A methodology for the prediction of Embryophyta protein functions using string kernels.

Andrés Felipe Cardona Escobar

Juan Camilo Pineda Iral

This document is presented in partial fulfillment of the requirement for the degree of:

Ingeniero(a) en Mecatrónica

Engineering's Department

Mechatronics engineering

Advisor

Jorge Alberto Jaramillo Garzón

INSTITUTO TECNOLÓGICO METROPOLITANO – ITM

MEDELLÍN – COLOMBIA

2015

Metodología para la predicción de funciones en proteínas Embryophyta usando kernels de secuencias.

Andrés Felipe Cardona Escobar

Juan Camilo Pineda Iral

Este trabajo se presenta como requisito parcial para obtener el grado de:

Ingeniero(a) en Mecatrónica

Facultad de Ingenierías

Ingeniería Mecatrónica

Asesor

Jorge Alberto Jaramillo Garzón

INSTITUTO TECNOLÓGICO METROPOLITANO – ITM

MEDELLÍN – COLOMBIA

2015

RESUMEN

Mediante este trabajo se automatizó el proceso de anotación de secuencias proteicas, a través del uso de técnicas de aprendizaje de máquina supervisado y kernels de secuencias conocidos también como string kernels, para ello se implementan tres tipos de kernel, en conjunto con una metodología para la clasificación supervisada de secuencias proteicas, que incluye máquinas de vectores de soporte (SVM) para resolver 14 problemas de clasificación que hacen referencia a funciones moleculares de plantas terrestres (Embryophyta). La metodología implementada utiliza algoritmos meta-heurísticos bio-inspirados para encontrar los parámetros óptimos de la SVM, a través de una validación cruzada de 10 particiones. Con el propósito de resolver el problema del desbalance de clases, se asignan pesos a las mismas y luego se introducen como hiperparámetros al clasificador, esto con el fin de evitar métodos de muestreo usados para adicionar o quitar muestras. Los resultados obtenidos fueron comparados con el kernel de base radial (RBF) bajo la misma metodología. La media geométrica entre la sensibilidad y la especificidad fue utilizada como medida de desempeño global, los resultados obtenidos muestran que el desempeño de los kernels de secuencias fue mejor en la mayoría de los problemas, mostrando que este tipo de kernels, son una herramienta adecuada para el problema tratado.

RECONOCIMIENTOS

Agradecemos de manera especial y sincera al profesor Jorge Alberto Jaramillo Garzón por habernos brindado su tiempo, conocimiento y paciencia en el desarrollo de esta tesis de grado. También agradecemos a la profesora Norma Patricia Guarnizo por toda la ayuda que nos brindó a lo largo de toda la ingeniería. Adicionalmente agradecemos al semillero de Inteligencia Artificial del laboratorio MIRP (Máquinas Inteligentes y Reconocimiento de Patrones) del ITM (Instituto Tecnológico Metropolitano) por habernos brindando el espacio y los equipos necesarios para probar y procesar los algoritmos desarrollados a lo largo de este trabajo, los cuales fueron vitales para la culminación de esta tesis. De igual manera damos gracias a nuestros padres por apoyarnos siempre en nuestra carrera de ingeniería y en nuestros sueños personales.

ACRÓNIMOS

SVM Máquinas de vectores de soporte

Embryophyta Base de datos de plantas terrestres

RBF Kernel de base radial

ADN Acido desoxirribonucleico

GO Ontología del gen

GOKey Herramienta computacional para la predicción de funciones de ontología del gen.

PoGO Herramienta computacional para la predicción de funciones de ontología del gen.

k-grams / k-mers Subsecuencias de longitud k

k-fold cross validation Validación cruzada de k particiones

LOO Leave one out

SMOTE Sobremuestreo de muestras sintéticas

RRFS Selección de características por relevancia y redundancia

TP Cantidad de Verdaderos positivos

TN Cantidad de Verdaderos negativos

FP Cantidad de falsos positivos

FN Cantidad de falsos negativos

PSO Optimización por enjambre de partículas

TABLA DE CONTENIDO

1. INTRODUCCIÓN

1.1 Generalidades

Con el éxito del proyecto del genoma humano, la secuenciación biológica ha ido creciendo de manera exponencial. Sin embargo, la secuenciación por sí sola, no es suficiente para extraer información relevante que contenga funcionalidades biológicas. La extracción de esta información a través de la experimentación es un proceso lento, costoso y en algunos casos se pueden generar desacuerdos de los resultados obtenidos entre expertos (Petrova, N. V., et al., 2006) (Saigo, H., et al., 2004)

El análisis de datos secuenciales de gran escala se ha convertido en una tarea importante en áreas como aprendizaje de máquina o minería de datos, en parte por las numerosas aplicaciones en la ciencia y la tecnología tales como el análisis biomédico y el análisis de secuencias biológicas (Pavel, 2013). A medida que el número de secuencias proteicas en las bases de datos biomédicas crece más rápido que nuestra capacidad para experimentalmente caracterizar sus funciones, la necesidad por la anotación precisa de proteínas desde una secuencia de aminoácidos, es más que nunca un problema central en la biología computacional (Saigo & Vert, 2004). Uno de los mayores retos biológicos que se tiene a nivel celular es comprender los procesos regulados por proteínas, los cuales podrían ayudar a entender mejor las causas de enfermedades y funciones celulares; para así obtener mecanismos de intervención más eficaces en el tratamiento de enfermedades con el diseño de drogas (Hernández, 2013, pág. 2).

La Bioinformática es una alternativa para confrontar esta problemática, debido a que implementa métodos matemáticos, estadísticos y computacionales para hallar información relevante contenida en datos biológicos. En consecuencia, la bioinformática puede

entenderse como una herramienta para la automatización de procesos de laboratorio, tales como la anotación de secuencias proteicas que incluye el tratamiento digital de variables biológicas y el uso de algoritmos inteligentes para el desarrollo de software de computación que resuelve el problema de caracterizar las funciones moleculares, minimizando el tiempo y el costo de trabajo experimental.

En particular, la predicción de funciones de proteínas, a partir de sus correspondientes secuencias de aminoácidos también conocidas como la estructura primaria, constituye un tema fundamental para el conocimiento de las especies, pues permite que la gran cantidad de datos almacenados se convierta en conocimiento biológico. La determinación de las funciones de las proteínas requiere en la mayoría de casos aproximaciones experimentales realizadas en un laboratorio, y estos procesos deben estar enfocados sobre proteínas, razón por la cual la experimentación es un proceso altamente costoso y demorado. Esto ha llevado a varios investigadores a proponer la predicción computacional como herramienta confiable de análisis, para dilucidar sobre las funciones de algunas proteínas importantes.

Uno de los métodos que ha alcanzado mayor éxito en la anotación de secuencias por medio de métodos no experimentales, es el aprendizaje de máquina; algunos métodos implementan técnicas tales como, redes neuronales (Jensen, L. J., et a. 2003), clasificadores Bayesianos multi-etiqueta (Jung, J., et al. 2008), máquinas de vectores de soporte (Cai, C. Z., et al. 2003), GOKey (Bi, R., et al. 2007) y PoGO (Jung, J., et al. 2010). Sin embargo, las técnicas de aprendizaje de máquina por lo general son entrenadas bajo atributos físico-químicos (Jaramillo-Garzón, J. A, et al, 2013), lo cual implica que si la etapa de caracterización no es llevada a cabo de forma correcta, el proceso de clasificación podría verse seriamente afectado es su desempeño.

Existen diferentes metodologías para realizar el proceso de anotación de secuencias, tales como, alineamiento de secuencias por parejas (Altschul, S. F., et al. 1990), perfiles para

familias de proteínas (Gribskov, M., et al. 1987), Modelos Ocultos de Markov (Krogh, A., et al. 1994). Por lo general, estos métodos requieren modelos generativos; los cuales implican construir un modelo para cada familia de proteínas y luego observar si cada una de las secuencias candidatas se ajusta a este modelo.

Una forma plausible de realizar el proceso de anotación de secuencias, es mediante el uso de kernels de secuencias como Spectrum kernel (Leslie, C. S., et al., 2002), Mismatch kernel (Eskin, E., et al., 2002) o Gappy Pair kernel (Kuksa, P., et al, 2009), la gran ventaja de este tipo de kernels es que permiten prescindir de la etapa de caracterización por medio de atributos físico-químicos.

Por otra parte, los kernel de secuencias usan un enfoque discriminativo; es decir, a partir de un conjunto de datos etiquetados de forma positiva si la secuencia pertenece a la familia o super-familia, y etiquetados de forma negativa si no pertenecen, se establece una frontera de decisión entre las clases, por lo tanto, el modelo es construido con los datos de ambas clases. El objetivo de los string kernels consiste en representar cada una de las secuencias en un espacio de alta dimensión, esto se logra mediante un mapeo que se realiza teniendo en cuenta la estructura primaria de las secuencias. Sin embargo, este mapeo se realiza de forma implícita, es decir, se deben comparar las secuencias por pares teniendo en cuenta su estructura primaria, esto permite calcular los productos punto entre ellas y formar una matriz kernel que posteriormente es utilizada para entrenar una SVM.

1.2 Objetivos

General

Implementar Kernels de secuencias (String Kernels) en conjunto con una metodología para la clasificación supervisada de secuencias proteicas que permita automatizar el proceso de anotación funcional de secuencias.

Específicos

Implementar una estrategia que permita encontrar los parámetros óptimos en cada uno de los problemas de clasificación, haciendo uso de algoritmos de optimización meta-heurísticos bio-inspirados.

Desarrollar un análisis comparativo de los diferentes kernels de secuencias propuestos en la literatura con el fin de determinar cuál de ellos se ajusta más a la aplicación específica.

Validar las estrategias propuestas sobre una base de datos real, por medio de técnicas de validación cruzada con el fin de asegurar la confiabilidad de los resultados obtenidos en la clasificación.

1.3 Organización de la tesis

Este trabajo está organizado de la siguiente manera:

En primer lugar se expone el marco teórico, donde se tratan los aspectos fundamentales y necesarios para la comprensión y asimilación de los métodos empleados durante el trabajo. Luego del marco teórico se procede a explicar de forma detallada y precisa la metodología

empleada, mencionando cuáles fueron las herramientas utilizadas para el cálculo de los resultados.

Seguidamente se dan a conocer los resultados obtenidos luego de seguir la metodología anteriormente mencionada, estos resultados se muestran mediante gráficas donde es posible identificar cual fue la técnica que obtuvo un mejor desempeño. De igual manera, se realiza un análisis objetivo de los resultados que demuestran la pertinencia de los métodos empleados.

Por último, se encuentran las conclusiones y el trabajo futuro donde se exponen los puntos principales del trabajo y se sintetiza de manera general si los objetivos fueron cumplidos de forma satisfactoria, además se da a conocer cuál será el siguiente paso a desarrollar en un futuro para complementar el trabajo.

2. MARCO TEÓRICO

2.1 Proteínas

Un aminoácido es una molécula orgánica cuya estructura está dada por un átomo de carbono central llamado carbono alpha, un grupo amino (NH2), un grupo carboxilo (COOH), un átomo de hidrógeno y un grupo de átomos llamados R, que conforman una cadena lateral. La diferencia entre cada aminoácido está dada por el grupo de átomos R (Hernandez, N. 2013, Pág. 7); en total existen 20 aminoácidos diferentes para construir proteínas.

Las proteínas son moléculas de gran tamaño formadas por secuencias de aminoácidos, como se puede observar en la Figura 1.

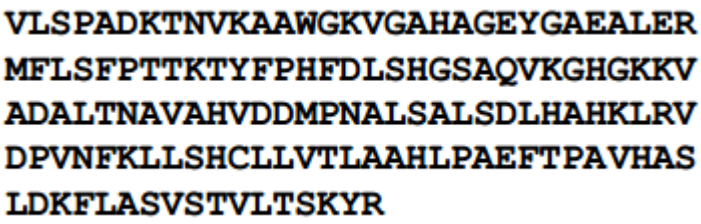

Figura 1. Estructura primaria de una proteína, representada mediante una cadena de aminoácidos. Tomado de: http://cbio.ensmp.fr/~jvert/kmb03/material/leslie.pdf

Una pequeña variación en la secuencia, puede influir drásticamente en el funcionamiento de la proteína. Por lo tanto, para comprender adecuadamente el funcionamiento de la misma es importante conocer su estructura primaria, la cual simplemente representa la secuencia lineal de aminoácidos basada en la información hereditaria proveniente de la

célula. Asimismo, es posible extraer las estructuras secundaria y terciaria, las cuales hacen referencia a las interacciones que se dan entre los aminoácidos.

2.2 Ontología del Gen (GO)

En los últimos años, científicos han hecho enormes contribuciones para categorizar lingüísticamente y formalmente funciones moleculares, procesos biológicos y componentes celulares (Botstein, D., et al., 2000), esto con el objetivo de generar vocablos que proporcionen descriptores consistentes para los dominios claves de la biología molecular. La ontología del Gen es un proyecto colaborativo encaminado a suplir tres necesidades; el desarrollo y mantenimiento de las ontologías por sí mismas, la anotación de productos génicos, y el desarrollo de herramientas que faciliten la creación, mantenimiento y uso de ontologías.

