SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO
Instituto Tecnológico de La Paz

INSTITUTO TECNOLÓGICO DE LA PAZ
DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN
MAESTRÍA EN SISTEMAS COMPUTACIONALES

ALGORITMO CONCURRENTE PARA LA PREDICCIÓN DE AFINIDAD DE PÉPTIDOS CON MHC-II

TESIS

QUE PARA OBTENER EL GRADO DE
MAESTRO EN SISTEMAS COMPUTACIONALES

PRESENTA:
ING. ALEJANDRO DIBENE DEL VALLE

DIRECTORES DE TESIS:
DR. MARCO ANTONIO CASTRO LIERA
DR. CARLOS ELIUD ANGULO VALADEZ

LA PAZ, BAJA CALIFORNIA SUR, MÉXICO, JUNIO 2017.

"Año del Centenario de la Promulgación de la Constitución Política de los Estados Unidos Mexicanos"

La Paz, Baja California Sur, **19/junio/2017**

OFICIO No. DEPI/237/2017

ASUNTO: Autorización de impresión

C. ALEJANDRO DIBENE DEL VALLE,
ESTUDIANTE DE LA MAESTRÍA EN
SISTEMAS COMPUTACIONALES,
P R E S E N T E .

Con base en el dictamen de aprobación emitido por el Comité Tutorial de la Tesis denominada: **"ALGORITMO CONCURRENTE PARA LA PREDICCIÓN DE AFINIDAD DE PÉPTIDOS CON MHC-II"**, mediante la opción de tesis (Proyectos de Investigación), entregado por usted para su análisis, le informamos que se **AUTORIZA** la impresión

A T E N T A M E N T E
"CIENCIA ES VERDAD, TÉCNICA ES LIBERTAD"

MC. MAURICIO SALVADOR RODRÍGUEZ OJEDA,
JEFE DE LA DIV. DE EST. DE POSGRADO E INV.

INSTITUTO TECNOLÓGICO DE LA PAZ
DIVISIÓN DE ESTUDIOS DE POSGRADO
E INVESTIGACIÓN

c.c.p. Archivo.
MSRO/icl*

DICTAMEN DEL COMITÉ TUTORIAL

SUBDIRECCIÓN ACADÉMICA
DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN.

La Paz, B.C.S., **19/junio/ 2017**

C. M.C. MAURICIO SALVADOR RODRÍGUEZ OJEDA,
JEFE DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN,
P R E S E N T E.

Por medio del presente, enviamos a usted dictamen del Comité Tutorial de tesis para la obtención del grado de Maestro, con los siguientes datos generales:

No. de Control M13310015	Nombre ALEJANDRO DIBENE DEL VALLE
Maestría en:	SISTEMAS COMPUTACIONALES
Título de la tesis: ALGORITMO CONCURRENTE PARA LA PREDICCIÓN DE AFINIDAD DE PÉPTIDOS CON MHC-II	
DICTAMEN: Se autoriza el trabajo de investigación, en virtud de que realizó las correcciones correspondientes conforme a las observaciones planteadas por este Comité Tutorial.	

A t e n t a m e n t e .
El Comité Tutorial

M.C. JESÚS ANTONIO CASTRO

DR. SAÚL MARTÍNEZ DÍAZ

DR. MARCO ANTONIO CASTRO LIERA

c.c.p. Coordinador de la Maestría.
c.c.p. Departamento de Servicios Escolares.
c.c.p. Estudiante.

AGRADECIMIENTOS

Quiero expresar mi más profundo agradecimiento a mis padres, a quienes les debo lo que soy, lo que he hecho y lo que seguiré haciendo durante toda mi vida.

Se agradece al CONACYT por el apoyo a través del proyecto SEP-CONACYT No. 151818: Identificación y Evaluación de Vacunas Comestibles en Alfalfa para Prevenir la Paratuberculosis (Mycobacterium avium subps. Paratuberculosis) de Rumiantes Domésticos de México.

Agradezco también a mis directores de tesis, Dr. Carlos Eliud Angulo Valadez y Dr. Marco Antonio Castro Liera quienes, con paciencia, estuvieron siempre dispuestos a ayudarme y solventar todas esas dudas que me asaltaron, por muy básicas que fuesen.

Un agradecimiento especial para el MSC. Joel Artemio Morales Viscaya, estimado amigo y compañero cuyo conocimiento no dudó nunca en compartir conmigo para la realización de este trabajo.

Por último y desde lo más profundo de mi corazón agradezco y dedico este trabajo a mi amada Daniela, quien con tiernos empujones me impulsó a concluir en seis meses lo que no pude hacer yo solo en un año y medio.

RESUMEN

El presente trabajo aborda un tema de suma importancia para el campo de la inmunología, el desarrollo de vacunas. Especialistas en la materia hacen uso día con día de diversas herramientas para la identificación de proteínas como candidatas ideales para la creación de vacunas efectivas.

La principal característica de todas estas herramientas disponibles es su funcionamiento bajo el esquema tradicional de la programación secuencial, por lo que en vista de la falta de propuestas alternativas y haciendo uso del enfoque actual que favorece la aplicación de nuevas tecnologías para mejorar procesos en diversos campos del conocimiento, en el transcurso de esta tesis se describe el proceso para el desarrollo de una herramienta informática que facilita la identificación de proteínas como posibles candidatas para el desarrollo de vacunas.

Los tópicos principales involucrados en la codificación de esta aplicación inmuno-informática son la computación paralela como enfoque de programación y una implementación del Método Estabilizador de Matrices (SMM por sus siglas en ingles) que opera gracias a los metodos de descenso máximo y bagging, con dicho método se logra una predicción eficaz y con un menor coste de tiempo con respecto a las herramientas ya existentes en la bibliografía actual.

ABSTRACT

The present work deals with a uttermost important topic for field immunology, vaccine development. Everyday specialist on the matter use diverse software tools aimed to identify good candidate proteins for the creation of efective vaccines.

The comun trait of the above mentioned software tools is the use of a single processor scheme for their operation. This work describes the development of a more efficient software that uses a newly available parrallel technology known as general purpose graphical processing units (GPGPU).

The main components of the proposed application are parallel computing and an implementation of the Stabilized Matrix Method (SMM) applying gradient-descend and bagging. Using this implementation an effective prediction of the binding of immunogenic peptides to major histocompatibility complex (MHC) molecules is achieved using less time than the currently available applications and without the need of Internet access.

INDICE GENERAL

Índice de figuras III

Índice de tablas IV

CAPITULO I. INTRODUCCIÓN 1

1.1 Antecedentes 1
1.2 Delimitación del problema 2
1.3 Justificación 2
1.4 Hipótesis 3
1.5 Objetivo general 3
1.6 Objetivos específicos 4
1.7 Organización del trabajo 4

CAPÍTULO II. CONCEPTOS GENERALES 6

2.1 El sistema inmune 6
2.2 Complejo mayor de histocompatibilidad 9
2.2.1 Proteína 10
2.2.2 Antígeno 11
2.2.3 Anticuerpos 12
2.2.4 Aminoácidos 12
2.2.5 Linfocito B 13
2.2.6 Péptido 13
2.2.7 Epítopo 13
2.2.8 Alelo 14
2.3 Inmunología computacional 14
2.3.1 Vacunología inversa 15
2.4 Arquitectura cliente – servidor 16
2.5 Lenguace de programación C 17
2.6 Colección de compiladores GNU 18
2.7 Linux mint 19
2.8 Biblioteca científica GNU 20

CAPÍTULO III. PREDICCIÓN DE AFINIDAD PÉPTIDO – MHC II **21**

3.1 Antecedentes 21
3.2 Base de datos IEDB 22
3.3 Servidor NetMHC-II 23

CAPÍTULO IV. CÓMPUTO DE PROPÓSITO GENERAL EN UNIDADES DE PROCESAMIENTO GRÁFICO **25**

4.1 Paralelismo 26
4.2 Lenguajes de programación para GPGPU 28
4.3 Estructura de un programa en CUDA 29

CAPÍTULO V. MÉTODO ESTABILIZADOR DE MATRICES CON DESCENSO MÁXIMO **31**

5.1 Principios del método 31
5.2 Validación cruz 32
5.3 Descenso de gradiente 33

CAPÍTULO VI. CODIFICACIÓN SECUENCIAL E IMPLEMENTACIÓN PARALELA **36**

6.1 Estructura de la aplicación 36
6.1.1 Origen de los datos 37
6.1.2 Pre-procesamiento 38
6.1.3 Construcción de la matriz H 39
6.1.4 Descenso máximo 40
6.1.5 Método de bagging 41
6.1.6 Bibliotecas 42
6.2 Proceso de paralelización 42
6.3 Acciones dentro de la GPU 43

CAPÍTULO VII. RESULTADOS Y CONCLUSIONES **45**

7.1 Resultados 45

BIBLIOGRAFÍA **57**

Índice de figuras

Figura 1: Conformación del sistema inmune 6

Figura 2: Interacción entre la CPA y el antígeno 9

Figura 3: Ejemplo de estructura proteínica 11

Figura 4: Interacción antígeno – anticuerpo 11

Figura 5: Interacción célula T – célula B 13

Figura 6: Epítopos localizados en la superficie celular 14

Figura 7: Diagrama cliente – servidor 17

Figura 8: Disposición lógica de espacios de memoria en tarjetas NVIDIA para CUDA 26

