción y repulsión. El modelo resulta eficiente para tratar con grandes cantidades de datos. Según (Gomez *et al*., 2003) resulta competitivo en la tarea de predecir interacciones proteína-proteína en levaduras.

En el modelo de atracción y repulsión se supone que ciertas características conservadas por la evolución en cada proteína de un par que interactúan son responsables de la interacción. Los pares que interactúan encontrados en una sección de una red frecuentemente tienen homólogos en otra región de la misma red. Se supone que la redundancia evolutiva implica que características que tienen un valor de predecir para una porción de una red grande también lo tienen en otra parte de la red. La red molecular es representada como un grafo $G= (V, A)$ con vértices V y aristas A, donde los vértices corresponden a las proteínas y las aristas a las interacciones físicas entre ellas. Se supone que cualquier par de proteínas en la red pueden interactuar y se asignan probabilidades a dichas interacciones (Gomez *et al*., 2003).

II.5.4 Predicción de interacción proteína-proteína de la estructura primaria y propiedades físico-químicas con vectores.

Para la predicción se entrena una SVM con la interacción en base a secuencias de proteínas y propiedades fisicoquímicas, alcanzando un 80% (en promedio) de interacciones correctamente predichas. Los datos son obtenidos de DIP[4] (Database of Interacting Proteins). Para cada secuencia de aminoácidos de un complejo proteína-proteína se tomó en cuenta información como: carga, hidrofobicidad y tensión superficial de cada residuo en la secuencia. Se utilizó la concatenación de vectores para representación de características (Bock y Gough, 2001).

II.5.5 Predicción de interacciones proteína-proteína usando productos característicos (signature products).

Los productos característicos de pares de proteína son implementados en un clasificador de máquinas de vectores de soporte como una función núcleo o kernel. Se aplica este método a datos públicos de levadura, *Helicobacter pylori*, humanos y ratón. Se utilizaron los conjuntos de datos de levadura y *H. pylori* para verificar la habilidad predictiva del método, alcanzando entre 70% y 80% de precisión usando validación cruzada de 10-fold. Se usaron los datos de humanos y ratones para demostrar la capacidad del método de predecir a través de diferentes especies (Martin *et al*., 2005).

Un producto característico es el resultado de subsecuencias y una expansión del descriptor característico (signature descriptor) para información química. Una firma (signature) consiste de un aminoácido y sus vecinos, así como el espacio característico (signature space) son todas las posibles firmas. Como un ejemplo, considere la secuencia de aminoácidos LVMTTM. Todas las firmas de altura 1 están basadas en triadas, por lo tanto, tenemos LVM, VMT, MTT y TTM. Cada firma tiene una raíz (la letra media) y sus dos vecinos ordenados

[4] DIP (Database of Interacting Proteins). Http://dip.doe-mbi.ucla.edu/

alfabéticamente. Entonces, las firmas correspondientes a las cuatro triadas anteriores serían: V(LM), M(TV), T(MT) y T(MT). Obviamente, este es un ejemplo sencillo que puede extenderse a manejar subsecuencias más largas y diferentes alturas. El núcleo de firmas (signature kernel) sería k(A, B) = s(A) • s(B). Pero, para trabajar con pares de proteínas se hace necesario el uso de productos entre vectores, dando lugar al producto de firmas (signature product) s⊗s (A, B) = s(A)⊗ s(B). Finalmente, el núcleo del producto de firmas (signature product kernel) es: $k((A,B),(C,D)) = (s(A) \otimes s(B)) \bullet (s(C) \otimes s(D))$ (Martin *et al*., 2005).

II.5.6 Detección de interacción proteína-proteína basado en la sensibilidad de subcadenas.

En Zaki *et al*. (2006) se propone un método para detectar la interacción proteína-proteína midiendo la similitud entre subcadenas. La idea es que dos secuencias de proteínas pueden interactuar dado el grado de similitud entre las subcadenas que contienen, ya que dicha similitud incorpora conocimiento biológico acerca de las relaciones evolutivas de las estructuras de proteínas. Se aplica este método sobre datos de levadura *Saccharomyces cerevisiae* del DIP, obteniéndose arriba del 80% de precisión.

Considerando dos secuencias de proteínas $p_1=(a_{11}, a_{12},\ldots,a_{1n})$ y $p_2=(a_{21}, a_{22},\ldots,a_{2m})$, donde a_{1n} se refiere al aminoácido *n* en la secuencia de proteína p_1, y son representados como $seq_{pos1}= (a_{11}, a_{12},\ldots,a_{1n}, a_{21}, a_{22},\ldots,a_{2n})$ si ambas secuencias sí interaccionan. Por lo tanto, seq_{pos1} es la nueva secuencia de proteínas creada mediante la concatenación de dos proteínas interactuantes. Y de manera similar se lleva a cabo para los pares de proteínas que no interaccionan. Posteriormente, se crean las subcadenas de las secuencias mediante una ventana k>1 (la mejor ventana es k=30 según los resultados del estudio) y se mide la similitud con el algoritmo Smith-Waterman. Para realizar la discriminación entre pares de proteínas que interaccionan y aquellas que no, se utiliza SVM, en particular LibSVM (Zaki *et al*., 2006).

II.5.7 Métodos núcleo para predicción de interacción proteína-proteína.

Se ha realizado un estudio de métodos núcleo para predicción de la interacción entre proteínas, usando una combinación de fuentes de datos, incluyendo secuencias de proteínas, anotaciones ontológicas, propiedades locales de la red e interacción homóloga con otras especies. Se ilustra la eficacia de los núcleos en conjunto con un clasificador SVM. La tarea de aprendizaje involucra la relación entre pares de secuencias de proteínas para determinar si un par de proteínas están interactuando o no. Pero, como es difícil predecir interacciones utilizando sólo secuencias, se incorporaron fuentes adicionales de datos. Se aplicó dicho método para predecir las interacciones físicas en levadura usando datos de la base BIND[5] (Biomolecular Interaction Network Database). Se reportan resultados del 80% de efectividad (Ben-Hur y Noble, 2005).

En nuestro estudio tomamos la forma de combinar núcleos o kernels para pares de proteínas de (Ben-Hur y Noble, 2005), pero usando sólo el núcleo de cadena o string kernel *p*-spectrum (Shawe-Taylor y Cristianini, 2004) y la SVM. Variando *p*, analizamos cual de ellos aporta la mayor precisión en la predicción de interacciones proteína-proteína basados sólo en su estructura primaria. Por lo tanto, un análisis como el que hace y que no se ha llevado a cabo con la perspectiva propuesta, es muy interesante y útil para varios sectores de la comunidad científica.

II.6 Los priones, un caso especial

En esta sección, hablaremos brevemente sobre los priones, unas proteínas con un comportamiento muy particular, y cuya mención se hace necesaria para definir los límites de nuestra investigación.

[5] BIND es, ahora, un componente de BOND (Biomolecular Object Network Databank).
URL: http://www.unleashedinformatics.com/index.php?pg=home

Hace unas décadas se descubrió que algunas proteínas pueden ser agentes infecciosos, y se les llamó priones. Son responsables de enfermedades como el mal de las vacas locas. En lugar de reproducirse como los virus o bacterias, provocan que otras proteínas cambien de forma, así el organismo no puede destruirlas y se acumulan en los tejidos del cerebro, dándole un aspecto de esponja (visible en los exámenes post mortem) (López Munguía, 2005).

En 1982 Stanley B. Prusiner dio el nombre de prion (partícula infecciosa de naturaleza proteica) al agente vinculado a un grupo de desórdenes degenerativos del sistema nervioso central, que comparten características patológicas crónicas y progresivas (Prusiner, 1982; Prusiner, 1998). Es transmisible por herencia o infección (ingesta de órganos contaminados, hormona de crecimiento inyectada, trasplante de córnea procedente de personas infectadas). La evidencia que las proteínas pueden transmitir una enfermedad infecciosa causó gran sorpresa entre la comunidad científica (Villegas de Olazábal y Villegas del Carpio, 1997).

Las enfermedades transmitidas por priones más conocidas, en animales son: encefalopatía espongiforme bovina (“vacas locas”), scrapie (ovejas), encefalopatía transmisible (visones), enfermedades crónicas de desgaste (mulas, ciervos y alces). Por otro lado, en los humanos serían: enfermedad de Cretzfeld-Jakon (CJD), síndrome de Gertmann-Straussler-Scheinker (GSS), kuru (propio de habitantes de Nueva Guinea por rituales de canibalismo), insomnio fatal familiar (FFI) y síndrome de Alpers (Villegas de Olazábal y Villegas del Carpio, 1997).

Los priones son agentes causantes de un grupo de patologías neuro-degenerativas letales características de mamíferos, también conocidas como encefalopatías espongiformes transmisibles. Estos agentes son capaces de propagarse dentro de un mismo huésped causando una lesión espongiótica y de transmitirse de huésped a huésped con tiempos elevados de incubación. La búsqueda de este agente reveló como componente mayoritario, si no único, a una

proteína PrPsc. Así, se denomina prion a la forma alterada de una proteína celular funcional (PrPc en mamíferos) que pierde su función normal y adquiere la capacidad de transformar la forma normal en patológica (Gasset y Westaway, 1998).

Los priones convierten proteínas normales en moléculas anormales mediante la modificación de su forma (ISCIII, 2001), ver Figura 6.

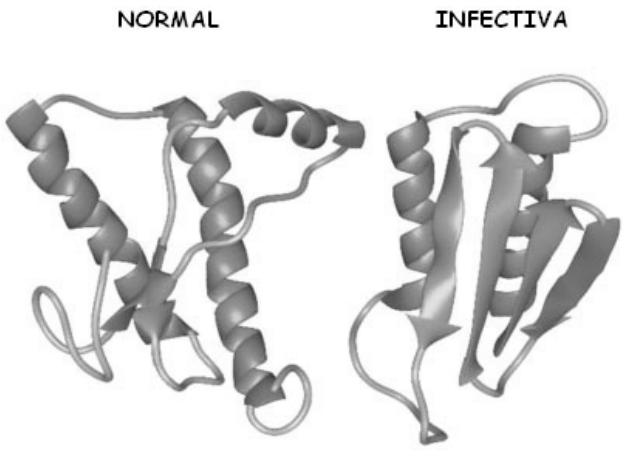

Figura 6: estructura tridimensional de proteína criónica (modelo de cintas). Obtenida de http://www.isciii.es/htdocs/centros/epidemiologia/salud_publica.jsp

El prion presenta dos isoformas: normal (PrPc) y anormal (PrPcs) o infecciosa. (Villegas de Olazábal y Villegas del Carpio, 1997). Ambas proteínas poseen la misma secuencia de aminoácidos, pero distinta conformación tridimensional. La teoría de los priones propuesta por Prusiner supone, entonces, la existencia de dos plegamientos para una única secuencia de aminoácidos (Raisman, 2007).

Ahora bien, como nuestra investigación se enfoca en el estudio de los algoritmos para predecir la posible interacción proteína-proteína (con máquinas de vectores de soporte y el núcleo de cadena *p*-spectrum), pero basados sólo en la

estructura primaria de las proteínas, es decir, su secuencia, se hacía necesario señalar el caso de los priones. Debido a que para una misma proteína, con la misma secuencia, presentan estructuras tridimensionales diferentes, afectando con ese cambio la posible interacción entre proteínas, no se puede diferenciar la forma normal de la anormal sólo observando la secuencia. Es por esto, que se ha observado que van más allá del alcance del presente estudio, y por ende, se excluyen del mismo. Abriéndose de esta manera, nuevas líneas de investigación para trabajos futuros.

Capítulo III

Métodos núcleo o kernel

III.1 Introducción a los métodos núcleo

Una función núcleo o kernel *K(x,y)* define la similitud entre un par de objetos. *K(x,y)* con un valor alto indica que *x* y *y* son similares y un valor bajo indica que son diferentes o menos parecidos (Lewis *et al*., 2006). Una de las primeras apariciones de los métodos núcleo fue en aplicaciones estadísticas realizadas Vapnik en base a la teoría desarrollada por Aronszajn y Parzen (Vert *et al*., 2004). Lo que convierte a los métodos núcleo en una herramienta relativamente joven. A continuación, ahondaremos en la cuestión de la representación de los datos que deseamos analizar por medio de núcleos. Sea $S=(x_1, ..., x_n)$ un conjunto de *n* objetos a ser analizados. Supóngase que cada objeto x_i es un elemento de un conjunto *X*; el cual puede ser, por ejemplo: un grupo de imágenes o un grupo de moléculas en un contexto biológico. Ahora bien, una de las primeras preguntas que debemos hacernos es cómo representar los datos del conjunto *S* para su procesamiento. La mayoría de los métodos de análisis de datos primero definen una representación para cada objeto, y luego todo el conjunto mediante la agrupación de las representaciones individuales. Es decir, una representación $ø(x) \in F$ es definida para cada posible objeto $x \in X$. El conjunto de datos *S* es, entonces, representado como el grupo de las representaciones individuales $ø(S)=(ø(x_1),..., ø(x_n))$ (Vert *et al*., 2004).

En cambio, en los métodos núcleo los datos no son representados individualmente, sino a través del conjunto de comparaciones de pares de datos.

En otras palabras, en lugar de usar $ø : X \rightarrow F$ para representar cada objeto $x \in X$ por $ø(x) \in F$, se utiliza una función de comparación con valor real $k : X \times X \rightarrow R$ y el conjunto de datos S es representado por la matriz $n \times n$ de comparaciones en pares $k_{i,j} = k(x_i, x_j)$ (Vert *et al*., 2004). En la Figura 7 se ejemplifican los dos tipos diferentes de representación. Todos los métodos núcleo son diseñados para procesar esas matrices cuadradas. A este respecto, es importante destacar algunas cuestiones, como:

1. La representación como matriz cuadrada no depende de la naturaleza de los datos a ser analizados. Pueden ser imágenes, moléculas o secuencias, y la representación de los datos es siempre una matriz cuadrada de valores reales.

2. El tamaño de la matriz empleada para representar el conjunto de *n* objetos es siempre de $n \times n$, independientemente de la naturaleza o complejidad de los datos.

3. Existen muchos casos en los que comparar objetos es más fácil que encontrar una representación explícita para cada objeto (Vert *et al*., 2004).

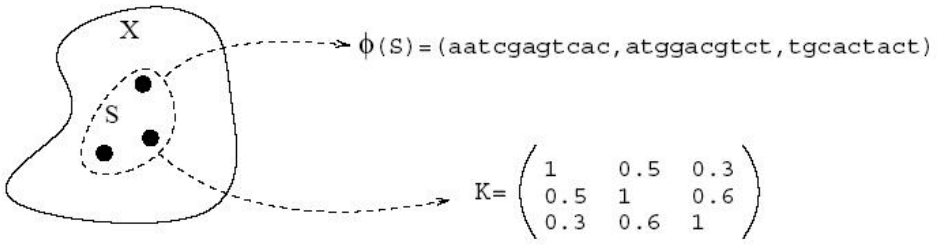

Figura 7: Dos representaciones diferentes de un mismo conjunto de datos: tradicional (parte superior) y kernel (matriz inferior). Obtenido de (Vert et al., 2004).

Los núcleos pueden ser vistos como productos internos en el espacio *F* (conocido como espacio de características). Usar un núcleo equivale a representar cada objeto $x \in X$ como un vector $ø(x) \in F$ y calcular productos internos. Pero la representación *ø(x)* no necesita ser calculada explícitamente para cada punto en el conjunto de datos *S*, sólo los productos internos por pares son necesarios. Cabe destacar que en muchos casos, calcular el producto interno en el espacio de características es mucho más fácil. Los núcleos son considerados como medidor de similitud, dado que *k (x, x')* es mayor cuando *x* y *x'* son más parecidos (Vert *et al*., 2004). La Figura 8 ilustra lo anterior.

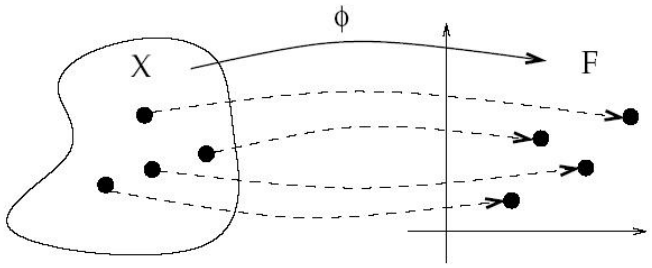

Figura 8: Núcleos y espacio de características. Cualquier kernel en un espacio *X* puede ser representado como un producto interno después de que el espacio *X* es transformado al espacio *F*, llamado espacio de características. Obtenido de (Vert et al., 2004).

Los métodos núcleo se basan en lo siguiente: cualquier algoritmo para datos vectoriales que puede ser expresado sólo en términos de productos internos entre vectores, puede trabajar implícitamente en el espacio de características asociado con un núcleo, reemplazando cada producto interno por una evaluación núcleo (Vert *et al*., 2004). Para entender a mayor profundidad esto, lo aplicaremos al problema de calcular distancias entre puntos. Sabemos que los núcleos pueden ser expresados como producto interno $k(x,x')=\langle ø(x),ø(x')\rangle$ en el espacio de características *F* para algún $ø : X \rightarrow F$. Tenemos dos objetos $x_1, x_2 \in X$. Estos puntos son trasladados a dos vectores $ø(x_1)$ y $ø(x_2)$ en *F*, por lo tanto, es natural definir la distancia $d(x_1, x_2)$ como la distancia de las imágenes:

$$d(x_1, x_2) := \| \phi(x_1) - \phi(x_2) \| \quad (1)$$

A primera vista, parece necesario calcular explícitamente las imágenes $\phi(x_1)$ y $\phi(x_2)$ antes de obtener la distancia. Sin embargo, como estamos en un espacio de producto interior, la distancia puede ser expresada en términos de productos internos en *F*:

$$\| \phi(x_1) - \phi(x_2) \|^2 = \langle \phi(x_1), \phi(x_2) \rangle + \langle \phi(x_1), \phi(x_2) \rangle - 2\langle \phi(x_1), \phi(x_2) \rangle \quad (2)$$

Ahora, aplicamos la representación del producto interno por el núcleo en (2) y el resultado en (1) mostrando que la distancia puede ser calculada sólo en términos del núcleo (3) (Vert *et al*., 2004). En la Figura 9 podemos ver la representación de estos conceptos.

$$d(x_1, x_2) = \sqrt{k(x_1, x_1) + k(x_2, x_2) - 2k(x_1, x_2)} \quad (3)$$

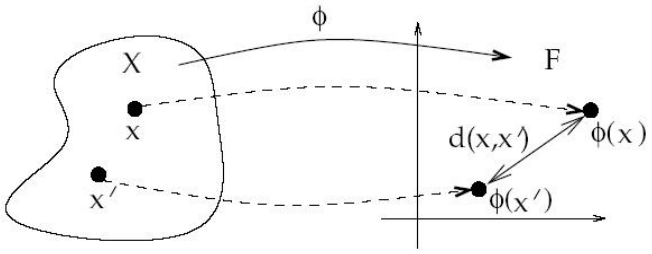

Figura 9: Representación producto interior del núcleo. Dado un espacio *X* y un núcleo, una distancia puede ser definida entre puntos de *X* trasladados al espacio de características *F* asociado con el núcleo. Esta distancia puede ser calculada sin conocimiento explícito de ø, gracias a la representación producto interior del núcleo. Obtenido de (Vert et al., 2004).

Debemos tener presente que los métodos núcleo tienen dos componentes que deben ser distinguidos: la máquina núcleo y la función núcleo. La máquina núcleo contiene la tarea de aprendizaje y la manera en que la solución será

buscada. Un ejemplo de máquina núcleo son las máquinas de vectores de soporte. En cambio, la función núcleo encapsula como puede construirse el grupo de soluciones. Nos referimos, generalmente, a las funciones núcleo simplemente como núcleos (kernels) (Gärtner, 2003).

III.2 Núcleos para cadenas (string kernels)

La familia de funciones núcleo definida sobre vectores en el espacio de características calculados para cadenas, se conoce como núcleos para cadenas o string kernels. Estos núcleos se basan en ocurrencias de ciertos tipos de subsecuencias en la cadena. Existe una gran variedad de núcleos para cadenas dependiendo de como se define la subsecuencia: puede ser contigua o no, con longitud limitada o ilimitada, las no coincidencias o espacios pueden ser penalizados de diferentes maneras (Rousu y Shawe-Taylor, 2005).

Para entrar en materia, es necesario conocer algunos conceptos básicos (Shawe-Taylor y Cristianini, 2004), como los siguientes:

- Un alfabeto es un conjunto ∑ finito de símbolos.
- Una cadena es una secuencia finita de símbolos del alfabeto incluyendo ε (cadena vacía).
- Una subcadena es contigua, mientras que las subsecuencias permiten espacios (gaps) entre los caracteres, aunque el orden de los mismos tiene suma importancia. Sin embargo, en ocasiones se utiliza el término subsecuencia para referirse a ambos conceptos, pero se aclara si existen espacios o no para diferenciar.
- Una cadena *t* es subcadena de *s*, si existen cadenas *u, v* tales que *s* = *utv*. Si *u* = *ε* se dice que *t* es un prefijo de *s*. Si *v* = *ε* se dice que *t* es un sufijo de *s*.
- Los *k*-gramas o *k*-meros son cadenas de la forma *s(i : j)* de longitud *k*.

- $u = s(i)$ denota que u es una subsecuencia de s en la posición dada por i.
- $l(i)$ es longitud de la subsecuencia ($i_{|u|} - i_l + 1$).
- Ejemplo: $\sum$ = { a, b, c, ..., z }; s=kernels; s(1:3)=ker; s(1,2,4,7)=kens; l(1,2,4,7)=7.

Los núcleos han sido desarrollados para calcular el producto interno entre imágenes de cadenas en espacios de características de alta dimensionalidad usando técnicas como la programación dinámica (Shawe-Taylor y Cristianini, 2004). Pueden ser definidos por un mapa embebido desde el espacio de todas las secuencias finitas sobre un alfabeto $\sum$ a un espacio de vectores F. Las coordenadas de F son indexadas por I (subconjunto I de cadenas sobre $\sum$), que es un subconjunto del espacio de entrada. En algunos casos I puede ser el conjunto $\sum^p$ de cadenas de longitud p dando un espacio de vectores de dimensión $|\sum|^p$, en otros puede ser el espacio de dimensiones infinitas indexado por $\sum^*$ (Shawe-Taylor y Cristianini, 2004).

Un núcleo es una función K tal que para todo x, z en X satisface: $k(x,z)=\langle ø(x),ø(z)\rangle$. Donde $ø$ representa, por ejemplo, cuántas veces ocurre u como subcadena o subsecuencia (con espacios) en s, o las coincidencias ponderando el número de errores (mismatches) (Shawe-Taylor y Cristianini, 2004).