2.3 Características Fisicoquímicas

Una forma de realizar el proceso de anotación funcional de secuencias es transformar la estructura primaria en un conjunto de características fisicoquímicas y luego escoger un algoritmo de aprendizaje de máquina para encontrar un modelo computacional que determine las diferentes funciones que pueda cumplir determinada secuencia.

En esta metodología las máquinas de vectores de soporte son ampliamente utilizadas (Zhang, Y., et al., 2014), sin embargo, el desempeño de cualquier clasificador depende estrictamente de la calidad de las características con las cuales es entrenado.

2.4 Selección de características

Las técnicas de selección de características o reducción de dimensiones son comúnmente utilizadas en aplicaciones donde existen cientos o miles de características. Generalmente muchos métodos utilizados en la selección de características están basados en la relevancia de las mismas, lo cual por si solo es insuficiente (Yu, L., et al. 2004), ya que, es necesario realizar un análisis de redundancia. La relevancia de una característica se puede clasificar en tres tipos; fuertemente relevante, débilmente relevante e irrelevante.

Una característica es fuertemente relevante cuando es siempre necesaria dentro del subconjunto óptimo de características a seleccionar, una característica débilmente relevante indica que no es siempre necesaria dentro del conjunto óptimo de características, y una característica irrelevante no es necesaria en ningún caso.

2.5 Máquinas de Vectores de Soporte

Las máquinas de vectores de soporte (SVM) se han vuelto muy populares en los últimos años para resolver problemas de clasificación y regresión. Las SVM emplean la siguiente idea: Se realiza un transformación de los vectores de entrada x, a un espacio de alta dimensión Z, a través de un mapeo no lineal escogido previamente; luego en este espacio de alta dimensión conocido como espacio de características es posible trazar una frontera de decisión lineal para separar los datos. Este hiperplano o frontera de decisión busca separar los datos de entrada maximizando el margen entre las clases y está delimitado por unas muestras conocidas como vectores de soporte; esto se puede apreciar en la Figura 2.

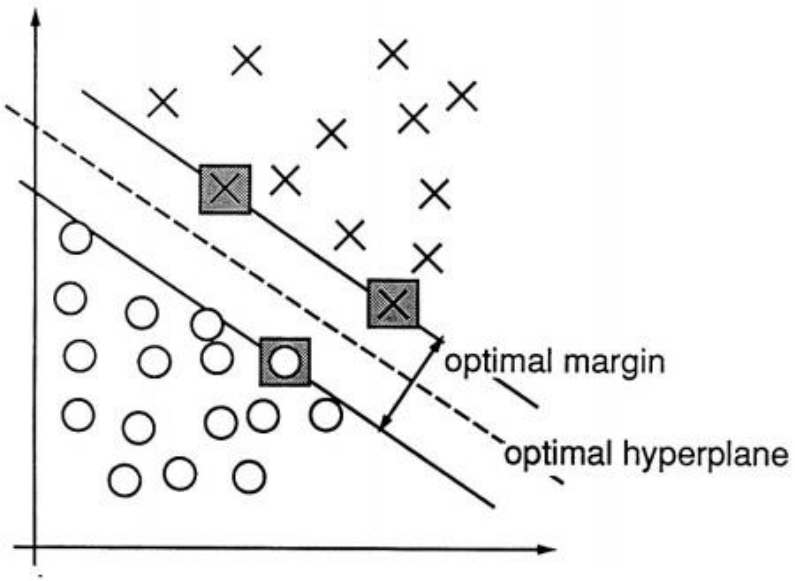

Figura 2. Problema de clasificación binaria linealmente separable. Los vectores de soporte están enmarcados en gris y determinan el máximo margen o separación entre las clases. Tomado de: Cortes, C., et al., 1995, Pág. 275.

Para encontrar la frontera de decisión óptima que maximice el margen entre las clases, no es necesario llevar los datos explícitamente al espacio de características, solo es necesario encontrar los productos punto en el espacio de características (Cortes, C., et al., 1995, pág. 274).

Dado un conjunto de entrenamiento

$$(y_1, x_1), \dots, (y_l, x_l), \quad x \in R^n, \quad y \in \{-1,1\}, \quad \text{(Ecuación 1)}$$

Se puede decir que los datos son linealmente separables si existe un vector **w** y una constante b tal que

$$\boldsymbol{w} \cdot \boldsymbol{x_i} + b \geq 1 \quad si \quad y_i = 1$$
$$\boldsymbol{w} \cdot \boldsymbol{x_i} + b \leq -1 \quad si \quad y_i = -1$$

(Ecuación 2)

Reescribiendo las igualdades anteriores, se obtiene la siguiente expresión

$$\boldsymbol{y_i}(\boldsymbol{w}\cdot\boldsymbol{x_i}+b)\geq 1, \quad \text{(Ecuación 3)}$$

Cuando los datos son linealmente separables debe existir un hiperplano que separe de forma óptima los datos con el máximo margen posible

$$\boldsymbol{w_0}\cdot\boldsymbol{x}+b_0=0, \quad \text{(Ecuación 4)}$$

El máximo margen posible se da cuando la distancia dada en la ecuación 5 es máxima

$$\rho(w,b)=\min_{\{x:y=1\}}\frac{\boldsymbol{x}\cdot\boldsymbol{w}}{|\boldsymbol{w}|}-\max_{x:y=-1}\frac{\boldsymbol{x}\cdot\boldsymbol{w}}{|\boldsymbol{w}|}, \quad \text{(Ecuación 5)}$$

Cuando se evalúa esta distancia utilizando los parámetros óptimos

$$\rho(w_0,b_0)=\frac{2}{|\boldsymbol{w_0}|}=\frac{2}{\sqrt{\boldsymbol{w_0}\cdot\boldsymbol{w_0}}}, \quad \text{(Ecuación 6)}$$

Los vectores x_i para los cuales $\boldsymbol{y_i}(\boldsymbol{w}\cdot\boldsymbol{x_i}+b)=1$ son conocidos como vectores de soporte, es posible expresar $\boldsymbol{w_0}$ como combinación lineal en términos de los vectores de soporte

$$\boldsymbol{w_0}=\sum_{i=1}^{l}y_i\alpha_i^0x_i, \quad \text{(Ecuación 7)}$$

Donde $\alpha_i^0\geq 0$. Si se construye un vector $\Lambda_{\boldsymbol{0}}^{\boldsymbol{T}}$ formado por los valores α_i^0, se obtiene el siguiente problema de optimización

$$W(\Lambda)=\Lambda^{\boldsymbol{T}}\boldsymbol{1}-\frac{\boldsymbol{1}}{\boldsymbol{2}}\Lambda^{\boldsymbol{T}}\boldsymbol{D}\Lambda, \quad \text{(Ecuación 8)}$$

Cuando los datos no son linealmente separables es necesario introducir unas variables de holgura que permiten un margen de error en la etapa de entrenamiento

$$\phi(\xi) = \sum_{i=1}^{l} \xi_i^{\sigma} \ , \text{ (Ecuación 9)}$$

Esta idea puede ser representada mediante el siguiente problema

$$\min \quad \frac{1}{2}\|w\|^2 + \ C \ \sum_{i=1}^{n} \xi_i \ , \text{ (Ecuación 10)}$$

Sujeto a las siguientes restricciones

$$\boldsymbol{y_i}(\boldsymbol{w} \cdot \boldsymbol{x_i} + b) \geq 1 - \xi_i$$
$$\xi_i > 0$$
(Ecuación 11)

2.6 Kernels de secuencias (String kernels)

En la detección de homólogos remotos, la cual consiste en determinar si dos secuencias vienen del mismo ancestro y en general para la clasificación de secuencias proteicas, se han propuesto en la literatura varios kernels conocidos como kernels de secuencias o string kernels. Este tipo de kernels comparan dos secuencias por medio de las sub-secuencias que contienen; a medida que se tengan más sub-secuencias en común, más similares serán las secuencias entre sí. El uso de kernels que puedan actuar en cadenas de símbolos permite hacer un análisis estadístico de los patrones (*Hernández, 2013, pág. 27*).

Un alfabeto ∑, se define como un conjunto finito de símbolos, la longitud de éste alfabeto se denota como $|\sum|$, por ejemplo; el alfabeto español se compone de 27 símbolos, análogamente para secuencias de aminoácidos el alfabeto tiene una longitud de 20.

Una cadena o string es una secuencia finita de símbolos pertenecientes a un alfabeto ∑, incluyendo la secuencia vacía representada por ε, el único string de longitud cero, por ejemplo, una proteína es una secuencia finita compuesta de aminoácidos la cual tiene longitud n (Lodhi, H.,et al.,2002, pág. 423).

$$s = s_1 \dots s_n \,, \quad \text{(Ecuación 12)}$$

Para $1 \le i \le j \le |s|$, el string representado por $s(i:j)$ es el sub-string $s_i \dots s_j$ de s. Los sub-strings de longitud k son también conocidos como k-grams o k-mers.

Se puede decir que u es una sub-secuencia de una cadena s, si existen subíndices $i = (i_1, \dots, i_{|u|})$, con $1 \le i_1 < \dots < i_{|u|} \le |s|$, tal que $u_j = s_{ij}$ para $j = 1, \dots, |u|$. Usando una representación corta $u = s(i)$ si u es subsecuencia de s en las posiciones dadas por i.

Para representar el conjunto de secuencias de longitud n, pertenecientes a un alfabeto ∑ se utiliza la notación $\sum^n$; de igual manera para representar el conjunto de todas las secuencias se utiliza $\sum^*$

$$\sum^* = \bigcup_{n=0}^{\infty} \sum^n \,, \quad \text{(Ecuación 13)}$$

2.6.1 Spectrum kernel

El Spectrum kernel (Leslie, C. S., et al., 2002) es una de las formas más simples para comparar dos secuencias, donde la idea es compararlas por medio del número de subsecuencias de longitud k que tienen en común.

El mapeo llevado a cabo para transformar todas las posibles subsecuencias desde espacio de entrada X formado por todas las posibles secuencias de caracteres provenientes de un alfabeto ∑, a un espacio de características R^{l^k}, está dado por

$$\phi_k(x) = (\phi_a(x))_{a \in \sum^k}, \quad \text{(Ecuación 14)}$$

Donde $\phi_a(x)$= número de veces que aparece el k-mer a en la secuencia x.

Para su implementación, lo primero es definir un parámetro k el cual indica la longitud de las sub-secuencias, luego se extrae de ambas cadenas todas las posibles sub-secuencias de longitud k que sean contiguas dentro de la secuencia, llamadas k-mers, en seguida para cada secuencia se crea un vector, el cual contiene el número de veces que aparece cada k-mer dentro de la secuencia, por último se calcula el producto punto entre ambos vectores. El kernel entre dos secuencias *x* y *y* está definido como

$$K_k(x, y) = \langle \phi_k(x), \phi_k(y) \rangle, \quad \text{(Ecuación 15)}$$

Por ejemplo, para los string “bar”,”bat”, ”car” y “cat”, se podría calcular el Spectrum kernel, tomando k=2; para ello se extraen todos los posibles subtrings de longitud 2, y se contabiliza cuantas veces se repiten en cada una de las secuencias, como se muestra en la siguiente tabla

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

Tabla 1. Mapeo de las secuencias a un espacio de características donde se contabiliza el número de veces que una determinada subsecuencia aparece dentro de la secuencia.

La matriz kernel resultante se muestra a continuación

ϕ	bar	bat	car	cat
bar	2	1	1	0
bat	1	2	0	1
car	1	0	2	1
cat	0	1	1	2

Tabla 2. Cálculo de la matriz kernel, donde cada elemento corresponde al producto punto entre los vectores en el espacio de características.

2.6.2 Mismatch kernel

El Mismatch kernel propuesto en (Eskin, E., et al., 2002) es muy similar al Spectrum kernel, la diferencia está en la introducción de un concepto biológicamente importante que es la disparidad, esto se hace por medio de un nuevo parámetro llamado m, este parámetro indica el número de disparidades entre cada uno de los k-mer de la secuencia y los k-mer extraídos, es decir, si entre dos k-mer existe un número de disparidades menor a m se podría decir que ambas sub-secuencias son iguales, para representar el número de disparidades entre dos subsecuencias *u* y *v* se utiliza la expresión *d(u,v)*, que simboliza el número de caracteres en que difieren *u* y *v,* esta medida es conocida como distancia Hamming (Kuksa, P. P., et al, 2008, pág. 4).

Para el mismatch kernel se realiza un mapeo parecido al mapeo realizado en el Spectrum kernel con una pequeña diferencia y es la inclusión de un parámetro m, que permite un número máximo de disparidades entre los k-mer, cuando m es igual a cero se obtiene el Spectrum kernel, por lo tanto el mapeo queda definido por

$$\phi_{k,m}(x) = (\phi_a(x))_{a \in \sum^k}\ ,\ \text{(Ecuación 16)}$$

Existen diversas formas de implementar el mismatch kernel, una de ellas se describe en (Kuksa, P. P., et al, 2008, pág 5) donde se plantea un algoritmo basado en la distancia Hamming, para ello se precalcula el tamaño de las intersecciones entre los k-mer de las secuencias. El enfoque tradicional usado en (Eskin, E.,et al, 2002) consiste en construir un árbol de profundidad k, donde cada uno de los nodos contiene l ramas , donde l es el tamaño del alfabeto (para el caso de aminoácidos l=20) y cada rama es etiquetada con un símbolo perteneciente al alfabeto. No obstante, en la implementación del algoritmo no es necesario almacenar todo el árbol, la manera de hacerlo es mediante una función recursiva.