Figura 9: Representación de matrices en CUDA 27

Figura 10: Ejemplo de grids aplicados a un programa en CUDA 30

Figura 11: Linealización de una matriz 43

Figura 12: Resultados de afinidad utlizando el método de descenso máximo para TrainInput2 46

Figura 13: Resultados de afinidad utlizando el método de bagging para TrainInput2 46

Figura 14: Resultados de afinidad utlizando el método de descenso máximo para TrainInput3 47

Figura 15: Resultados de afinidad utlizando el método de bagging para TrainInput3 47

Figura 16: Resultados de afinidad utlizando el método de descenso máximo para TrainInput4 48

Figura 17: Resultados de afinidad utlizando el método de bagging para TrainInput4 48

Índice de tablas

Tabla 1: Aminoácidos presentes en el proceso de predicción 12

Tabla 2: Resumen de datos disponibles en IEDB 23

Tabla 3: Resultados de predicción arrojados por NetMHCII 24

Tabla 4: Comparativa entre descenso máximo y bagging para TrainInput2 51

Tabla 5: Comparativa entre descenso máximo y bagging para TrainInput3 53

Tabla 6: Comparativa entre descenso máximo y bagging para TrainInput4 55

CAPÍTULO I. INTRODUCCIÓN

1.1 Antecedentes

El desarrollo de vacunas ha supuesto un importante avance en el campo de la medicina, específicamente en el área de la inmunología. Ha permitido eliminar enfermedades funestas como la viruela, estableciéndose como un método de prevención contra cada nueva enfermedad que aparece con el paso del tiempo.[1]

La inmunología es una sub-disciplina de la medicina que se encarga específicamente del estudio de los procesos moleculares y celulares implicados en la defensa de la integridad biológica del organismo a través de la identificación de sustancias propias y la detección de las sustancias extrañas. Todo ello con el objetivo principal de destruir a los microorganismos patógenos y evitar así las infecciones.[2]

La vacunología es el área dentro de la inmunología que se encarga del desarrollo de las vacunas aplicando principios del sistema inmune en un ambiente controlado con pruebas *in vivo* (realizadas directamente en el tejido interno del sujeto experimental), *in vitro* (realizadas a nivel celular y fuera del organismo) y mas recientemente *in silico* (realizadas mediante herramientas informáticas).

Actualmente la vacunología, con ayuda de herramientas informáticas, se vale de un novedoso proceso denominado vacunología inversa. Dicho proceso consiste en el análisis bio-informático de secuencias completas para la identificación de proteínas que después serán utilizadas en procesos de purificación e inmunización dentro de pruebas *in vitro* o *in vivo* para confirmar su exposición de superficie y eficiencia bactericida y así seleccionar finalmente los componentes idóneos para el desarrollo de las vacunas.[3]

El presente proyecto, abarca el proceso de desarrollo de una nueva herramienta inmuno-informática auxiliar para la identificación de proteínas como potenciales candidatos para el posterior desarrollo de vacunas, basada en los mecanismos de la respuesta inmuno-específica mediada por células. Esta herramienta será para uso por parte del Centro de Investigaciones Biológicas del Noroeste (CIBNOR), centro especializado en la generación y divulgación de conocimiento científico dentro del campo de las ciencias biológicas; así como en la formación de científicos y expertos en el uso , manejo y preservación de los recursos naturales.

1.2 Delimitación del problema

En el campo de la Vacunología se han venido desarrollando y mejorando diversas herramientas inmuno-informáticas, entre ellas las de predicción de afinidad de epítopos (péptidos) al complejo mayor de histocompatibilidad presente en la respuesta inmune específica, como una alternativa para la identificación de proteínas candidatas a vacunas. Sin embargo, dicho proceso conlleva la utilización de diferentes servidores web independientes entre sí, que deben ser utilizados de forma secuencial y que generan resultados heterogéneos, que deben ser comparados y homogeneizados para su uso en el laboratorio. Lo anterior implica largos periodos de procesamiento, esfuerzos invertidos en actividades que podrían optimizarse y una dependencia total de la conexión a internet.

1.3 Justificación

Dado el proceso actual de identificación de epítopos afines a la molécula MHC-II para la selección de proteínas como potenciales candidatos para el diseño de vacunas, se propone una aplicación paralela de escritorio que tendrá varios beneficios como permitir establecer un punto inicial para el futuro desarrollo de toda una plataforma de herramientas inmuno-informáticas integradas para el auxilio en la realización de diversas tareas en el laboratorio.

Al ser una aplicación de escritorio se prescinde por completo de una conexión a internet para su funcionamiento, todo el proceso de predicción funciona entonces a nivel local, es decir, en sitio y sin necesidad de procesamiento externo, transmisión o recepción de datos vía web, todo esto representa un ahorro de costo en tiempo y una aumento en la eficiencia de la aplicación.

Por último, se apuesta por la utilización del paradigma de programación paralela para la construcción de esta aplicación de predicción, dicho paradigma será abordado detalladamente más adelante, pero en términos generales significa que se le da a la aplicación la capacidad de utilizar la tecnología de una GPGPU (General Purpose Graphic Processing Unit) para procesar todos los datos de forma concurrente, es decir, al mismo tiempo usando todas las ventajas ofrecidas por la arquitectura de la tarjeta gráfica NVIDIA a través de CUDA, lo anterior representa un aumento considerable en la capacidad de procesamiento general de la aplicación.

1.4 Hipótesis

Con el desarrollo de la aplicación de escritorio paralela, para la predicción de afinidad de epítopos con el complejo mayor de histocompatibilidad clase II, será posible obtener dichas predicciones de manera eficaz, eficiente, con bajo costo y sin depender del acceso a Internet.

1.5 Objetivo general

Desarrollar, mediante el lenguaje de programación Cuda C, una aplicación paralela de escritorio que implemente un algoritmo basado en el Método Estabilizador de Matrices (SMM) y combinado con el método numérico de Descenso de Gradiente, para la predicción de enlace de péptidos al complejo mayor de histocompatibilidad clase II.

1.6 Objetivos específicos

- Analizar el contenido del servidor web para obtener los algoritmos implementados en el proceso de predicción del grado de afinidad de epítopos con MHC II.
- Desarrollar la aplicación paralela en base al esquema de paralelización apropiado que permita la predicción eficiente del grado de afinidad de epítopos con MHC II.
- Validar la aplicación paralela empleando secuencias de epítopos cuya compatibilidad ha sido previamente verificada de manera biológica y comparando su afinidad *in vitro* con la predicha por la aplicación *in silico*.

1.7 Organización del trabajo

A continuación se listan los temas tratados en cada uno de los capítulos subsecuentes contenidos en la presente tesis:

- Capítulo 2.- Comprende un compendio de los conceptos clave para la construcción de una base teórica fundamentada que permita esclarecer el contexto biológico implicado en el desarrollo de este proyecto.
- Capítulo 3.- Describe los antecedentes más importantes relacionados con el proyecto, aplicaciones inmuno-informáticas y herramientas auxiliares para predicción de enlace utilizando diversos algoritmos.
- Capítulo 4.- Establece los conceptos más importantes respecto al tema de la computación paralela con CUDA, parte fundamental de la aplicación desarrollada.
- Capítulo 5.- Explica la estructura, secuencia y funcionamiento del algoritmo principal “Stabilized Matrix Method” en el que se basa el proceso de predicción de la aplicación descrita en este proyecto.

- Capítulo 6.- Trata todos los puntos respecto al desarrollo de la parte paralela del proyecto, desde la implantación de la estrategia de paralelización hasta la codificación en CUDA C.
- Capítulo 7.- El apartado final que compila todos los datos acerca de los resultados obtenidos en las pruebas de software, las validaciones estadísticas y las conclusiones obtenidas respecto a todo lo anterior.

CAPÍTULO II. CONCEPTOS GENERALES

2.1 El sistema inmune

Para poder defenderse de todo aquello que sea ajeno, los organismos, cuentan con barreras naturales de aislamiento tales como la piel y la mucosa, así como de un *sistema inmune* con la capacidad de identificar y destruir cualquier elemento extraño que invade nuestro organismo.[4]

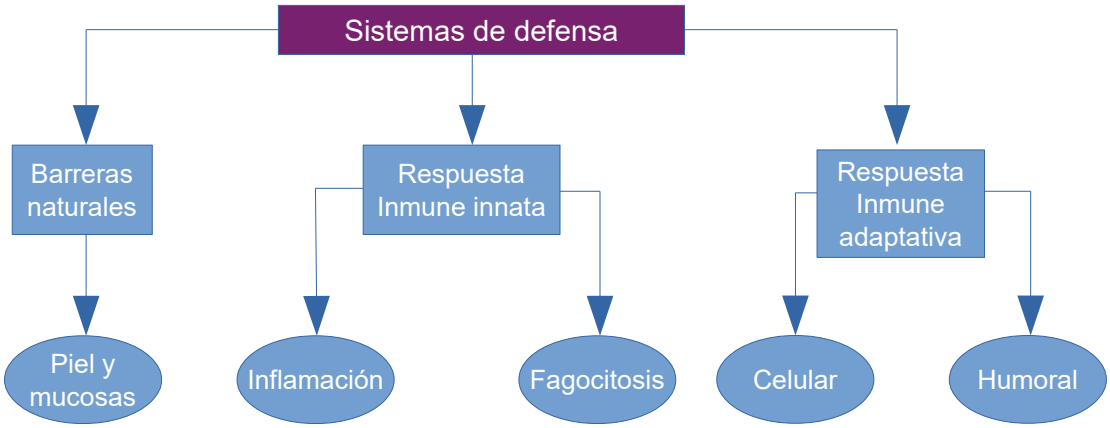

Figura 1: Conformación del sistema inmune

La inmunología es precisamente la ciencia que estudia los procesos moleculares y celulares implicados en la defensa de la integridad biológica del organismo a través de la identificación de las sustancias propias y la detección de las sustancias extrañas. Todo lo anterior con el objetivo principal de detectar y destruir microorganismos patógenos para evitar infecciones.[4]

Cada organismo además, cuenta con dos tipos de mecanismos de defensa diferentes, innatos y adaptativos.