Uno de los enfoques utilizados para calcular núcleos para cadenas eficientemente, es la programación dinámica; y se basa en componer la solución a partir de subproblemas, es decir, desde valores núcleo de subsecuencias más cortas y prefijos de las dos cadenas. La complejidad en tiempo suele ser Ω(p||s|||t||) dado que se necesitan calcular los resultados intermedios para cada par de caracteres s_i, t_i para cada subsecuencia de longitud $1 \leq l \leq p$ (Rousu y Shawe-Taylor, 2005). Hablamos de un costo cuadrático en la longitud de las secuencias (Shawe-Taylor y Cristianini, 2004).

Otro enfoque es el basado en el árbol trie (contracción de “retrieval tree”, es decir, árbol de recuperación), este consiste en hacer la toma de valor recorriendo una estructura trie implícita de manera transversal. Generalmente, la complejidad del tiempo suele ser lineal (Rousu y Shawe-Taylor, 2005). Se verá más sobre trie en la sección final de este capítulo. En el estudio realizado por (Rousu y Shawe-Taylor, 2005) sugieren que si el tamaño del alfabeto es pequeño (menor de 10) la programación dinámica es el mejor método; si el alfabeto es de tamaño mediano (menor de 1000), entonces el enfoque trie es competitivo si el número de espacios (gaps) es restringido.

En los siguientes módulos hablaremos un poco acerca de algunos núcleos para cadenas que existen. Pero, en nuestra investigación sólo trabajaremos con el *p*-spectrum.

III.3 Núcleo *p*-spectrum

Una de las definiciones de espectro es: distribución de la intensidad de una radiación en función de una magnitud característica, como la longitud de onda, la energía, la frecuencia o la masa. O la representación gráfica de cualquiera de estas distribuciones (RAE, 2007). Es posible que en el contexto en el que nos encontramos, dichas definiciones no apliquen textualmente, pero si nos permiten darnos una idea acerca del concepto de espectro.

La forma más natural de comparar dos cadenas es contar cuántas subcadenas contiguas de longitud *p* tienen en común (*p*-spectrum). Podemos así definir un núcleo como el producto interno de sus *p*-spectra. Comparando los *p*-spectra de dos cadenas puede obtenerse información importante acerca de su similitud en aplicaciones donde la contigüidad juega un rol importante (Shawe-Taylor y Cristianini, 2004).

El espacio de características *F* asociado con el *p*-spectrum está dado por (Shawe-Taylor y Cristianini, 2004):

$$\phi_u^p(s) = \left|\{(v_1, v_2) : s = v_1 u v_2\}\right|, u \in \sum^p \quad (4)$$

El núcleo asociado se define como (Shawe-Taylor y Cristianini, 2004):

$$k_p(s,t) = \left\langle \phi^p(s), \phi^p(t) \right\rangle = \sum_{u \in \Sigma^p} \phi_u^p(s) \phi_u^p(t) \quad (5)$$

Ejemplo: Para calcular un núcleo 2-spectrum sobre las cadenas “bar”, “bat”, “car” y “cat”. Obtenemos la siguiente tabla (Shawe-Taylor y Cristianini, 2004):

Tabla I: Ejemplo de núcleo 2-spectrum, donde se tienen los valores de ø para las cadenas en cuestión.

ø	ar	at	ba	ca
bar	1	0	1	0
bat	0	1	1	0
car	1	0	0	1
cat	0	1	0	1

Además, de otras dimensiones indexadas por otras cadenas de longitud 2 que tienen valor 0. La matriz núcleo es (Tabla II):

Tabla II: Matriz núcleo para 2-spectrum.

K	bar	bat	car	cat
bar	2	1	1	0
bat	1	2	0	1
car	1	0	2	1
cat	0	1	1	2

Para obtener los valores de la matriz núcleo debemos usar su definición, obteniendo los productos internos. Por ejemplo: para calcular el valor de ⟨ø(bar)·ø(bar)⟩, obtenemos su producto interior:

$[1\ 0\ 1\ 0] \cdot [1\ 0\ 1\ 0]^T = 2$

La complejidad del núcleo *p*-spectrum es *O(p* |*s*| |*t*|*)* (Shawe-Taylor y Cristianini, 2004).

El código en Matlab del núcleo *p*-spectrum se encuentra disponible en forma gratuita en la página web http://www.kernel-methods.net/matlab_tools.html que corresponde al sitio del libro “Kernel Methods for Pattern Analysis” (Shawe-Taylor y Cristianini, 2004),

III.4 Núcleo all-subsequences

Para este núcleo todas las subsecuencias, *ø* cuenta el número de veces que la cadena *u* ocurre como una subsecuencia (no contigua) en la cadena *s*. El espacio de características *F* está dado por (Shawe-Taylor y Cristianini, 2004):

$$\phi_u(s) = \left|\{i : u = s(i)\}\right|, u \in I \tag{6}$$

El núcleo all-subsequences está definido por (Shawe-Taylor y Cristianini, 2004):

$$k(s,t) = \left\langle \phi(s), \phi(t) \right\rangle = \sum_{u \in \Sigma^*} \phi_u(s)\phi_u(t) \tag{7}$$

Por ejemplo: tenemos las cadenas “bar”, “baa”, “car” y “cat” (Shawe-Taylor y Cristianini, 2004). Ver Tabla III.

Tabla III: Ejemplo de núcleo all-subsequences, donde se tienen los valores de ø para las cadenas en cuestión.

ø	ε	a	b	c	r	t	aa	ar	at	ba	br	bt	ca	cr	ct	bar	baa	car	cat
bar	1	1	1	0	1	0	0	1	0	1	1	0	0	0	0	1	0	0	0
baa	1	2	1	0	0	0	1	0	0	2	0	0	0	0	0	0	1	0	0
car	1	1	0	1	1	0	0	1	0	0	0	0	1	1	0	0	0	1	0
cat	1	1	0	1	0	1	0	0	1	0	0	0	1	0	1	0	0	0	1

La matriz núcleo sería (Tabla IV):

Tabla IV: Matriz núcleo para all-subsequences.

K	bar	baa	car	cat
bar	8	6	4	2
baa	6	12	3	3
car	4	3	8	4
cat	2	3	4	8

La programación dinámica resuelve problemas combinando las soluciones de subproblemas. Esta técnica evita explorar todas las posibilidades por medio de la resolución de subproblemas de tamaño creciente y almacenamiento en una tabla de las soluciones óptimas de esos subproblemas (Cormen *et al*., 2001). Si usamos programación dinámica para calcular un núcleo all-subsequences, debemos construir una tabla con los valores previos (ver Tabla V), el núcleo de las secuencias *s, t* es *k(s,t)= DP (m, n)* y la complejidad es *O(*|s| |*t*|*)* (Shawe-Taylor y Cristianini, 2004).

Tabla V: Matriz núcleo all-subsequences con programación dinámica.

DP	ε	t_1	t_2	...	t_m
ε	1	1	1	...	1
s_1	1	k_{11}	k_{12}	...	k_{1m}
s_2	1	k_{21}	k_{22}	...	k_{2m}
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
s_n	1	k_{n1}	k_{n2}	...	$k_{m\,n}$

Por ejemplo, si *s* =“cata” y *t* =“gatta”, tendríamos la Tabla VI (Shawe-Taylor y Cristianini, 2004):

Tabla VI: Ejemplo de matriz núcleo all-subsequences con programación dinámica.

DP	ε	g	a	t	t	a
ε	1	1	1	1	1	1
c	1	1	1	1	1	1
a	1	1	2	2	2	3
t	1	1	2	4	6	7
a	1	1	3	5	7	14

Por lo tanto, *k(s, t)* = 14. El algoritmo para resolver este ejemplo y demás núcleos all-subsequences es el siguiente (Shawe-Taylor y Cristianini, 2004):

Entrada	cadenas *s* y *t* de longitud *n* y *m*
1	for j = 1: *m*
2	DP(0, j)=1;
3	end
4	for i = 1:*n*
5	last = 0; P(0) = 0;
6	for k = 1:m
7	P(k) = P(last);
8	if $t_k = s_i$ then
9	P(k) = P(last) + DP (i-1, k-1)
10	last = k;
11	end
12	end
13	for k = 1:*m*
14	DP(i,k) = DP(i-1, k)+ P(k);
15	end
16	end
Salida	*k(s, t)* = DP *(n, m)*

III.5 Núcleo fixed length subsequences

El núcleo fixed length subsequences o núcleo de subsecuencias con longitud fija, es una adaptación al all-subsequences que reduce la dimensionalidad del espacio de características considerando sólo subsecuencias no contiguas de longitud fija usando la variable *p*. El espacio de características *F* y el núcleo están dados por (Shawe-Taylor y Cristianini, 2004):

$$\phi_u^p(s) = \left|\{i : u = s(i)\}\right|, u \in \Sigma^p \tag{8}$$

$$k_p(s,t) = \left\langle \phi^p(s), \phi^p(t) \right\rangle = \sum_{u \in \Sigma^p} \phi_u^p(s)\phi_u^p(t) \tag{9}$$

El algoritmo para el núcleo fixed length subsequences sería el siguiente y su complejidad es *O(p |s| |t|)* (Shawe-Taylor y Cristianini, 2004):

Entrada	cadenas *s* y *t* de longitud *n* y *m*, longitud *p* de las subsecuencias
1	DP(0:*n*, 0:*m*)=1;
2	for *l*=1:*p*
3	DPrec = DP;
4	for j=1:*m*
5	DP(0,j)=1;
6	end
7	for i = 1:*n*-*p*+l
8	last = 0; P(0) = 0;
9	for k = 1:*m*
10	P(k) = P(last);
11	if $t_k = s_i$ then
12	P(k) = P(last) + DPrec (i-1, k-1)
13	last = k;
14	end
15	end
16	for k = 1:*m*
17	DP(i,k) = DP(i-1, k)+ P(k);
18	end
19	end
20	end
Salida	$k_p(s, t)$ = DP *(n, m)*

III.6 Núcleo gap-weighted subsequences

En éste se toman en cuenta los espacios o gaps presentes en las cadenas. La idea es todavía comparar cadenas mediante las subsecuencias que contienen (más subsecuencias en común, más similares son); sin embargo, como se ponderan todas las subsecuencias igual, el grado de contigüidad de las subsecuencias en la cadena de entrada *s* determina el grado de contribución. Por ejemplo: “gon” es una subsecuencia de las cadenas “gone”, “going” y “galleon”, la de mayor similitud es la primera, y la de menor es la tercera. El espacio de características *F* y su núcleo están dados por (Shawe-Taylor y Cristianini, 2004):

$$\phi_u^p(s) = \sum_{i:u=s(i)} \lambda^{l(i)}, u \in \Sigma^p \tag{10}$$

$$k_p(s,t) = \left\langle \phi^p(s), \phi^p(t) \right\rangle = \sum_{u \in \Sigma^p} \phi_u^p(s) \phi_u^p(t) \tag{11}$$

Donde $l(i) = (i_{|u|} - i_1 + 1)$ y λ es un factor de decaimiento $\lambda \in (0, 1)$, si $\lambda = 0$ maneja subcadenas (contiguas) y si $\lambda = 1$ maneja subsecuencias (no contiguas). El algoritmo que calcula el núcleo gap-weighted subsequences se muestra a continuación, y su complejidad es $O(p\,|s|\,|t|)$ (Shawe-Taylor y Cristianini, 2004):

Entrada	cadenas *s* y *t* de longitud *n* y *m*, longitud *p* de las subsecuencias, parámetro λ
1	DPS(1:*n*, 1:*m*)=0;
2	for i =1:*n*
3	for j = 1:*m*
4	if $s_i = t_j$ then
5	DPS(i,j)= λ^2;
6	end
7	end
8	end
9	DP(0,0:*m*)=0;
10	DP(1:*n*,0)=0;
11	for *l* = 2:*p*
12	Kern(*l*) = 0;
13	for i = 1:*n*-1
14	for j = 1:*m*-1
15	DP(i,j)= DPS(i,j)+ λDP(i-1,j)+
16	λDP(i,j-1)- λ^2 DP(i-1,j-1);
17	if $s_i = t_j$ then
18	DPS(i,j)=λ^2DP(i-1,j-1);
19	Kern(*l*)=Kern(*l*)+ DPS(i,j);
20	end
21	end
22	end
23	end
Salida	$k_p(s, t)$ = Kern*(p)*

III.7 Núcleo soft matching string

El núcleo soft matching string se aplica cuando las cadenas contienen símbolos distintos, pero que de alguna forma estén relacionados. Para lo anterior, se supone que se ha dado una matriz A de similitud entre los símbolos. Su implementación requerirá modificar el algoritmo del núcleo gap-weighted subsequences en sus líneas 4, 5, 16 y 17 como se muestra a continuación (Shawe-Taylor y Cristianini, 2004):

4	if A(s_i, t_j) $\neq$ 0
5	DPS(i,j)= λ^2A(s_i, t_j);
16	if A(s_i, t_j) $\neq$0
17	DPS(i,j)= λ^2A(s_i, t_j)DP(i-1,j-1);

III.8 Núcleo *p*-spectrum basado en trie

Un trie sobre un alfabeto ∑ es un árbol cuyos nodos internos tienen sus hijos indexados por ∑. Las aristas que conectan un padre con sus hijos tienen etiquetas con el símbolo correspondiente de ∑. El nombre de trie proviene de la contracción de “retrieval tree” (árbol de recuperación). En un trie completo existen una correspondencia 1-1 entre los nodos a la profundidad k y las cadenas de longitud k. La cadena asociada con el nodo raíz es el caracter nulo o vacío ε (Shawe-Taylor y Cristianini, 2004). Por ejemplo: un trie en el que se encuentran almacenadas las palabras “atun”, “atar”, “atado”, “pio”, “pato” y “pava”, ver Figura 10.

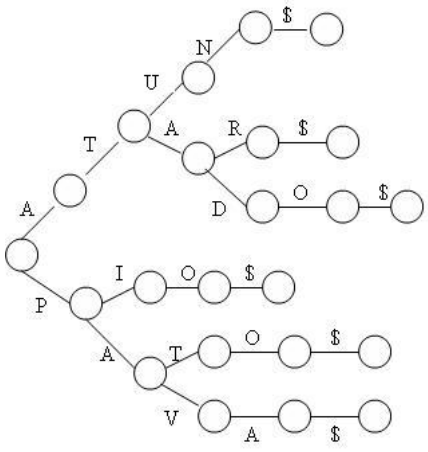

Figura 10: Ejemplo de estructura de datos trie.

En el núcleo *p*-spectrum basado en trie se tiene una complejidad de *O(p(|s|+|t|))* y su algoritmo es el siguiente (Shawe-Taylor y Cristianini, 2004):

Entrada	cadenas *s* y *t*, parametro *p*
1	Sea $L_s(\varepsilon) = \{(s(i : i + p - 1), 0) : i = 1 : \|s\| - p + 1\}$
2	Sea $L_t(\varepsilon) = \{(t(i : i + p - 1), 0) : i = 1 : \|t\| - p + 1\}$
3	Kern=0;
4	processnode(ε,0)
donde	processnode (v,depth)
6	Sean $L_s(v)$, $L_t(v)$ las listas asociadas con v;
7	If depth=p
8	Kern= Kern + $\|L_s(v)\|\|L_t(v)\|$;
9	end
10	Else if $L_s(v)$ y $L_t(v)$ están no vacías
11	while existan (u,i) en la lista $L_s(v)$
12	sumar (u,i+1) a la lista $L_s(vu_{i+1})$;
13	end
14	while existan (u,i) en la lista $L_t(v)$
15	sumar (u,i+1) a la lista $L_t(vu_{i+1})$;
16	end
17	for $\alpha \in \Sigma$
18	processnode(vα, depth+1);
19	end
20	end
Salida	$K_p(s, t)$ = Kern

El código en Matlab del núcleo *p*-spectrum basado en trie se encuentra disponible en forma gratuita en la página web del libro “Kernel Methods for Pattern Analysis” (Shawe-Taylor y Cristianini, 2004), que es http://www.kernel-methods.net/matlab_tools.html.

El núcleo *p*-spectrum basado en trie obtiene los mismos resultados que el *p*-spectrum tradicional, pero su tiempo de ejecución es mucho menor. No obstante, el funcionamiento en sí es el mismo, variando sólo la forma de implementación.

Capítulo IV

Máquinas de vectores de soporte

IV.1 Introducción a las máquinas de vectores de soporte

Este capítulo versará sobre las máquinas de vectores de soporte o SVM (Support Vector Machines, en inglés); sin embargo, no se pretende abordar exhaustivamente el tema, ya que ciertos detalles no son estrictamente necesarios para utilizar un programa que ejecute una SVM. Las máquinas de vectores de soporte son una pieza clave de la presente investigación al utilizarse junto con ciertos núcleos de cadenas para los experimentos realizados. Existen diversas herramientas que llevan a cabo las funciones de una máquina de vectores de soporte, entre ellas destaca LIBSVM (Chang y Lin, 2001) por su sencillez y flexibilidad, además es utilizada en algunos estudios previos como en (Zaki *et al*., 2006). Ahondaremos en LIBSVM en el Apéndice A.

Los métodos de clasificación (también conocidos como métodos de aprendizaje de máquina, aprendizaje automático o “Machine Learning”) (Shoemaker y Panchenko, 2007) utilizados para la predicción de la posible interacción entre un par de proteínas, entre otros muchos problemas bioinformáticos, pueden dividirse en dos grupos: enfoque supervisado y no supervisado. El aprendizaje supervisado es cuando a un algoritmo de clasificación se le proporciona un conjunto de ejemplos de datos con la clase a la que pertenecen y se prueba en un conjunto de datos cuyas clases desconocemos (Bobadilla *et al*., 2003).

Los métodos no supervisados no reciben información sobre la clase a la que pertenecen los datos, entonces deben encontrar relaciones entre ellos de manera automática. Un ejemplo de un algoritmo de aprendizaje no supervisado para determinar la interacción entre proteínas, es el propuesto por Mamitsuka. Dicho enfoque se basa en un algoritmo EM (Expectation Maximization) (Mamitsuka, 2005).

Dentro de los clasificadores supervisados, podemos mencionar los árboles de decisión (Decision Trees), el clasificador Bayesiano simple, los bosques aleatorios de decisión (Random Forest Decision) y métodos de kernel o núcleo como las máquinas de vectores de soporte (Kohutyuk y Honavar, 2006).

Un método núcleo representa una colección de datos complejos usando una función núcleo que define la similitud entre cualquier par de datos u objetos dados. Las máquinas de vectores de soporte son el método de núcleo más popular, dado que tiene un desempeño robusto y permite el tratamiento de datos no heterogéneos como los existentes en las bases de datos biológicas (Lewis *et al*., 2006).

En el siguiente módulo, se presenta la descripción de las máquinas de vectores de soporte y algunos de sus fundamentos más importantes.

IV.2 Máquinas de vectores de soporte

Las máquinas de vectores de soporte, o SVM constituyen un método de aprendizaje supervisado, en el contexto del reconocimiento de patrones y el aprendizaje automático (Lewis *et al*., 2006).

Las máquinas de vectores de soporte son un algoritmo de aprendizaje estadístico que representa los datos en un espacio vectorial y considera que es

razonable que los elementos de una misma clase estén cercanos y los que pertenezcan a diferentes clases se localicen en regiones diferentes del espacio de representación (Cortijo Bon, 2001).

Las máquinas de vectores de soporte están caracterizadas por el uso de núcleos o kernels, la ausencia de mínimos locales, entre otros. Fueron inventadas por Vladimir Vapnik (Vapnik, 1995) y colaboradores. Aunque se dieron a conocer en 1992 como tal, su fundamento ya se conocía desde la década de los sesenta en aprendizaje automático. El uso de núcleos fue propuesto por Wahba, Poggio, y otros en 1964, introduciendo la interpretación geométrica de los núcleos como productos interiores en el espacio de características (Cristianini y Shawe-Taylor, 2000). En fechas más recientes, la aplicación de las máquinas de vectores de soporte a la resolución de problemas biológicos se ha incrementado significativamente, para una revisión del estado del arte a este respecto ver (Noble, 2004), (Cristianini y Shawe-Taylor, 2000) y (Vert *et al*., 2004).

Estas máquinas fueron ideadas en un principio para la resolución de clasificación binaria con clases linealmente separables. También se les conocía como “hiperplano óptimo de decisión” debido a que la solución es aquella que clasifica correctamente las muestras, colocando el hiperplano[6] de separación lo más lejos posible de todas ellas. A las muestras más cercanas al hiperplano óptimo se les conoce como “vectores soporte”, de ahí su nombre. El hiperplano es buscado en un espacio de decisión llamado “espacio de características”, al que previamente se han transformado los datos. Para realizar dicha transformación sólo es necesario conocer su núcleo reproductor o kernel, con lo que se simplifica la obtención de funciones de decisión no lineales (Sánchez-Vítores, 2004). El

[6]El hiperplano es un concepto geométrico que, en términos globales, es una generalización del plano. En un espacio bidimensional R^2 un hiperplano es una recta. En un espacio tridimensional R^3, un hiperplano es un plano corriente, divide el espacio en dos mitades (Poole, 2006).

funcionamiento de las SVM, grosso modo, consiste en la transición del problema original a un plano de mayor dimensionalidad mediante funciones núcleos, para tener mayores probabilidades de que las clases sean linealmente separables en el espacio de características (Bobadilla *et al*., 2003).

Mucho del poder de estas máquinas viene de su criterio para seleccionar el plano de separación cuando hay varios planos candidatos: la máquina de vectores de soporte elegirá el plano que mantenga un margen máximo entre los puntos de entrenamiento (Zaki *et al*., 2006). El problema de dicha elección puede ser formulada y resuelta mediante programación cuadrática (Lewis *et al*., 2006). Ver Figura 11.

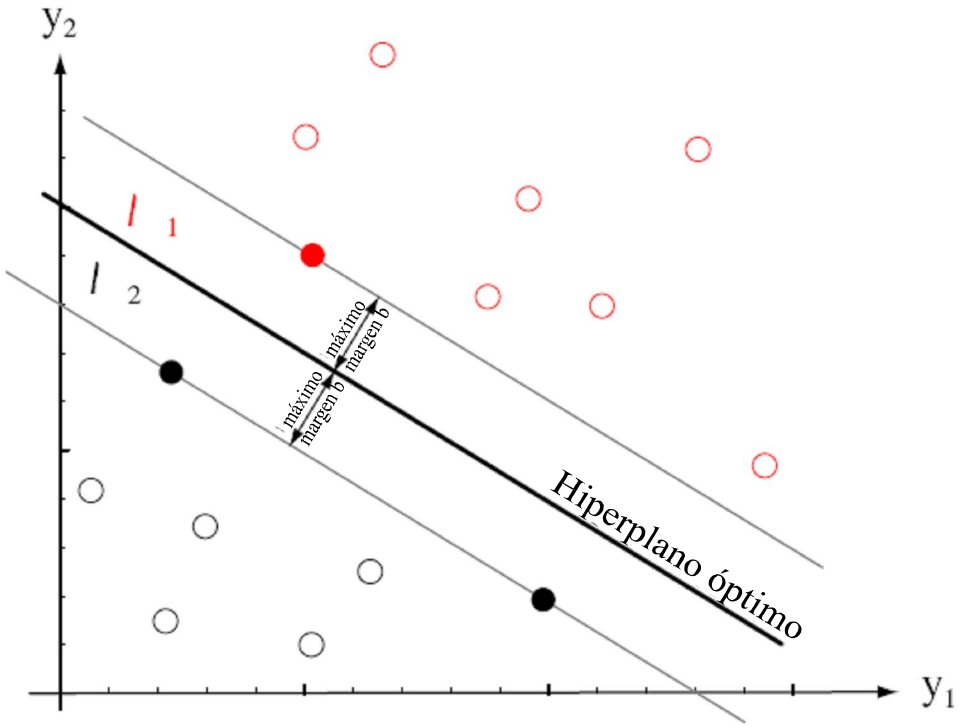

Figura 11: Hiperplano óptimo. Entrenar una máquina de vectores de soporte consiste en encontrar el hiperplano óptimo, es decir, el que maximice la distancia entre los patrones más cercanos. Los vectores de soporte están representados por los círculos rellenos. Obtenida de (Duda et al, 2000).