2.6.3 Gappy Pair kernel

El Gappy Pair kernel (Kuksa, P., et al, 2008) tiene dos parámetros k y m, y tiene como objetivo buscar pares de k-mers separados cierta distancia dentro de una secuencia. El parámetro m tiene diferente significado que en el mismatch kernel, ya que, este parámetro representa la distancia máxima a la que pueden estar separados dos k-mer.

$$\underbrace{kmer1}_{k} \underbrace{. . . . . .}_{0 \leq i \leq m} \underbrace{kmer2}_{k}$$

Figura 3. Pares de k-mers separados una distancia m. Tomado de: Palme, J., Hochreiter, S., & Bodenhofer, U. (2015). KeBABS: an R package for kernel-based analysis of biological sequences. Bioinformatics, btv176.

Además tiene en consideración diferentes mutaciones o transformaciones biológicas tales como inserciones o supresiones. Este kernel no solo toma en consideración la frecuencia en que se encuentra un k-mer dentro de una secuencia, sino que además tienen en cuenta como es la ubicación de los k-mer. La inclusión de información espacial ha demostrado buen desempeño incluso para subsecuencias pequeñas, por ejemplo k=1 (Kuksa, P., et al, 2008, pág. 31). El Gappy Pair kernel está dado por la siguiente expresión

$$k(s,t) = \sum_{\substack{(a_1,\,k,a_2) \\ a_1,\ a_2\ \in\ \sum^k}} C(a_1,\,k,a_2\ |s) \cdot C(a_1,\,k,a_2\ |t)\ ,\ \text{(Ecuación 17)}$$

Donde $C(a_1,\,k,a_2\ |s)$, representa el número de veces que aparece el substring a_1 separado de a_2 en k caracteres, dentro de la secuencia s.

El tamaño del espacio de características está dado por $|M| = (m+1)\ |\sum|^{2k}$, el cual es considerablemente más grande que para el caso del mismatch y Spectrum kernel con los mismos valores de k. A manera de ejemplo se considera la secuencia CAGAT, la cual se proyecta en el espacio de características, utilizando el Gappy Pair kernel con k=1 y m=2, y seleccionando todos los posibles pares de monómeros (CA, C.G, C..A, AG, A.A, A..T, GA, G.T y AT) que estén separados con una distancia no mayor a m=2, luego se contabiliza el número de ocurrencias de cada par dentro de la secuencia.

2.7 Técnicas de balanceo de clases

Es muy común en áreas como el aprendizaje de maquina encontrar problemas de clasificación binaria, ya que, este tipo de problemas surge de la necesidad de indicar pertenencia o no a determinada clase; asimismo problemas más complejos de clasificación como por ejemplo problemas de aprendizaje multi-etiqueta donde a cada muestra se le puede asignar más de una clase, se pueden descomponer fácilmente en varios problemas de tipo binario; utilizando técnicas como relevancia binaria donde se crea un problema binario para cada una de las clases indicando la pertenencia o no a cada una de las clases. Sin embargo, estos métodos por sí solos generan desbalance de clases, el cual puede afectar considerablemente el desempeño de un clasificador.

Para mejorar el desempeño de un clasificador, se utilizan técnicas de balanceo de clases, estas técnicas se dividen principalmente en dos enfoques diferentes; uno de ellos consiste en hacer un sobre-muestreo o sub-muestreo de las base de datos original, es decir, agregar o quitar muestras; algunas de estas técnicas son; sub-muestreo aleatorio, sobre-muestreo con reemplazamiento o sobre-muestreo sintético (SMOTE) propuesto en (Chawla, N. V., et al, 2002). El otro enfoque utilizado consiste en asignarle mayor importancia a la clase minoritaria con el fin de erradicar el efecto del desbalance de clases, esto se conoce como aprendizaje por sensibilidad de costo, y se realiza a través de la inclusión de una ponderación de las muestras desde el proceso de entrenamiento. En este último caso no se cambia el número de muestras en la base de datos.

Las máquinas de vectores de soporte (SVM), han demostrado ser una herramienta muy útil en problemas de clasificación donde no exista un desbalance muy alto entre las clases, ya que, su principio de funcionamiento se basa en la minimización del riesgo estructural; esto quiere decir que deben ser ajustadas para manejar un equilibrio adecuado entre la complejidad del modelo y el error de entrenamiento, esto otorga una capacidad de generalización alta, sobretodo en comparación con algunos clasificadores estándar (Tang, Y., *et al, 2009, pág. 1*). Sin embargo, como se explicó anteriormente, técnicas como relevancia binaria pueden generar problemas donde el desbalance entre las clases es considerable. Específicamente las SVM son muy sensibles al problema del desbalance de clases; por esta razón se han probado distintos métodos para corregir este problema.

Uno de ellos consiste en quitar muestras de la clase mayoritaria de forma aleatoria, lo cual se conoce como sub-muestreo aleatorio. Otra técnica muy utilizada consiste en repetir muestras de la clase minoritaria en forma aleatoria hasta igualar ambas clases, lo cual se conoce como sobre-muestreo.

2.7.1 Sobre-muestreo sintético (SMOTE)

Uno de los métodos más utilizados en los últimos años, ha sido el SMOTE, este método consiste en generar muestras sintéticas a partir de la clase minoritaria, esto se hace en el espacio de características y no desde el espacio de datos como en los métodos anteriormente mencionados; para ello se calculan los k vecinos más cercanos de cada uno de los ejemplos de la clase minoritaria, utilizando la distancia euclídea; luego para cada muestra se escoge uno de los k vecinos aleatoriamente y partiendo de este vecino se genera la muestra sintética, cada una de las muestras sintéticas se ubica en la línea que une la muestra a partir de la cual se creó la nueva y el vecino correspondiente; este proceso se realiza hasta igualar las clases.

Cada una de las muestras sintéticas se crea a partir de la siguiente ecuación:

$$S_i = x_i + \lambda \circ (x_\varepsilon - x_i), \text{ (Ecuación 18)}$$

Donde S_i representa la nueva muestra sintética, x_i es la muestra a partir de la cual se genera la nueva muestra, x_ε es uno de los vecinos más cercanos de x_i escogido de forma aleatoria y λ es un vector con valores entre 0 y 1, la operación ∘ hace referencia al producto Hadamard entre vectores.

2.7.2 Aprendizaje por sensibilidad de costo

La mayoría de los clasificadores son incapaces de establecer cuál es la diferencia entre el costo de los falsos positivos o el costo de los falsos negativos (Thai-Nghe, N., et al., 2010., pág. 2). Sin embargo, en muchas situaciones comunes no es correcto asignar el mismo costo a falsos positivos y falsos negativos; por ejemplo, en un examen de cáncer es mucho más

grave pasar por alto un resultado positivo, que diagnosticar una falsa alarma, ya que esta podría ser descartada con otro examen.

El aprendizaje por sensibilidad de costo tiene como objetivo asignar pesos a las clases, de forma tal que no sea necesario crear u omitir muestras en la etapa de entrenamiento. Para realizar el aprendizaje por sensibilidad de costo utilizando máquinas de vectores de soporte, se utiliza un clasificador conocido como weighted-SVM (Osuna, E., et al., 1997), el cual asigna un peso diferente a cada clase por medio del coeficiente de regularización C de la SVM, es decir, las muestras de la clase positiva tienen un coeficiente de regularización diferente a las muestras negativas, este enfoque queda representado por el siguiente problema de optimización.

$$\min \quad \frac{1}{2}\|w\|^2 + C^+\left(\sum_{i:yi=1}\xi_i\right) + C^-\left(\sum_{i:yi=-1}\xi_i\right), \quad \text{(Ecuación 18)}$$

Es importante señalar que por medio de este enfoque, no se incrementa la complejidad del problema, pero aun así, es necesario sintonizar el peso de las clases para obtener un buen desempeño.

2.8 Optimización por enjambre de partículas (PSO)

La optimización por enjambre de partículas (PSO), que fue propuesto por (Kennedy & Eberhart, 1995), es una técnica de optimización estocástica basada poblaciones e inspirada en el comportamiento social de bandadas de aves o cardumen, para la búsqueda de una solución óptima en espacios de búsqueda complejos. Debido a su efectividad e implementación simple en la solución de problemas multidimensionales, los algoritmos PSO y sus variantes han sido usados en muchas áreas aplicadas (Li et al, 2011).

El PSO comparte muchas similitudes con las técnicas de computación evolutivas, en las cuales una población de soluciones potenciales al problema en consideración es usada para sondear el espacio de búsqueda. Sin embargo, en el PSO cada individuo de la población tiene una velocidad adaptable, de acuerdo a la cual se mueve en el espacio de búsqueda. Adicionalmente, cada individuo tiene memoria, lo que le permite recordar la mejor posición que este ha visitado en el espacio de búsqueda (Eberhart et al., 1996). Este valor de la mejor posición de la partícula se llama *pbest*. Otros valores que son tomados en cuenta en la optimización por enjambre de partículas, son las mejores posiciones globales encontradas por otras partículas a medida que recorren el espacio de búsqueda. Estas localizaciones son llamadas *lbest*. Cuando una partícula llama a toda la población a su vecindad topológica, el mejor valor obtenido es el mejor valor global y este es llamado *gbest*.

El comportamiento de una partícula puede ser descrita de la siguiente manera:

$$V_{id} = \omega \times V_{id} + c_1 \times r_1 \times (pbest_{id} - X_{id}) + c_2 \times r_2 \times (gbest_d - X_{id}), \quad \text{(Ecuación 19)}$$

$$X_{id} = X_{id} + V_{id}\ , \text{(Ecuación 20)}$$

Donde V_{id} y X_{id} representan la velocidad y la posición de la partícula i in la *d*th dimensión, ω es el peso inercial que permite un equilibrio entre las capacidades globales y locales; c_1 y c_2 son constantes de aceleración, r_1 y r_2 son números aleatorios en el rango [0,1], $pbest_{id}$ es la mejor posición con respecto a la partícula *i* encontrada hasta el momento en la dimensión *d*th y el $gbest_d$ es la mejor posición global que ha sido visitada por todas las partículas hasta el momento. En la Figura 4, se muestra el comportamiento de las partículas para encontrar la solución óptima de una función, utilizando el algoritmo PSO.

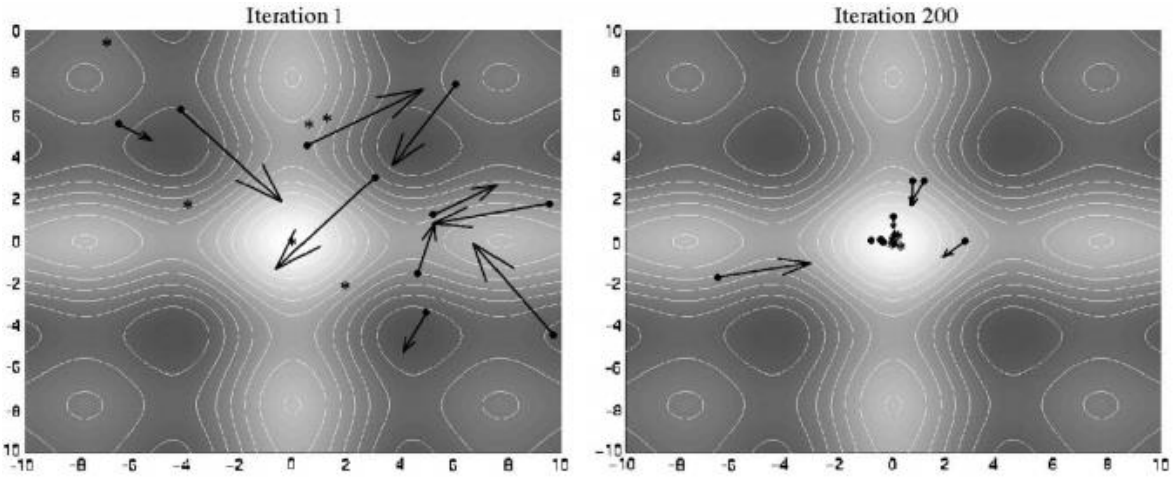

Figura 4. Ejemplo de Búsqueda del algoritmo PSO en un espacio de dos dimensiones; la primera imagen muestra la iteración número uno del algoritmo donde las partículas están dispersas, en la segunda imagen se muestra la iteración número 200 donde las partículas están cercanas al punto de solución óptima. Tomado de: Robinson, J., & Rahmat-Samii, Y. (2004), Pág. 401.

2.9 Técnicas de validación

Para comprobar el desempeño de un clasificador se utilizan diferentes técnicas de validación las cuales dan información acerca del error generalizado o error total cometido en la clasificación de los datos (Duan, K., et al., 2003, pág. 41). Algunos de estos métodos son muy utilizados en la sintonización de un clasificador, este proceso consiste en encontrar los parámetros óptimos para entrenar un clasificador. Por esta razón, existen varias técnicas para encontrar el error cometido por el clasificador, tales como el Bootstrapping o la validación cruzada, no obstante, la técnica que será tenida en cuenta para este trabajo será validación cruzada.