Los mecanismos de tipo innato, también llamados respuesta inmune innata, son aquellos con los que el organismo cuenta desde el momento de su nacimiento y representan la primera línea de defensa. Este tipo de respuesta inmune es no-específica y no genera una memoria del antígeno, se pueden clasificar de la siguiente manera:[5]

- Mecanismos físicos y químicos como la piel, la mucosa y la saliva, es decir, las barreras naturales.
- Fagocitosis, el proceso de engullir y destruir a los organismos invasores, llevada a cabo por macrófagos y células dendríticas.
- Respuesta molecular, por parte de Citoquinas, Quimioquinas y el sistema complemento.
- Respuesta inflamatoria, que consiste en la acumulación de fluido y células en el sitio de la infección.

Los mecanismos de tipo adaptativo, en cambio, son el segundo mecanismos de defensa contra cualquier cosa que sea reconocida como extraña y provee protección contra la re-exposición al mismo patógeno. Las características de la inmunidad adaptativa son:

- Especificidad, dado que cada respuesta es específica al antígeno atacante.
- Tolerancia, que permite al sistema inmune diferenciar entre lo que es propio y extraño.
- Memoria, para tener una respuesta más rápida y fuerte en caso de una exposición subsecuente a un mismo antígeno.[6]

A su vez la respuesta inmune adaptativa se divide en dos categorías:

- Celular: es aquella en la que las llamadas Células-T generan otras células especiales llamadas APC (Antigen Presenting Cell por sus siglas en inglés) como los macrófagos y las células dendríticas.
- Humoral: mediada por las Células-B para generar moléculas inmunoglobulinas llamadas “anticuerpos”.[6]

El caso de este proyecto para predicción de afinidad de epítopos a la molécula del complejo mayor de histocompatibilidad clase II entra, específicamente, en el proceso de respuesta inmune adaptativa celular dado de la siguiente forma:

1. Cuando un antígeno no identificado ataca, un tipo de Célula-T llamado Colaboradora o CD4+ envía a una CPA (Célula Presentadora de Antígeno) hacía la zona donde se produce el ataque.
2. Una vez detectado, se comienza con un proceso de endocitosis, es decir, la CPA introduce al antígeno en su medio celular y lo devora para desintegrarlo.
3. Una vez desintegrado, el antígeno arroja ciertos fragmentos de sí mismo denominados péptidos, que son proteínas que contienen información propia del antígeno en cuestión, dichos fragmentos se someten a un proceso llamado fagocitosis, consistente en aislarlos dentro de un fagocito.
4. Este fagocito se fusiona con otro que contiene a las denominadas moléculas MHC-II.
5. Estando ya en el mismo medio se da el proceso de enlace entre los péptidos y las moléculas MHC-II, recordando que no todos los péptidos pueden lograr dicho enlace.
6. Cada vez que se forma un complejo péptido-MHCII es llevado a la superficie de la CPA mediante exocitosis.

7. Por último la CPA regresa hacia la Celula-T colaboradora y se da la presentación del antígeno, donde la Celula-T recibe la información del antígeno atacante para poder desencadenar las reacciones necesarias que den una respuesta satisfactoria para la defensa del organismo.[4]

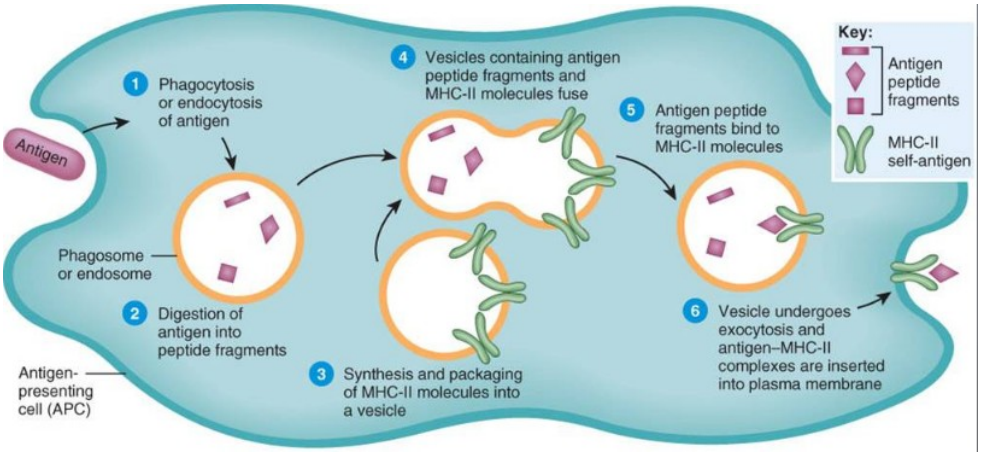

Figura 2: Interacción entre la CPA y el antígeno

Dado que no todos los péptidos se enlazan con la molécula MHC-II, es necesario el desarrollo y constante mejoramiento de herramientas inmuno-informáticas de predicción que puedan ayudar a detectar qué péptidos se enlazarán efectivamente para la identificación de proteínas candidatas a vacunas, la actividad principal realizada en el campo de la vacunología.

2.2 Complejo mayor de histocompatibilidad

El complejo mayor de histocompatibilidad (CMH o MHC, acrónimo para el inglés major histocompatibility complex), o complejo principal de histocompatibilidad, es una familia de genes ubicados en el brazo corto del cromosoma 6 cuyos productos están implicados en la presentación de antígenos a los linfocitos T y en la diferenciación de lo propio y lo ajeno en el sistema inmunitario.[7]

En humanos, los genes MHC conforman el denominado sistema HLA (por human leukocyte antigen), porque estas proteínas se descubrieron como antígenos en los leucocitos, que podían detectarse con anticuerpos.[8] Los genes MHC son fundamentales en la defensa inmunológica del organismo frente a los patógenos, y por otro lado, constituyen también la principal barrera al trasplante de órganos y de células madre.[9]
La región del brazo corto del cromosoma 6 que contiene los genes del MHC posee la información de:

- Glucoproteínas de la membrana plasmática involucradas en los mecanismos de presentación y procesamiento de antígenos a los linfocitos T, agrupándose en los genes de clase II (que codifican las proteínas MHC-II) y los genes de clase I (que codifican las proteínas MHC-I).[10]
- Citocinas y proteínas del sistema del complemento, importantes en la respuesta inmunológica, estos genes se agrupan en la clase III.[10]

Las proteínas mencionadas anteriormente participan en la respuesta inmunitaria, que permite la identificación de las moléculas propias y extrañas (invasoras), para eliminar estas últimas mediante diferentes mecanismos.

2.2.1 Proteína

Las proteínas son biomoléculas formadas por cadenas lineales de aminoácidos. Se dividen en proteínas simples (holoproteidos), proteínas conjugadas (heteroproteidos) y proteínas derivadas. Las proteínas son necesarias para la vida pues constituyen el 80% del protoplasma deshidratado de toda célula y tiene funciones biorreguladoras (enzimas) y defensivas (anticuerpos).[11]

Prácticamente todos los procesos biológicos dependen de la presencia o actividad de éstas moléculas. Bastan algunos ejemplos para vislumbrar la variedad y trascendencia de funciones que desempeñan. Son proteínas casi todas las enzimas, catalizadores de reacciones químicas en organismos vivientes; la

hemoglobina y otras moléculas con funciones de transporte en la sangre; los anticuerpos, encargados de acciones de defensa natural contra infecciones o agentes patógenos; los receptores de las células, a los que se fijan moléculas capaces de desencadenar una respuesta determinada; la actina y la miosina, responsables finales del acortamiento del músculo durante la contracción, etc.[11]

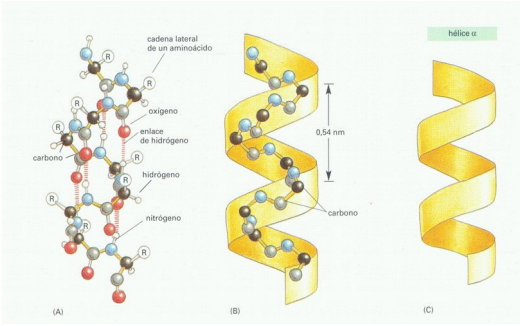

Figura 3: Ejemplo de estructura proteínica

2.2.2 Antígeno

Un antígeno es cualquier sustancia que provoca que el sistema inmunitario produzca anticuerpos contra sí mismo. Esto significa que su sistema inmunitario no reconoce la sustancia, y está tratando de combatirla [12]. La definición moderna abarca todas las sustancias que pueden ser reconocidas por el sistema inmune adaptativo, bien sean propias o ajenas. Los antígenos son usualmente proteínas o polisacáridos. Esto incluye partes de bacterias (cápsula, pared celular, flagelos, fimbrias, y toxinas), de virus y otros microorganismos.[13]