Un beneficio destacable de SVM versus otros métodos es que la complejidad del clasificador, obtenido mediante este enfoque, es caracterizado por

el número de vectores de soporte independientemente de la dimensionalidad del espacio de transformación (Duda *et al*., 2000).

Aplicar una máquina de vectores de soporte a un problema de clasificación consta de dos fases: entrenamiento y predicción o prueba. En el entrenamiento, el conjunto de datos de entrada debe tener asociado una etiqueta binaria, como +1 o 1 para la clase positiva y -1 o 0 para la negativa. El algoritmo de entrenamiento buscará un hiperplano que separe los ejemplos positivos de los negativos. Después, viene la etapa de la predicción, donde la máquina de vectores de soporte tratará de establecer cuáles etiquetas corresponden a los datos del nuevo conjunto de entrada, es decir, determinar en qué lado del hiperplano de separación caen los datos (Lewis *et al*., 2006).

Supóngase que el conjunto de entrenamiento *S* consiste de los vectores de entrada etiquetados (x_i, y_i), $i = 1, \ldots, m$, donde $x_i \in \Re^n$ y $y_i \in \{\pm 1\}$. Se puede especificar una regla de clasificación *f* para un par (w, b), donde $w \in \Re^n$ y $b \in \Re$ mediante (Leslie *et al*., 2002):

$$f(x) = \langle w, x \rangle + b, \tag{12}$$

donde un punto *x* es clasificado como positivo si $f(x) > 0$ y negativo si $f(x) < 0$. El hiperplano de separación está dado por (Leslie *et al*., 2002):

$$\{x \in \Re^n : \langle w, x \rangle + b = 0\}, \tag{13}$$

donde *w* es un vector normal al hiperplano y *b* es el parámetro de desplazamiento. Ver Figura 12.

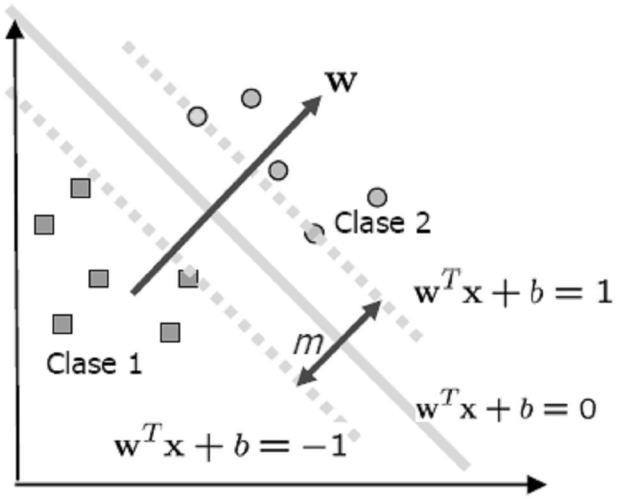

Figura 12: Separación de clases. Se puede observar la separación de las clases mediante el hiperplano óptimo, procurando maximizar la distancia entre los vectores de soporte. Obtenida de (Betancourt, 2005).

Un punto clave de las máquinas de vectores de soporte es que la optimización se simplifica resolviendo el problema dual de optimización minimizando $\frac{\|w^2\|}{2}$, sujeto a $y(w^T x + b) \geq 1$, la solución a (13) se encuentra resolviendo este problema. Para el caso linealmente separable, el clasificador de máximo margen es encontrado mediante la solución de los "pesos" óptimos α_i, i=1,…, m, en el problema dual (Leslie *et al*., 2002):

$$\begin{aligned} &Maximizar \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \\ &región : \alpha_i \geq 0 \forall i \end{aligned} \tag{14}$$

Entonces, los parámetros *(w, b)* del clasificador son determinados por los óptimos α_i, los datos de entrenamiento y la función resuelta $f(x) = \sum_i \alpha_i \langle x_i, x \rangle + b$.

El problema dual no sólo hace a la SVM propicia para varios algoritmos eficientes de optimización, sino que como depende sólo de los productos internos $\langle x_i, x_j \rangle$, permite la introducción de núcleos o kernels. Para ello, suponemos que los datos de entrenamiento son ejemplos etiquetados *(x_i, y_i)*, donde x_i pertenece a un espacio de entrada *X,* el cual puede ser un espacio vectorial o un espacio de estructuras discretas como secuencias de caracteres de un alfabeto. Dado cualquier mapeo de características Φ de *X* a un espacio vectorial llamado el espacio de características, tenemos (Leslie *et al*., 2002):

$$\Phi : X \to \Re^N \tag{15}$$

Obtenemos un núcleo *K* en *X* x *X* definido por:

$$K(x, y) = \langle \Phi(x), \Phi(y) \rangle \tag{16}$$

Reemplazando $\langle x_i, x_j \rangle$ por *K(x_i, y_i)* en el problema dual, podemos usar la máquina de vectores de soporte en el espacio de características. Calculando, incluso, directamente los valores del núcleo sin conocer explícitamente los vectores de características (Leslie *et al*., 2002).

Existen varias funciones kernel o núcleo, destacando las siguientes cuatro, consideradas básicas (Hsu *et al*., 2007):

- Lineal: $K(x_i, x_j) = x_i^T x_j$
- Polinomial: $K(x_i, x_j) = (\gamma x_i^T x_j + r)^d, \gamma > 0$
- Función Base Radial (RBF): $K(x_i, x_j) = \exp\left(-\gamma \left\| x_i - x_j \right\|^2\right), \gamma > 0$
- Sigmoidal: $K(x_i, x_j) = \tanh(\gamma x_i^T x_j + r)$

Donde γ, r y d son parámetros de los núcleos.

En el caso de la presente investigación, se construyó un kernel de pares para núcleos de cadena, que fue explicado en el capítulo anterior. Para mayor detalle sobre los cuatro núcleos enlistados anteriormente, consultar (Hsu *et al*., 2007) y (Vert *et al*., 2004).

Según (Noble, 2006), la esencia de las máquinas de vectores de soporte puede ser entendida sin el uso de fórmulas, para lo cual propone el conocimiento de cuatro conceptos básicos: el hiperplano de separación, el hiperplano óptimo, el margen suave y la función kernel o núcleo. El término general para referirse a una línea en un espacio de alta dimensionalidad es el hiperplano y el hiperplano de separación es, obviamente, aquel que permite la división de las clases de datos. Ahora bien, el hiperplano óptimo es el que maximice la distancia entre los datos de diferentes clases. En ocasiones, los datos no pueden ser separados con una simple línea recta, por lo que el algoritmo de la máquina de vectores de soporte debe ser modificado para permitir el manejo de márgenes suaves, es decir, tolerar que algunos puntos de datos se localicen en el lado erróneo del hiperplano, ver Figura 13. La función núcleo o kernel proyecta los datos a un espacio de mayor dimensionalidad, conocido como espacio de características en búsqueda del hiperplano óptimo suponiendo que en este espacio, los datos son linealmente separables, ver Figura 14 (Noble, 2006).

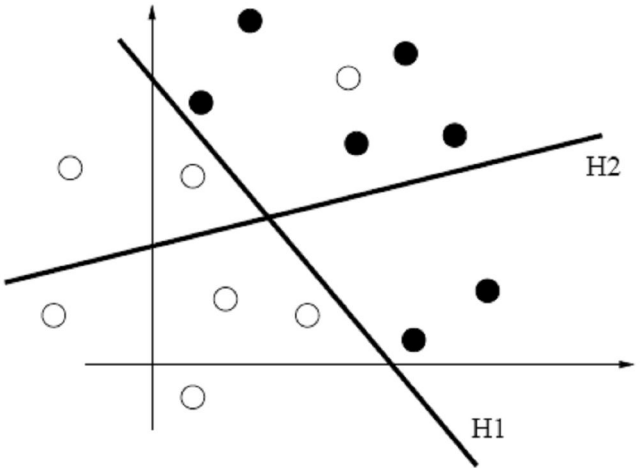

Figura 13: Discriminación de los hiperplanos. El hiperplano H1 discrimina los círculos negros de los blancos con un error. El hiperplano H2 separa estos puntos con cinco errores. El principio de la minimización del riesgo empírico indica que se debe elegir el hiperplano con menos errores posibles, en este caso H1 (suavización del margen). Obtenida de (Vert et al., 2004).

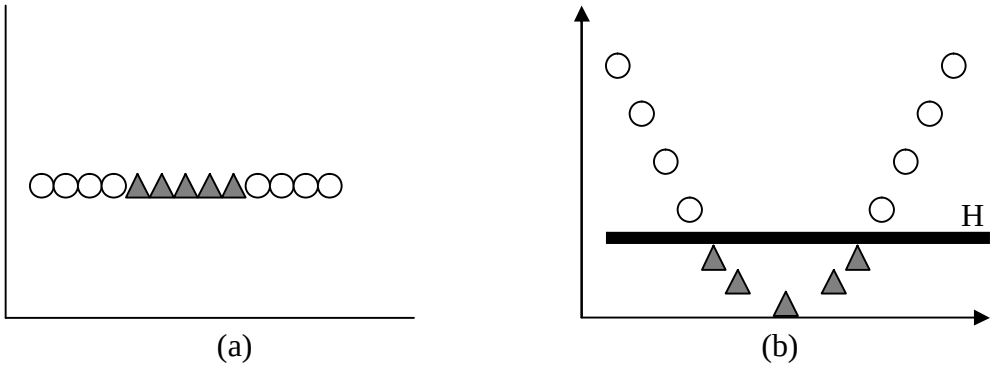

Figura 14: Separación de un conjunto de datos en dos dimensiones mediante un hiperplano H, que en una dimensión no era posible. (a) Grupo de datos unidimensionales no separables. (b) datos separados linealmente. Inspirada en (Noble, 2006).

En términos generales, estas serían las bases del funcionamiento de las máquinas de vectores de soporte. Para mayor detalle, sobre todo, la cuestión matemática, remitirse a (Cristianini y Shawe-Taylor, 2000) y (Vert *et al*., 2004). En el Apéndice A se describe brevemente el programa en específico que lleva a cabo las funciones de la máquina de vectores de soporte utilizada en este estudio.

Capítulo V

Experimentos y análisis de resultados

V.1 Experimentos

En nuestro estudio tomaremos la forma de combinar núcleos para pares de proteínas de (Ben-Hur y Noble, 2005), pero usando sólo el núcleo de cadena (string kernel) denominado *p*-spectrum (Shawe-Taylor y Cristianini, 2004) y una SVM. Variando *p*, analizaremos qué valor de este aporta la mayor precisión en la predicción de interacciones proteína-proteína basados sólo en estructura primaria.

Para los experimentos se utilizó la implementación del núcleo para cadenas *p*-spectrum de Shawe-Taylor y Cristianini (2004) disponible en: http://www.kernel-methods.net/matlab_tools.html. Para mayor información ver el Capítulo III sobre métodos núcleo.

El núcleo de pares (Pairwise Kernel: Kp) de (Ben-Hur y Noble, 2005), se expresa de la manera siguiente:

$$K\left((X_1, X_2),\left(X_1', X_2'\right)\right) = K'\left(X_1, X_1'\right)K'\left(X_2, X_2'\right) + K'\left(X_1, X_2'\right)K'\left(X_2, X_1'\right) \quad (17)$$

Donde *K* corresponde a Kp o núcleo de pares, que mide la similitud entre los pares (X_1, X_2) y (X_1', X_2'). Al núcleo que opera sobre genes o proteínas individuales, se le llama “núcleo genómico” o “genomic kernel”; y al que compara pares de genes o proteínas “núcleo de pares” o “pairwise kernel” (Ben-Hur y Noble, 2005).

El conjunto de prueba es un subconjunto de los datos de BIND[7] de interacción física en levadura (Ben-Hur y Noble, 2005).

Dado que el método de clasificación de los datos es aprendizaje de máquina supervisado, en específico, máquina de vectores de soporte, se hace necesario el contar con dos grupos de datos: uno con un conjunto de pares de proteínas que interactúen (ejemplos positivos) y otro conjunto de pares de proteínas que se considere no interactúan (ejemplos negativos). Ambos, deben ser elegidos con cuidado en beneficio de la exactitud en nuestras predicciones. Pero, los ejemplos negativos requieren especial atención, ya que no existe un lineamiento formal para elegirlos, aunque dos métodos han prevalecido en la literatura. Algunos autores sugieren que ejemplos negativos de alta calidad pueden ser generados considerando pares de proteínas cuya localización celular es diferente, para prevenir que esas proteínas participen en una interacción biológicamente relevante. Otros autores usan un esquema más sencillo, seleccionan pares de forma aleatoria del conjunto de todos los pares de proteínas cuya interacción se desconoce (Ben-Hur y Noble, 2006).

Aparentemente, el primer enfoque resultaría mejor, pero como se demostró en (Ben-Hur y Noble, 2006) esto no es necesariamente cierto. Restringir ejemplos negativos a proteínas no co-localizadas puede llegar a mermar la exactitud en la predicción de interacciones proteína-proteína, ya que se puede inducir una distribución diferente de las características que son usadas para la clasificación. Por lo tanto, la distribución resultante podría ser demasiado optimista en la estimación de la exactitud de clasificación (Ben-Hur y Noble, 2006).

En cambio, una visión más simple (elegir ejemplos negativos aleatorios uniformes) tiende a tener una distribución más real. Aunque, como las redes de

[7] BIND (Biomolecular Interaction Network Database)
http://bond.unleashedinformatics.com/Action?pg=23299.

interacción no están completas, el grupo de ejemplos negativos puede ser contaminado con pares de proteínas que sí interactúan. Sin embargo, dicha contaminación suele ser muy pequeña y como, también, se demostró en el estudio antes mencionado, las máquinas de vectores de soporte son robustas a grados, incluso mayores, de contaminación (Ben-Hur y Noble, 2006).

Por lo anterior, el conjunto de datos que se decidió utilizar para nuestro estudio, se basa en la selección aleatoria uniforme de pares de proteínas cuyas interacciones desconocemos para los ejemplos negativos, utilizando los datos del artículo (Ben-Hur y Noble, 2005) tanto para los ejemplos positivos como negativos[8]. Se seleccionaron los primeros 100 pares de proteínas para cada tipo de ejemplo (200 pares, 366 proteínas diferentes) de un conjunto de datos de 10 517 interacciones físicas en levadura y 10 517 ejemplos negativos extraídos de la base de datos BIND, que involucran a 4233 proteínas diferentes.

Las longitudes de las secuencias del conjunto de datos varían desde 77 aminoácidos, la más pequeña, hasta 2748 aminoácidos, la más larga. A continuación mostramos un extracto (primeros 10 pares de ejemplos positivos y primeros 10 pares de ejemplos negativos) de la tabla que contiene las longitudes de todas las secuencias de nuestro conjunto, que puede consultarse completa en el Apéndice B.

[8] Estos datos se encuentran disponibles en la página Web: http://noble.gs.washington.edu/proj/sppi/.

Tabla VII: Extracto del conjunto de datos empleado en esta investigación con sus respectivas longitudes de secuencias, agrupados por pares de proteínas de acuerdo a su interacción.

# par	Longitud proteína 1	Longitud proteína 2	Interacción	# par	Longitud proteína 1	Longitud proteína 2	Interacción
1	274	345	Sí	101	433	341	No
2	685	560	Sí	102	304	207	No
3	612	156	Sí	103	733	726	No
4	175	279	Sí	104	150	604	No
5	310	113	Sí	105	334	669	No
6	610	1091	Sí	106	328	119	No
7	591	901	Sí	107	287	683	No
8	352	248	Sí	108	1755	404	No
9	499	370	Sí	109	1391	884	No
10	368	552	Sí	110	577	341	No

Una vez seleccionado el conjunto de datos con el que se trabajará, se procedió a crear un programa en python para extraer de la base de datos SwissProt[9] las secuencias y darle formato de manera automática.

Una de las problemáticas que se resolvieron, fue que en el conjunto de datos original de (Ben-Hur y Noble, 2005) los pares de proteínas están dados en OLN (OrderedLocusName), que es una abreviación informalmente asignada en el proceso de designar un gen. Y con sólo el OLN no es posible extraer la secuencia de la base de datos SwissProt, por lo que previamente fue necesario establecer una equivalencia del OLN con el AC (Accession number). Para detalles ver Apéndice C.

Las secuencias extraídas constan de caracteres que representan aminoácidos de acuerdo al estándar IUPAC[10] con código de una letra. Para ver la

[9] Swiss-Prot, Protein knowledgebase. URL: http://www.expasy.org/sprot/

[10] IUPAC-IUB Joint Commission on Biochemical Nomenclature (JCBN). Nomenclature and Symbolism for Amino Acids and Peptides. Recommendations 1983. Eur. J. Biochem. 138:9-37(1984).

tabla con los caracteres y su correspondencia en aminoácidos, consultar el Apéndice D.

Posteriormente, se creó un programa en Matlab que calcula la matriz del núcleo de pares (Pairwise Kernel: Kp (Ben-Hur y Noble, 2005)) con el núcleo *p*-spectrum, para *p*=2, 3, 4, 5, 6, 7 y 8. Y se ejecutaron con las secuencias obtenidas para las 400 proteínas. Cabe señalar, que esta sección es la que mayor tiempo consume dada la longitud de las cadenas de aminoácidos de las proteínas y la naturaleza del núcleo *p*-spectrum. Por ejemplo, para calcular la matriz Kp de 200 x 200 elementos del núcleo *p*-spectrum para *p*=8 en un equipo de cómputo promedio, tarda poco menos de dos semanas, ya que para calcular cada uno de los elementos de dicha matriz es necesario ejecutar la fórmula (17), que implica obtener el núcleo *p*-spectrum cuatro veces (comparando bloques continuos de tamaño *p*) y las longitudes de las secuencias pueden ser considerables. Sin embargo, ya con la experiencia que obtuvimos al realizar los experimentos, podemos sugerir ciertas medidas que pueden disminuir el tiempo de cálculo, como: fraccionar el cálculo de la matriz Kp en secciones y realizarlo en varios equipos de cómputo al mismo tiempo; se podrían traducir los algoritmos a un lenguaje de programación más rápido como C, Java o Python; también, podría usarse un núcleo *p*-spectrum basado en la técnica del árbol trie (ver sección final del Capítulo III); o bien, intentar realizar todo lo anterior, cuya combinación reducirá el tiempo de ejecución de forma considerable.

También, fue necesaria la creación de un programa en Matlab para dar formato coherente con LibSVM 2.86 (Hsu *et al*., 2007), nuestra máquina de vectores de soporte (Library for Support Vector Machines, http://www.csie.ntu.edu.tw/~cjlin/libsvm/) a los datos de la matriz Kp. Ya que tenemos una matriz de 200 x 200 para cada *p*, ejecutamos este último programa para obtener los conjuntos de datos que se le proveerán a LibSVM 2.86 para la predicción. Para detalles ver Apéndice E.

Se realizaron básicamente tres experimentos con diferentes proporciones para los datos de entrenamiento y prueba. Esto, con la finalidad de estudiar cómo influyen las proporciones en los resultados obtenidos.

1. 50-50. Ya con la matriz de datos para *p*=2, 3, 4, 5, 6, 7 y 8 en formato LibSVM, ejecutamos entrenamiento (primeros 70 pares positivos y primeros 70 pares negativos) y prueba (los restantes 30 pares positivos y 30 negativos). Es decir, se tomaron el mismo número de pares positivos que negativos (70) para el entrenamiento y el resto para la sección de prueba (30) de cada tipo. La predicción se realizará en un conjunto con igual número de pares etiquetados como positivos y negativos.

2. 30-70. Con las mismas matrices de datos para *p* desde 2 hasta 8, se tomaron para entrenamiento los primeros 30 pares positivos y los primeros 70 negativos. Para la prueba se tomaron los 70 últimos pares positivos y 30 negativos. En este experimento la muestra está desbalanceada, teniendo más pares positivos que negativos para la prueba.

3. 70-30. Con las matrices Kp de *p*=2 a 8, el entrenamiento se realizó con los primeros 70 pares positivos y los primeros 30 pares negativos; en la prueba, se usaron los restantes 30 pares positivos y 70 negativos. Esta muestra de datos está desbalanceada con menos elementos positivos que negativos para la etapa de predicción.

Ahora bien, para cada experimento con *p*, se varió el parámetro nu (0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0). Nu es un parámetro dentro de la opción clasificador nu-SVM en LibSVM 2.86, que controla el número de vectores de soporte y errores de entrenamiento. El valor de nu va de 0 a 1, siendo una cota superior a la fracción de errores de entrenamiento y una cota inferior de la fracción de vectores de soporte. Mientras más grande sea el valor de nu se permite a más

puntos caer dentro del margen; es decir, aumentar nu permite más errores e incrementar el margen (Schölkopf *et al*., 2000), ver Figura 15. Para más detalle, ver (Chang y Lin, 2001) y (Schölkopf *et al*., 2000).

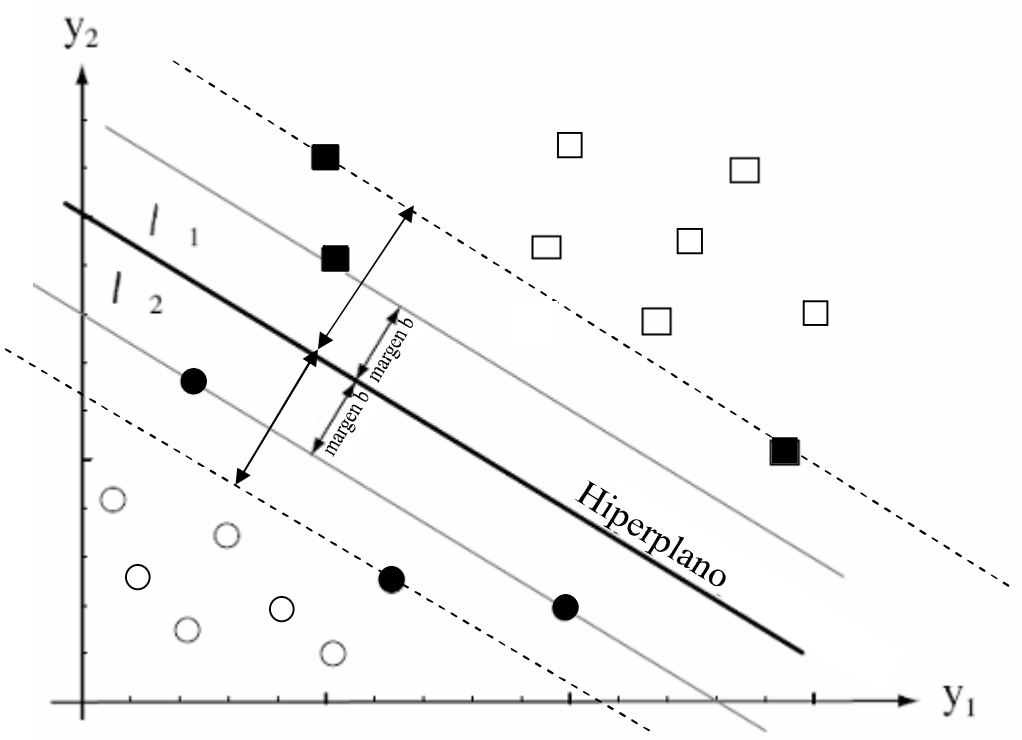

Figura 15: Parámetro nu. Mientras más grande sea el valor de nu se permite a más puntos caer dentro del margen, es decir, aumentar nu permite incrementar el margen b. El margen mayor está indicado en línea punteada.