Existen diferentes versiones de validación cruzada, una de ellas es k-fold cross validation la cual consiste en dividir los datos en k particiones, luego se separan k-1 particiones y se utilizan para entrenar el clasificador, después se valida el modelo utilizando la última partición, este proceso se realiza k veces, y en cada iteración se utiliza un conjunto de datos

diferente para validar. También existe otro tipo de validación cruzada conocida como leave-one-out (LOO), la cual es simplemente una versión en extremo de k-fold cross validation, esta técnica consiste en hacer k igual al número de muestras de la base de datos, dejando así una sola muestra para la validación; si lo que se requiere es realizar la sintonización de un clasificador, el método LOO es muy costoso computacionalmente (Duan, K., et al., 2003, pág. 43).

Como se dijo anteriormente lo que se busca con las técnicas de validación es encontrar los parámetros de entrada adecuados para entrenar el clasificador, estos parámetros normalmente son conocidos como hiper-parámetros. En la sintonización de un clasificador es muy común el uso de algoritmos meta heurísticos bio-inspirados, que si bien no garantizan de forma absoluta la mejor solución o la convergencia hacia el valor más óptimo, si pueden lograr un alto desempeño a la hora de encontrar los hiper-parámetros.

2.10 Medidas de desempeño

Una de las formas más comunes de estimar el error en un problema de clasificación es promediando el error de cada una de las iteraciones de una validación cruzada. Sin embargo, en el contexto del desbalance de clases algunas medidas de desempeño no representan de forma correcta el desempeño global de un clasificador. Una forma muy común de cuantificar el desempeño de un algoritmo de aprendizaje, es mediante una matriz de confusión (Chawla, N. V., et al, 2002, pág. 323); la cual está representada en la siguiente tabla

		Clase Estimada	
		-	+
Clase Real	-	TN	FP
	+	FN	TP

Tabla 3. Matriz de Confusión para Clasificación Binaria

Donde

TN es el número de verdaderos negativos e indica el número de muestras de la clase negativa que fueron clasificadas correctamente.

TP es el número de verdaderos positivos e indica el número de muestras de la clase positiva que fueron clasificadas correctamente.

FN es el número de falsos negativos e indica el número de muestras de la clase negativa que fueron clasificadas incorrectamente.

FP es el número de falsos positivos e indica el número de muestras de la clase positiva que fueron clasificadas incorrectamente.

Para medir el desempeño global en un problema de clasificación, se han creado algunas medidas que estiman de forma adecuada el error en problemas de clasificación que involucran desbalance de clases, estas medidas son creadas a partir de la sensibilidad y la especificidad; también conocidas como tasa de verdaderos positivos y tasa de verdaderos negativos respectivamente.

$$sensibilidad = \frac{TP}{TP+FN} \quad \text{(Ecuación 21)}$$

$$Especificidad = \frac{TF}{TN+FP} \quad \text{(Ecuación 22)}$$

Una medida utilizada para estimar el desempeño de un clasificador es la media geométrica entre la sensibilidad y la especificidad; esta medida representa el balance entre las muestras bien clasificadas de la clase positiva y las muestras bien clasificadas de la clase negativa (Tang, Y., et al., 2010, pág. 2). La media geométrica está definida como la raíz n-ésima del producto de n valores.

$$G(x_1, \dots, x_n) = \left(\prod_{i=1}^{n} x_i\right)^{1/n}, \quad \text{(Ecuación 23)}$$

Si se quiere calcular la media geométrica entre la sensibilidad y la especificidad, se utiliza la siguiente expresión

$$G(sensibilidad, especificidad) = \sqrt{sensibilidad * especificidad} \quad \text{(Ecuación 24)}$$

3. METODOLOGÍA

3.1 Base de datos

Los datos implementados en este trabajo está compuesto por todas las proteínas *Embryophyta* disponibles en la base de datos UniProtKB/Swiss-Prot (Versión del archivo: 10/01/2013) con al menos una anotación en el proyecto Gene Ontology Annotation (Versión del archivo: 07/01/2013). Los grupos resultantes están comprendidos por proteínas de 189 plantas distintas. Con el fin de evitar posibles sesgos debidos a la presencia de familias funcionales dentro de la base de datos, se filtraron las secuencias (utilizando el software Cd-Hit) de manera que sólo se conservaron aquellas que presentaron una identidad inferior al 30%. El grupo principal está comprendido por un total de 3368 secuencias proteicas, de las cuales 2544 secuencias están anotadas con funciones moleculares, 2210 secuencias anotadas con componentes celulares y 2798 con procesos biológicos.

Función molecular	**Acrónimo**	**Tamaño clase positiva**	**Tamaño clase negativa**	**ClasePositiva/ ClaseNegativa**
Nucleotide binding	Ntbind	47	1506	0.0312
Molecular function*	MF*	268	1705	0.1572
DNA binding	DnaBind	107	1430	0.0748
Transcription factor activity	TranscFact	307	1402	0.2190
RNA binding	RnaBind	43	1493	0.0288
Catalytic activity*	Catal*	334	1372	0.2434

Receptor binding	RecBind	38	943	0.0403
Transporter activity	Transp	125	1583	0.0790
Binding*	Bind*	173	1534	0.1128
Protein binding*	ProtBind*	630	968	0.6508
Kinase activity	Kinase	68	1147	0.0593
Transferase activity*	Transf*	173	1204	0.1437
Hydrolase activity	Hydrol	190	1193	0.1593
Enzyme regulator activity	EnzReg	41	1665	0.0246

Tabla 4. Funciones moleculares, con su respectivo número de muestras que pertenecen o no pertenecen a dicha función.

3.2 Metodología general

La metodología implementada se describe en los siguientes pasos:

(i) Se carga la base de datos que contiene las proteínas con funciones moleculares.

(ii) Se genera una validación cruzada externa de diez particiones, la cual recibe como entrada la base de datos carga en el paso (i).

(iii) La base de datos dentro de la validación cruzada externa de diez particiones es dividida en dos grupos: los datos de entrenamiento (90%) y los datos de prueba (10%).

(iv) Se implementa un algoritmo de optimización (PSO) para sintonizar los hiper-parámetros de los problemas de clasificación. Para aplicar este algoritmo PSO (Particle Swarm

Optimization) se necesita implementar una validación cruzada interna de diez particiones. Los datos de entrenamiento obtenidos en el paso (iii) son divididos en otro 90% para los datos de entrenamiento de la validación cruzada interna y otro 10% para los datos de prueba de la validación cruzada interna.

El algoritmo PSO inicializa algunos parámetros como candidatos para que los hiper-parámetros sean sintonizados. Luego con estos parámetros una SVM es entrenada y probada para cada partición. Cuando la validación cruzada interna de diez particiones se completa, se calcula una matriz de confusión.

Luego la media geométrica entre la sensibilidad y la especificidad es retornada hacia el algoritmo PSO. Este algoritmo PSO se repite hasta que se logra maximizar el valor de la media geométrica. Los parámetros que dieron los mejores resultados para el modelo de predicción de la SVM interna y dieron el valor máximo de la media geométrica, son retornados por el algoritmo PSO.

(v) Los valores retornados por el PSO son los hiper-parámetros óptimos que servirán como parámetros de entrada para realizar el entrenamiento de la SVM (con los datos de entrada iniciales), después este modelo se utiliza para predecir la clase de los datos iniciales de prueba en cada iteración de la validación cruzada externa.

(vi) Cuando se finaliza con la validación cruzada externa de diez particiones, se calcula una matriz de confusión con los valores predichos por los modelos de la SVM. La media geométrica entre la sensibilidad y la especificidad se retorna como medida global de desempeño obtenido en el problema de clasificación.

3.3 Información general de los kernels de secuencias

Para todos los kernels de secuencias implementados en este trabajo se pre-calculan las respectivas matrices kernels, que sirven como parámetro de entrada para realizar la validación cruzada tanto interna como externa.

Para la implementación de estos kernels de secuencias no se necesitaron de las etapas de caracterización ni de selección de características. Las mejores matrices kernel para cada problema son seleccionadas por el algoritmo PSO, además este mismo encuentra los mejores valores para el parámetro de penalización C y los pesos de las clases de la SVM en cada problema.

Todas las simulaciones se llevan a cabo en el programa para computación estadística R (R Core Team, 2015). Los diferentes kernels de secuencias se calculan utilizando el paquete “kebabs”. La SVM es entrenada con el paquete “kernlab”. El algoritmo de PSO se implementa con el paquete “pso”. Todos los paquetes están disponibles de manera gratuita en el proyecto R-CRAN.

3.4 Metodología del Spectrum Kernel

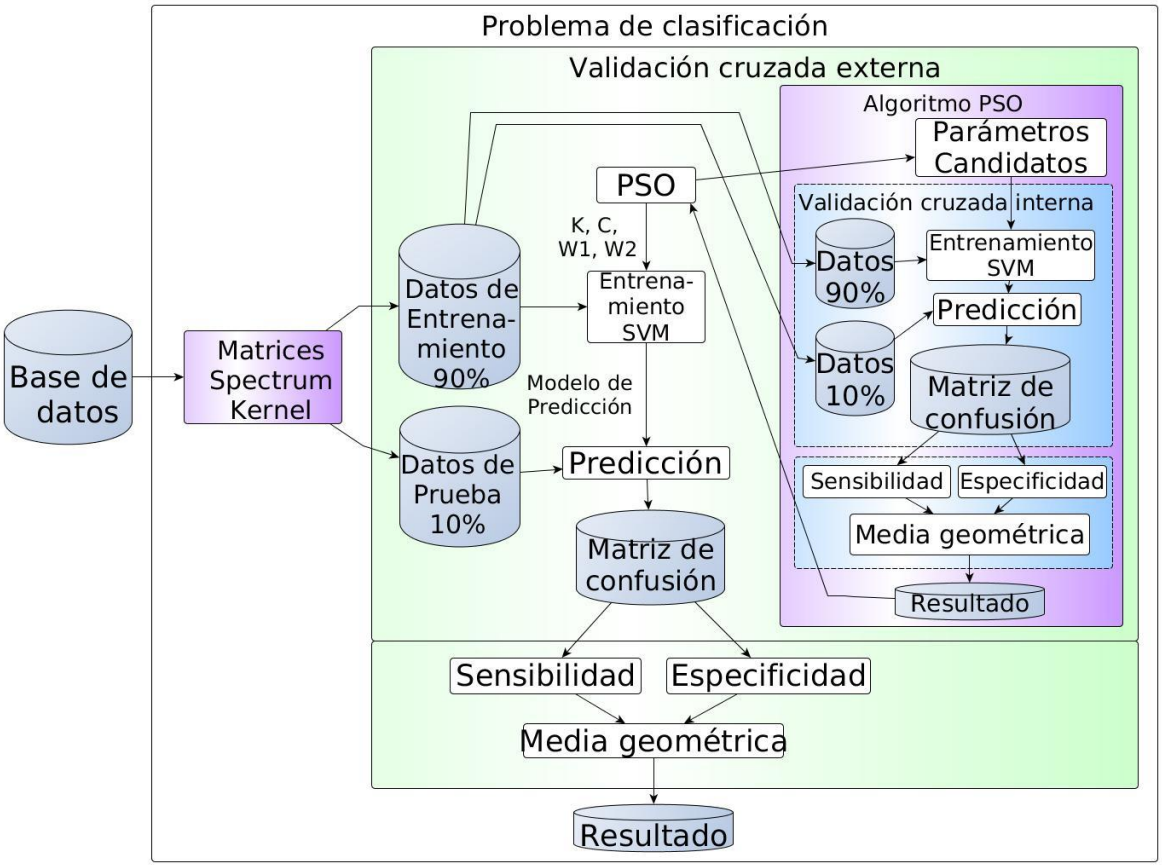

Figura 5. Metodología usada para el Spectrum kernel

La Figura 5 muestra la implementación del Spectrum kernel en conjunto con la metodología descrita en la sección anterior. Los parámetros a sintonizar en esta metodología son: *k* que define la longitud de las sub-secuencias del Spectrum kernel, *C* que es la constante de penalización de la SVM, y finalmente *W1* y *W2* que son los pesos de las clases del problema de clasificación.

En esta metodología, se pre-calculan las matrices kernel para el parámetro *k*= 3, 4, 5, 6.