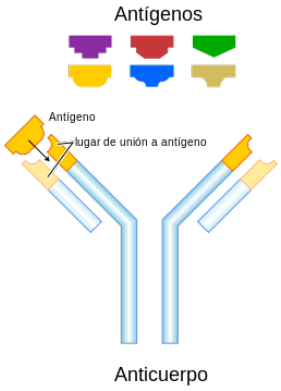

Figura 4: Interacción antígeno - anticuerpo

2.2.3 Anticuerpo

Los anticuerpos (también conocidos como inmunoglobulinas, abreviado Ig) son glicoproteínas del tipo gamma globulina. Pueden encontrarse de forma soluble en la sangre u otros fluidos corporales de los vertebrados, disponiendo de una forma idéntica que actúa como receptor de los linfocitos B y son empleados por el sistema inmunitario para identificar y neutralizar elementos extraños tales como bacterias, virus o parásitos.[14]

2.2.4 Aminoácido

Un aminoácido es una molécula orgánica con un grupo amino ($-NH_2$) y un grupo carboxilo (-COOH). La unión de varios aminoácidos da lugar a cadenas llamadas péptidos o polipéptidos, se denominan proteínas cuando la cadena polipeptídica supera una cierta longitud (entre 50 y 100 residuos aminoácidos) o la masa molecular total supera las 5000 uma y, especialmente, cuando tienen una estructura tridimensional estable definida.[15]

Numero	*Aminoácido*	*Abreviatura*	*Símbolo*
1	Alanina	Ala	A
2	Cisteína	Cys	C
3	Ácido Aspártico	Asp	D
4	0	Glu	E
5	Fenilanina	Phe	F
6	Glicina	Gly	G
7	Histidina	His	H
8	Isoleucina	Ile	I
9	Lisina	Lys	K
10	Leucina	Leu	L
11	Metionina	Met	M
12	Asparagina	Asn	N
13	Pirrolisina	Pyl	O
14	Prolina	Pro	P
15	Glutamina	Gln	Q
16	Arginina	Arg	R
17	Serina	Ser	S
18	Treonina	Thr	T
19	Selenocisteína	Sec	U
20	Valina	Val	V
21	Triptofano	Trp	W
22	Tirosina	Tyr	Y

Tabla 1: Aminoácidos presentes en el proceso de predicción

2.2.5 Linfocito B

Los linfocitos B son los leucocitos de los que depende la inmunidad mediada por anticuerpos con actividad específica de fijación de antígenos. Las células B, que constituyen un 5 a 15% del total de linfocitos, dan origen a las células plasmáticas que producen anticuerpos.[16]

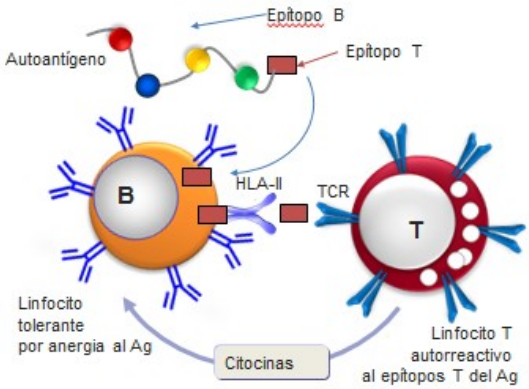

Figura 5: Interacción célula T – célula B

2.2.6 Péptido

Los péptidos (del griego πεπτός, peptós, digerido) son un tipo de moléculas formadas por la unión de varios aminoácidos mediante enlaces peptídicos.[17]

2.2.7 Epítopo

Los anticuerpos, bien sean libres o fijos a la matriz extracelular, se unen a moléculas antigénicas en una superficie de unión, para así formar complejos antígeno-anticuerpo. Esas superficies de unión sobre las macromoléculas constan de complejas secuencias específicas llamadas determinantes antigénicos o epítopos. La mayoría de los epítopos reconocidos por anticuerpos o células B se pueden pensar como relieves de superficies tridimensionales de una molécula antígeno; estos relieves encajan con precisión y así se unen a anticuerpos. La parte de un anticuerpo que reconoce el epítopo se llama parátopo.[18]

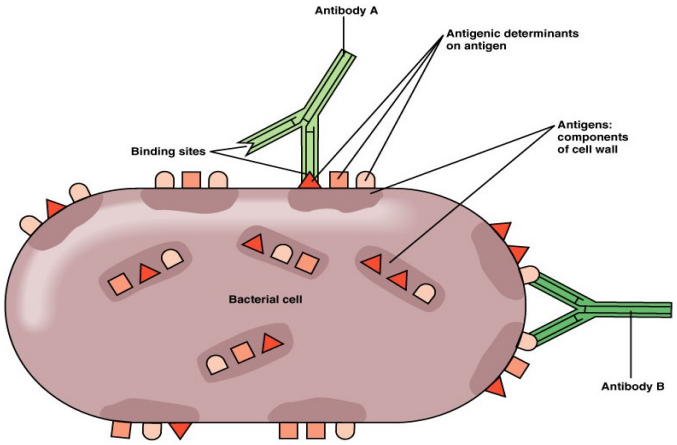

Figura 6: Epítopos localizados en la superficie celular

2.2.8 Alelo

Un alelo o aleloide es cada una de las formas alternativas que puede tener un mismo gen, se diferencian en su secuencia y se puede manifestar en modificaciones concretas de la función de ese gen. (Producen variaciones en características heredadas como, por ejemplo, el color de ojos o el grupo sanguíneo).

Por alelo debe entenderse el valor de dominio que se otorga a un gen cuando rivaliza contra otro gen por la ocupación de posición final en los cromosomas durante la separación que se produce durante la meiosis celular.[19]

2.3 Inmunología Computacional

Inmunología Computacional inició hace más de 90 años con el modelado teórico de la epidemiología de la malaria, haciendo uso de las matemáticas para guiar el estudio de la transmisión de la enfermedad. Desde entonces, el campo se ha ampliado para cubrir todos los aspectos de los procesos del sistema inmunológico y enfermedades.

Actualmente el principal objetivo del campo es convertir los datos inmunológicos en problemas de cálculo, resolver estos problemas usando matemáticas y enfoques computacionales y luego convertir estos resultados en interpretaciones inmunológicamente significativas.[20]

2.3.1 Vacunología inversa

La vacunología inversa es una mejora en el campo de la vacunología , que emplea la bio-informática, fue Rino Rappuoli quien utilizó este enfoque por primera vez contra el serogrupo B del meningococo. Desde entonces, se ha utilizado en varias otras vacunas bacterianas.

La idea básica detrás de la vacunología inversa es que el genoma de un patógeno puede ser examinado mediante enfoques bio-informáticos para encontrar genes. Algunos de los rasgos de dichos genes son controlados para que puedan indicar antigenicidad, se incluyen genes codificados para proteínas con localización extracelular, péptidos señalizados, y epítopos de células B. Una vez localizados, estos genes se filtran en base a atributos deseables para una buena vacuna, tales como proteínas exteriores de la membrana. Finalmente dichas proteínas se someten a pruebas de laboratorio para verificar una respuesta inmune.

Aunque el uso de la tecnología bio-informática para desarrollar vacunas se ha popularizado en los últimos diez años, los laboratorios generales a menudo no tienen el software adecuado para realizar estas actividades. Sin embargo, hay un número creciente de programas bio-informáticos que hacen la aplicación de la vacunología inversa más accesible.[20]

2.4 Arquitectura cliente - servidor

La arquitectura cliente-servidor es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Un cliente realiza peticiones a otro programa, el servidor, quien da respuesta. Esta idea también se puede aplicar a programas que se ejecutan sobre una sola computadora, aunque es más ventajosa en un sistema operativo multiusuario distribuido a través de una red de computadoras.

En esta arquitectura la capacidad de proceso está repartida entre los clientes y los servidores, aunque son más importantes las ventajas de tipo organizativo debidas a la centralización de la gestión de la información y la separación de responsabilidades, lo que facilita y clarifica el diseño del sistema.

La separación entre cliente y servidor es una separación de tipo lógico, donde el servidor no se ejecuta necesariamente sobre una sola máquina ni es necesariamente un sólo programa. Los tipos específicos de servidores incluyen los servidores web, los servidores de archivo, los servidores del correo, etc. Mientras que sus propósitos varían de unos servicios a otros, la arquitectura básica seguirá siendo la misma.

Una disposición muy común son los sistemas multicapa en los que el servidor se descompone en diferentes programas que pueden ser ejecutados por diferentes computadoras aumentando así el grado de distribución del sistema.