V.2 Resultados

Los resultados de los experimentos fueron los siguientes y están agrupados por el tipo de proporción y a su vez, por el valor de *p*. La exactitud está dada en porcentaje y los valores entre paréntesis corresponden al número de pares de proteínas correctamente clasificados sobre el total de la muestra de prueba.

V.2.1 Proporción 50-50

Para la proporción 50-50 y *p* = 2, se presentan resultados del 50% para todos los valores de nu, ver Tabla VIII.

Tabla VIII: Resultado del experimento para *p*=2 con proporción 50-50 y valores de nu desde 0.1 a 1.0.

p=2	
nu	Exactitud
0.1	50% (30/60)
0.2	50% (30/60)
0.3	50% (30/60)
0.4	50% (30/60)
0.5	50% (30/60)
0.6	50% (30/60)
0.7	50% (30/60)
0.8	50% (30/60)
0.9	50% (30/60)
1.0	50% (30/60)

En la Tabla IX puede verse que el mejor resultado para *p*=3 fue con nu=0.4, por lo que se realizaron pruebas utilizando más decimales y se llegó a una exactitud de 56.67% (34/60) para un valor de nu = 0.414.

Tabla IX: Resultado del experimento para *p*=3 con proporción 50-50 y valores de nu desde 0.1 a 1.0.

p=3	
nu	Exactitud
0.1	43.33% (26/60)
0.2	48.33% (29/60)
0.3	53.33% (32/60)
0.4	55% (33/60)
0.5	51.67% (31/60)
0.6	50% (30/60)
0.7	50% (30/60)
0.8	50% (30/60)
0.9	50% (30/60)
1.0	50% (30/60)

Sin embargo, para p = 4, 5, 6, 7 y 8 el valor de la exactitud es de 50% en todos los casos, ver Tabla X.

Tabla X: Resultado del experimento para *p*=4, 5, 6, 7y 8 con proporción 50-50 y valores de nu desde 0.1 a 1.0.

nu	*p*=4	*p*=5	*p*=6	*p*=7	*p*=8
0.1	50% (30/60)	50% (30/60)	50% (30/60)	50% (30/60)	50% (30/60)
0.2	50% (30/60)	50% (30/60)	50% (30/60)	50% (30/60)	50% (30/60)
0.3	50% (30/60)	50% (30/60)	50% (30/60)	50% (30/60)	50% (30/60)
0.4	50% (30/60)	50% (30/60)	50% (30/60)	50% (30/60)	50% (30/60)
0.5	50% (30/60)	50% (30/60)	50% (30/60)	50% (30/60)	50% (30/60)
0.6	50% (30/60)	50% (30/60)	50% (30/60)	50% (30/60)	50% (30/60)
0.7	50% (30/60)	50% (30/60)	50% (30/60)	50% (30/60)	50% (30/60)
0.8	50% (30/60)	50% (30/60)	50% (30/60)	50% (30/60)	50% (30/60)
0.9	50% (30/60)	50% (30/60)	50% (30/60)	50% (30/60)	50% (30/60)
1.0	50% (30/60)	50% (30/60)	50% (30/60)	50% (30/60)	50% (30/60)

V.2.2 Proporción 30-70

Para p = 2 en la proporción 30-70 se obtienen resultados parecidos a los de p = 4, 5, 6, 7, y 8 para esta misma proporción. Al inicio 70% y luego una tendencia al 45%, ver Tabla XI.

Tabla XI: Resultado del experimento para *p*=2 con proporción 30-70 y valores de nu desde 0.1 a 1.0.

p=2	
nu	Exactitud
0.1	70% (70/100)
0.2	70% (70/100)
0.3	45% (45/100)
0.4	50% (50/100)
0.5	50% (50/100)
0.6	45% (45/100)
0.7	45% (45/100)
0.8	45% (45/100)
0.9	45% (45/100)
1.0	45% (45/100)

P = 3 no sigue la proporción y tiene como valor más alto 54%, como puede observarse en la Tabla XII.

Tabla XII: Resultado del experimento para *p*=3 con proporción 30-70 y valores de nu desde 0.1 a 1.0.

p=3	
nu	Exactitud
0.1	52% (52/100)
0.2	52% (52/100)
0.3	52% (52/100)
0.4	47% (47/100)
0.5	54% (54/100)
0.6	35% (35/100)
0.7	52% (52/100)
0.8	52% (52/100)
0.9	52% (52/100)
1.0	52% (52/100)

En la Tabla XIII se ve una clara tendencia de los datos a seguir la proporción de los datos de prueba, siendo una constante 70% y 30% para *p* = 4, 5, 6, 7 y 8.

Tabla XIII: Resultado del experimento para *p*=4, 5, 6, 7 y 8 con proporción 30-70 y valores de nu desde 0.1 a 1.0.

nu	*p*=4	*p*=5	*p*=6	*p*=7	*p*=8
0.1	70% (70/100)	70% (70/100)	70% (70/100)	70% (70/100)	70% (70/100)
0.2	70% (70/100)	70% (70/100)	70% (70/100)	70% (70/100)	70% (70/100)
0.3	30% (30/100)	30% (30/100)	30% (30/100)	30% (30/100)	30% (30/100)
0.4	30% (30/100)	30% (30/100)	30% (30/100)	30% (30/100)	30% (30/100)
0.5	30% (30/100)	30% (30/100)	30% (30/100)	30% (30/100)	30% (30/100)
0.6	30% (30/100)	30% (30/100)	30% (30/100)	30% (30/100)	30% (30/100)
0.7	30% (30/100)	30% (30/100)	30% (30/100)	30% (30/100)	30% (30/100)
0.8	30% (30/100)	30% (30/100)	30% (30/100)	30% (30/100)	30% (30/100)
0.9	30% (30/100)	30% (30/100)	30% (30/100)	30% (30/100)	30% (30/100)
1.0	30% (30/100)	30% (30/100)	30% (30/100)	30% (30/100)	30% (30/100)

V.2.3 Proporción 70-30

Para p = 2, los resultados obtenidos rondan el 42%, es decir, muestran menor tendencia a seguir la proporción, ver Tabla XIV.

Tabla XIV: Resultado del experimento para *p*=2 con proporción 70-30 y valores de nu desde 0.1 a 1.0.

p=2	
nu	Exactitud
0.1	42% (42/100)
0.2	42% (42/100)
0.3	58% (58/100)
0.4	42% (42/100)
0.5	42% (42/100)
0.6	42% (42/100)
0.7	42% (42/100)
0.8	42% (42/100)
0.9	42% (42/100)
1.0	42% (42/100)

En esta proporción y p =3 el resultado más alto de exactitud es 61%, como se muestra en la Tabla XV.

Tabla XV: Resultado del experimento para *p*=3 con proporción 70-30 y valores de nu desde 0.1 a 1.0.

p=3	
nu	Exactitud
0.1	34% (34/100)
0.2	61% (61/100)
0.3	59% (59/100)
0.4	34% (34/100)
0.5	33% (33/100)
0.6	30% (30/100)
0.7	34% (34/100)
0.8	34% (34/100)
0.9	34% (34/100)
1.0	34% (34/100)

Para la proporción 70-30, *p* = 4, 5, 6, 7 y 8 muestran de nuevo una tendencia a seguir las proporciones de los datos de prueba, con 30% y 70% como constantes, ver Tabla XVI.

Tabla XVI: Resultado del experimento para *p*=4, 5, 6, 7 y 8 con proporción 70-30 y valores de nu desde 0.1 a 1.0.

nu	*p*=4	*p*=5	*p*=6	*p*=7	*p*=8
0.1	30% (30/100)	30% (30/100)	30% (30/100)	30% (30/100)	30% (30/100)
0.2	30% (30/100)	30% (30/100)	30% (30/100)	30% (30/100)	30% (30/100)
0.3	70% (70/100)	70% (70/100)	70% (70/100)	70% (70/100)	70% (70/100)
0.4	70% (70/100)	70% (70/100)	70% (70/100)	70% (70/100)	70% (70/100)
0.5	70% (70/100)	70% (70/100)	70% (70/100)	70% (70/100)	70% (70/100)
0.6	70% (70/100)	70% (70/100)	70% (70/100)	70% (70/100)	70% (70/100)
0.7	70% (70/100)	70% (70/100)	70% (70/100)	70% (70/100)	70% (70/100)
0.8	70% (70/100)	70% (70/100)	70% (70/100)	70% (70/100)	70% (70/100)
0.9	70% (70/100)	70% (70/100)	70% (70/100)	70% (70/100)	70% (70/100)
1.0	70% (70/100)	70% (70/100)	70% (70/100)	70% (70/100)	70% (70/100)

V.3 Análisis de resultados

Para la proporción 50-50, *p* = 3 tiene la mayor exactitud correspondiente al valor de nu = 0.414 con 56.67%, es decir, 34 pares de proteínas correctamente clasificadas de una muestra de prueba de 60 pares.

Recordemos que el valor de nu va de 0 a 1, siendo una cota superior a la fracción de errores de entrenamiento y una cota inferior de la fracción de vectores de soporte.

Los valores obtenidos para *p* = 4, 5, 6, 7 y 8 se deben a que las matrices de datos Kp con las que se realiza tanto el entrenamiento como la prueba contienen gran cantidad de ceros, lo cual no proporciona información que sea de utilidad a la máquina de vectores de soporte para realizar la clasificación de forma correcta. Ahora bien, ese valor de 50% se mantiene constante porque la decisión que toma

la máquina de vectores de soporte frente al caso de no tener suficiente información es dar una sola respuesta, por ejemplo, asignar la etiqueta de sólo ceros a todos los pares, de esta forma y al existir únicamente dos opciones de clasificación, habrá acertado en la mitad de los casos.

En esta proporción y *p* = 2, los porcentajes reflejan que, a pesar de que la matriz de datos Kp contiene datos cuyas cifras son diferentes de cero, el núcleo *p*-spectrum para *p* = 2 en combinación con el núcleo de pares basándose sólo en estructura primaria de proteínas, no está proporcionando información relevante para realizar una correcta discriminación de interacción entre proteínas.

Para la proporción 30-70 con *p* = 4, 5, 6, 7 y 8, los resultados siguen la proporción de los datos de prueba ya que son los valores de *p* cuyas matrices Kp contienen gran cantidad de ceros. La máquina de vectores de soporte reacciona de forma similar que con la proporción 50-50, tomando una decisión de sólo ceros o sólo unos y al existir únicamente dos opciones de respuesta acierta en la cantidad de datos de prueba.

La tendencia de los resultados de *p* = 2 tiene por causa que, a pesar de que la matriz de datos Kp para esta *p* contiene datos cuyas cifras son diferentes de cero, no está proporcionando información relevante para realizar una correcta discriminación de interacción entre proteínas con el kernel de pares con *p*-spectrum de sólo secuencia de proteínas.

En esta proporción, *p* = 3 obtiene como mejor exactitud, 54% con nu = 0.5.

Para la proporción 70-30, *p* = 4, 5, 6, 7 y 8 muestran de nuevo una tendencia a seguir las proporciones de los datos de prueba, debido a las mismas circunstancias de los casos anteriores, se trabaja con matrices de datos con una

gran cantidad de ceros y la máquina de vectores de soporte opta por tomar una decisión de sólo ceros o unos y, por ende, acierta en la proporción en cuestión.

Como ya hemos mencionado, consideramos que el valor *p* = 2 no es particularmente apropiado para realizar una correcta discriminación de interacción entre proteínas con el kernel de pares con *p*-spectrum de sólo secuencia de proteínas, ya que, a pesar de que la matriz de datos Kp para *p* = 2 contiene datos cuyas cifras son diferentes de cero, no proporciona información relevante para dicha discriminación.

Finalmente, para *p* =3 el mejor resultado de exactitud es 61% con nu = 0.2.

En términos generales, podríamos decir que *p*=3 provee los resultados más confiables de los experimentos, sin ser satisfactorios. En seguida mostramos un fragmento de la matriz Kp para *p*=3 (Tabla XVII), donde podemos ver los valores del kernel de pares del núcleo *p*-spectrum para dicha *p*, los cuales son mayores a cero. Además, es una matriz simétrica y los valores de la diagonal son los más grandes dado que son el producto de la comparación del par de proteínas contra sí mismo.

Tabla XVII: Fragmento de la matriz Kp para p=3, primeros 10 x 9 elementos.

116500	3559	1087	314	277	6342	4198	512	1202
3559	574758	4535	1372	859	25350	14421	2136	6072
1087	4535	123769	375	271	5608	4879	666	1544
314	1372	375	51649	80	3762	2592	623	770
277	859	271	80	38198	1904	1187	146	262
6342	25350	5608	3762	1904	1176579	46751	6007	11305
4198	14421	4879	2592	1187	46751	815326	4123	6440
512	2136	666	623	146	6007	4123	105585	1429
1202	6072	1544	770	262	11305	6440	1429	232530
1947	6439	2046	951	409	8300	8457	1532	4274

Para los casos *p* = 4, 5, 6, 7 y 8, el hecho de que las matrices de datos de entrenamiento y prueba contengan una cantidad considerable de ceros se debe a que el núcleo *p*-spectrum compara secciones continuas, y conforme *p* es mayor, la posibilidad de coincidencia disminuye. Ver el ejemplo dado en el capítulo sobre métodos kernel en el apartado de *p*-spectrum.

Aunque existieran ciertas coincidencias en las cadenas y como el núcleo de pares implica realizar productos, puede obtenerse al final cero debido a la no coincidencia de otro factor del producto, que es el núcleo de un par de cadenas no coincidente (ver ecuación (17) al inicio de este capítulo). A continuación se presenta un fragmento de la matriz de datos Kp para *p*=8 con el objeto de acentuar nuestros comentarios al respecto.

Tabla XVIII: Fragmento de la matriz Kp para p=8, primeros 10 x 9 elementos.

90246	0	0	0	0	0	0	0	0
0	374934	0	0	0	0	0	0	0
0	0	90145	0	0	0	0	0	0
0	0	0	45696	0	0	0	0	0
0	0	0	0	32118	0	0	0	0
0	0	0	0	0	653652	0	0	0
0	0	0	0	0	0	522096	0	0
0	0	0	0	0	0	0	83145	0
0	0	0	0	0	0	0	0	178596
0	0	0	0	0	0	0	0	0

En nuestro estudio tomamos la forma de combinar núcleos o kernels para pares de proteínas de (Ben-Hur y Noble, 2005), pero usando sólo el núcleo de cadena o string kernel *p*-spectrum (Shawe-Taylor y Cristianini, 2004) y la máquina de vectores de soporte LibSVM 2.86. Variando *p*, analizamos cual de ellos aporta la mayor precisión en la predicción de interacciones proteína-proteína basados sólo en estructura primaria, la *p* con resultados más confiables aunque no satisfactorios fue *p* = 3.

La exactitud de los resultados no es alta, esto puede deberse a factores como:

1) trabajar sólo con secuencia de aminoácidos y para la predicción de interacción proteína-proteína pueden ser suficiente dada la naturaleza del problema.

2) el hecho de que el núcleo *p*-spectrum considera secciones contiguas para las comparaciones. Los residuos encargados de la interacción no están necesariamente contiguos en la secuencia.

En (Ben-Hur y Noble, 2005) se ha realizado un estudio de métodos núcleo para predicción de la interacción entre proteínas, usando una combinación de fuentes de datos, incluyendo secuencias de proteínas, anotaciones ontológicas, propiedades locales de la red e interacción homóloga con otras especies, es decir, no sólo estructura primaria. Se ilustra la eficacia de los núcleos en conjunto con un clasificador SVM. La tarea de aprendizaje involucra la relación entre pares de secuencias de proteínas para determinar si un par de proteínas están interactuando o no. En (Ben-Hur y Noble, 2005) se afirma que es difícil predecir interacciones utilizando sólo secuencia (cuestión que hemos comprobado), por lo que se incorporaron fuentes adicionales de datos. Se reportan resultados del 80% de efectividad (Ben-Hur y Noble, 2005). Podemos ver claramente, que nuestra investigación difiere bastante con la de (Ben-Hur y Noble, 2005), de la cual básicamente, hemos retomado el núcleo de pares.

Ahora, tras el estudio del tema y los experimentos realizados, podemos pensar en ciertas formas que ayudarían a elevar la exactitud de los resultados, estas ideas las hemos concentrado en la sección de Propuesta.

V.4 Propuesta para mejorar los resultados

Tras el desarrollo de esta investigación, hemos visto que la exactitud de los resultados de los experimentos realizados es baja, lo cual consideramos puede

deberse a varios factores como: el núcleo utilizado, *p*-spectrum, considera secciones contiguas para las comparaciones, utilizar sólo secuencia de aminoácidos, y dada la naturaleza de la interacción proteína-proteína estos aspectos pueden no ser suficientes para la predicción correcta de posibles interacciones entre proteínas. Sin embargo, mediante la experiencia e información adquirida durante este estudio, nos permitimos sugerir algunas ideas que consideramos podrían incrementar la exactitud de los resultados.

Primeramente, seleccionar una muestra más grande de ejemplos positivos y negativos. Nuestra muestra fue de 400 proteínas agrupadas en pares, 100 pares como ejemplos positivos y 100 como ejemplos negativos. Pero, al utilizar una máquina de vectores de soporte, al ser esta un método supervisado de aprendizaje automático, debemos separar la muestra en dos conjuntos: entrenamiento y prueba. Esto hace que la muestra disminuya. Por lo tanto, creemos que aumentar considerablemente el tamaño de la muestra puede proveer de mayor información a la máquina de vectores de soporte, y que ésta puede hacerse de más elementos con los que discernir con mayor eficacia.

El núcleo *p*-spectrum se basa en comparaciones de secciones contiguas de longitud *p*, esto puede que no sea muy útil para predecir la posible interacción entre proteínas debido a la naturaleza de dichas interacciones que se llevan a cabo cuando las proteínas están en su estructura terciaria o cuaternaria. Sería muy interesante ver el impacto que tiene en la exactitud de los resultados el utilizar otros núcleos de cadena como: el kernel fixed length subsequences o núcleo de subsecuencias con longitud fija (Shawe-Taylor y Cristianini, 2004), que es una adaptación al kernel all-subsequences que reduce la dimensionalidad del espacio de características considerando sólo subsecuencias no contiguas de longitud fija usando la variable *p*. En este núcleo se permite la no continuidad en la secuencia, lo cual proporciona mayor flexibilidad al comparar las cadenas. Otra posibilidad es usar el kernel soft matching string o núcleo de empatamiento suave de cadenas

(Shawe-Taylor y Cristianini, 2004), que se aplica cuando las cadenas contienen símbolos distintos, pero que de alguna forma estén relacionados, se supone que se ha dado una matriz A de similitud entre los símbolos. Este núcleo puede ser de utilidad, por ejemplo, cuando sabemos de la interacción entre proteínas en una especie y se dan ligeros cambios en la secuencia en otra especie similar o bien, cuando ha sucedido una mutación. Para más información de estos y otros núcleos, ver el Capítulo III.

En nuestro estudio utilizamos sólo estructura primaria de proteínas, es decir, su secuencia de aminoácidos. En (Ben-Hur y Noble, 2005) se afirma que es difícil predecir interacciones utilizando sólo secuencia, por lo que incorporan fuentes adicionales de datos, incluyendo secuencias de proteínas, anotaciones ontológicas, propiedades locales de la red e interacción homóloga con otras especies. Consideramos que si se agrega más información, además de la secuencia de aminoácidos, como lo hacen en (Ben-Hur y Noble, 2005), y otras características físico-químicas de las interacciones entre proteínas como hidrofobicidad, se tendría un panorama más completo de los elementos presentes en las interacciones proteína-proteína, y esto aportaría mayor información para realizar una mejor discriminación.

Consideramos que con estas sugerencias podría incrementarse en gran medida la exactitud de los resultados obtenidos.

Capítulo VI

Conclusiones y trabajo futuro

VI.1 Sumario

El poder determinar si un par de proteínas pueden interactuar, es decir, relacionarse para llevar a cabo alguna función, es uno de los problemas más importantes del estudio de las mismas. La información acerca de esas interacciones contribuye a entender mejor las enfermedades y puede proporcionar datos relevantes para el desarrollo de innovaciones en la farmacología (Rojas Vahos, 2007). Existen diferentes algoritmos de clasificación para el estudio de interacción proteína-proteína.

El problema que abordamos consiste en determinar experimentalmente el efecto que tiene el valor del parámetro *p* en el núcleo *p*-spectrum sobre la exactitud para predecir la interacción o no entre un par de proteínas de acuerdo a esta forma de representación de los datos. La exactitud se refiere al éxito de clasificación para un conjunto predefinido de proteínas.

El conjunto de datos que se decidió utilizar para nuestro estudio, se basa en la selección aleatoria uniforme de pares de proteínas cuyas interacciones desconocemos para los ejemplos negativos, utilizando los datos propuestos por Ben-Hur y Noble, (2005) tanto para los ejemplos positivos como negativos. Se seleccionaron los primeros 100 pares de proteínas para cada tipo de ejemplo (200 pares, 366 proteínas diferentes).

Se realizaron tres experimentos con diferentes proporciones de ejemplos positivos y negativos para los datos de entrenamiento y prueba: 50-50, 30-70 y 70-30. Calculando el núcleo de pares del *p*-spectrum para *p* = 2, 3, 4, 5, 6, 7 y 8 para cada caso.

Ahora bien, para cada experimento con *p*, se varió el parámetro nu (0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0). Nu es un parámetro dentro de la opción nu-SVM classification en LibSVM 2.86 (Hsu *et al*., 2007), que controla el número de vectores de soporte y errores de entrenamiento.

VI.2 Conclusiones

Los resultados experimentales nos llevan a concluir lo siguiente:

- Los valores de *p* = 4, 5, 6, 7 y 8 no son adecuados para la predicción de interacción proteína-proteína ya que las matrices de datos Kp contienen gran cantidad de ceros.