3.5 Metodología del Mismatch Kernel

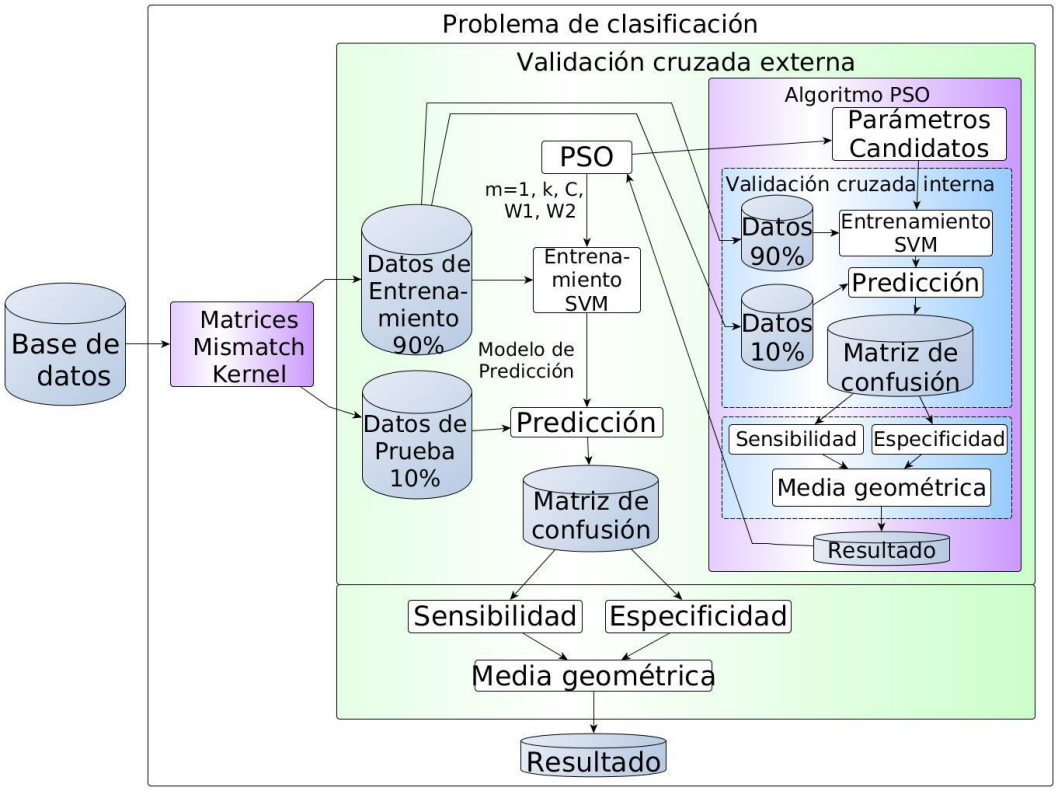

Figura 6. Metodología usada para el Mismatch kernel

La Figura 6 muestra la implementación del Mismatch kernel en conjunto con la metodología general descrita anteriormente. Los parámetros a sintonizar en esta metodología son: *k* que define la longitud de las sub-secuencias del Mismatch kernel, *C* que es la constante de penalización de la SVM, y finalmente *W1* y *W2* que son los pesos de las clases del problema de clasificación. El parámetro *m* que es la cantidad de disparidades permitidas entre las sub-secuencias de longitud *k*, es dejada por defecto con un valor de uno (*m*= 1).

En esta metodología, se pre-calculan las matrices kernel para los parámetros *k*= 3, 4, 5, 6 y *m*= 1. El parámetro m no fue cambiado porque el cálculo del kernel con m> 1 para la base de datos implementada, es muy costoso computacionalmente hablando.

3.6 Metodología del Gappy Pair Kernel

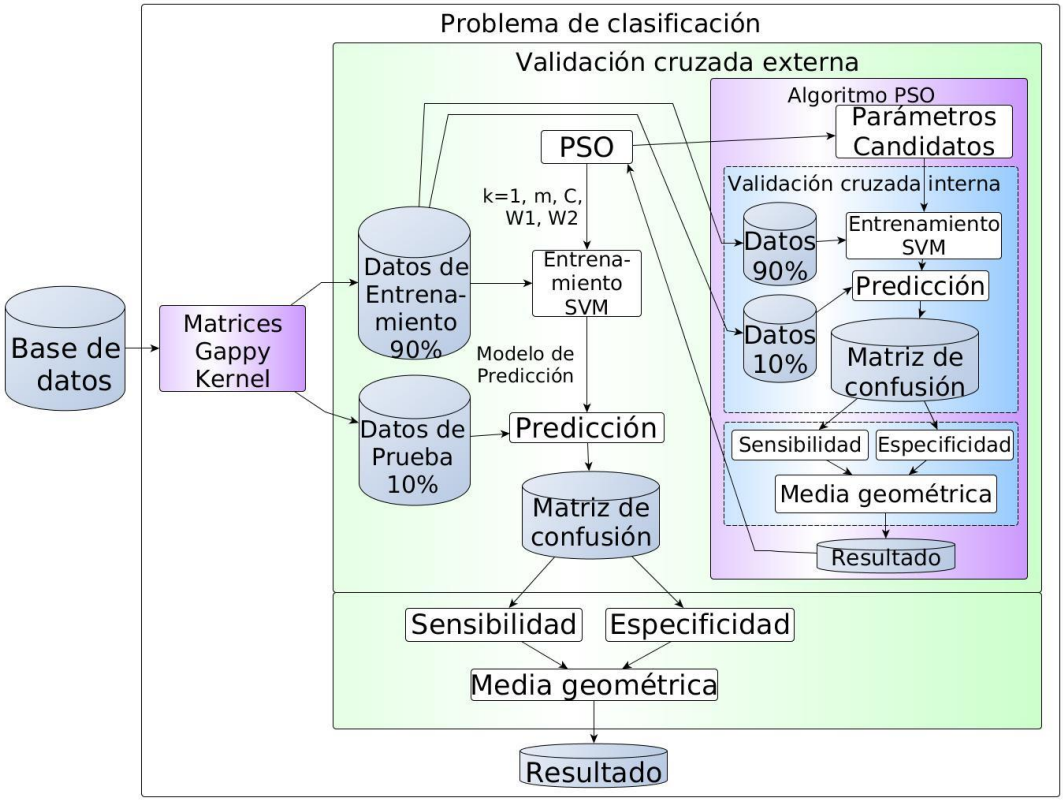

Figura 7. Metodología usada para el Gappy Pair kernel

La Figura 7 muestra la implementación del Gappy Pair kernel en conjunto con la metodología general descrita anteriormente. Los parámetros a sintonizar en esta metodología son: *m* que es la máxima cantidad de espacios permitidos entre las sub-secuencias de longitud *k*, *C* que es la constante de penalización de la SVM, y finalmente *W1* y *W2* que son los pesos de las clases del problema de clasificación. El parámetro *k* que define la longitud de las sub-secuencias del Gappy Pair kernel es dejada por defecto con un valor de uno (*k*= 1).

En esta metodología, se pre-calculan las matrices kernel para los parámetros *k*= 1 y *m*= 5, 6, 7, 8, 9, 10.

3.7 Metodología del kernel de base radial (RBF)

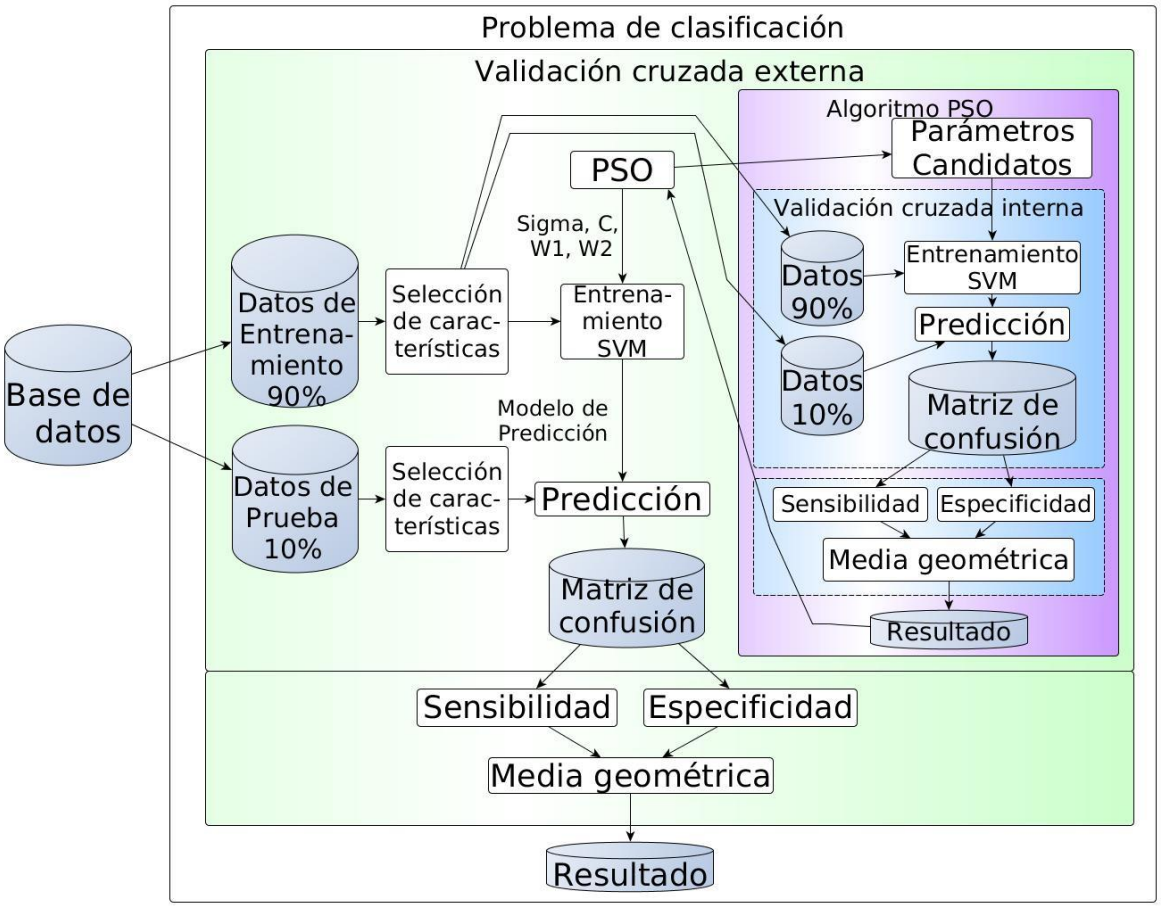

Figura 8. Metodología usada para el kernel de base radial (RBF)

La Figura 8 nos muestra la implementación del kernel RBF en conjunto con la metodología general previamente descrita. Los parámetros a sintonizar en esta metodología son: σ que es una constante perteneciente al kernel RBF, *C* que es la constante de penalización de la SVM, y finalmente *W1* y *W2* que son los pesos de las clases del problema de clasificación.

Dado que el kernel RBF incluye una etapa de caracterización, es necesario realizar un proceso de selección de características en cada iteración de la validación cruzada externa.

Se usó un algoritmo de máquina de vectores de soporte (SVM) con kernel Gaussiano (RBF) para realizar las pruebas de clasificación. Esta SVM fue entrenada con el paquete “kernlab” disponible en el proyecto R-CRAN.

4. RESULTADOS Y DISCUSIÓN

Los resultados de la sensibilidad y la especificidad para el Spectrum kernel, el Mismatch kernel, el Gappy Pair kernel y el kernel RBF, con cada problema de la función molecular, son presentados en las figuras 9, 10, 11 y 12 respectivamente.

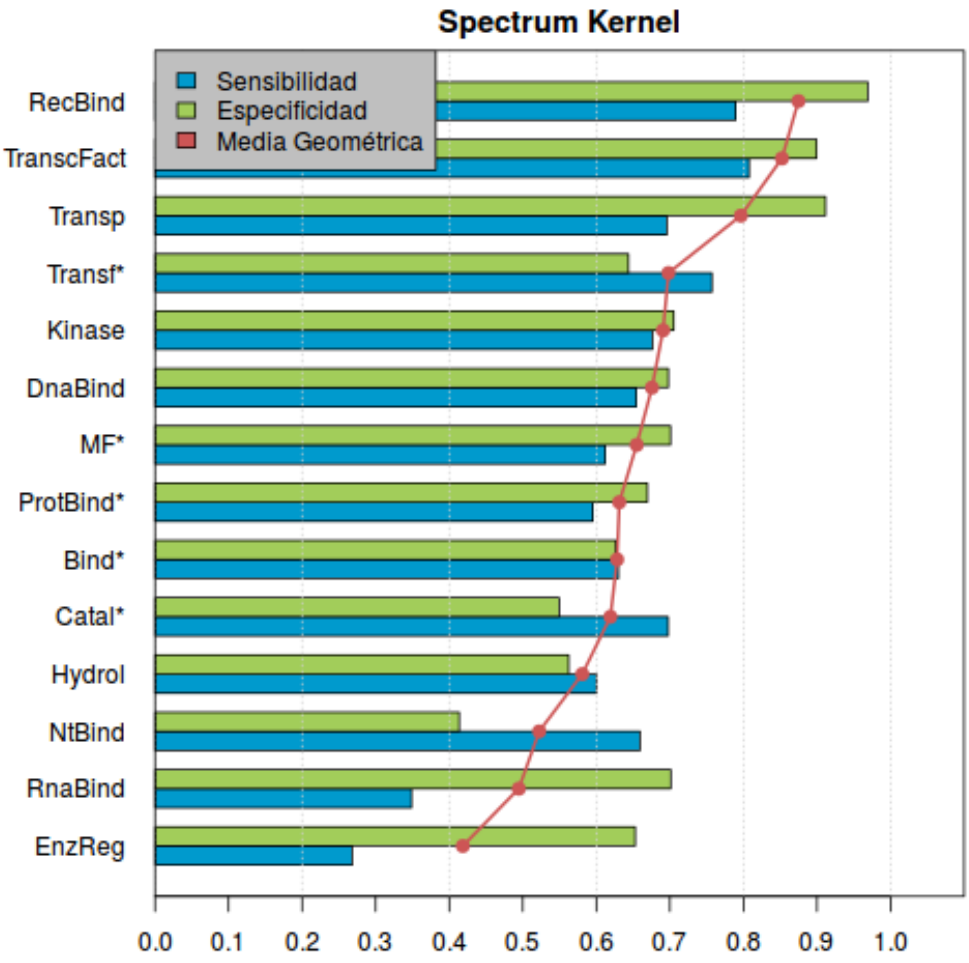

Figura 9. Sensibilidad y especificidad obtenidas en el Spectrum kernel para cada una de las funciones moleculares, la línea roja indica el valor de la media geométrica obtenida entre la sensibilidad y la especificidad y utilizada como medida de desempeño global.