La red cliente-servidor es una red de comunicaciones en la cual los clientes están conectados a un servidor, en el que se centralizan los diversos recursos y aplicaciones con que se cuenta; y que los pone a disposición de los clientes cada vez que estos son solicitados. Esto significa que todas las gestiones que se realizan se concentran en el servidor, de manera que en él se disponen los requerimientos provenientes de los clientes que tienen prioridad, los archivos que son de uso público y los que son de uso restringido, los archivos que son de sólo

lectura y los que, por el contrario, pueden ser modificados, etc. Este tipo de red puede utilizarse conjuntamente en caso de que se esté utilizando en una red mixta.[21]

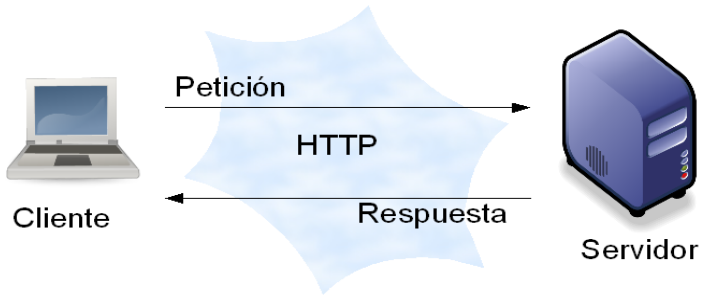

Figura 7: Diagrama de arquitectura cliente – servidor

2.5 Lenguaje de programación C

C es un lenguaje de programación originalmente desarrollado por Dennis M. Ritchie entre 1969 y 1972 en los Laboratorios Bell,2 como evolución del anterior lenguaje B, a su vez basado en BCPL.

Se trata de un lenguaje de tipos de datos estáticos, débilmente tipificado, de medio nivel pero con muchas características de bajo nivel. Dispone de las estructuras típicas de los lenguajes de alto nivel pero, a su vez, dispone de construcciones del lenguaje que permiten un control a muy bajo nivel. Los compiladores suelen ofrecer extensiones al lenguaje que posibilitan mezclar código en ensamblador con código C o acceder directamente a memoria o dispositivos periféricos.

La primera estandarización del lenguaje C fue en ANSI, con el estándar X3.159-1989. El lenguaje que define este estándar fue conocido vulgarmente como ANSI C. Posteriormente, en 1990, fue ratificado como estándar ISO (ISO/IEC 9899:1990). La adopción de este estándar es muy amplia por lo que, si los programas creados lo siguen, el código es portable entre plataformas y/o arquitecturas.[22]

2.6 Colección de compiladores GNU

La colección de compiladores GNU o GCC por sus siglas en inglés (GNU Compiler Collection), es una distribución integrada para varios lenguajes de programación, entre los que se incluyen:

- C
- C++
- Objetive-C
- Objetive-C++
- Java
- Fortran
- Ada, entre otros.

El componente independiente de lenguajes de GCC incluye la mayoría de los optimizadores (conocidos como “back ends”) que generan código máquina para diversos procesadores.

La parte del compilador que es específica para un lenguaje de programación particular se denomina “front end”. Además de los “front ends” integrados a GCC, hay diversos “front ends” separados que permiten soporte para lenguajes como Pascal, Mercury y COBOL.[23]

Estos compiladores se consideran estándar para los sistemas operativos derivados de UNIX, de código abierto y también de propietarios, como Mac OS X. GCC requiere el conjunto de aplicaciones conocido como binutils para realizar tareas como identificar archivos objeto u obtener su tamaño para copiarlos, traducirlos o crear listas, enlazarlos, o quitarles símbolos innecesarios.

Originalmente GCC significaba GNU C Compiler (compilador GNU de C), porque sólo compilaba el lenguaje C, luego cambió debido al soporte de los lenguajes mencionados anteriormente.[24]

2.7 Linux Mint

Linux Mint es una sistema operativo para computadoras diseñado para trabajar en la mayoría de los sistemas modernos, incluyendo PCs típicos x86 y x64. Se puede pensar en Linux Mint como un sistema que cumple con el mismo rol que Windows de Microsoft, Mac OS de Apple y el Sistema Operativo BSD. Linux Mint también está diseñado para trabajar en conjunción con otros sistemas operativos (incluyendo los antes mencionados) y que puede configurar automáticamente un entorno de «arranque dual» o «multi-arranque» (en el que al usuario se le pregunta con cuál sistema operativo desea iniciar en cada arranque) durante su instalación.

Su desarrollo se inició en 2006. Sin embargo, está construido sobre capas de software muy consolidado y probado, incluyendo el kernel Linux, las herramientas GNU y el escritorio Cinnamon, MATE, KDE y Xfce. También está apoyado en los grandes proyectos como Ubuntu y Debian, y utiliza sus sistemas como base.[25]

Linux Mint mantiene un inventario actualizado, un sistema operativo estable para el usuario medio, con un fuerte énfasis en la usabilidad y facilidad de instalación. Es reconocido por ser fácil de usar, especialmente para los usuarios sin experiencia previa en Linux.

Linux Mint se compone de muchos paquetes de software, la mayor parte de los cuales se distribuyen bajo una licencia de software libre. La principal licencia utilizada es la GNU General Public License (GNU GPL) que, junto con la GNU Lesser General Public License (GNU LGPL), declara explícitamente que los usuarios tienen libertad para ejecutar, copiar, distribuir, estudiar, cambiar, desarrollar y mejorar el software. Linux Mint es financiada por su comunidad de usuarios. Los usuarios individuales y empresas que utilizan el sistema operativo pueden actuar como donantes, patrocinadores y socios de la distribución. El apoyo financiero de la comunidad y la publicidad en el sitio web ayuda a mantener Linux Mint libre y abierto.[26]

2.8 GNU Scientific Library

GNU Scientific Library (GSL) es una colección de rutinas diseñadas para cómputo numérico. Estas rutinas han sido escritas en lenguage C, y presentan una moderna interface de aplicaciónes para programación (API por sus siglas en ingles) para programadores C, permitiendo la codificación de subrutinas en lenguajes de muy alto nivel.

Incorpora, entre otras, rutinas para el manejo de números complejos, funciones elementales y funciones especiales, combinatoria, álgebra lineal, integración y derivación numéricas, transformada rápida de Fourier, transformada wavelet discreta, generación de números aleatorios y estadística.[27]

CAPÍTULO III. PREDICCIÓN DE AFINIDAD PÉPTIDO – MHC II

3.1 Antecedentes

Desarrollos en el campo de la inmuno-informática han dejado a su paso una gran cantidad de proyectos y aplicaciones para la predicción de enlace entre epítopos y la molécula del complejo mayor de histocompatibilidad clase II. Una de las personalidades más destacadas dentro de este campo de estudio es Morten Nielsen, afamado físico danés quien es pionero en el diseño de modelos computacionales del sistema inmunológico. A continuación se presentan algunos proyectos previos, incluidos aquellos en los que Nielsen se ha involucrado y que sirven como base para el desarrollo de este proyecto.

Uno de los primeros trabajos relacionados con la predicción de enlace a la molécula MHC-II por parte de Morten Nielsen fue en el año 2004 titulado “Improved prediction of MHC class I and class II epitopes using a novel Gibbs sampling approach”. En éste se habla de un método basado en el muestreo de Gibbs y características específicamente optimizadas para el reconocimiento de patrones de enlace de MHC clase I y clase II, se localizan patrones de enlace en un conjunto de secuencias y se caracterizan en términos de una matriz de pesos. [28]

En 2007 también aportó otro trabajo llamado “Prediction of MHC class II binding affinity using SMM-align, a novel stabilization matrix alignment method”, en el que se describe el uso del método estabilizador de matrices como herramienta para la predicción de patrones de enlace para la molécula MHC-II. Cabe destacar que este método estabilizador de matrices es uno de los puntos fundamentales en el funcionamiento de la herramienta de predicción descrita en la presente tesis. [29]

Otros proyectos referentes a MHC-II por parte de Nielsen son NetMHCIIpan-2.0 (2010) y su mejora NetMHCIIpan-3.0 (2013), método pan específico basado en redes neuronales que funciona sin la necesidad de ningún tipo de rep-alineamiento para los datos de entrada.[30][31]

Finalmente algunos otros métodos encontrados en el estado del arte referente a predicción de enlace a la molécula MHC-II incluyen optimización por enjambre de partículas, uso de múltiples instancias regresivas, secuencias de pesos, entre otros.[32][33][34]

Después de una exhaustiva búsqueda de todos aquellos proyectos disponibles hasta el día de hoy, se encontró que ninguno de los actualmente propuestos incluye el uso de la tecnología GPGPU bajo el paradigma de la programación paralela, lo que supone una novedosa característica incluida en la nueva herramienta de predicción propuesta en esta tesis en comparación con todo lo anteriormente descrito.

3.2 Base de datos IEDB

La IEDB (Immune Epitope Database por sus siglas en ingles), es una colección de datos relacionados a anticuerpos y epitopes de Celulas-T para humanos, primates no-humanos, roedores y otras especies animales. Todos los datos de epítopos peptídicas y no-peptídicas relacionados a padecimientos infecciosos, alergias, padecimientos auto-inmunes y trasplantes se encuentran en constante actualización.

Esta base de datos también contiene datos relacionados a enlace con MHC de una variedad de fuentes antigénicas e inmuno-epitopes provenientes de otras bases de datos tales como FIMM, HLA Ligand, TopBank, y MHC Binding.[35]

Categoría	*Cantidad*
Epítopos peptídicas	122661
Epítopos no péptídicas	2091
Ensayos de célula T	255056
Ensayos de célula B	173927
Ensayos de ligando MHC	305740
Organismos fuente de epítopo	3304
Restricción de alelos MHC	692
Referencias	16384

Tabla 2: Resumen de datos disponibles en IEDB

Lo anterior pone a IEDB como una de las bases de datos más utilizadas en el campo de la inmuno-informática, específicamente en aplicaciones de predicción tales como NetMHC-II o BediPred.