- Para *p* = 2, se presentan resultados similares a los valores anteriores lo cual refleja que, a pesar de que la matriz Kp contiene datos diferentes de cero, no está proporcionando información relevante para realizar una correcta discriminación.

- Los resultados más confiables, aunque no satisfactorios corresponden a *p* = 3, teniendo como mejores exactitudes las siguientes:
    - Para la proporción 50-50, 56.67%.
    - Para la proporción 30-70, 54%.
    - Para la proporción 70-30, 61%.

- En general, podemos decir que los núcleos que consideren fragmentos contiguos de aminoácidos no proporcionarán información suficiente para una correcta discriminación entre interacción y no interacción.

VI.3 Trabajo futuro

En términos generales, podemos ver tres áreas en las cuales se puede profundizar, obtener mejoras, complementar o ampliar el trabajo realizado. Éstas serían: velocidad de cálculo, mejora de exactitud en los resultados y el caso de los priones.

La sección que calcula la matriz del kernel de pares (pairwise kernel: Kp (Ben-Hur y Noble, 2005)) con el núcleo *p*-spectrum, es la que mayor tiempo consume, las sugerencias para reducir el tiempo de ejecución se encuentran en el apartado V.1 Experimentos, página 58.

La exactitud de los resultados de los experimentos es pobre. Para incrementarla sugerimos algunas ideas que se plantearon en la sección de propuestas (V.4): seleccionar una muestra más grande de ejemplos positivos y negativos, el utilizar otros núcleos de cadena e incorporan fuentes adicionales de datos.

Finalmente, existen casos como el de los priones, en los que, para una misma proteína, con la misma secuencia, los estos presentan estructuras tridimensionales diferentes, afectando con este cambio la posible interacción entre proteínas. Esto motiva a estudiar la interacción entre proteínas considerando la estructura tridimensional de las mismas.

Referencias

ACCEFYN (Academia Colombiana de Ciencias Exactas, Físicas y Naturales). 2000. La bioinformática como herramienta en el descubrimiento de nuevos medicamentos.
Disponible en: http://www.accefyn.org.co/bioinfo/pdf/bioinformatica.PDF
Consultado: marzo de 2008.

Alfarano, C., Andrade, C., Anthony, K., Bahroos, N., Bajec, M., Bantoft, K., Betel, D., Bobechko, B., Boutilier, K., Burgess, E., Buzadzija, K., Cavero, R., D'Abreo, C., Donaldson, I., Dorairajoo, D., Dumontier, M., Dumontier, M., Earles, V., Farrall, R., Feldman, H., Garderman, E., Gong, Y., Gonzaga, R., Grytsan, V., Gryz, E., Gu, V., Haldorsen, E., Halupa, A., Haw, R., Hrvojic, A., Hurrell, L., Isserlin, R., Jack, F., Juma, F., Khan, A., Kon, T., Konopinsky, S., Le, V., Lee, E., Ling, S., Magidin, M., Moniakis, J., Montojo, J., Moore, S., Muskat, B., Ng, I., Paraiso, J., Parker, B., Pintilie, G., Pirone, R., Salama, J., Sgro, S., Shan, T., Shu, Y., Siew, J., Skinner, D., Snyder, K., Stasiuk, R., Strumpf, D., Tuekam, B., Tao, S., Wang, Z., White, M., Willis, R., Wolting, C., Wong, S., Wrong, A., Xin, C., Yao, R., Yates, B., Zhang, S., Zheng, K., Pawson, T., Ouellette B. y Hogue, C. 2005. The Biomolecular Interaction Network Database and related tools 2005 update. Nucleic Acids Research, 33: 418-424.

Bader, G., Donaldson, I., Wolting, C., Ouellette, B. F., Pawson, T. y Hogue, C. W. 2001. BIND—the biomolecular interaction network database. Nucleic Acids Research, 29: 242–245.

Ben-Hur, A. y Noble, W. S. 2005. Kernel methods for predicting protein–protein interactions. Bioinformatics, 21: i38-i46.

Ben-Hur, A. y Noble, W. S. 2006. Choosing negative examples for the prediction of protein-protein interactions. BMC Bioinformatics, 7: S2-S7.

Betancourt, G. 2005. Las máquinas de soporte vectorial (SVMs). Scientia et Technica, 11(27): 67-72.

Bobadilla, J., Mojica, T., y Niño, L. 2003. Identificación de sitios en proteínas usando máquinas con vectores de soporte. Nova, 1(1): 65-71.

Bock, J. y Gough, D. 2001. Predicting protein-protein interactions from primary structure. Bioinformatics, 17(5): 455–460.

Bolívar Zapata, F. 2004. Moléculas informacionales de la célula viva: ácidos nucleicos y proteínas. En: Bolívar Zapata, F. (ed.). Fundamentos y casos exitosos de la biotecnología moderna. El Colegio Nacional, México, 19-56 p.

Bolívar Zapata, F. 2004b. Ciencia genómica, proteómica y bioinformática. El genoma y el proteoma humano. En: Bolívar Zapata, F. (ed.). Fundamentos y casos exitosos de la biotecnología moderna. El Colegio Nacional, México, 85-116 p.

Redacción BorNet. 2000. La utilización de biochips se extiende a la detección de péptidos y proteínas. BorNet, Revista de divulgación sobre ciencias. Disponible en: http://www.bornet.es/notic/Biologia_y_Biotecnologia/010800131500.shtml Consultado: marzo de 2008.

Cañedo Andalia, R. y Arencibia Jorge, R. 2004. Bioinformática: en busca de los secretos moleculares de la vida. Acimed (Revista Cubana de los profesionales de la información y de la comunicación en salud), 12(6). Disponible en: http://bvs.sld.cu/revistas/aci/vol12_6_04/aci02604.htm Consultado: marzo de 2008.

Carugo, O. y Franzot, G. 2004. Prediction of protein-protein interactions based on surface match comparison. Proteomics, 4(6): 1727-1736.

Chagolla López, A. y Barba de la Rosa, A. 2006. Proteómica y bioinformática. Ciencia y Desarrollo, 32(198): 50-55.

Chang, C. y Lin, C. 2001. LIBSVM: a library for support vector machines. Software disponible en: http://www.csie.ntu.edu.tw/~cjlin/libsvm. Consultado: noviembre de 2007.

Chung, J., Wang, W. y Bourne, P. 2006. Exploiting Sequence and Structure homologs to identify protein-protein binding sites. Proteins: Structure, Function, and Bioinformatics, 62(3): 630-640.

Cormen, T., Leiserson, C., Rivest, R. y Stein, C. 2001. Introduction to Algorithms. Chapter 15: Dynamic Programming. MIT Press. Segunda edición. Cambridge, Massachusetts, E. U. A. 1202 pp.

Cortijo Bon, F. 2001. Tema 1 Introducción al reconocimiento de formas. Reconocimiento de formas. Universidad de Granada. E. T. S. Ingeniería Informática, Departamento de Ciencias de la Computación. Notas de curso 2001-2002. 50 pp.

Cristianini, N. y Shawe-Taylor, J. 2000. An introduction to support vector machines (and other kernel-based learning methods). Cambridge University Press. 208 pp.

De la Torre Russis, V., Valles, A., Gómez, R., Chinea, G. y Pons, T. 2003. Interacciones proteína-proteína: bases de datos y métodos teóricos de predicción. Biotecnología aplicada, 20(3): 201-208.

Duda, R., Hart, P. y Stork, D. 2000. Pattern Classification. Wiley Interscience. Segunda edición. 678 pp.

Eidhammer, I., Jonassen, I. y Taylor, W. 2004. Protein Bioinformatics: An Algorithmic Approach to Sequence and Structure Analysis. John Wiley and Sons. 376 pp.

Gärtner, T. 2003. A survey of Kernels for Structured Data. ACM SIGKDD Explorations Newsletter archive, 5(1): 49-58.

Gasset, M. y Westaway, D. 1998. Los priones y su biología. I Congreso Virtual Iberoamericano de Neurología.
Disponible en: http://svneurologia.org/congreso/priones-1.html Consultado: marzo de 2008.

Gavilanes, J. 2001. ¿Qué es una proteína? Avizora Publicaciones. Argentina. Disponible en:
http://www.avizora.com/publicaciones/ciencias/textos/que_es_proteina_una_enzima_0002.htm Consultado: marzo de 2008.

Gomez, S., Noble, W. S. y Rzhetsky, A. 2003. Learning to predict protein-protein interactions from protein sequence. Bioinformatics, 19(15): 1875-1881.

Gosset Lagarda, G. y Bolívar Zapata, F. 2004. Capítulo VIII, Ingeniería celular microbiana: La importancia del conocimiento del transcriptoma y el proteoma de la célula para la comprensión final del metabolismo celular. En: Bolívar Zapata, F. (ed.). Fundamentos y casos exitosos de la biotecnología moderna. El Colegio Nacional, México, 234-235 p.

Hsu C., Chang, C. y Lin, C. 2007. A practical guide to SVM classification. Disponible en: http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf
Consultado: noviembre de 2007.

Instituto de Salud Carlos III (ISCIII). Servicios de Información Internet. 2001. Las Encefalopatías Espongiformes Transmisibles Humanas. Una visión desde la Salud Pública. Unidad de Vigilancia de Encefalopatías Espongiformes Transmisibles Humanas. Centro Nacional de Epidemiología. Madrid. Disponible en: http://www.isciii.es/htdocs/centros/epidemiologia/salud_publica.jsp Consultado: marzo de 2008.

Kohutyuk, O. y Honavar, V. 2006. Protein interaction prediction with Supervised Learning Methods. International Conference on Intelligent Systems for Molecular Biology (ISMB), Fortaleza, Brasil.
Disponible en: http://ismb2006.cbi.cnptia.embrapa.br/poster_abstract.php?id=J-29
Consultado: marzo de 2008.

Leslie, C., Eskin, E. y Noble, W. S. 2002. The spectrum kernel: A string kernel for SVM protein classification. Proceedings of the Pacific symposium on Biocomputing, 564-575 p.

Lewis, D., Jebara, T. y Noble, W. S. 2006. Support vector machine learning from heterogeneous data: an empirical analysis using protein sequence and structure. Bioinformatics, 22(22): 2753 - 2760.

López Munguía Canales, A. 2005. Las proteínas. Consejo Nacional para la Cultura y las Artes, México, D. F., 24 pp.

Maino, R. 2001. Proteómica. El desafío de la post-genómica. Revista de la Sociedad de Medicina Interna de Buenos Aires, Argentina. 2(1):7-8.

Mamitsuka, H. 2005. Essential Latent Knowledge for Protein-Protein Interactions: Analysis by an unsupervised Learning Approach. IEEE/ACM Transactions on Computational Biology and Bioinformatics, 2(2): 119-130.

Martin, S., Roe, D. y Faulon, J. 2005. Predicting protein–protein interactions using signature products. Bioinformatics, 21(2): 218–226.

Martínez Hormazábal, J. 2006. Función e importancia de la Bioinformática en el desarrollo de las Ciencias, especialmente en Biotecnología y Medicina Molecular. Ciencia y Trabajo, 8(22): 159-163.

Noble, W. S. 2004. Support vector machine applications in computational biology. En: Schölkopf, B., Tsuda, K. y Vert, J. (eds.). Kernel Methods in Computational Biology. MIT Press. Cambridge, Massachusetts, E. U. A. 71-92 p.

Noble, W. S. 2006. What is a support vector machine? Nature Biotechnology, 24(12): 1565-1567.

Nooren, I. y Thornton, J. 2003. Diversity of protein-protein interactions. The EMBO Journal, 22(14): 3486-3492.

Parrish, J., Gulyas, K. y Finley, R. 2006. Yeast two-hybrid contributions to interactome mapping. Current Opinion in Biotechnology, 17(4): 387-393.

Poole, D. 2006. Algebra lineal: Una introducción moderna. Cengage Learning Editores. Segunda edición. 744 pp.

Prusiner, S. 1982. Novel proteinaceous infectious particles cause scrapie. Science, 216(4542): 136-144.

Prusiner, S. 1998. Prions. Proc. Natl. Acad. Sci. USA, 95:13363–13383.

Real Academia Española. Diccionario de la lengua española. Vigésima segunda edición. Disponible en: http://www.rae.es/ Consultado: diciembre de 2007.

Raisman, J. 2007. Priones. Facultad de Agroindustrias, Universidad Nacional del Nordeste, Argentina. Basado en textos de Dra. Ingeborg M. M. van Leeuwen. Departamento de Biología Teórica, Vrije Universiteit, Ámsterdam, Paises Bajos. Disponible en: http://video.google.es/videoplay?docid=-278337970598189514 9 Consultado: marzo de 2008.

Rojas Vahos, D. 2007. La Biotecnología: Herramienta para desarrollar nuevos medicamentos. CECIF (Centro de la Ciencia y la Investigación Farmacéutica) Magazine. Disponible en: http://www.cecif.org/magazine/index.php?option=com_content&task=view&id=107&Itemid=1 Consultado: marzo de 2008

Rousu, J. y Shawe-Taylor, J. 2005. Efficient Computation of Gapped Substring Kernels on Large Alphabets. Journal of Machine Learning Research, 6: 1323–1344.

Salwinski, L. y Eisenberg, D. 2003. Computational methods of analysis of protein-protein interactions. Current Opinion in Structural Biology, 13:377-382.

Sánchez-Vítores, R. 2004. Sistemas de Reconocimiento Facial. Disponible en: http://www.faq-mac.com/mt/archives/009001.php Consultado: marzo de 2008

Shawe-Taylor, J. y Cristianini, N. 2004. Kernels for structured data: strings, trees, etc. Kernel Methods for Pattern Analysis. Cambridge University Press, 488 pp.

Shoemaker, B. y Panchenko, A. 2007. Deciphering Protein-Protein Interactions. Part II. Computational methods to predict protein and domain interaction partners. PLoS Computational Biology, 3(4): e43-50.

Soberón Mainero, X y Montero Morán, G. 2004. Ingeniería de proteínas y evolución dirigida. En: Bolívar Zapata, F. (ed.). Fundamentos y casos exitosos de la biotecnología moderna. El Colegio Nacional, México, 195-218 p.

Tramontano, A. 2005. The Ten Most Wanted Solutions in Protein Bioinformatics. CRC Press Science, 232 pp.

Valencia, A. y Pazos, F. 2002. Computational methods for the prediction of protein-protein interactions. Current Opinion in Structural Biology, 12:368-373.

Vert, J., Tsuda, K. y Schölkopf, B. 2004. Kernel Methods in Computational Biology. A primer on Kernel Methods. MIT Press. Cambridge, Massachusetts, E. U. A. 410 pp.

Vapnik, V. 1995. The nature of statistical learning theory. Springer. Primera edición. 188pp.

Villegas de Olazával, H. y Villegas del Carpio, R. 1997. Enfermedades transmitidas por priones. Biblioteca Nacional de Salud y Seguridad Social (BINASSS), Caja Costarricense de Seguro Social (CCSS). Costa Rica. Disponible en: http://www.binasss.sa.cr/sitios/priones.htm Consultado: marzo de 2008.

Zaki, N., Deris, S. y Alashwal, H. 2006. Protein-Protein Interaction Detection Based on Substring Sensitivity Measure. International Journal of Biomedical Sciences, 1(2): 148-154.

Xenarios, I. y Eisenberg, D. 2001. Protein interaction databases. Current Opinion in Biotechnology, 12: 334-339.

Apéndice A: LIBSVM

LIBSVM (Library for Support Vector Machines), como su nombre lo dice, es una biblioteca de software para máquinas de vectores de soporte y su finalidad es que la máquina de vectores de soporte sea utilizada fácilmente como una herramienta (Chang y Lin, 2001). En su página web (http://www.csie.ntu.edu.tw/~cjlin/libsvm/) se encuentra mucha información valiosa como tutoriales, consejos y herramientas complementarias; por lo que se recomienda ampliamente visitarla, así como leer (Hsu *et al*., 2007), el archivo incluido en el software y la sección de preguntas frecuentes.

LIBSVM resuelve clasificación C-SVM, nu-SVM y SVM para una clase (one-class-SVM); también regresión epsilon-SVM y nu-SVM. Los dos primeros realizan clasificación, con la diferencia que el rango del parámetro "C" va de cero a infinito, y "nu" está entre cero y uno. One-class-SVM se utiliza para estimar los soportes de distribución en alta dimensión, es más una estimación de distribución. Los últimos dos casos se refieren a regresión. Así mismo, maneja cinco tipos de núcleos: lineal, polinomial, función base radial (RBF), sigmoidal y núcleos pre-calculados. Básicamente, hay que entrenar los datos con la función “train” y luego hacer la prueba con “predict”, obviamente agregando los nombres de los archivos y los parámetros a usarse. Los archivos de datos deben tener, previamente, el formato específico en que LIBSVM los lee (Chang y Lin, 2001).

Entre las ventajas de LIBSVM se destacan: la facilidad de instalación (en sistemas operativos tipo Unix, Linux y Windows) y uso, rapidez de respuesta y flexibilidad; ya que cuenta con varios núcleos o kernels predefinidos, pero también con la opción de aceptar matrices de núcleos pre-calculados. Sin embargo, en cuanto a núcleos de cadena (string kernels) falta aún mucho trabajo por realizar, es un área no muy desarrollada, y esta es la razón por la que en la presente

investigación recurrimos a la alternativa de entregarle al programa una matriz de núcleos calculados previamente.

Si se desea conocer más, por ejemplo, acerca de la solución del problema cuadrático de la máquina de vectores de soporte dada por LIBSVM, revisar (Chang y Lin, 2001). Aunque dicho conocimiento no es imprescindible para empezar a trabajar con el programa, sí es aconsejable. Para detalles como los parámetros y demás, específicos a los experimentos realizados durante este estudio, favor de remitirse al capítulo sobre experimentos y análisis de resultados.

Apéndice B: Conjunto de datos

Tabla XIX: Conjunto de datos empleado en esta investigación con sus respectivas longitudes de secuencias, agrupados por pares de proteínas de acuerdo a su interacción.

# par	Longitud proteína 1	Longitud proteína 2	Interacción	# par	Longitud proteína 1	Longitud proteína 2	Interacción
1	274	345	Sí	101	433	341	No
2	685	560	Sí	102	304	207	No
3	612	156	Sí	103	733	726	No
4	175	279	Sí	104	150	604	No
5	310	113	Sí	105	334	669	No
6	610	1091	Sí	106	328	119	No
7	591	901	Sí	107	287	683	No
8	352	248	Sí	108	1755	404	No
9	499	370	Sí	109	1391	884	No
10	368	552	Sí	110	577	341	No
11	206	122	Sí	111	392	254	No
12	144	114	Sí	112	177	181	No
13	899	143	Sí	113	350	353	No
14	505	619	Sí	114	770	782	No
15	571	245	Sí	115	693	316	No
16	420	671	Sí	116	884	347	No
17	279	686	Sí	117	917	270	No
18	523	429	Sí	118	117	108	No
19	106	474	Sí	119	520	491	No
20	515	396	Sí	120	554	148	No
21	438	884	Sí	121	153	332	No
22	345	102	Sí	122	1301	1031	No
23	280	506	Sí	123	327	249	No
24	469	298	Sí	124	164	571	No
25	183	607	Sí	125	426	796	No
26	121	1017	Sí	126	175	518	No
27	336	210	Sí	127	438	511	No
28	172	281	Sí	128	1195	267	No
29	409	650	Sí	129	147	514	No
30	77	188	Sí	130	668	446	No
31	420	842	Sí	131	785	915	No
32	1447	769	Sí	132	354	191	No
33	753	279	Sí	133	1037	635	No
34	596	346	Sí	134	462	345	No

35	217	213	Sí	135	633	1083	No
36	122	417	Sí	136	492	478	No
37	2167	842	Sí	137	123	684	No
38	528	469	Sí	138	232	672	No
39	338	312	Sí	139	1592	1619	No
40	633	308	Sí	140	289	140	No
41	518	749	Sí	141	246	363	No
42	420	1131	Sí	142	423	695	No
43	114	663	Sí	143	334	272	No
44	668	1056	Sí	144	211	628	No
45	335	600	Sí	145	542	1557	No
46	327	400	Sí	146	551	117	No
47	654	1091	Sí	147	153	138	No
48	1091	122	Sí	148	149	318	No
49	639	1056	Sí	149	122	184	No
50	334	245	Sí	150	337	511	No
51	1224	959	Sí	151	189	286	No
52	1381	892	Sí	152	133	379	No
53	163	373	Sí	153	376	619	No
54	255	696	Sí	154	409	661	No
55	354	172	Sí	155	120	1223	No
56	539	380	Sí	156	165	271	No
57	393	1802	Sí	157	107	405	No
58	733	729	Sí	158	635	413	No
59	149	389	Sí	159	629	284	No
60	111	741	Sí	160	281	546	No
61	123	428	Sí	161	140	970	No
62	876	768	Sí	162	340	515	No
63	119	205	Sí	163	199	502	No
64	469	123	Sí	164	849	297	No
65	568	400	Sí	165	450	394	No
66	285	120	Sí	166	2748	256	No
67	169	109	Sí	167	151	810	No
68	438	393	Sí	168	113	1178	No
69	571	124	Sí	169	442	242	No
70	542	733	Sí	170	1155	332	No
71	123	434	Sí	171	319	280	No
72	279	842	Sí	172	420	304	No
73	368	114	Sí	173	1306	456	No
74	196	245	Sí	174	360	499	No
75	894	406	Sí	175	394	761	No
76	610	109	Sí	176	107	238	No
77	216	738	Sí	177	379	880	No

78	317	741	Sí	178	914	931	No
79	105	345	Sí	179	448	183	No
80	1091	345	Sí	180	875	844	No
81	293	393	Sí	181	415	535	No
82	1025	970	Sí	182	573	1024	No
83	537	642	Sí	183	481	387	No
84	368	388	Sí	184	104	382	No
85	312	1953	Sí	185	133	637	No
86	417	407	Sí	186	213	250	No
87	380	477	Sí	187	180	694	No
88	507	205	Sí	188	107	279	No
89	236	359	Sí	189	221	340	No
90	440	749	Sí	190	348	303	No
91	357	245	Sí	191	439	429	No
92	382	807	Sí	192	926	232	No
93	110	1149	Sí	193	458	213	No
94	480	373	Sí	194	241	444	No
95	1091	435	Sí	195	351	113	No
96	687	129	Sí	196	471	502	No
97	589	1056	Sí	197	1142	353	No
98	704	479	Sí	198	721	178	No
99	187	743	Sí	199	436	142	No
100	215	395	Sí	200	127	1549	No

Tabla XX: Pares de proteínas en AC que son el conjunto de datos empleado en esta investigación.