Se puede apreciar que los valores de la sensibilidad y la especificidad en el Spectrum kernel están próximos entre sí, es decir el resultado de la especificidad no dista mucho del valor de la sensibilidad en la mayoría de los problemas de clasificación. Por el contrario, los problemas Enzyme regulator activity, Nucleotic binding y RNA binding con las clases más desbalanceadas (muchas más muestras de la clase negativa que de la clase positiva) presentan resultados en los cuales la Sensibilidad y la especificidad están significativamente distantes.

En la Figura 9, se logra observar que los problemas Receptor binding, Transcription factor activity y Transporter activity alcanzaron los mejores desempeños, y los problemas Enzime regulator acrivity, RNA binding, y Nucleotide binding obtuvieron los desempeños más bajos. En la mayoría de los problemas (con excepción de Enzyme regulator activity) se obtuvo una media geométrica mayor o igual a 0.5.

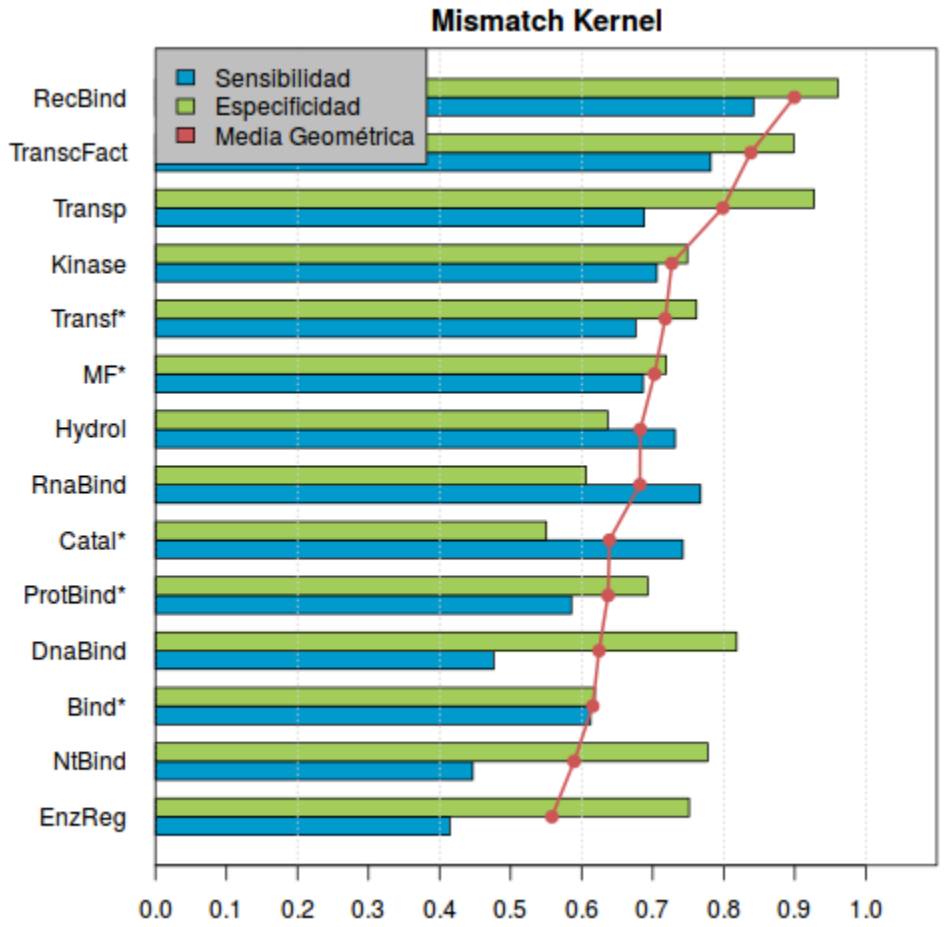

Figura 10. Sensibilidad y especificidad obtenidas en el Mismatch kernel para cada una de las funciones moleculares, la línea roja indica el valor de la media geométrica obtenida entre la sensibilidad y la especificidad y utilizada como medida de desempeño global.

En la Figura 10, se observa que los problemas Receptor binding, Transcription factor activity y Transporter activity obtuvieron los mejores desempeños, y los problemas Enzime regulator acrivity, Nucleotide binding, y Binding lograron los desempeños más bajos.

También se puede observar que los casos en los cuales la sensibilidad y la especificidad se distancian considerablemente una de la otra son Enzyme regulator activity, Nucleotic binding y DNA binding. En todos los problemas de clasificación, el valor de la media geométrica para el mismatch kernel fue superior a 0.5.

Para ambos kernels (Mismatch y Spectrum), las funciones moleculares Receptor binding, Transcription factor activity y Transporter activity obtuvieron los mayores desempeños, de igual manera, las funciones Enzime regulator activity y Nucleotide binding estuvieron entre los problemas con los desempeños más bajos. La función molecular RNA binding mejoró su desempeño de manera significativa utilizando el Mismatch kernel en comparación con el Spectrum kernel.

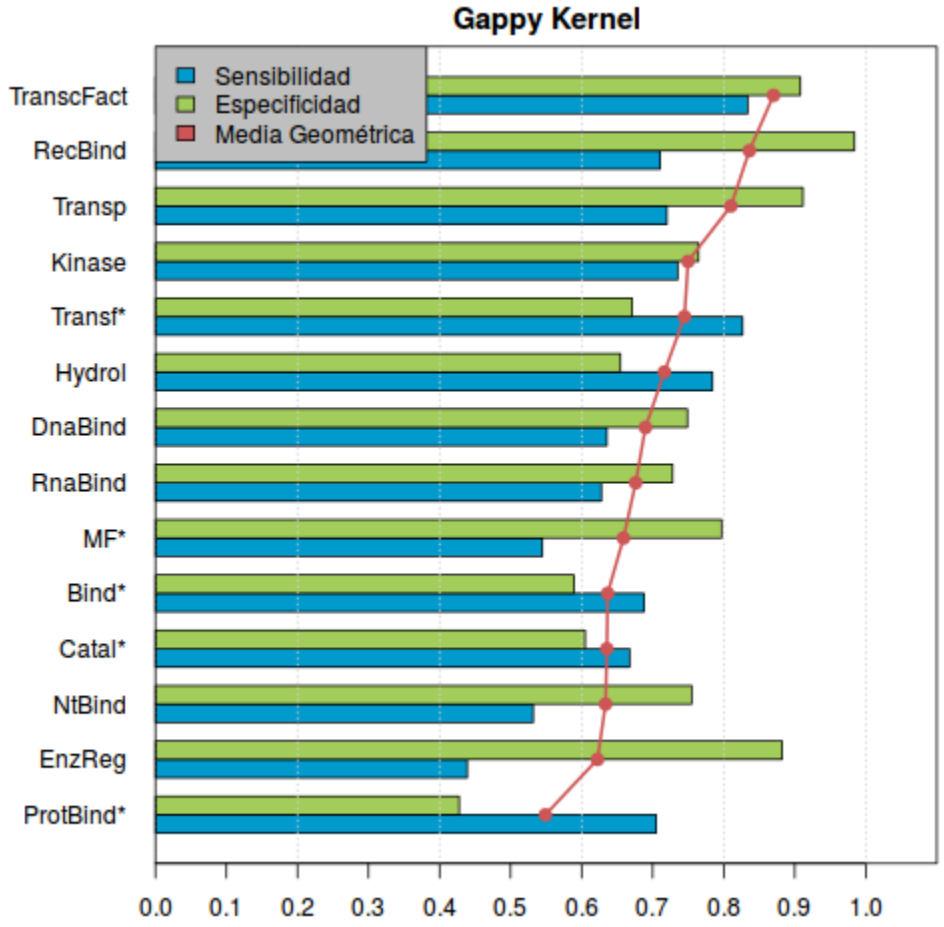

Figura 11. Sensibilidad y especificidad obtenidas en el Gappy Pair kernel para cada una de las funciones moleculares, la línea roja indica el valor de la media geométrica obtenida entre la sensibilidad y la especificidad y utilizada como medida de desempeño global.

Se puede apreciar que en todos los problemas se alcanzó un desempeño superior a 0.5, además los problemas Transcription factor activity, Receptor binding y Transporter activity obtuvieron los mejores desempeños. Igualmente los problemas Nucleotide binding y Enzime regulator activity siguen estando entre los resultados con los desempeños más bajos. A diferencia de los kernels anteriormente mencionados, el Gappy Pair kernel obtiene un resultado más bajo para la función molecular Protein binding.

En los problemas de clasificación Enzyme regulator activity, Nucleotic binding, Protein Binding, Molecular function y Receptor binding se puede apreciar una separación algo considerable entre los resultados de la sensibilidad y la especificidad, especialmente en el problema de clasificación de Enzyme regulator activity.

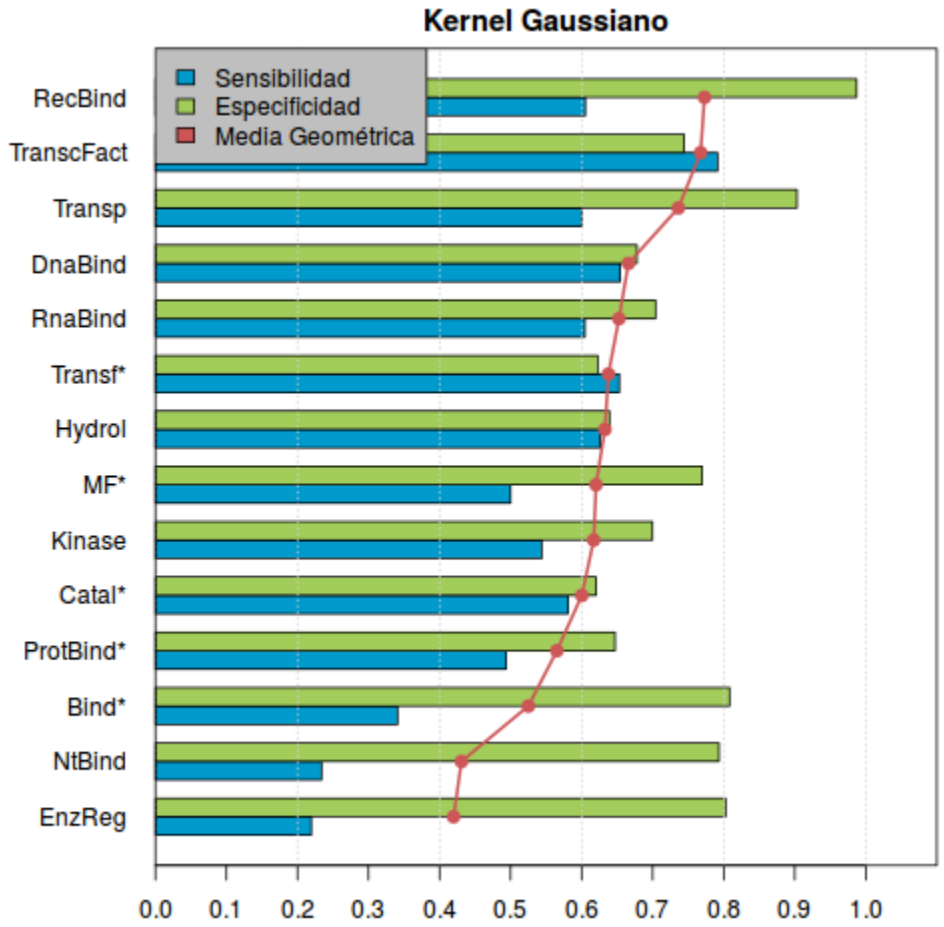

Figura 12. Sensibilidad y especificidad obtenidas en el kernel de base radial para cada una de las funciones moleculares, la línea roja indica el valor de la media geométrica obtenida entre la sensibilidad y la especificidad y utilizada como medida de desempeño global.

El comportamiento de los resultados de la sensibilidad y especificidad en los problemas de clasificación, utilizando el kernel Gaussiano presenta valores de especificidades muy altas y sensibilidades considerablemente más bajas.

Para el kernel Gaussiano las funciones moleculares que obtuvieron el mejor desempeño fueron Receptor binding, Transcription factor activity y Transporter activity conservando la tendencia obtenida en los kernels de secuencias, asimismo los problemas Nucleotide binding y Enzime regulator activity tienen los desempeños más bajos.

Realizando una comparación entre todos los kernels de secuencias las funciones moleculares con los valores de sensibilidad y especificidad más próximos fueron Hydrolase activity, Transcription factor activity, Kinase activity y Binding.

Se logra observar que para todos los kernels mencionados en este trabajo, las funciones moleculares que obtuvieron los mejores desempeños de clasificación fueron Receptor binding, Transcription factor activity y Transcription factor activity. También se puede notar que las funciones moleculares que obtuvieron los desempeños de clasificación más bajos, en todos los kernels, fueron Enzyme regulator activity y Nucleotide binding.

La comparación de la media geométrica entre los kernels se muestra en la Figura 13.