3.3 Servidor NetMHC-II

El servidor NetMHC-II predice la afinidad de enlace de péptidos a la molécula MHC-II de los alelos HLA-DR, HLA-DQ, HLA-DP y de ratón. Los métodos utilizados para la predicción son SMM (Stabilized Matrix Method) y Gibbs Sample. Este servidor usa el enfoque de cualitativo de afinidad, es decir, no sólo predice si un péptido es enlazador con la molécula MHC-II, sino también su grado de afinidad para determinar si será un enlazador fuerte (strong binder) o enlazador débil (weak binder). [36]

A continuación se muestra un ejemplo de los resultados arrojados por el servidor, al ser ejecutado en su versión local:

NetMHCII version 2.2.

Strong binder threshold 50.00. Weak binder threshold 500.00.

Allele	pos	peptide	core	1-log50k(aff)	affinity(nM)	MeasuredAff	Bind Level	%Random	Identity
HLA-DRB10101	0	AANWILRGTSFVYVP	NWILRGTSF	0.719	20.9	0.188	SB	16.00	PEPLIST
HLA-DRB10101	1	AGDLGRDELMELASD	GDLGRDELM	0.237	3848.3	0.030		50.00	PEPLIST
HLA-DRB10101	2	ALLIIPPKIHISIEL	LLIIPPKIH	0.574	100.6	0.787	WB	32.00	PEPLIST
HLA-DRB10101	3	ASRELERFALNPGLL	ERFALNPGL	0.575	99.5	0.571	WB	32.00	PEPLIST
HLA-DRB10101	4	AYAAQGYKVLVLNPSVAA	YKVLVLNPS	0.850	5.0	0.623	SB	0.90	PEPLIST
HLA-DRB10101	5	CHFITKETPDRLTDQ	ITKETPDRL	0.408	605.4	0.760		50.00	PEPLIST
HLA-DRB10101	6	CQFLKVEKSQLLNEF	LKVEKSQLL	0.822	6.9	0.570	SB	4.00	PEPLIST
HLA-DRB10101	7	DCIMTSYQYLIIQNT	YQYLIIQNT	0.718	21.2	0.808	SB	16.00	PEPLIST
HLA-DRB10101	8	DIVIYSKYGGTEIKY	YSKYGGTEI	0.516	188.9	0.074	WB	50.00	PEPLIST
HLA-DRB10101	9	DSKHQLDMIITAVNS	DMIITAVNS	0.553	125.9	0.438	WB	50.00	PEPLIST
HLA-DRB10101	10	EDPYWGNGDRHSDYQPLGTQDQSLY	YQPLGTQDQ	0.777	11.1	0.149	SB	8.00	PEPLIST
HLA-DRB10101	11	EHDLERGPPGPRRPP	HDLERGPPG	0.259	3021.6	0.149		50.00	PEPLIST
HLA-DRB10101	12	EIPSFRWTQSLRRGL	FRWTQSLRR	0.761	13.2	0.458	SB	8.00	PEPLIST
HLA-DRB10101	13	ELGEWVFSAIKSPQA	FSAIKSPQA	0.686	29.7	0.168	SB	16.00	PEPLIST
HLA-DRB10101	14	ERFALNPSLLETTEGCQQI	FALNPSLLE	0.563	113.7	0.374	WB	50.00	PEPLIST
HLA-DRB10101	15	EYLILSARDVLAVVS	YLILSARDV	0.876	3.8	0.138	SB	0.05	PEPLIST
HLA-DRB10101	16	FIHFFTWGTMFVPKY	FIHFFTWGT	0.684	30.5	0.449	SB	16.00	PEPLIST
HLA-DRB10101	17	FMRMAWGGSYIALDS	FMRMAWGGS	0.704	24.7	0.147	SB	16.00	PEPLIST
HLA-DRB10101	18	FSLECIMDVGEIQNK	FSLECIMDV	0.430	474.9	0.489	WB	50.00	PEPLIST
HLA-DRB10101	19	GFVGLCRTLGSKCVR	CRTLGSKCV	0.742	16.3	0.320	SB	16.00	PEPLIST

Allele: HLA-DRB10101. Number of high binders 11. Number of weak binders 6. Number of peptides 20

Tabla 3: Resultados de predicción arrojados por NetMHCII

Para efectos de uso en el laboratorio, los datos más importantes de la tabla son el core o núcleo de enlace del péptido, el valor de afinidad del péptido y el bind level o nivel de enlace que determina si el péptido sera un enlazador fuerte o débil.

CAPÍTULO IV. CÓMPUTO DE PROPÓSITO GENERAL EN UNIDADES DE PROCESAMIENTO GRÁFICO

El cómputo paralelo, en ocasiones llamado paralelismo, es una forma de cómputo en la que muchas instrucciones se ejecutan al mismo tiempo, operando bajo el principio de que un problema grande puede dividirse en sub-problemas más pequeños, que al ser resueltos llevan a la solución del problema completo. Este enfoque de paralelismo se ha aplicado específicamente en la programación de aplicaciones, generando un nuevo paradigma de programación conocido como programación paralela.[37]

La programación paralela es, como su nombre lo indica, programar de forma tal que muchas instrucciones puedan ser ejecutadas al mismo tiempo y se diferencia del paradigma tradicional de programación debido a que éste es de carácter secuencial, es decir, cada una de las instrucciones se programan de forma tal que se ejecutan una a una sucesivamente. Actualmente no hay un común acuerdo acerca de qué enfoque de programación es mejor, se piensa más bien que cada uno es adecuado para situaciones o requerimientos específicos de programación, de modo que por ejemplo, la programación secuencial aplica para algunos casos y la programación paralela aplica para otros.

En un principio la programación paralela, así como su contra-parte secuencial, se aplicaban sobre la CPU de la computadora. Herramientas como MPI o PVM permitían al usuario programar aplicaciones que trabajaran sobre los hilos de la CPU para generar un comportamiento de paralelismo. Después se descubrió que podían utilizarse las capacidades de las tarjetas gráficas (GPU por sus siglas en inglés) para trabajar sobre programas paralelos. La empresa NVIDIA, gigante del mercado de tarjetas gráficas, fue uno de los pioneros en el campo del cómputo paralelo sobre GPUs, quien posteriormente creó las llamadas GPGPU (General Purpose Graphic Proccesor Unit), tarjetas gráficas hechas específicamente para programar aplicaciones paralelas.[38]

La arquitectura CUDA para programación paralela se ilustra a continuación:

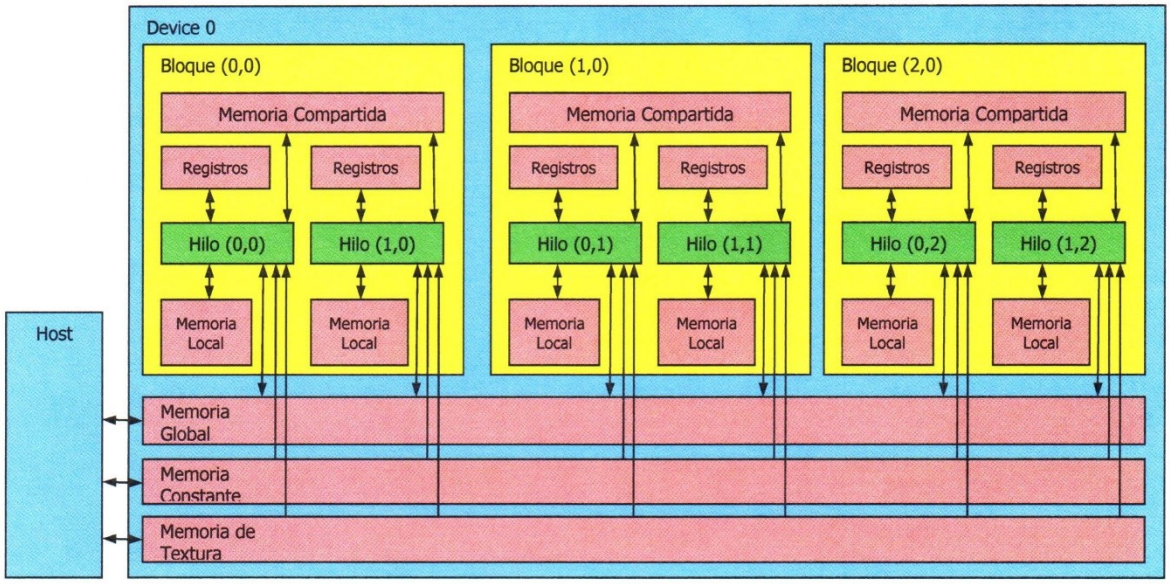

Figura 8: Disposición lógica de espacios de memoria en tarjetas NVIDIA para CUDA

4.1 Paralelismo de los datos

Muchas aplicaciones de software que procesan enormes cúmulos de datos y por lo tanto se ejecutan durante largos periodos de tiempo, son diseñados para modelar fenómenos del mundo real. Imágenes y video son retratos de un mundo físico donde diferentes partes de una imagen capturan de forma simultanea eventos físicos independientes. La física de cuerpos rígidos, dinámica de fluidos y movimientos pueden evaluarse de forma independiente sobre pequeños pasos de tiempo. Entonces, el paralelismo de los datos se refiere a la cualidad de un programa para realizar, de forma exitosa, muchas operaciones aritméticas sobre estructura de datos y de forma simultanea.