# par	Proteína 1	Proteína 2	Tipo de ejemplo	# par	Proteína 1	Proteína 2	Tipo de ejemplo
1	P21825	P29055	positivo	101	P38211	P53290	negativo
2	P37838	Q12514	positivo	102	Q06188	Q03079	negativo
3	Q02866	Q06592	positivo	103	P46995	P36000	negativo
4	P38334	P47153	positivo	104	Q12513	P14747	negativo
5	P38238	P53307	positivo	105	P38803	P23900	negativo
6	P40024	P47135	positivo	106	P52920	P40895	negativo
7	P54113	Q05934	positivo	107	P13517	P32432	negativo
8	P35197	P53039	positivo	108	O13527	P21957	negativo
9	P09624	P53167	positivo	109	P40064	Q06163	negativo
10	P53295	P40988	positivo	110	P38932	P14908	negativo
11	P01123	P27999	positivo	111	P36081	Q12339	negativo
12	P13434	P52553	positivo	112	P12962	Q08962	negativo
13	P19735	Q06819	positivo	113	Q12449	Q12123	negativo
14	P32566	Q99234	positivo	114	P21192	Q03653	negativo
15	P38871	P38987	positivo	115	P32590	P25586	negativo

16	P40091	Q12114	positivo	116	Q03466	P47172	negativo
17	P32447	P35817	positivo	117	P32917	P16523	negativo
18	P40016	Q06103	positivo	118	P47174	P32911	negativo
19	P36092	Q02574	positivo	119	P38207	P25381	negativo
20	P25382	P40956	positivo	120	P38707	P40565	negativo
21	Q12509	Q06163	positivo	121	P36010	Q02796	negativo
22	P32366	O13558	positivo	122	P46674	P07276	negativo
23	Q03433	Q08923	positivo	123	P38063	Q12512	negativo
24	P40054	P40529	positivo	124	P46973	P21264	negativo
25	P38116	Q04632	positivo	125	P39706	P43573	negativo
26	P38701	Q07830	positivo	126	Q03323	P38860	negativo
27	Q04629	P53604	positivo	127	Q99303	P35191	negativo
28	P47017	P53965	positivo	128	P36168	Q05016	negativo
29	Q05543	Q08032	positivo	129	P53074	Q06214	negativo
30	Q03919	P52491	positivo	130	P34217	P38757	negativo
31	P40091	P48562	positivo	131	P50089	P36117	negativo
32	P35187	P04786	positivo	132	P38251	P87266	negativo
33	P38254	P32447	positivo	133	P25389	P09232	negativo
34	Q06325	Q12118	positivo	134	Q12069	P46946	negativo
35	Q12035	Q99176	positivo	135	P40497	P40480	negativo
36	Q06333	Q06410	positivo	136	P25345	P47018	negativo
37	P39960	P48562	positivo	137	Q07738	Q04779	negativo
38	P04385	P40054	positivo	138	P38064	P32831	negativo
39	P38232	P32598	positivo	139	P53855	Q08562	negativo
40	P40497	P53728	positivo	140	P39731	P40162	negativo
41	P39937	P40957	positivo	141	P46948	Q08213	negativo
42	P16387	P08004	positivo	142	P25591	Q07980	negativo
43	P52553	P24720	positivo	143	P25453	P50113	negativo
44	Q06677	P40395	positivo	144	P38307	Q08641	negativo
45	P25583	P41833	positivo	145	P53189	P38873	negativo
46	P45976	P53934	positivo	146	Q07657	O13574	negativo
47	P17121	P47135	positivo	147	P36010	Q08789	negativo
48	P47135	P53887	positivo	148	Q12156	P38620	negativo
49	P38196	P40395	positivo	149	P38253	P05740	negativo
50	P25453	P38987	positivo	150	Q05568	P40959	negativo
51	P52918	Q02629	positivo	151	P53910	Q12017	negativo
52	P34216	P53858	positivo	152	P38350	P04803	negativo
53	Q04493	P36224	positivo	153	P18898	Q99234	negativo
54	P31377	P34226	positivo	154	P47013	P53835	negativo
55	Q12154	Q12184	positivo	155	P38155	Q02979	negativo
56	P39705	P25559	positivo	156	P38362	Q08558	negativo
57	P50108	P41809	positivo	157	O13532	Q03687	negativo
58	Q12748	P17119	positivo	158	P31383	P32478	negativo
59	Q07541	Q04951	positivo	159	Q12333	Q12343	negativo
60	P40005	P49956	positivo	160	P03069	P43621	negativo
61	Q06406	P26725	positivo	161	P47822	P53550	negativo

62	P53953	P15303	positivo	162	Q03703	P53741	negativo
63	P39938	Q06672	positivo	163	P40212	Q12505	negativo
64	P40054	P47146	positivo	164	P21372	P25368	negativo
65	P53309	P53934	positivo	165	P38141	Q12385	negativo
66	Q03835	P53082	positivo	166	Q00402	Q03691	negativo
67	Q07509	Q12187	positivo	167	P38824	Q08960	negativo
68	Q12509	Q06697	positivo	168	Q99345	Q12527	negativo
69	P36076	Q12349	positivo	169	P25646	Q02767	negativo
70	P53179	P46995	positivo	170	P53046	P00358	negativo
71	P46984	P33297	positivo	171	P29509	P46961	negativo
72	P32447	P48562	positivo	172	P31385	P38324	negativo
73	Q06675	P52553	positivo	173	P32334	P53050	negativo
74	P40892	P38987	positivo	174	P21190	P41948	negativo
75	P11325	P53891	positivo	175	Q12385	Q12453	negativo
76	Q12136	Q12506	positivo	176	P47020	P50946	negativo
77	P18562	Q06628	positivo	177	P28004	P47043	negativo
78	Q04659	P49956	positivo	178	P39936	P33894	negativo
79	P38311	P29055	positivo	179	P07286	P29340	negativo
80	P47135	P36163	positivo	180	P32480	P48582	negativo
81	P38786	P16622	positivo	181	P39727	P53296	negativo
82	P38065	P53550	positivo	182	P40077	P22515	negativo
83	P37302	P32776	positivo	183	Q05515	P42947	negativo
84	P43568	P41808	positivo	184	Q08486	P33300	negativo
85	Q04949	P41832	positivo	185	P38350	Q08484	negativo
86	P25293	P53939	positivo	186	P23968	P40561	negativo
87	P25559	P40015	positivo	187	P36088	Q12753	negativo
88	P06243	Q06672	positivo	188	P87281	Q12164	negativo
89	Q12163	P53088	positivo	189	P43592	Q12524	negativo
90	P11709	P40957	positivo	190	P38074	P33333	negativo
91	P35497	P38987	positivo	191	Q03330	P53729	negativo
92	P26793	Q06479	positivo	192	P32571	P32656	negativo
93	Q04307	P22276	positivo	193	P38907	P36129	negativo
94	P39981	P38998	positivo	194	Q03941	P49960	negativo
95	P47135	P30283	positivo	195	P17106	O13542	negativo
96	P32569	P40461	positivo	196	P24868	P39941	negativo
97	Q05567	P40395	positivo	197	Q12263	P35735	negativo
98	P32325	P50874	positivo	198	P40498	Q05933	negativo
99	P38828	P38699	positivo	199	P38349	P22804	negativo
100	P38166	Q06208	positivo	200	P53221	P40438	negativo

Apéndice C: Extracción de secuencias

Para lograr la extracción de las secuencias de la base de datos SwissProt, se debió hacer una conversión previa, ya que en el conjunto de datos original de (Ben-Hur y Noble, 2005) los pares de proteínas están dados en OLN (OrderedLocusName), que es una abreviación informalmente asignada en el proceso de designar un gen. Y con sólo el OLN no nos fue posible extraer las secuencias del SwissProt. Para lograrlo, fue necesario establecer una equivalencia del OLN con AC (Accession number) previamente.

Primero, se obtuvo de la página de SwissProt el archivo “yeast.txt” que contiene toda la información de levadura en columnas como: designaciones del gen, OLN, AC, entre otros, de un total de 6657 proteínas (consultado en abril de 2008). Enseguida, se muestra una parte de ese archivo, 50 primeras proteínas y cuatro de sus columnas más importantes, en la Tabla XXI.

Tabla XXI: Extracto del archivo “yeast.txt” obtenido de SwissProt. Se muestran las primeras 50 proteínas del archivo con cuatro de sus columnas más relevantes.

Gene designations (designaciones del gen)	OLN	AC	Length (longitud)
AAC1; YM9796.09C	YMR056C	P04710	309
AAC2; PET9; YBL0421	YBL030C	P18239	318
AAC3; YBR0753	YBR085W	P18238	307
AAD3	YCR107W	P25612	363
AAD4	YDL243C	Q07747	329
AAD6	YFL056C	P43547	212
AAD10; J2245	YJR155W	P47182	288
AAD14; N0300	YNL331C	P42884	376
AAD15	YOL165C	Q08361	143
AAD16	YFL057C	P43546	152
AAH1; N1208; N1825	YNL141W	P53909	347
AAP1	YHR047C	P37898	856
AAR2; YBL0611; YBL06.06	YBL074C	P32357	355
AAT1; YKL461	YKL106W	Q01802	451
AAT2; ASP5; L1746	YLR027C	P23542	418
ABC1	YGL119W	P27697	501
ABD1; YBR1602	YBR236C	P32783	436
ABF2; HIM1; YM9916.11	YMR072W	Q02486	183

ABP140	YOR239W	Q08641	628
ABP1; YCR88W	YCR088W	P15891	592
ABZ1; N3286	YNR033W	P37254	787
ABZ2; YM8021.15	YMR289W	Q03266	374
ACB1; ACB	YGR037C	P31787	87
ACE1; CUP2; G1810	YGL166W	P15315	225
ACE2; L3123; L9606.10	YLR131C	P21192	770
ACF2; ENG2; PCA1; L3180	YLR144C	Q12168	779
ACH1; YBL0304; YBL03.18	YBL015W	P32316	526
ACK1; D1066	YDL203C	Q07622	623
ACM1	YPL267W	Q08981	209
ACN9; D9719.16	YDR511W	Q04401	133
ACO1; GLU1; L8003.22	YLR304C	P19414	778
ACO2; J0327	YJL200C	P39533	789
ACP1	YKL192C	P32463	125
ACR2; ARR2; P9677.16	YPR200C	Q06597	130
ACR3; ARR3; P9677.2	YPR201W	Q06598	404
ACS1; FUN44	YAL054C	Q01574	713
ACS2; L9634.10	YLR153C	P52910	683
ACT1; ABY1; END7	YFL039C	P60010	375
ADA2; D9461.33	YDR448W	Q02336	434
ADA3; NGG1; YD9395.09	YDR176W	P32494	702
ADD66; PBA2; POC2	YKL206C	P36040	267
ADE1	YAR015W	P27616	306
ADE2; O3293; YOR3293C	YOR128C	P21264	571
ADE3; G7733	YGR204W	P07245	946
ADE4; YM9952.02C	YMR300C	P04046	510
ADE5,7	YGL234W	P07244	802
ADE6	YGR061C	P38972	1358
ADE8; D9509.26	YDR408C	P04161	214
ADE12; N1290	YNL220W	P80210	433

Posteriormente, se eliminaron las columnas excedentes del archivo “yeast.txt”, dejando solamente OLN y AC en un archivo llamado “locusyacsolo.txt”. Ver Tabla XXII.

Tabla XXII: Fragmento del archivo "locusyacsolo.txt". Se muestran los primeros diez elementos en columnas correspondientes al OLN y AC, respectivamente.

YMR056C	P04710
YBL030C	P18239
YBR085W	P18238
YCR107W	P25612
YDL243C	Q07747
YFL056C	P43547
YJR155W	P47182
YNL331C	P42884
YOL165C	Q08361
YFL057C	P43546

El conjunto de pares de proteínas a utilizar en la presente investigación se almacenó en un archivo que se le llamó "cienpares_locus.txt" ya que contiene 100 pares de proteínas como ejemplos positivos y 100 pares como ejemplos negativos en OLN. Ver Tabla XXIII.

Tabla XXIII: Fragmento del archivo "cienpares_locus.txt". Se muestran los primeros diez pares de proteínas en OLN separadas por columnas.

YPL094C	YPR086W
YPL043W	YPR072W
YPL070W	YPR193C
YBR254C	YJR116W
YBR061C	YGR219W
YER036C	YJR091C
YLR028C	YLR373C
YDL226C	YGR172C
YFL018C	YGL063W
YGR173W	YMR319C

Luego, se creó un programa en python que lee el archivo "cienpares_locus.txt" y busca en el archivo "locusyacsolo.txt" la equivalencia en

AC del SwissProt a los pares en OLN, obteniéndose otro archivo “pares_ac.txt”. Con esto ya logramos tener nuestro conjunto de datos en AC. Ver Tabla XXIV.

Tabla XXIV: Fragmento del archivo “pares_ac.txt”. Se muestran los primeros diez pares de proteínas, ahora, en AC separadas por columnas.

P21825	P29055
P37838	Q12514
Q02866	Q06592
P38334	P47153
P38238	P53307
P40024	P47135
P54113	Q05934
P35197	P53039
P09624	P53167
P53295	P40988

Finalmente, se creó un programa en python que lee el archivo “pares_ac.txt” y consulta la base de datos de SwissProt, generando un archivo nuevo “seq.txt”, que contiene sólo las secuencias en pares, separadas por coma. A continuación, se muestra un fragmento del archivo mencionado, las comas se encuentran resaltadas en gris.

Tabla XXV: Fragmento del archivo “seq.txt”. Se muestran los primeros diez pares de secuencias de proteínas separadas por comas.

MSAVGPGSNAGASVNGGSATAIATLLRNHKELKQRQGLFQAKQTDFFRYKRFV
RALHSEEYANKSARQPEIYPTIPSNKIEDQLKSREIFIQLIKAQMVIPVKKLHSQEC
KEHGLKPSKDFPHLIVSNKAQLEADEYFVWNYNPRTYMDYLIVIGVVSIILALVC
YPLWPRSMRRGSYYVSLGAFGILAGFFAVAILRLILYVLSLIVYKDVGGFWIFPN
LFEDCGVLESFKPLYGFGEKDTYSYKKKLKRMKKKQAKRESNKKKAINEKAEQ
N,MMTRESIDKRAGRRGPNLNIVLTCPECKVYPPKIVERFSEGDVVCALCGLVLS
DKLVDTRSEWRTFSNDDHNGDDPSRVGEASNPLLDGNNLSTRIGKGETTDMRF
TKELNKAQGKNVMDKKDNEVQAAFAKITMLCDAAELPKIVKDCAKEAYKLCH
DEKTLKGKSMESIMAASILIGCRRAEVARTFKEIQSLIHVKTKEFGKTLNIMKNIL
RGKSEDGFLKIDTDNMSGAQNLTYIPRFCSHLGLPMQVTTSAEYTAKKCKEIKEI
AGKSPITIAVVSIYLNILLFQIPITAAKVGQTLQVTEGTIKSGYKILYEHRDKLVDP
QLIANGVVSLDNLPGVEKK

MEETIENVEVPSSNVSKQNDDGLDMKTLFVRSIPQDVTDEQLADFFSNFAPIKH
AVVVKDTNKRSRGFGFVSFAVEDDTKEALAKARKTKFNGHILRVDIAKRRDRS
KKTSEVVEKSTPESSEKITGQNNEDEDDADGEDSMLKGKPKLIIRNMPWSCRDP
VKLKKIFGRYGTVVEATIPRKRDGKLCGFAFVTMKKISNCRIALENTKDLKIDGR
KVAVDFAVQKNRWEDYKKAQPEMNDKDDNESGNEDAEENHDDEEDENEEED
RQVDQASKNKESKRKAQNKREDFSVFVRNVPYDATEESLAPHFSKFGSVKYAL
PVIDKSTGLAKGTAFVAFKDQYTYNECIKNAPAAGSTSLLIGDDVMPEYVYEGR
VLSITPTLVREDAGRMAEKNAAKRKEALGKAPGEKDRRNLYLLNEGRVVEGSK
MADLLTNTDMEIREKSYKLRVEQLKKNPSLHLSMTRLAIRNLPRAMNDKALKA
LARKAVVEFATEVKNKERHPLSKEEIIRSTKEKYKFMGPDEIEAQKKKDKKSGV
VKQAKVIMEVKGSTAGRSRGYGFVEFRDHKNALMGLRWLNCHAVTSDEILEG
LNDDEKKQVDNDLGKGRRLCVEFAIENSNVVKRRREQLKQARTKRTRPDNEDT
GDVGESENKKPKKEEATTPTNPDDKKMGDDIKRIIGFKRKRKHAKK,MSQRKLQ
QDIDKLLKKVKEGIEDFDDIYEKFQSTDPSNSSHREKLESDLKREIKKLQKHRDQ
IKTWLSKEDVKDKQSVLMTNRRLIENGMERFKSVEKLMKTKQFSKEALTNPDII
KDPKELKKRDQVLFIHDCLDELQKQLEQYEAQENEEQTERHEFHIANLENILKK
LQNNEMDPEPVEEFQDDIKYYVENNDDPDFIEYDTIYEDMGCEIQPSSSNNEAPK
EGNNQTSLSSIRSSKKQERSPKKKAPQRDVSISDRATTPIAPGVESASQSISSTPTP
VSTDTPLHTVKDDSIKFDNSTLGTPTTHVSMKKKESENDSEQQLNFPPDRTDEIR
KTIQHDVETNAAFQNPLFNDELKYWLDSKRYLMQPLQEMSPKMVSQLESSLLN
CPDSLDADSPCLYTKPLSLPHPTSIFFPNEPIRFVYPYDVPLNLTNNENDTDNKFG
KDSKAKSKKDDDIYSRTSLARIFMKFDLDTLFFIFYHYQGSYEQFLAARELFKNR
NWLFNKVDRCWYYKEIEKLPPGMGKSEEESWRYFDYKKSWLARRCGNDFVY
NEEDFEKL

MARQLFTPPITNPRFDPNQSIRESYKNTTGGMQFQQNLHEDQNDNERSSCDGDE
NSTTGERLENNKSPILTKQEIDEALNTVTNLPPELSKLIDIFIDDLKQPKYVRPLSV
LQLSSLFQSFYIKFDKASFQHVSSANNNGYYFSGGGSSSFLAAKETLSSGLSGIFG
RSRSSSGNSLMRPRRSSSLFSNESISNSTNATQMLSPEEIKKQLKINELNNMKIEK
YMELCERDVFKKILIVGTSVSSPNKMKTFKPHQLQTFKVGNLFRNSVEFTEYNK
LLNEKILCLSKLSTMNKINLIKFLSLNNGIDPEPKFEEIKDILYEFTYHSISPCEKIK
ALLKLHEIMTYSQEMSNDDYLSLLIYYIITIVPRDIFLNAEFIRLFRYKKKLVETES
FALTNLEAALVFVEGLTKNDFSNELQDKLTVNESKILENSISSRVSLPSKTAIMH
KNNGNNGSNLGDIVTPTIQRPDVTRSNSYDGFRTVFDSSLKNIIGKIRSYTPPHPN
NTSNNNLHSSNNLNIPRSSSQLSMELSNRDTTEMSRDGSRSTSSSSRSSASLEHG
NREFTGDLTVTASINGADKKEFQKSWKKYKGYKFEDLTICELRDLFEIYQKMM
Q,MSNTSEDNITVRFVTENDKEGWQRLWKSYQDFYEVSFPDDLDDFNFGRFLDP
NIKMWAAVAVESSSEKIIGMINFFNHMTTWDFKDKIYINDLYVDENSRVKGAG
GKLIQFVYDEADKLGTPSVYWCTDESNHRAQLLYVKVGYKAPKILYKRKGY

MPQYFAIIGKKDNPVYEIEFTNAENPQGFPQDLKELNPFILHASLDIVEDLQWQIN
PTSQLNGNGGNGSNGGGGFLRSRAVNNTDNCYLGKVDHFYGLAITAYISYSGM
KFVMIHGNSANSSVVIDDNNMRSFYQEVHELYVKTLMNPFYKITDPIRSPAFDS
RVRTLARKHLSK,MNANSTTTAIGLTSPFEKLSFFPHSSNLILAHLHEIIFSFVFYQ
LAFSVVAPFLNKVVFRKHYTTIRDPLLKIDFNVHTVSMIQAVVSNTVLLPTLTTP
MHYNVVTYTDSYSSMVSSLSAGYFIWDLTMCVRYFKLYGLEFTGHAIGSVYV

MLLSLRPFCQPWIGRFLIYEASTPFVNINWFIMQCNAKSKNSIPLWFNVVNGLLL MTVFFVVRICWGSIASALLFRQMWKVRDELPKFSAVTMMSLNIFMNLLNVLWF KKMIRIAKKLAKPAPTSKLD
MGKSSKDKRDLYYRKAKEQGYRARSAFKLLQLNDQFHFLDDPNLKRVVDLCA APGSWSQVLSRKLFDESPSSDKEDRKIVSVDLQPMSPIPHVTTLQADITHPKTLA RILKLFGNEKADFVCSDGAPDVTGLHDLDEYVQQQLIMSALQLTACILKKGGTF VAKIFRGRDIDMLYSQLGYLFDKIVCAKPRSSRGTSLEAFIVCLGYNPPSNWTPK LDVNTSVDEFFQGCFLNKLCISDKLSHWNEEERNIAEFMACGSLQSFDSDATYH DLPSSVAGTSSSLDPVQSPTNPPYKKALELKRSGKLTRSV,MYMIVVKYLYALCS SFFDCILNFNETVFGPSHRAFNPNNIIFIVDFQNFNILDSNMLVSHMTRHLSSWQY PTWVLVLTIRTTVSMHNRCTMRCSQTLESIPFHHTSKTFAFAD
MPPVSASKAKRDAKKAEREAKKAAAGKTIRKLGRKKEAAAEESEVDAAAREIK MMKLQQDKDGLSDRVVTGVLSSLETSRDIKLSSVSLLFHGKVLIQDSGLELNYG RRYGLLGENGCGKSTFLKALATREYPIPEHIDIYLLDEPAEPSELSALDYVVTEA QHELKRIEDLVEKTILEDGPESELLEPLYERMDSLDPDTFESRAAIILIGLGFNKKT ILKKTKDMSGGWKMRVALAKALFVKPTLLLLDDPTAHLDLEACVWLEEYLKR FDRTLVLVSHSQDFLNGVCTNMIDMRAQKLTAYGGNYDSYHKTRSELETNQM KQYNKQQEEIQHIKKFIASAGTYANLVKQAKSRQKILDKMEADGLVQPVVPDK VFSFRFPQVERLPPPVLAFDDISFHYESNPSENLYEHLNFGVDMDSRIALVGPNG VGKSTLLKIMTGELTPQSGRVSRHTHVKLGVYSQHSQDQLDLTKSALEFVRDK YSNISQDFQFWRGQLGRYGLTGEGQTVQMATLSEGQRSRVVFALLALEQPNVL LLDEPTNGLDIPTIDSLADAINEFNGGVVVVSHDFRLLDKIAQDIFVVENKTATR WDGSILQYKNKLAKNVVL,MDKSKQMNINNLSNIPEVIDPGITIPIYEEEYENNGE SNSQLQQQPQKLGSYRSRAGKFSNTLSNLLPSISAKLHHSKKNSHGKNGAEFSSS NNSSQSTVASKTPRASPSRSKMMESSIDGVTMDRPGSLTPPQDMEKLVHFPDSS NNFLIPAPRGSSDSFNLPHQISRTRNNTMSSQITSISSIAPKPRTSSGIWSSNASAND PMQQHLLQQLQPTTSNNTTNSNTLNDYSTKTAYFDNMVSTSGSQMADNKMNT NNLAIPNSVWSNTRQRSQSNASSIYTDAPLYEQPARASISSHYTIPTQESPLIADEI DPQSINWVTMDPTVPSINQISNLLPTNTISISNVFPLQHQQPQLNNAINLTSTSLAT LCSKYGEVISARTLRNLNMALVEFSSVESAVKALDSLQGKEVSMIGAPSKISFAK ILPMHQQPPQFLLNSQGLPLGLENNNLQPQPLLQEQLFNGAVTFQQQGNVSIPVF NQQSQQSQHQNHSSGSAGFSNVLHGYNNNNSMHGNNNNSANEKEQCPFPLPPP NVNEKEDLLREIIELFEANSDEYQINSLIKKSLNHKGTSDTQNFGPLPEPLSGREF DPPKLRELRKSIDSNAFSDLEIEQLAIAMLDELPELSSDYLGNTIVQKLFEHSSDII KDIMLRKTSKYLTSMGVHKNGTWACQKMITMAHTPRQIMQVTQGVKDYCTPL INDQFGNYVIQCVLKFGFPWNQFIFESIIANFWVIVQNRYGARAVRACLEAHDIV TPEQSIVLSAMIVTYAEYLSTNSNGALLVTWFLDTSVLPNRHSILAPRLTKRIVEL CGHRLASLTILKVLNYRGDDNARKIILDSLFGNVNAHDSSPPKELTKLLCETNYG PTFVHKVLAMPLLEDDLRAHIIKQVRKVLTDSTQIQPSRRLLEEVGLASPSSTHN KTKQQQQQHHNSSISHMFATPDTSGQHMRGLSVSSVKSGGSKHTTMNTTTTNG SSASTLSPGQPLNANSNSSMGYFSYPGVFPVSGFSGNASNGYAMNNDDLSSQFD MLNFNNGTRLSLPQLSLTNHNNTTMELVNNVGSSQPHTNNNNNNNNTNYNDD NTVFETLTLHSAN
MGKYTKTAILSVYDKTGLLDLAKGLVENNVRILASGGTANMVREAGFPVDDVS