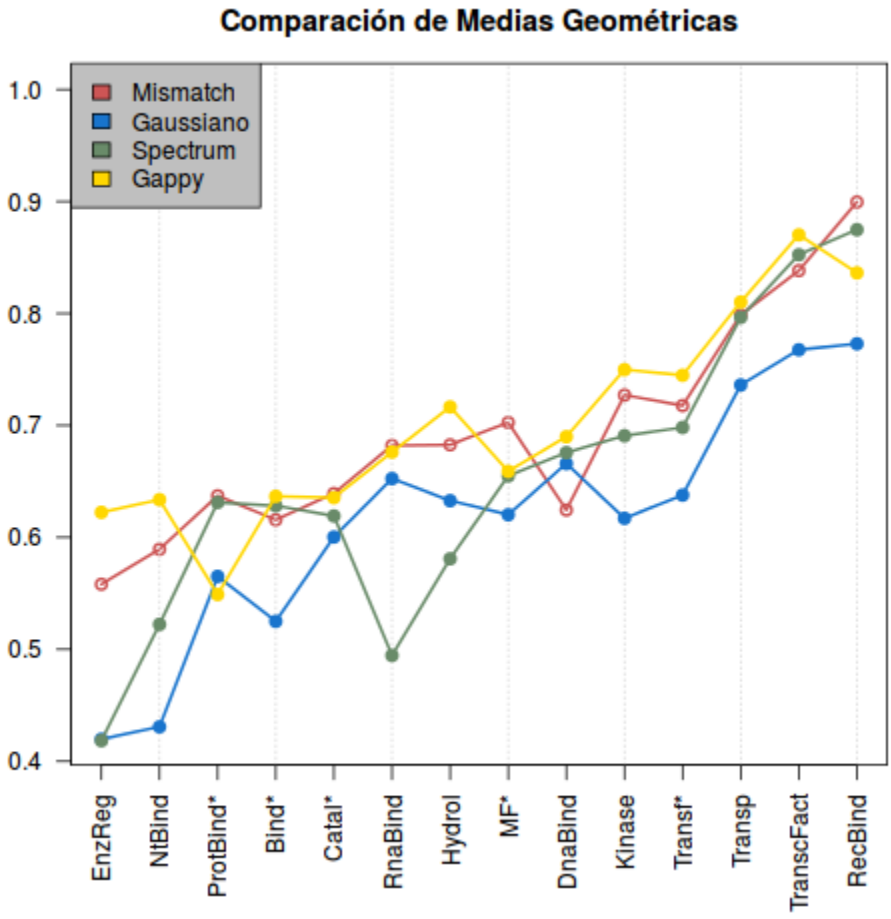

Figura 13. Comparación entre las medias geométricas obtenidas en cada kernel para cada uno de los problemas de clasificación.

Se logra observar en la Figura 13 que el kernel que tuvo los valores más altos en la media geométrica, en la mayoría de los problemas de clasificación fue el Gappy Pair kernel. Un número reducido de clases tuvieron un desempeño menor al 50%.

De todas las clases Receptor Binding obtuvo los resultados de clasificación más altos.

Se demostró que la metodología propuesta usando los kernels de secuencias, para la base de datos previamente descrita, tiene un mejor desempeño que el kernel RBF.

La siguiente tabla muestra cada uno de los problemas de clasificación, con la recomendación de que kernel aplicar para cada problema de clasificación de las funciones moleculares, con el fin de garantizar un buen desempeño en estos.

Función molecular	**Acrónimo**	**Aplicar Kernel**
Nucleotide binding	Ntbind	Gappy Pair
Molecular function*	MF*	Mismatch
DNA binding	DnaBind	Gappy Pair, Spectrum o Rbf
Transcription factor activity	TranscFact	Gappy Pair, Spectrum o Mismatch
RNA binding	RnaBind	Gappy Pair o Mismatch
Catalytic activity*	Catal*	Gappy Pair o Mismatch
Receptor binding	RecBind	Spectrum o Mismatch
Transporter activity	Transp	Gappy Pair, Spectrum o Mismatch
Binding*	Bind*	Gappy Pair, Spectrum o Mismatch
Protein binding*	ProtBind*	Spectrum o Mismatch
Kinase activity	Kinase	Gappy Pair o Mismatch
Transferase activity*	Transf*	Gappy Pair o Mismatch
Hydrolase activity	Hydrol	Gappy Pair
Enzyme regulator activity	EnzReg	Gappy Pair

Tabla 5. Recomendaciones de kernels para resolver los problemas de clasificación de las funciones moleculares

Siguiendo estas recomendaciones, para estos problemas planteados, es posible obtener para todos los problemas de clasificación un desempeño mayor a un 60%.

También se puede observar que con la implementación de la metodología se resolvió el problema del desbalance de clases, ya que muchos de los problemas que estaban más desbalanceados, obtuvieron desempeños bastante buenos, incluso mejores que aquellos problemas que no presentaban de manera tan marcada el problema del desbalance.

5. CONCLUSIONES, RECOMENDACIONES Y TRABAJO FUTURO

Se implementó una metodología que involucra máquinas de vectores de soporte para la predicción de funciones de proteínas usando algunos tipos de kernels de secuencias o string kernels, que miden la similitud entre dos secuencias sin depender de una etapa de caracterización. Esta metodología utiliza algoritmos meta-heurísticos bio-inspirados para encontrar los parámetros óptimos del clasificador. Para mostrar la eficacia de este método, los resultados experimentales se comparan con un kernel de base radial (RBF) utilizando la misma metodología. Para ello se establece la media geométrica entre la sensibilidad y la especificidad del clasificador como medida de desempeño global.

Los resultados indican que los kernels de secuencias obtienen un mejor resultado en la mayoría de los problemas, adicionalmente estos kernels permiten omitir la etapa de caracterización evitando técnicas de reducción de dimensiones, por lo tanto, proveen una poderosa herramienta para la predicción de funciones en secuencias proteicas de plantas terrestres. De igual manera, no es necesario utilizar técnicas convencionales para balancear las clases; tales como SMOTE o sub-muestreo aleatorio, ya que, con esta metodología basada en aprendizaje por sensibilidad de costo, se sintonizó el peso de cada clase utilizando un algoritmo de optimización meta-heurístico.

Como trabajo futuro se pretende utilizar este tipo de kernels bajo un enfoque semi-supervisado utilizándolos como métricas de distancia, en el cálculo de un neighborhood kernel, de igual manera, se pretende utilizar técnicas de aprendizaje multi-etiqueta que permitan transformar el problema de forma eficiente teniendo en cuenta la correlación entre las clases y de esta manera lograr un mejor desempeño.

REFERENCIAS

Altschul, S. F., Gish, W., Miller, W., Myers, E. W., & Lipman, D. J. (1990). Basic local alignment search tool. Journal of molecular biology, 215(3), 403-410.

Altschul, S. F., Madden, T. L., Schäffer, A. A., Zhang, J., Zhang, Z., Miller, W., & Lipman, D. J. (1997). Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. Nucleic acids research, 25(17), 3389-3402.

Bi, R., Zhou, Y., Lu, F., & Wang, W. (2007). Predicting Gene Ontology functions based on support vector machines and statistical significance estimation. Neurocomputing, 70(4), 718-725.

Botstein, D., Cherry, J. M., Ashburner, M., Ball, C. A., Blake, J. A., Butler, H., ... & Eppig, J. T. (2000). Gene Ontology: tool for the unification of biology. Nat Genet, 25(1), 25-29.

Cai, C. Z., Han, L. Y., Ji, Z. L., Chen, X., & Chen, Y. Z. (2003). SVM-Prot: web-based support vector machine software for functional classification of a protein from its primary sequence. Nucleic acids research, 31(13), 3692-3697.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. Journal of artificial intelligence research, 321-357.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine learning,20(3), 273-297.

Duan, K., Keerthi, S. S., & Poo, A. N. (2003). Evaluation of simple performance measures for tuning SVM hyperparameters.Neurocomputing, 51, 41-59.

Eberhart, R., Simpson, P., & Dobbins, R. (1996). Computational intelligence PC tools. Academic Press Professional, Inc..

Eskin, E., Weston, J., Noble, W. S., & Leslie, C. S. (2002). Mismatch string kernels for SVM protein classification. In Advances in neural information processing systems (pp. 1417-1424).

Gribskov, M., McLachlan, A. D., & Eisenberg, D. (1987). Profile analysis: detection of distantly related proteins. Proceedings of the National Academy of Sciences, 84(13), 4355-4358.

Hernández González, N. (2013). Métodos de Kernels en secuencias para la clasificación de residuos catalíticos en sitios activos de enzimas (Doctoral dissertation, Universidad Nacional de Colombia).

J. Kennedy and R. C. Eberhart, “Particle swarm optimization,” in Proc. IEEE Int. Conf. Neural Netw., 1995, pp. 1942–1948

Jaramillo-Garzón, J. A., Gallardo-Chacón, J. J., Castellanos-Domínguez, C. G., & Perera-Lluna, A. (2013). Predictability of gene ontology slim-terms from primary structure information in Embryophyta plant proteins. BMC bioinformatics, 14(1), 68.

Jensen, L. J., Gupta, R., Staerfeldt, H. H., & Brunak, S. (2003). Prediction of human protein function according to Gene Ontology categories. Bioinformatics, 19(5), 635-642.

Jung, J., & Thon, M. R. (2008, December). Gene function prediction using protein domain probability and hierarchical gene ontology information. In Pattern Recognition, 2008. ICPR 2008. 19th International Conference on (pp. 1-4). IEEE.

Jung, J., Yi, G., Sukno, S. A., & Thon, M. R. (2010). PoGO: Prediction of Gene Ontology terms for fungal proteins. BMC bioinformatics, 11(1), 215.

Krogh, A., Brown, M., Mian, I. S., Sjölander, K., & Haussler, D. (1994). Hidden Markov models in computational biology: Applications to protein modeling. Journal of molecular biology, 235(5), 1501-1531.

Kuksa, P., Huang, P. H., & Pavlovic, V. (2008). A fast, large-scale learning method for protein sequence classification. In 8th Int. Workshop on Data Mining in Bioinformatics (pp. 29-37).

Kuksa, P. P., Huang, P. H., & Pavlovic, V. (2009). Scalable algorithms for string kernels with inexact matching. In Advances in Neural Information Processing Systems (pp. 881-888).

Leslie, C. S., Eskin, E., & Noble, W. S. (2002, January). The spectrum kernel: A string kernel for SVM protein classification. In Pacific symposium on biocomputing (Vol. 7, pp. 566-575).

Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., & Watkins, C. (2002). Text classification using string kernels. The Journal of Machine Learning Research, 2, 419-444.

Osuna, E., Freund, R., & Girosi, F. (1997). Support vector machines: Training and applications.

Palme, J., Hochreiter, S., & Bodenhofer, U. (2015). KeBABS: an R package for kernel-based analysis of biological sequences. Bioinformatics, btv176.

Pandey, G., Kumar, V., & Steinbach, M. (2006). Computational approaches for protein function prediction: A survey. Twin Cities: Department of Computer Science and Engineering, University of Minnesota.

Petrova, N. V., & Wu, C. H. (2006). Prediction of catalytic residues using Support Vector Machine with selected protein sequence and structural properties. BMC bioinformatics, 7(1), 312.

R Core Team, R: A Language and Environment for Statistical Computing, R Foundation for Statistical Computing, Vienna, Austria, 2015. [Online]. Available: http://www.R-project.org/

Robinson, J., & Rahmat-Samii, Y. (2004). Particle swarm optimization in electromagnetics. Antennas and Propagation, IEEE Transactions on, 52(2), 397-407.

Saigo, H., Vert, J. P., Ueda, N., & Akutsu, T. (2004). Protein homology detection using string alignment kernels. Bioinformatics, 20(11), 1682-1689.

Shawe-Taylor, J., & Cristianini, N. (2004). Kernel methods for pattern analysis. Cambridge university press.

Tang, Y., Zhang, Y. Q., Chawla, N. V., & Krasser, S. (2009). SVMs modeling for highly imbalanced classification. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, 39(1), 281-288.

Thai-Nghe, N., Gantner, Z., & Schmidt-Thieme, L. (2010, July). Cost-sensitive learning methods for imbalanced data. In Neural Networks (IJCNN), The 2010 International Joint Conference on (pp. 1-8). IEEE.

Yu, L., & Liu, H. (2004). Efficient feature selection via analysis of relevance and redundancy. The Journal of Machine Learning Research, 5, 1205-1224.

Zhang, Y., Xu, J., Zheng, W., Zhang, C., Qiu, X., Chen, K., & Ruan, J. (2014). newDNA-Prot: Prediction of DNA-binding proteins by employing support vector machine and a comprehensive sequence representation. Computational biology and chemistry, 52, 51-59.

APÉNDICES

Apéndice A: *Pseudocódigo de la Metodología para los kernels de secuencias*

Inicio Programa principal

Entrada1: base de datos con las posiciones y las etiquetas de cada problema de clasificación
Entrada2: librerías necesarias para implementar el entrenamiento y la optimización.
Entrada3: matrices kernel del kernel de secuencias que se esté probando

Para *j* **desde** 1 **hasta** número de problemas de clasificación, **hacer:**

Seleccionar las etiquetas correspondientes al Problema de clasificación *j*

Seleccionar las posiciones de las muestras que pertenecen al Problema de clasificación *j*

Dividir la base de datos en 10 particiones iguales, y enumerarlas de 1 hasta 10

Validación cruzada de 10 particiones
Para *i* **desde** 1 **hasta** 10, **hacer:**

Seleccionar las particiones diferentes a i, para realizar el entrenamiento

Seleccionar las etiquetas de los datos pertenecientes a las particiones diferentes a i, para realizar el entrenamiento

Seleccionar la partición i, para realizar la prueba

ParametrosOptimos = Optimización de la función "ValidaciónInterna" para encontrar los parámetros óptimos, con los cuales de entrenará la SVM, esta función "ValidaciónInterna" toma como parámetro de entrada los datos de entrenamiento, las etiquetas de entrenamiento,

las matrices kernel y valores candidatos de los parámetros a optimizar.