Un buen ejemplo acerca de paralelismo en los datos se presenta a continuación, donde una matriz P es el resultado del producto punto entre las filas de una matriz M y las columnas de una matriz N. Se ve que cada una de las operaciones ejecutadas entre los elementos correspondientes a las filas de M y las columnas de N pueden realizarse de forma simultanea e independiente entre sí. Entonces,

para matrices muy grandes la cantidad de multiplicaciones punto es enorme, para una matriz de 1,000 por 1,000 serían por ejemplo 1,000,000 de operaciones, cada una de ellas involucra 1,000 multiplicaciones y 1,000 operaciones aritméticas acumuladas.

Son situaciones como ésta en las que se hace evidente que la utilización de dispositivos GPU, como NVIDIA por ejemplo, auxiliados de herramientas de programación paralela como CUDA C, pueden acelerar la ejecución de dichas operaciones en comparación con su contra-parte tradicional sobre una CPU.[39]

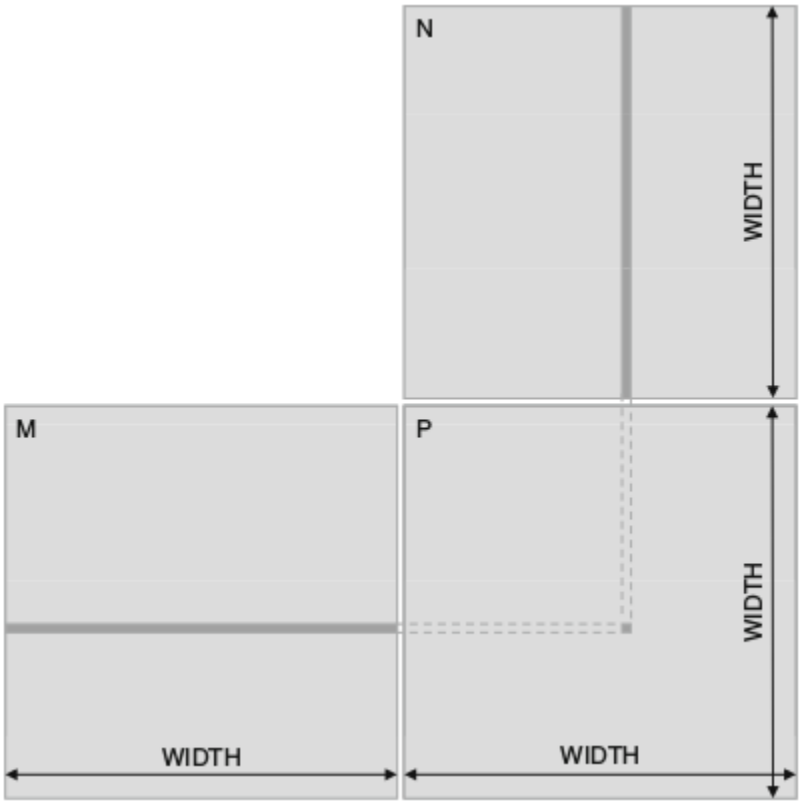

Figura 9: Representación de matrices en CUDA

4.2 Lenguajes de programación para GPGPU

Existen muchos entornos de desarrollo que posibilitan la aplicación del paradigma de programación paralela usando GPGPU como recurso de procesamiento. Al principio, uno de los primeros acercamientos fue gracias al lenguaje ensamblador, después surgirían diversos lenguajes como lo son GLSL, Cg y HLSL, que si bien no usaban la GPU desde la perspectiva de operaciones aritméticas, si lo hacían para el desarrollo de aplicaciones gráficas.

Posteriormente se descubriría el gran potencial de procesamiento de las GPUs y así nacerían las GPGPUs mencionadas anteriormente, para las cuales se han desarrollado otros lenguajes de programación especializados como Brook GPU que es una extensión de ANSI C, Sh como extensión d C++, OpenCL que puede usarse sobre diversos dispositivos de procesamiento como CPU multi-núcleo o GPU y por último CUDA, que trabaja exclusivamente sobre dispositivos de la marca NVIDIA y actualmente se posiciona como una de las herramientas de desarrollo de aplicaciones paralelas más extendidas del mercado.[40]

Para el presente proyecto, específicamente para la sección de desarrollo paralelo de la aplicación se eligió CUDA C como herramienta principal, debido a su gran alcance como lenguaje predilecto, lo que permite acceso a un sinfín de material de apoyo, tanto bibliográfico como por parte de foros especializados en la materia, además CUDA C, como su nombre lo indica, viene siendo una especie de añadido o extensión del lenguaje de programación C, que viene siendo el más utilizado en el mundo, lenguaje base para la aparición de otros, con mucha funcionalidad, una enorme influencia y aplicación aun en la actualidad.

4.3 Estructura de un programa en CUDA

Un programa en CUDA consiste en una o más fases que son ejecutadas ya sea por el CPU o un dispositivo, una GPU por ejemplo. Aquellas fases que exhiban poco o nada de paralelismo son ejecutadas en el llamado “host code” o código anfitrión, mientras que las fases que si presentan una gran cantidad de paralelismo en sus datos son ejecutadas por el “device code” o código de dispositivo. Un programa desarrollado en CUDA es entonces un código fuente unificado que contiene tanto “host code” como “device code”. Entonces el compilador de NVIDIA C, conocido como nvcc, separa ambas partes durante la etapa de compilación de código. El código anfitrión es ANSI C, por lo que es compilado por el compilador tradicional para lenguaje C y ejecutado como un proceso CPU ordinario. En cambio, el código de dispositivo es escrito en lenguaje ANSI C pero extendido con palabras clave para aquellos métodos específicos para datos paralelos, estas etiquetas se denominan “kernels”, compilado en nvcc y ejecutado en la GPU.

Estas funciones kernels generan una gran cantidad de hilos, que son instancias de ejecución, para explotar el paralelismo en los datos. En el ejemplo anterior de multiplicación punto de matrices, se puede implementar un kernel que compute todas las operaciones matriciales utilizando estos hilos. Siendo una operación con matrices de 1,000 por 1,000 se pueden lanzar 1,000,000 de hilos, donde cada uno de ellos es capas de computar la operación correspondiente a un sólo elemento de la matrix P. Es importante notar que los hilos de CUDA son mucho más ligeros que los hilos de la CPU, por lo tanto los programadores CUDA pueden asumir que la generación y organización de estos hilos tomará muchos menos ciclos gracias a la eficiencia del hardware GPU aplicado. Lo anterior contrasta con los hilos de la CPU que generalmente toman miles de ciclos de reloj para generarse y organizarse.[41]

Entonces, la ejecución de un típico programa en CUDA puede ilustrarse en la siguiente imagen donde el proceso comienza con la ejecución del anfitrión (CPU), después se lanza el kernel que mueve la ejecución al dispositivo (GPU), donde se generan una gran cantidad de hilos que toman ventaja del gran paralelismo de los datos. Todos los hilos generados durante la ejecución de un kernel se denominan como red o “grid”.

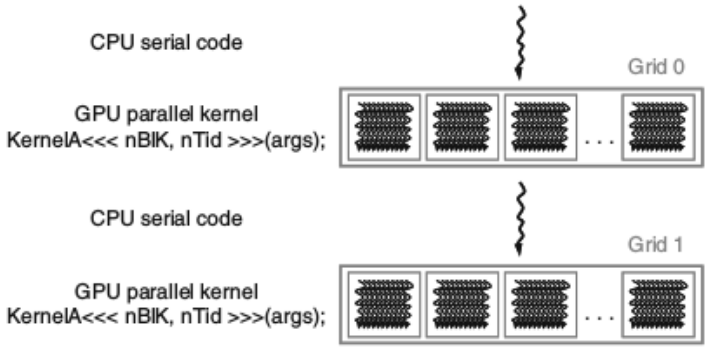

Figura 10: Ejemplo de grids aplicados a un programa en CUDA

CAPÍTULO V. MÉTODO ESTABILIZADOR DE MATRICES CON DESCENSO MÁXIMO

5.1 Principios del método

Cada péptido sometido a pruebas de afinidad de enlace con la molécula MHC-II puede ser codificado como un vector binario con ceros en todas sus posiciones, excepto en aquellas que codifican cada uno de sus aminoácidos específicos.