SITHAPEMLGGRVKTLHPAVHAGILARNLEGDEKDLKEQHIDKVDFVVCNLYPF KETVAKIGVTVQEAVEEIDIGGVTLLRAAAKNHSRVTILSDPNDYSIFLQDLSKD GEISQDLRNRFALKAFEHTADYDAAISDFFRKQYSEGKAQLPLRYGCNPHQRPA QAYITQQEELPFKVLCGTPGYINLLDALNSWPLVKELSASLNLPAAASFKHVSPA GAAVGLPLSDVERQVYFVNDMEDLSPLACAYARARGADRMSSFGDFIALSNIV DVATAKIISKEVSDGVIAPGYEPEALNILSKKKNGKYCILQIDPNYVPGQMESRE VFGVTLQQKRNDAIINQSTFKEIVSKNKALTEQAVIDLTVATLVLKYTQSNSVCY AKNGMVVGLGAGQQSRIHCTRLAGDKTDNWWLRQHPKVLNMKWAKGIKRA DKSNAIDLFVTGQRIEGPEKVDYESKFEEVPEPFTKEERLEWLSKLNNVSLSSDA FFPFPDNVYRAVQSGVKFITAPSGSVMDKVVFQAADSFDIVYVENPIRLFHH,MR AMDTQVQSAERGLVLPPMNSTVSSATAATTATNTDTDTDGDRDEERESLAEDG SEWVPAYMLTRDRSRYLGHFLGVDKMLEAVKCKYCGVIIRRQGNSISMAEASQ THLWSTHKIDPNANYYSGWTGVEAGSTFMVRPPLKNHQGGSATTNSIANLLEID EDFLKRTREKEMALPLVQSLAIIIASENLPLSFVDNTAVRLLINQNANSLSFIDHD LILNAIRSIAYNLDRIIQRTALRNNSDLSLIIDKNYLLMDPTDRSNQLSNRLKNQL FEMQKINFFSLSHSVWNNTISILSIQYYDDFHSQVKTLPLIIQNLHEYNNDPKLSIP AQLLKISQELPGLQNTVISITLPRSQIVDLLNVMDSQPFFPNTYTNAKNYYHNCII SIINSAILPLFGTPKSADITHPRQSSFSKEPLTLLDSLIDLSNIDISNSIFSRINSFLDD LQSNSWQLDKFRSLCEKFGFEFVCSKFDLSRYSTATVSLQTFLNLRPIIEEYQSSI QIEKFNEIDFQIIDYLLITLNSINRILKFFTSSKSLNFTYVLFAIMSIEKHLLSTLGSL QFQRLIAPFETFLSKIQEFKTILFSDDMNLLAMFLCPAILFEREVLEYSFHTISLSEI VDKLSTSIFSLLKRFLNLHTIGNVNNSHNTSNHSNMNIHTDNQTNNINNRSGNNS DNNDNEHDNDNDNHSNSNTPASRIDIDPTGGENSVLPEQQPQNSNNNLSFGSLS DTHHLSDSTISKEIDSIFLQIIQEDLYDYLSTVNSIVPISYRSYCEQSNFIRDSGRFK KRIITEDSIIGELEQPMNFIEELLDIHVPVCNAFWSQYLDNDAGPIIRILFKIMQCQS SSSIRGEYSFLNDFIPRVHPDLTQEIIKIKLFNDQFVASKVDYDLDTLQTASQYLP
MSDWKVDPDTRRRLLQLQKIGANKKCMDCGAPNPQWATPKFGAFICLECAGIH RGLGVHISFVRSITMDQFKPEELLRMEKGGNEPLTEWFKSHNIDLSLPQKVKYD NPVAEDYKEKLTCLCEDRVFEEREHLDFDASKLSATSQTAASATPGVAQSREGT PLENRRSATPANSSNGANFQKEKNEAYFAELGKKNQSRPDHLPPSQGGKYQGF GSTPAKPPQERSAGSSNTLSLENFQADPLGTLSRGWGLFSSAVTKSFEDVNETVI KPHVQQWQSGELSEETKRAAAQFGQKFQETSSYGFQAFSNFTKNFNGNAEDSS TAGNTTHTEYQKIDNNDKKNEQDEDKWDDF,MSFYNTSNNANNGGGFYQPSA QFAVPQGSMSFQNTVGSSNTGNDNNLGVAPDPLPVGILHALSTKGYPHEPPLLE EIGINFDHIITKTKMVLIPIRFGSGVPQEILNDSDLAGPLIFFLLFGLFLLMAGKVHF GYIYGVALFGTISLHNLSKLMSNNDTSTQTNLQFFNTASILGYCFLPLCFLSLLGI FHGLNNTTGYVVSVLFVIWSTWTSSGFLNSLLQLQNARLLIAYPLLIFYSVFALM VIFV
MLRIRSLLNNKRAFSSTVRTLTINKSHDVVIIGGGPAGYVAAIKAAQLGFNTACV EKRGKLGGTCLNVGCIPSKALLNNSHLFHQMHTEAQKRGIDVNGDIKINVANFQ KAKDDAVKQLTGGIELLFKKNKVTYYKGNGSFEDETKIRVTPVDGLEGTVKED HILDVKNIIVATGSEVTPFPGIEIDEEKIVSSTGALSLKEIPKRLTIIGGGIIGLEMGS VYSRLGSKVTVVEFQPQIGASMDGEVAKATQKFLKKQGLDFKLSTKVISAKRN DDKNVVEIVVEDTKTNKQENLEAEVLLVAVGRRPYIAGLGAEKIGLEVDKRGR

LVIDDQFNSKFPHIKVVGDVTFGPMLAHKAEEEGIAAVEMLKTGHGHVNYNNIP SVMYSHPEVAWVGKTEEQLKEAGIDYKIGKFPFAANSRAKTNQDTEGFVKILID SKTERILGAHIIGPNAGEMIAEAGLALEYGASAEDVARVCHAHPTLSEAFKEAN MAAYDKAIHC,MLLGYCGSGYYGMQYNPPHKTIEGEILTKLFDVGAISEENSLA PKKNSFMAAARTDKGVHAMLNLLSLKITLREDTVAKLNAALPPEIRVWGIQPV NKKFNARSACDSRWYQYLIPEFILIGPPRSSLLHRNVGGSYREDGSQEVWDTFLE QTRGRFSGDELCRLQDTAQKLSESDPLVQDYVGLLSGTLSGYCLSPSKLDAFEA AMQEYVGTHNFHNFTTGKLWGDPSAQRHIKKVVVSQASPGWICVRIHGQSFML HQIRRMVALAVLAARCQLPPNIVRNYFNAGPRKYIPRAPAQGLLLEGPVFDGYN TKLRNLLYCEIRPDDITLERMCRFRERQICTAIAHEETQRHVFCHFVRQMNRLAT PLI
MGIIDKIKAIEEEMARTQKNKATEHHLGLLKGKLARYRQQLLADEAGSGGGGG SGFEVAKSGDARVVLIGYPSVGKSSLLGKITTTKSEIAHYAFTTLTSVPGVLKYQ GAEIQIVDLPGIIYGASQGKGRGRQVVATARTADLVLMVLDATKSEHQRASLEK ELENVGIRLNKEKPNIYYKKKETGGVKVTFTSPPKTNLTEQAIKMILRDYRIHNA EVLVRDDQCTIDDFIDVINEQHRNYVKCLYVYNKIDAVSLEEVDKLAREPNTVV MSCEMDLGLQDVVEEIWYQLNLSRVYTKKRGVRPVFDDPLVVRNNSTIGDLCH GIHRDFKDKFKYALVWGSSAKHSPQKCGLNHRIDDEDVVSLFAK,MGKIAEFLG NPGARPDVHHRAPTVDCKQYEEFGDSNDYKNDDVVRVVSHSDESTDDELCNV NLTETGAIFTSKGFTGLSKGFTDKTLDFLVRVAGSQAVFFIVWIILIIWVVIGIVYN APFNWQVVMQDGQSIQSYVWDTLLMRQQLMSTHEQILICGRLKSRLASFKNYL TRSTPEEEKADCTVEANEVSSVENHIDPSAINGELPVENWYDRLSNVASRYMGSI AAMVIFWIGIFVWIGCGAIPKDAGNTPPYTGETTGSNPRLKKFSDAWQMYINTA VAVSLLICTTFLQNIRARHDYFTGRFLVDIFDMDEKIDYRIRKHFNDFETPHPVV TIESKKRSTGRKMIDWYADIIGTGIGVLIGVAVFATWIGIGSPMKWDDNWWLIIG TYTGLIGFLDGFVLREVYFRIVQHEEKNYSDVAKEDLELFQELGIECPEEFSGKA PEINTIGYRTSQYINRICSTPWSVLVSVIIIIGLICIASGLRWSTTGQLIANTPTMIIEE FFLLVLLQAHNWADRQRRVEVTALYARRRILLSYVEKRFPEVMMLEK

Apéndice D: Código de aminoácidos

Tabla XXVI: Código de aminoácidos obtenido de http://www.expasy.org/sprot/userman.html#AA_codes.

One-letter code (Código de una letra)	Three-letter code (Código de tres letras)	Amino-acid name (Nombre del aminoácido)
A	Ala	Alanine
R	Arg	Arginine
N	Asn	Asparagine
D	Asp	Aspartic acid
C	Cys	Cysteine
Q	Gln	Glutamine
E	Glu	Glutamic acid
G	Gly	Glycine
H	His	Histidine
I	Ile	Isoleucine
L	Leu	Leucine
K	Lys	Lysine
M	Met	Methionine
F	Phe	Phenylalanine
P	Pro	Proline
S	Ser	Serine
T	Thr	Threonine
W	Trp	Tryptophan
Y	Tyr	Tyrosine
V	Val	Valine
O	Pyl	Pyrrolysine
U	Sec	Selenocysteine
B	Asx	Aspartic acid or Asparagine
Z	Glx	Glutamic acid or Glutamine
X	Xaa	Any amino acid

Apéndice E: Formato en LIBSVM

En LibSVM 2.86 (Hsu *et al*., 2007) existe la opción de proveerle a la SVM una matriz de datos precalculados, que en nuestro caso es la matriz Kp de cada *p*. Pero para que LibSVM 2.86 pueda leer los datos, se hace necesario ponerlos en el formato adecuado. Por lo que fue necesaria la creación de un programa en Matlab para ello.

Por ejemplo, la matriz Kp de *p*=3 sería como la mostrada en la siguiente tabla, aunque completa consta de 200 x 200 datos.

Tabla XXVII: Fragmento de la matriz Kp para p=3. Se presentan los primeros 10 x 9 elementos.

116500	3559	1087	314	277	6342	4198	512	1202
3559	574758	4535	1372	859	25350	14421	2136	6072
1087	4535	123769	375	271	5608	4879	666	1544
314	1372	375	51649	80	3762	2592	623	770
277	859	271	80	38198	1904	1187	146	262
6342	25350	5608	3762	1904	1176579	46751	6007	11305
4198	14421	4879	2592	1187	46751	815326	4123	6440
512	2136	666	623	146	6007	4123	105585	1429
1202	6072	1544	770	262	11305	6440	1429	232530
1947	6439	2046	951	409	8300	8457	1532	4274

Ahora, tras la ejecución del programa para darle formato a los datos, se vería como se muestra a continuación, para una proporción de 50-50. De una matriz Kp de 200 x 200 datos, se tomaron los primeros 1 al 70 como ejemplos positivos y del 101 al 170 como ejemplos negativos para entrenamiento (Tabla XXVIII). Y el resto para prueba (Tabla XXIX).

Tabla XXVIII: Fragmento de la matriz Kp para p=3 en formato coherente con LibSVM 2.86, primeros 10 x 140 datos de entrenamiento.

1 0:1 1:116500 2:3559 3:1087 4:314 5:277 6:6342 7:4198 8:512 9:1202 10:1947 11:145 12:113 13:708 14:1551 15:978 16:3612 17:1065 18:1980 19:490 20:1080 21:2177 22:185 23:944 24:971 25:1295 26:684 27:403 28:518 29:2429 30:234 31:2377 32:10931 33:1435 34:1938 35:706 36:717 37:9841 38:1914 39:642 40:1788 41:4252 42:4138 43:743 44:7169 45:1872 46:907 47:4625 48:916 49:6724 50:414 51:7056 52:11659 53:416 54:1596 55:234 56:1739 57:3833 58:5087 59:792 60:878 61:381 62:3960 63:184 64:466 65:1656 66:297 67:138 68:1019 69:543 70:3044 101:766 102:418 103:4702 104:558 105:1675 106:456 107:966 108:4662 109:9463 110:913 111:1176 112:195 113:1051 114:5676 115:1690 116:2382 117:2035 118:130 119:2724 120:1008 121:516 122:11792 123:614 124:861 125:3690 126:1125 127:1767 128:3513 129:446 130:1788 131:4707 132:871 133:6800 134:1840 135:6796 136:2238 137:516 138:1544 139:22755 140:346 141:778 142:2418 143:432 144:766 145:4649 146:602 147:172 148:519 149:230 150:1369 151:682 152:425 153:1055 154:1914 155:966 156:274 157:487 158:1800 159:1464 160:1453 161:1317 162:2104 163:723 164:2608 165:1150 166:7395 167:584 168:1032 169:910 170:2943
1 0:2 1:3559 2:574758 3:4535 4:1372 5:859 6:25350 7:14421 8:2136 9:6072 10:6439 11:778 12:662 13:4764 14:9603 15:6596 16:19178 17:17294 18:9478 19:1197 20:5853 21:14661 22:1210 23:9072 24:6402 25:6930 26:3898 27:2451 28:2069 29:17293 30:610 31:15580 32:70989 33:15048 34:8142 35:4721 36:2831 37:69286 38:9966 39:4548 40:9158 41:19787 42:12905 43:2748 44:40334 45:6709 46:3944 47:29075 48:1875 49:21905 50:2146 51:40868 52:71607 53:4462 54:9504 55:1977 56:9196 57:30776 58:22908 59:4032 60:3198 61:1008 62:20895 63:1231 64:2244 65:5008 66:1134 67:762 68:7187 69:4344 70:22269 101:3976 102:2763 103:26809 104:3700 105:7212 106:1966 107:4968 108:33397 109:47577 110:6790 111:5961 112:1726 113:5247 114:30871 115:13742 116:13077 117:10952 118:377 119:10496 120:3300 121:1474 122:74012 123:2436 124:3528 125:19548 126:3778 127:6966 128:15142 129:3545 130:13955 131:36113 132:2670 133:39818 134:9672 135:38992 136:14702 137:3484 138:6376 139:116328 140:1550 141:3022 142:14028 143:2543 144:4781 145:33792 146:3516 147:534 148:1865 149:559 150:7051 151:2262 152:1441 153:7186 154:6902 155:2517 156:893 157:1206 158:9891 159:7740 160:8010 161:10671 162:11523 163:3072 164:14757 165:4677 166:30687 167:6166 168:4751 169:3330 170:14068
1 0:3 1:1087 2:4535 3:123769 4:375 5:271 6:5608 7:4879 8:666 9:1544 10:2046 11:86 12:114 13:966 14:3036 15:1232 16:4662 17:1497 18:2783 19:588 20:2004 21:5395 22:238 23:1563 24:2321 25:1255 26:351 27:768 28:307 29:4279 30:213 31:5558 32:16523 33:1725 34:1748 35:568 36:738 37:28583 38:2525 39:1276 40:2190 41:7013 42:3660 43:399 44:11259 45:1476 46:1464 47:6158 48:625 49:5985 50:525 51:11868 52:13726 53:406 54:2031 55:571 56:1821 57:10634 58:4341 59:1012 60:768 61:364 62:4183 63:129 64:623 65:2096 66:238 67:164 68:1988 69:340 70:5362 101:978 102:603 103:6724 104:747 105:1533 106:84 107:1284 108:6210 109:16615 110:1808 111:1071 112:89 113:761 114:7194 115:1916 116:2335 117:2289 118:63 119:1905

120:900 121:580 122:16068 123:407 124:623 125:3888 126:1116 127:1552 128:2546 129:982 130:6360 131:7835 132:485 133:9266 134:1104 135:6568 136:2595 137:1449 138:1824 139:32311 140:470 141:989 142:4720 143:600 144:1376 145:15042 146:572 147:71 148:705 149:133 150:1164 151:437 152:514 153:2024 154:2230 155:504 156:262 157:380 158:1725 159:3527 160:1522 161:1670 162:1515 163:696 164:2494 165:1337 166:5424 167:1947 168:673 169:1506 170:3502
1 0:4 1:314 2:1372 3:375 4:51649 5:80 6:3762 7:2592 8:623 9:770 10:951 11:54 12:39 13:421 14:1008 15:312 16:1102 17:438 18:1217 19:172 20:539 21:1664 22:81 23:282 24:662 25:483 26:284 27:197 28:91 29:1216 30:88 31:1117 32:3275 33:366 34:1205 35:228 36:328 37:6486 38:998 39:287 40:806 41:2779 42:1398 43:285 44:2997 45:615 46:291 47:3410 48:644 49:3479 50:215 51:5979 52:5362 53:204 54:844 55:231 56:950 57:3860 58:2021 59:230 60:182 61:182 62:3312 63:71 64:166 65:526 66:102 67:130 68:472 69:138 70:1347 101:662 102:354 103:1593 104:308 105:794 106:209 107:924 108:2664 109:5952 110:776 111:385 112:135 113:560 114:2898 115:838 116:1542 117:731 118:74 119:1111 120:456 121:165 122:5008 123:262 124:501 125:1188 126:160 127:1006 128:1310 129:294 130:2005 131:2748 132:527 133:3410 134:518 135:2912 136:776 137:288 138:649 139:13740 140:100 141:227 142:864 143:315 144:460 145:3292 146:222 147:77 148:215 149:72 150:512 151:136 152:288 153:858 154:792 155:542 156:192 157:156 158:1151 159:519 160:540 161:396 162:899 163:331 164:1138 165:968 166:3320 167:391 168:188 169:330 170:1539
1 0:5 1:277 2:859 3:271 4:80 5:38198 6:1904 7:1187 8:146 9:262 10:409 11:42 12:9 13:448 14:857 15:496 16:898 17:556 18:532 19:121 20:604 21:1126 22:83 23:331 24:318 25:132 26:264 27:175 28:180 29:615 30:111 31:1190 32:3572 33:796 34:268 35:104 36:149 37:6098 38:328 39:207 40:518 41:1392 42:1130 43:123 44:2042 45:151 46:405 47:2080 48:552 49:2012 50:304 51:4530 52:2295 53:117 54:599 55:194 56:608 57:1928 58:1492 59:179 60:93 61:111 62:1270 63:83 64:96 65:621 66:79 67:73 68:420 69:62 70:1530 101:369 102:86 103:2022 104:130 105:628 106:107 107:678 108:2454 109:3593 110:451 111:258 112:36 113:272 114:1417 115:364 116:840 117:622 118:72 119:871 120:96 121:117 122:3520 123:288 124:130 125:607 126:192 127:673 128:496 129:192 130:919 131:1986 132:294 133:2316 134:153 135:1842 136:814 137:260 138:514 139:7060 140:117 141:94 142:912 143:264 144:352 145:2293 146:86 147:51 148:150 149:40 150:578 151:30 152:87 153:870 154:615 155:232 156:121 157:145 158:263 159:410 160:361 161:230 162:312 163:199 164:701 165:311 166:983 167:339 168:391 169:146 170:1132
1 0:6 1:6342 2:25350 3:5608 4:3762 5:1904 6:1.17658e+006 7:46751 8:6007 9:11305 10:8300 11:870 12:1820 13:9504 14:15024 15:12738 16:21968 17:11104 18:18072 19:3222 20:13455 21:27035 22:2394 23:13893 24:11436 25:9002 26:11166 27:5035 28:2890 29:23380 30:1209 31:37860 32:73330 33:12120 34:21462 35:3517 36:4288 37:173442 38:15993 39:6132 40:13482 41:39999 42:25746 43:7124 44:65118 45:11438 46:11548 47:163269 48:22392 49:39210 50:6734 51:160490 52:109424 53:6288 54:23279 55:5084 56:18945 57:64482 58:38511 59:5864 60:4460 61:3936 62:55363 63:1110 64:3171 65:20900 66:2248 67:1101 68:11485 69:5259 70:31699 101:11748 102:3615 103:45918 104:5902 105:25234 106:2822 107:20853 108:77086 109:107108 110:13603 111:5525 112:1687 113:7524 114:61905 115:14342 116:17724