Se entrena la SVM con los valores óptimos de la matriz kernel, C, W1 y W2 que se encuentran en la variable *ParametrosOptimos*.

Se prueba el modelo calculado de la SVM con los datos de prueba.

Se guardan los valores de las predicciones realizadas a los datos de prueba de la iteración *i* de la validación cruzada, perteneciente al problema de clasificación *j*.

FinPara

Después de haber realizado la validación cruzada de 10 iteraciones, se calcula una matriz de confusión entre las etiquetas predichas en la validación y las etiquetas reales del problema de clasificación.

De la matriz de confusión calculada, se extraen los valores de los TP, TN, FN y FP para calcular la sensibilidad y la especificidad del problema de clasificación *j*.

Se calcula la media geométrica entre la sensibilidad y la especificidad calculada anteriormente.

Retorno el valor de la media geométrica obtenida, como el valor de desempeño de la clasificación del problema *j*.

FinPara

Fin Programa principal

Función ValidaciónInterna (Datos de entrenamiento de la Validación cruzada externa, Etiquetas de entrenamiento de la Validación cruzada externa, Matrices kernel, valores candidatos de los parámetros a optimizar**):**

Datos = Datos de entrenamiento de la Validación cruzada externa.

Etiquetas = Etiquetas de entrenamiento de la Validación cruzada externa.

ParametrosCandidatos = valores candidatos de los parámetros a optimizar.

```
# Validación cruzada de 10 particiones
```

Dividir los datos en 10 particiones iguales, y enumerarlas de 1 hasta 10

Para *i* **desde** 1 **hasta** 10, **hacer:**

Seleccionar las particiones diferentes a i, para realizar el entrenamiento

Seleccionar las etiquetas de los datos pertenecientes a las particiones diferentes a i, para realizar el entrenamiento

Seleccionar la partición i, para realizar la prueba

Se entrena la SVM con los valores de ParametrosCandidatos de la matriz kernel, C, W1 y W2.

Se prueba el modelo calculado de la SVM con los datos de prueba.

Se guardan los valores de las predicciones realizadas a los datos de prueba de la iteración *i* de la validación cruzada.

FinPara

Después de haber realizado la validación cruzada de 10 iteraciones, se calcula una matriz de confusión entre las etiquetas predichas en la validación y las etiquetas reales del problema de clasificación.

De la matriz de confusión calculada, se extraen los valores de los TP, TN, FN y FP para calcular la sensibilidad y la especificidad del problema de clasificación.

Se calcula la media geométrica entre la sensibilidad y la especificidad calculada anteriormente.

Retorno el valor de (1 - la media geométrica obtenida), como el valor de desempeño de la clasificación del problema.

Fin de la función.

Apéndice B: *Pseudocódigo de la Metodología con kernel de base radial*

Inicio del Programa principal

Entrada1: base de datos con las posiciones y las etiquetas de cada problema de clasificación.

Entrada2: librerías necesarias para implementar el entrenamiento y la optimización.

Entrada3: base de datos con las características físico-químicas.

Para *j* **desde** 1 **hasta** el número de problemas de clasificación, **hacer**:

Selecciono las etiquetas correspondientes al Problema de clasificación *j*

Selecciono las posiciones de las muestras que pertenecen al Problema de clasificación *j*

Características = datos con las características físico-químicas.

Dividir la base de datos en 10 particiones iguales, y enumerarlas de 1 hasta 10

Validación cruzada de 10 particiones

Para *i* **desde** 1 **hasta** 10, **hacer**:

Seleccionar las particiones diferentes a i, para realizar el entrenamiento

Seleccionar las etiquetas de los datos pertenecientes a las particiones diferentes a i, para realizar el entrenamiento

Seleccionar la partición i, para realizar la prueba

SelecciónaCaracterísticas = Realizar selección de características al conjunto de entrenamiento con las características físico-químicas.

ParametrosOptimos = Optimización de la función "ValidaciónInterna" para encontrar los parámetros óptimos, con los cuales de entrenará la SVM, esta función "ValidaciónInterna" toma como parámetro de entrada los datos de entrenamiento, las etiquetas de entrenamiento, la variable

"Características" y valores candidatos de los parámetros a optimizar.

Se entrena la SVM con los valores óptimos de sigma, C, W1 y W2 que se encuentran en la variable ParametrosOptimos y con los datos de "SeleccionáCaracterísticas".

Se prueba el modelo calculado de la SVM con los datos de prueba.

Se guardan los valores de las predicciones realizadas a los datos de prueba de la iteración i de la validación cruzada, perteneciente al problema de clasificación *j*.

Fin

Después de haber realizado la validación cruzada de 10 iteraciones, se calcula una matriz de confusión entre las etiquetas predichas en la validación y las etiquetas reales del problema de clasificación.

De la matriz de confusión calculada, se extraen los valores de los TP, TN, FN y FP para calcular la sensibilidad y la especificidad del problema de clasificación *j*.

Se calcula la media geométrica entre la sensibilidad y la especificidad calculada anteriormente.

Retorno el valor de la media geométrica obtenida, como el valor de desempeño de la clasificación del problema *j*.

Fin

Fin del Programa principal

Función ValidaciónInterna (Datos de entrenamiento de la Validación cruzada externa, Etiquetas de entrenamiento de la Validación cruzada externa, Datos con las características físico-químicas, valores candidatos de los parámetros a optimizar**):**

Datos = Datos de entrenamiento de la Validación cruzada externa.

Etiquetas = Etiquetas de entrenamiento de la Validación cruzada externa.

ParametrosCandidatos = valores candidatos de los parámetros a optimizar.

Validación cruzada de 10 particiones

Dividir los datos en 10 particiones iguales, y enumerarlas de 1 hasta 10

Para *i* **desde** 1 **hasta** 10, **hacer:**

Seleccionar las particiones diferentes a i, para realizar el entrenamiento

Seleccionar las etiquetas de los datos pertenecientes a las particiones diferentes a i, para realizar el entrenamiento

Seleccionar la partición i, para realizar la prueba

Se entrena la SVM con los valores de ParametrosCandidatos C, Sigma, W1, W2 y con los datos de "SeleccionaCaracterísticas".

Se prueba el modelo calculado de la SVM con los datos de prueba.

Se guardan los valores de las predicciones realizadas a los datos de prueba de la iteración *i* de la validación cruzada.

Fin

Después de haber realizado la validación cruzada de 10 iteraciones, se calcula una matriz de confusión entre las etiquetas predichas en la validación y las etiquetas reales del problema de clasificación.

De la matriz de confusión calculada, se extraen los valores de los TP, TN, FN y FP para calcular la sensibilidad y la especificidad del problema de clasificación *j*.

Se calcula la media geométrica entre la sensibilidad y la especificidad calculada anteriormente.

Retorno el valor de (1 - la media geométrica obtenida), como el valor de desempeño de la clasificación del problema.

Fin de la función.

Apéndice C: *Gráficas que comparan las sensibilidades y especificidades obtenidas entre kernels, para cada función molecular.*

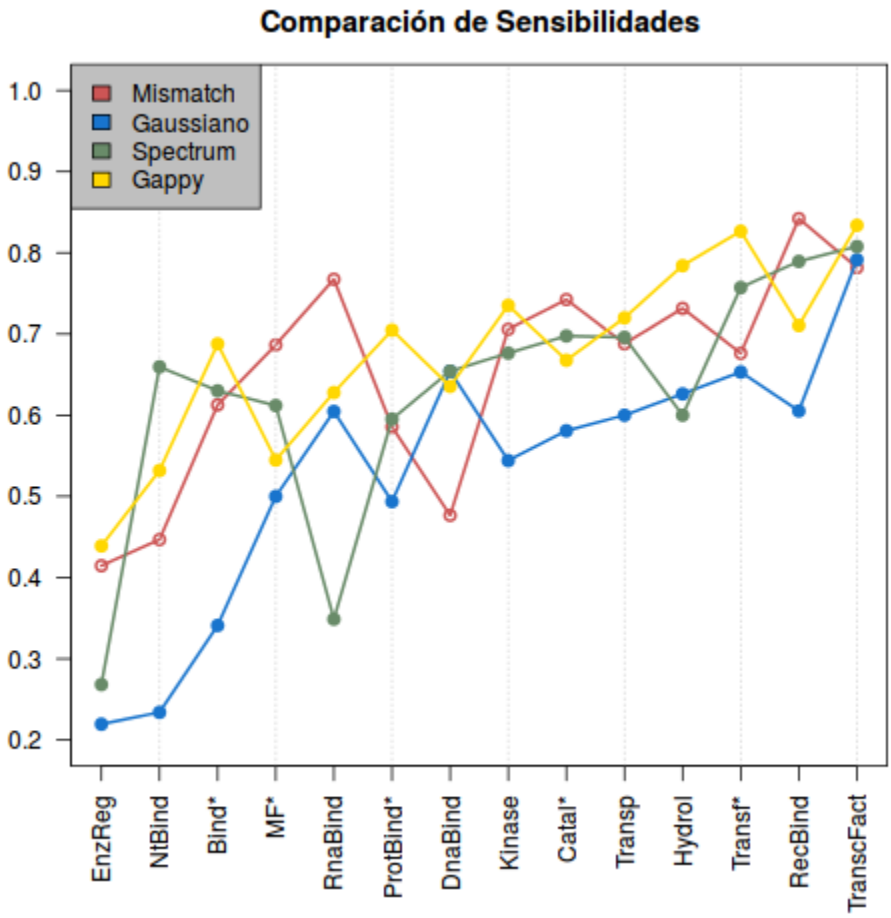

Figura 1. Comparación entre las sensibilidades obtenidas en cada kernel para cada uno de los problemas de clasificación.

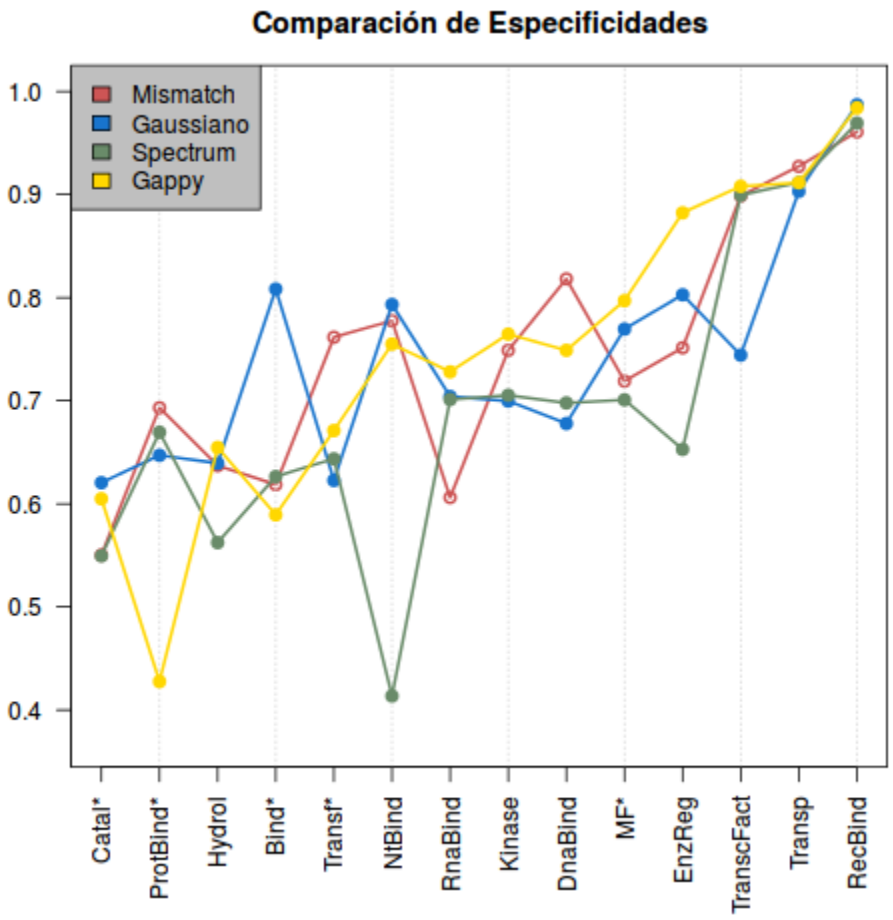

Figura 2. Comparación entre las especificidades obtenidas en cada kernel para cada uno de los problemas de clasificación.

FIRMA ESTUDIANTES 

Juan Camilo Pineda Iral

FIRMA ASESOR Jorge A. Jaramillo G.

FECHA ENTREGA: 12/11/2015

FIRMA COMITÉ TRABAJO DE GRADO DE LA FACULTAD_____

RECHAZADO___ ACEPTADO____ ACEPTADO CON MODIFICACIONES_____

ACTA NO._____

FECHA ENTREGA: _____

FIRMA CONSEJO DE FACULTAD_____

ACTA NO._____

FECHA ENTREGA: _____