117:23001 118:551 119:15942 120:3787 121:4396 122:99594 123:4712 124:8904 125:23164 126:7364 127:12322 128:35640 129:6552 130:34276 131:57681 132:7766 133:53970 134:8830 135:54460 136:19867 137:5565 138:12631 139:224284 140:3438 141:7092 142:26667 143:6040 144:7068 145:82184 146:3030 147:947 148:3555 149:1190 150:10564 151:3854 152:4609 153:12970 154:19038 155:10580 156:3063 157:2412 158:21058 159:12420 160:11746 161:16631 162:12063 163:8020 164:18029 165:8671 166:50352 167:6363 168:5339 169:5651 170:31260
1 0:7 1:4198 2:14421 3:4879 4:2592 5:1187 6:46751 7:815326 8:4123 9:6440 10:8457 11:775 12:414 13:5682 14:13593 15:7656 16:15047 17:9198 18:13890 19:1782 20:8860 21:23576 22:1139 23:8652 24:10743 25:7366 26:7050 27:2803 28:2208 29:14275 30:788 31:23581 32:57164 33:8674 34:14180 35:2184 36:4453 37:112538 38:13766 39:6045 40:9491 41:25392 42:17249 43:2579 44:43334 45:8119 46:7411 47:50180 48:7305 49:31500 50:4417 51:78426 52:88236 53:3888 54:11065 55:3082 56:11217 57:44513 58:31196 59:3734 60:3585 61:2056 62:36627 63:790 64:4176 65:9412 66:1346 67:746 68:9240 69:4393 70:23580 101:7925 102:3864 103:25200 104:5653 105:9159 106:1523 107:11967 108:39397 109:88124 110:15295 111:5158 112:732 113:5033 114:33460 115:11674 116:16958 117:19184 118:565 119:13119 120:4824 121:3042 122:68140 123:3483 124:5514 125:20857 126:4555 127:9224 128:20843 129:3954 130:24466 131:44550 132:6951 133:39580 134:6072 135:42218 136:12948 137:5292 138:6826 139:169338 140:2468 141:4759 142:19985 143:7146 144:5608 145:48010 146:3232 147:926 148:3099 149:614 150:8917 151:1890 152:2397 153:9593 154:12138 155:8435 156:1668 157:2158 158:21947 159:10749 160:9996 161:9008 162:8562 163:5618 164:21996 165:8184 166:41914 167:7980 168:4917 169:7482 170:26163
1 0:8 1:512 2:2136 3:666 4:623 5:146 6:6007 7:4123 8:105585 9:1429 10:1532 11:100 12:180 13:733 14:1431 15:1725 16:2676 17:1187 18:1320 19:335 20:978 21:2514 22:189 23:1016 24:2052 25:710 26:1358 27:915 28:150 29:2315 30:150 31:3447 32:7191 33:1901 34:1924 35:259 36:341 37:15987 38:1922 39:663 40:1601 41:4410 42:2531 43:790 44:6769 45:1082 46:1048 47:8901 48:986 49:4320 50:776 51:19999 52:11654 53:384 54:2172 55:513 56:1561 57:8664 58:4295 59:727 60:456 61:409 62:5210 63:40 64:511 65:1762 66:253 67:51 68:1284 69:192 70:2996 101:884 102:384 103:3723 104:435 105:1676 106:204 107:1876 108:3973 109:10994 110:1414 111:724 112:350 113:1072 114:6030 115:935 116:2728 117:1764 118:59 119:1428 120:426 121:346 122:14253 123:465 124:695 125:2252 126:747 127:1736 128:3213 129:810 130:3220 131:7526 132:854 133:7395 134:1338 135:7076 136:1775 137:634 138:1375 139:21539 140:402 141:679 142:2391 143:728 144:971 145:12457 146:415 147:106 148:338 149:53 150:1497 151:266 152:384 153:1401 154:2185 155:1691 156:248 157:468 158:2672 159:1696 160:1436 161:1248 162:1407 163:600 164:2184 165:998 166:7146 167:1398 168:812 169:796 170:2016
1 0:9 1:1202 2:6072 3:1544 4:770 5:262 6:11305 7:6440 8:1429 9:232530 10:4274 11:278 12:291 13:1433 14:3487 15:1735 16:5542 17:5328 18:4288 19:504 20:2133 21:5116 22:679 23:1796 24:1944 25:2419 26:1876 27:746 28:733 29:5183 30:337 31:6154 32:14659 33:5004 34:3904 35:546 36:1170 37:28179 38:4460 39:1543 40:3771 41:7360 42:8391 43:1342 44:9748 45:2939 46:1794 47:10407 48:1678

49:6674 50:1290 51:14900 52:15594 53:1404 54:3161 55:932 56:2673 57:9882 58:7535 59:1107 60:572 61:832 62:10392 63:436 64:488 65:3536 66:612 67:143 68:2665 69:1150 70:5994 101:1560 102:1131 103:8432 104:1836 105:2450 106:790 107:1771 108:11568 109:17365 110:2504 111:982 112:506 113:1628 114:10552 115:3790 116:4059 117:4286 118:140 119:4460 120:846 121:552 122:24651 123:980 124:1512 125:6515 126:1300 127:3316 128:4824 129:1396 130:3827 131:8045 132:1710 133:13487 134:1995 135:13857 136:3454 137:848 138:2069 139:46516 140:467 141:1122 142:6573 143:1616 144:2282 145:9606 146:1465 147:152 148:615 149:528 150:2684 151:652 152:777 153:2781 154:4263 155:1909 156:558 157:626 158:5616 159:3318 160:2153 161:2165 162:2062 163:1246 164:3580 165:1660 166:9275 167:1543 168:1355 169:1598 170:7708
1 0:10 1:1947 2:6439 3:2046 4:951 5:409 6:8300 7:8457 8:1532 9:4274 10:257337 11:562 12:330 13:2295 14:4925 15:2064 16:5410 17:5088 18:3615 19:834 20:1632 21:4573 22:240 23:1606 24:2635 25:3082 26:2199 27:990 28:930 29:3950 30:283 31:5248 32:21232 33:5025 34:4103 35:990 36:1280 37:24364 38:6193 39:1202 40:3008 41:6003 42:5015 43:1604 44:13842 45:2471 46:2030 47:7956 48:1276 49:11550 50:1422 51:20667 52:18448 53:994 54:3869 55:992 56:3252 57:12542 58:8570 59:1324 60:1168 61:1112 62:9938 63:563 64:899 65:3088 66:594 67:424 68:2355 69:1560 70:6348 101:2212 102:793 103:9744 104:1040 105:2796 106:630 107:1994 108:13482 109:17003 110:3386 111:2482 112:732 113:1638 114:9333 115:3651 116:4006 117:3398 118:194 119:3405 120:748 121:524 122:23612 123:1039 124:1848 125:6682 126:1664 127:3717 128:4758 129:1758 130:4021 131:11760 132:815 133:12951 134:1846 135:13203 136:4330 137:1329 138:2480 139:42194 140:831 141:1864 142:4840 143:1758 144:2605 145:11356 146:957 147:212 148:1111 149:617 150:2472 151:648 152:663 153:3155 154:5519 155:2611 156:480 157:909 158:5688 159:2574 160:2880 161:1974 162:2903 163:1652 164:5082 165:2206 166:14700 167:1625 168:1268 169:1553 170:6885

Tabla XXIX: Fragmento de la matriz Kp para p=3 en formato coherente con LibSVM 2.86, primeros 10 x 140 datos de prueba.

1 0:71 1:640 2:2365 3:575 4:107 5:125 6:3414 7:3488 8:205 9:768 10:1465 11:150 12:143 13:568 14:1412 15:692 16:2208 17:1259 18:1772 19:224 20:616 21:2146 22:188 23:786 24:486 25:986 26:229 27:154 28:265 29:2047 30:67 31:1652 32:6939 33:1091 34:728 35:328 36:256 37:6422 38:1088 39:428 40:1300 41:1712 42:1812 43:755 44:3767 45:975 46:641 47:1943 48:284 49:3022 50:508 51:4104 52:7681 53:368 54:1203 55:393 56:980 57:3918 58:2484 59:250 60:378 61:172 62:3571 63:124 64:339 65:996 66:124 67:159 68:1435 69:384 70:2058 101:593 102:353 103:3140 104:540 105:980 106:201 107:450 108:4176 109:4396 110:750 111:575 112:273 113:424 114:4356 115:1700 116:1979 117:930 118:59 119:1007 120:485 121:348 122:6609 123:400 124:635 125:2216 126:590 127:820 128:1499 129:378 130:1337 131:4798 132:223 133:5220 134:886 135:3956 136:1164 137:636 138:510 139:17001 140:202 141:552 142:1840 143:780 144:500 145:3050 146:212 147:120 148:392 149:31 150:1367 151:165 152:95 153:1068 154:744 155:825 156:72 157:129 158:2224

159:498 160:1164 161:946 162:1533 163:520 164:1546 165:462 166:3810 167:468 168:1312 169:742 170:2105
1 0:72 1:1470 2:15208 3:1941 4:414 5:920 6:16040 7:10661 8:2244 9:5520 10:5874 11:357 12:800 13:1904 14:7890 15:4378 16:13778 17:89240 18:8812 19:568 20:6152 21:7795 22:624 23:6197 24:6049 25:6362 26:4144 27:1644 28:1874 29:17170 30:300 31:47456 32:59964 33:72898 34:7082 35:990 36:993 37:170174 38:6738 39:2762 40:4720 41:10608 42:7901 43:1895 44:24565 45:4469 46:3009 47:36045 48:2140 49:15266 50:1902 51:32544 52:73408 53:3309 54:8550 55:2036 56:8162 57:21832 58:15350 59:3308 60:1372 61:760 62:24107 63:1426 64:2353 65:4356 66:2252 67:167 68:3075 69:3960 70:12393 101:2408 102:1782 103:14939 104:2067 105:5994 106:828 107:6574 108:55277 109:31918 110:5148 111:3016 112:1942 113:3689 114:15814 115:7722 116:8030 117:7140 118:398 119:5870 120:2448 121:1296 122:40446 123:1285 124:2905 125:20804 126:2474 127:4788 128:9835 129:1824 130:10137 131:22308 132:1436 133:18432 134:4042 135:35990 136:27014 137:1026 138:6660 139:67272 140:756 141:1636 142:10566 143:1284 144:3366 145:31376 146:668 147:357 148:1234 149:527 150:2356 151:1996 152:906 153:4398 154:3066 155:2076 156:822 157:484 158:6115 159:2786 160:5126 161:5173 162:5540 163:2628 164:6414 165:3272 166:20062 167:5064 168:3883 169:1724 170:9082
1 0:73 1:398 2:2232 3:285 4:141 5:71 6:4732 7:1367 8:480 9:775 10:1126 11:81 12:2297 13:1200 14:1393 15:990 16:4521 17:1776 18:1410 19:258 20:423 21:1408 22:161 23:923 24:496 25:810 26:470 27:28 28:279 29:1114 30:132 31:3091 32:6469 33:1371 34:752 35:496 36:479 37:8086 38:1110 39:643 40:1186 41:4415 42:1112 43:7156 44:3932 45:1160 46:434 47:3098 48:485 49:3086 50:493 51:5375 52:8218 53:712 54:1392 55:346 56:992 57:2240 58:3930 59:367 60:758 61:111 62:3556 63:50 64:239 65:1088 66:149 67:105 68:730 69:377 70:3464 101:723 102:328 103:5456 104:693 105:450 106:206 107:786 108:4059 109:5648 110:1540 111:414 112:158 113:790 114:4361 115:1612 116:1590 117:1968 118:51 119:1488 120:179 121:224 122:13021 123:141 124:480 125:2430 126:551 127:746 128:2368 129:596 130:1035 131:3778 132:246 133:3728 134:890 135:4578 136:1367 137:264 138:608 139:17286 140:577 141:417 142:1328 143:402 144:723 145:2908 146:360 147:72 148:296 149:18 150:780 151:521 152:48 153:910 154:827 155:599 156:142 157:297 158:983 159:670 160:682 161:783 162:670 163:505 164:1510 165:740 166:3219 167:228 168:442 169:688 170:1931
1 0:74 1:325 2:1361 3:259 4:248 5:64 6:3730 7:1982 8:478 9:700 10:1185 11:61 12:44 13:448 14:1342 15:4906 16:1052 17:720 18:1339 19:165 20:446 21:1476 22:163 23:386 24:676 25:763 26:845 27:188 28:221 29:1174 30:43 31:1660 32:4308 33:696 34:1202 35:117 36:111 37:8762 38:760 39:350 40:662 41:1304 42:1204 43:228 44:4199 45:894 46:304 47:3716 48:546 49:4329 50:3689 51:5730 52:4742 53:171 54:777 55:426 56:823 57:3960 58:2634 59:300 60:448 61:156 62:2126 63:133 64:217 65:746 66:96 67:19 68:471 69:197 70:1499 101:718 102:225 103:1928 104:164 105:636 106:208 107:1152 108:2464 109:5676 110:932 111:292 112:122 113:407 114:2693 115:433 116:1210 117:825 118:34 119:927 120:282 121:82 122:6405 123:366 124:276 125:1365 126:383 127:834 128:1228 129:316 130:1246 131:3754 132:505 133:4256 134:913 135:2908 136:789 137:252 138:578 139:13942 140:278

141:410 142:1820 143:282 144:825 145:4832 146:168 147:47 148:193 149:71 150:468 151:360 152:186 153:534 154:1428 155:742 156:82 157:156 158:2385 159:646 160:824 161:566 162:644 163:416 164:975 165:715 166:2527 167:558 168:468 169:825 170:2325
1 0:75 1:2792 2:11551 3:1986 4:1326 5:544 6:20202 7:17343 8:2622 9:5615 10:5656 11:551 12:353 13:3155 14:8764 15:4302 16:10151 17:5616 18:8104 19:1497 20:4480 21:13047 22:1348 23:4986 24:4601 25:4107 26:4022 27:1926 28:1152 29:7845 30:325 31:13055 32:29673 33:6010 34:5994 35:1608 36:1868 37:51850 38:7290 39:4053 40:6968 41:14323 42:12290 43:2183 44:23416 45:5170 46:3486 47:23117 48:4329 49:19448 50:2476 51:44757 52:42475 53:1527 54:5762 55:1206 56:6120 57:29300 58:17158 59:2171 60:1740 61:1032 62:17080 63:690 64:1663 65:6020 66:905 67:434 68:5620 69:2376 70:12114 101:4875 102:1447 103:15351 104:2754 105:4588 106:1124 107:5494 108:22978 109:39680 110:6215 111:2520 112:450 113:4480 114:16446 115:5950 116:8487 117:9864 118:156 119:6874 120:2078 121:1996 122:44712 123:1678 124:3097 125:7636 126:3093 127:4997 128:5888 129:1298 130:9222 131:22138 132:2256 133:17334 134:4043 135:24513 136:5541 137:2139 138:4224 139:83031 140:911 141:2210 142:9848 143:2884 144:4052 145:28302 146:986 147:683 148:916 149:628 150:5865 151:1383 152:1930 153:6954 154:6982 155:3696 156:1532 157:1330 158:8064 159:4279 160:5256 161:3681 162:5135 163:2187 164:7036 165:4058 166:29955 167:3024 168:3138 169:2476 170:11640
1 0:76 1:767 2:3349 3:848 4:288 5:344 6:5202 7:4678 8:832 9:895 10:1526 11:108 12:75 13:1275 14:1602 15:1013 16:3498 17:4393 18:1964 19:263 20:945 21:2185 22:267 23:1714 24:1132 25:1600 26:477 27:275 28:360 29:3392 30:70 31:2652 32:15542 33:2982 34:1621 35:495 36:399 37:10473 38:1628 39:1204 40:1602 41:3672 42:1393 43:552 44:4252 45:442 46:817 47:4448 48:1017 49:5359 50:489 51:7146 52:11604 53:286 54:1473 55:502 56:1834 57:5291 58:4398 59:381 60:672 61:315 62:7134 63:422 64:436 65:1032 66:365 67:189 68:933 69:1053 70:3706 101:1329 102:831 103:4622 104:583 105:1408 106:368 107:896 108:8612 109:11776 110:1801 111:732 112:140 113:564 114:6222 115:2229 116:2196 117:1964 118:136 119:2061 120:515 121:236 122:11818 123:346 124:804 125:2965 126:471 127:716 128:978 129:452 130:2626 131:4108 132:615 133:6582 134:889 135:9000 136:6292 137:494 138:706 139:21218 140:539 141:639 142:2630 143:729 144:1516 145:5024 146:553 147:122 148:188 149:184 150:1346 151:147 152:542 153:1117 154:1610 155:880 156:196 157:426 158:3374 159:2092 160:869 161:834 162:1064 163:848 164:2036 165:1091 166:7250 167:709 168:891 169:658 170:3547
1 0:77 1:931 2:6278 3:2203 4:344 5:221 6:13705 7:9511 8:1239 9:2088 10:2766 11:263 12:333 13:1367 14:5866 15:1884 16:5186 17:2795 18:4260 19:723 20:7278 21:5051 22:635 23:3505 24:3286 25:2225 26:3138 27:1270 28:1206 29:6256 30:260 31:5492 32:12864 33:3444 34:3304 35:1184 36:775 37:40138 38:4036 39:1662 40:3560 41:7653 42:6721 43:1161 44:11737 45:3479 46:2278 47:12730 48:2695 49:11658 50:945 51:27328 52:20352 53:1451 54:5317 55:1284 56:2330 57:12251 58:6460 59:819 60:1495 61:861 62:12101 63:545 64:1198 65:5754 66:538 67:90 68:2017 69:2403 70:5377 101:2213 102:889 103:7028 104:774 105:3204 106:602 107:2301 108:25628 109:19886 110:2820 111:1186 112:283 113:1572 114:10788

115:3237 116:4270 117:3476 118:91 119:4199 120:860 121:1202 122:18970 123:401 124:941 125:5542 126:1373 127:3247 128:6421 129:808 130:8061 131:11174 132:952 133:8154 134:2225 135:11280 136:3572 137:1119 138:7064 139:43645 140:465 141:1744 142:5804 143:1281 144:1936 145:19017 146:864 147:219 148:628 149:258 150:2466 151:420 152:469 153:4363 154:3751 155:1463 156:384 157:610 158:2481 159:3676 160:2145 161:1612 162:2746 163:1492 164:6364 165:2370 166:8245 167:1199 168:1656 169:1328 170:6201
1 0:78 1:1893 2:11271 3:1420 4:1051 5:545 6:18910 7:11751 8:1480 9:4117 10:3954 11:658 12:448 13:3993 14:6756 15:3147 16:8670 17:8867 18:7330 19:972 20:2096 21:9037 22:755 23:3772 24:3888 25:2913 26:2636 27:1194 28:935 29:8802 30:386 31:8942 32:33428 33:8496 34:4440 35:1806 36:1227 37:39198 38:5436 39:2066 40:5760 41:14928 42:5820 43:1875 44:19496 45:3970 46:3098 47:17322 48:2057 49:12843 50:1521 51:32305 52:29214 53:1712 54:6303 55:1212 56:6081 57:15734 58:15919 59:2049 60:7514 61:1169 62:12974 63:483 64:1359 65:3264 66:702 67:264 68:3619 69:1406 70:9027 101:2844 102:1918 103:13251 104:2285 105:4419 106:1093 107:4164 108:18590 109:35722 110:4821 111:2083 112:650 113:3250 114:18584 115:7081 116:6803 117:5000 118:420 119:5904 120:1415 121:1420 122:39402 123:1631 124:2552 125:11800 126:2255 127:2975 128:5954 129:2229 130:8072 131:20476 132:1284 133:19530 134:4452 135:22950 136:5075 137:1237 138:3312 139:75751 140:1158 141:2303 142:9938 143:1599 144:3132 145:22688 146:1178 147:525 148:990 149:424 150:3550 151:947 152:1373 153:4920 154:5964 155:3695 156:576 157:755 158:3715 159:4445 160:4176 161:6099 162:4974 163:1267 164:7858 165:3834 166:14979 167:2378 168:3383 169:2513 170:7890
1 0:79 1:2195 2:1023 3:299 4:159 5:139 6:2627 7:1782 8:304 9:398 10:1064 11:63 12:31 13:266 14:621 15:573 16:1434 17:364 18:753 19:99 20:351 21:595 22:60 23:218 24:409 25:398 26:457 27:242 28:220 29:1033 30:15 31:1079 32:4364 33:376 34:760 35:64 36:232 37:4205 38:795 39:299 40:976 41:1866 42:1158 43:348 44:2787 45:442 46:152 47:2128 48:483 49:2876 50:246 51:2734 52:4530 53:104 54:604 55:130 56:716 57:1171 58:1901 59:138 60:273 61:132 62:1932 63:120 64:230 65:376 66:110 67:56 68:307 69:126 70:946 101:425 102:186 103:1853 104:162 105:730 106:103 107:528 108:2150 109:4461 110:482 111:280 112:37 113:476 114:1738 115:559 116:893 117:818 118:40 119:676 120:471 121:226 122:4948 123:147 124:201 125:1240 126:222 127:454 128:600 129:266 130:670 131:2483 132:291 133:3020 134:809 135:3092 136:674 137:242 138:788 139:9699 140:150 141:326 142:1144 143:184 144:404 145:1729 146:300 147:82 148:206 149:93 150:726 151:152 152:135 153:601 154:854 155:513 156:114 157:222 158:549 159:708 160:401 161:479 162:580 163:346 164:632 165:564 166:4009 167:125 168:168 169:342 170:1377
1 0:80 1:2873 2:10875 3:2364 4:1683 5:1184 6:108009 7:20749 8:3960 9:4611 10:3712 11:318 12:443 13:3482 14:8133 15:6184 16:10990 17:5800 18:7769 19:1456 20:5700 21:11711 22:786 23:4605 24:6351 25:4014 26:4904 27:2925 28:1264 29:14619 30:383 31:19075 32:31544 33:6000 34:11532 35:1201 36:1427 37:103190 38:6557 39:2541 40:6522 41:16013 42:14051 43:1804 44:27624 45:5488 46:5218 47:120807 48:13363 49:19612 50:3256 51:83234 52:56045 53:2466 54:10618 55:2186 56:7825 57:39523 58:22638 59:1642 60:1517 61:1694 62:24071 63:488 64:2199

65:8686 66:1182 67:468 68:6161 69:2389 70:12375 101:6502 102:2061 103:18423
104:2503 105:9031 106:1740 107:9535 108:36152 109:51777 110:5065 111:2795
112:710 113:4430 114:33453 115:6302 116:11524 117:11253 118:317 119:7168
120:1296 121:1791 122:49962 123:2343 124:3111 125:11258 126:3036 127:6442
128:12440 129:1989 130:19822 131:28416 132:4056 133:21906 134:3903 135:24780
136:9241 137:2884 138:10973 139:103530 140:1390 141:3983 142:13114 143:2714
144:2946 145:46431 146:1744 147:614 148:1790 149:677 150:6318 151:1677
152:3062 153:6899 154:8758 155:5077 156:1868 157:1548 158:5954 159:5748
160:6286 161:6305 162:6949 163:4482 164:5532 165:4494 166:15532 167:3822
168:3101 169:2940 170:13260

Ya con los datos de entrenamiento y prueba se trabaja en LibSVM 2.86 para la predicción.