Entonces, por ejemplo, un péptido de longitud 20 puede ser codificado como un vector binario de N posiciones que se denota como $N*20$ bajo el contexto del método estabilizador de matrices. Para un conjunto de secuencias (péptidos), cada vector x^t puede ser apilado para construir una matriz H resultante. La matriz H es multiplicada por un vector w que asigna pesos a cada residuo posible en cada posición de la secuencia, esto genera un vector de predicción y_{pred} .

$$H*w=y_{pred} \tag{1}$$

De un conjunto de entrenamiento de secuencias codificado en H con valores medidos previamente y_{meas} , se pueden encontrar los valores correctos del vector w minimizando la diferencia entre los valores predichos y_{pred} y los valores medidos y_{meas} acorde a la norma L_2 . Para suprimir el efecto de ruido en los datos experimentales, se añade un segundo término a la minimización:

$$\| (H*w)- y_{meas}\| + w^t*\Lambda*w \rightarrow mínimo \tag{2}$$

En el cual Λ es un escalar o una matriz diagonal. El efecto del segundo término añadido es disminuir los valores de w que no contribuyen significativamente a minimizar la norma. Esta técnica llamada regularización, suprime el efecto de ruido. Siendo utilizada la norma L_2 o la suma del error cuadrado entonces la ecuación anterior puede resolverse analíticamente para cualquier valor dado de Λ como sigue:

$$w=\left(H^t*H+\wedge\right)^{-1}*H^t*y_{meas} \tag{3}$$

5.2 Validación cruz

El valor óptimo para Λ puede determinarse por validación cruz, los datos experimentales correspondientes a las filas de (H, y_{meas}) son separados aleatoriamente en conjuntos de entrenamiento $T_i=(H, y_{meas})_{train\,i}$ y conjuntos de prueba $(H, y_{meas})_{blind\,i}$. Para una Λ dada, la ecuación (3) es utilizada para calcular una w para cada conjunto de entrenamiento T_i . Entonces estás w_i pueden ser utilizadas para realizar predicciones para sus correspondientes conjuntos ciegos. Sumando las distancias de todos los conjuntos ciegos se obtiene la distancia y_{pred} validada por cruz:

$$\Phi(\Lambda)=\sum \| H_{blind\,i}*w_i(\Lambda)-y_{blind\,i}\| \tag{4}$$

Minimizando Φ como función de Λ corresponde a minimizar la distancia validada por cruz “aminorando” la influencia de los coeficientes en w susceptibles a ruido.

Dado que tanto el valor óptimo resultante para Λ y la correspondiente w_i son de algún modo influenciados por la forma cómo se dividen los conjuntos de entrenamiento y los conjuntos ciegos, el mismo procedimiento se repite varias veces con diferentes particiones, dicho procedimiento se denomina “bagging”.

La w final se obtiene promediando todas las w_i óptimas generadas en la validación cruz y el bagging repetidos.

5.3 Descenso de gradiente

Una parte fundamental dentro del proceso para estabilizar la matriz H es hallar una Λ óptima qué permita encontrar el vector de pesos w. La Λ será un valor escalar añadido a la diagonal de la matriz H. El descenso de gradiente como tal se aplica a un conjunto de valores, originalmente se pensó de esta forma para obtener varios valores que serían aplicados a la diagonal principal de H, después se determinó que para agilizar los tiempos de procesamiento de la aplicación era necesario cambiar un poco la perspectiva, por lo que se optó por manejar la derivada en lugar del gradiente, para un solo valor de Λ.

El algoritmo de descenso máximo se estructura de la siguiente manera:

1. Se tiene un número máximo de iteraciones para obtener el descenso máximo del valor Λ y un valor E .muy pequeño que sirve como paso para incrementar o decrementar el valor de Λ dependiendo de hacia donde tienda el mínimo al que se pretende acercar.
2. En cada iteración se obtiene la derivada que comprende primeramente obtener el tamaño del paso que dará E , después se calcula un valor "suma" de $x+\mathrm{E}$ y un valor "resta" $x-\mathrm{E}$ donde x es el valor actual al cual se le aplicara el paso de ascenso o descenso, por último se obtiene la resta entre la función de evaluación para $x+\mathrm{E}$ y $x-\mathrm{E}$.
3. A su vez, cada función de evaluación utiliza la ecuación (3) mostrada anteriormente para obtener un y_{pred} al que se le resta los valores de y_{meas} , después se obtiene la norma L_2 de dicha resta.
4. La norma obtenida en el paso tres se devuelve tanto para $x+\mathrm{E}$ como para $x-\mathrm{E}$, ambas normas son restadas para obtener el numerador del gradiente el cual se divide por el denominador del gradiente que será $2*\mathrm{E}$, dicha división corresponde a la derivada.
5. Teniendo la derivada se puede determinar la dirección de descenso y se procede a calcular el backtracking que es un criterio utilizado para reajustar el tamaño del paso E para el avance de la siguiente x . El backtracking primero multiplica el valor de descenso por el tamaño del paso alfa, este

valor es sumado a la x actual, este valor de $x+(pk*\alpha)$ se le suma una constante C de 0.5. Después se compara la norma de x siguiente contra la x actual, mientras sea mayor se reduce el tamaño de α a la mitad, en caso contrario de que alfa necesite ser más grande entonces se duplica su valor.

6. Finalmente se obtiene una x tendiendo al valor mínimo, gracias a la dirección de descenso. Como se mencionó anteriormente, esta x servirá como nuestra Λ inicial para la ejecución del paso de bagging.[42]

CAPÍTULO VI. CODIFICACIÓN SECUENCIAL E IMPLEMENTACIÓN PARALELA

6.1 Estructura de la aplicación

Una vez analizados el estado del arte y diversas fuentes bibliográficas se procedió a codificar la aplicación. De primer momento y dada la naturaleza del proceso de *bagging,* se determinó la necesidad de programar todas las funciones respecto a operaciones con matrices y vectores, dichas operaciones son:

- Traspuesta de una matriz
- Producto entre matrices
- Suma de vectores
- Resta de vectores
- Producto entre vector y escalar
- Producto punto entre dos vectores
- Producto entre matriz y vector

Después se implementaron los métodos para extraer los datos directamente del archivo original que contiene los péptidos que serán utilizados posteriormente para el entrenamiento y las prueba. Una vez obtenidos los datos es necesario transformarlos en una matriz H para proceder a su tratamiento utilizando el método de bagging, se obtienen las dimensiones de H, se determina el alfabeto de aminoácidos presentes en el conjunto de datos y después se construye la matriz encendiendo en uno sólo aquellas posiciones en H que corresponden a los aminoácidos presentes en cada uno de los péptidos, es importante recordar que la construcción de la matriz H se basa específicamente en el criterio de transformar cada péptido del conjunto de datos en un vector que cubra todas las posiciones posibles para cada aminoácido presente, finalmente cada vector péptido se apila para construir la matriz H que se utilizará constantemente en la aplicación.

Posteriormente, al tener las matrices H y H traspuesta se programó el método de bagging, que depende en parte del azar, por lo que se creó una librería llamada “alea.h” que contiene algunos métodos para generación de aleatorios pero principalmente un método de barajeo o “shuffle” que se utiliza al momento de entrenar los datos dentro del bagging.

En el SMM o método estabilizador de matrices, la formula para obtener el vector w de pesos conlleva realizar ciertas operaciones entre matrices, de las que la inversa es la más laboriosa, de primer momento y dado que H jamás será una matriz cuadrada se optó por la aplicación de la pseudo-inversa, específicamente por el método de Penrose. Posteriormente se decidió utilizar las herramientas provistas por las cabeceras de la GSL (GNU Scientific Library), las que se usaron fueron:

- gsl_matrix.h: librería para operaciones matriciales y gestión de memoria.
- gsl_linalg.h: librería para operaciones con álgebra lineal, como por ejemplo la descomposición LU y su pseudo-inversa.
- gsl_cblas.h: librería para operaciones de permutación.

A partir de aquí se explicará más detalladamente cada parte de la aplicación:

6.1.1 Origen de los datos

Como se comento en un apartado anterior, los datos utilizados tanto para el entrenamiento como para las pruebas se obtiene de una de las fuentes de datos más reconocidas en el campo de la inmuno-informática. Nos referimos a la IEDB (Immune Epitope Database). Nuestro primer conjunto de datos proviene del archivo “H-2-Iab_test_random.txt” que contiene péptidos tipo alelos H-2-Iab 15 de ratón.

6.1.2 Pre-procesamiento

El primer detalle notorio al revisar el archivo de origen de los datos fue que muchos de los péptidos listados se repetían presentando diferentes valores de medición para su afinidad con la molécula MHC-II. Entonces se procedió a identificar qué péptidos presentaban este comportamiento y dado que es necesario que cada uno tenga sólo un valor de medición relacionado para efecto de ejecutar el método de predicción, se procedió a promediar todos los valores ligados al péptido para obtener un único valor promedio qué tome en cuenta las variantes presentadas en el archivo originalmente.

Una vez realizado lo anterior se obtuvo un nuevo archivo conteniendo un total de 97 péptidos sin incluir alguno más de una vez. A este nuevo archivo se le dio el nombre de “TrainInput2.txt” y es el que se utiliza tanto para los datos de entrenamiento como los de prueba.

El siguiente aspecto a considerar es el “alfabeto”, cada péptido está formado a su vez de aminoácidos, existen entonces los llamados 20 aminoácidos esenciales, que son representados mediante una letra que no se repite. Por lo anterior es importante para cada conjunto de datos particular, determinar el alfabeto de aminoácidos presentes para ese caso específico, puesto que el criterio utilizado para construir la matriz H a partir de los datos de entrada es que debe determinarse qué aminoácidos de los 20 esenciales están realmente presentes, es decir, se considera cada posible variación de un aminoácido para cada posición en el arreglo de un peṕtido.

También es importante determinar en base a los datos de entrada las dimensiones que serán consideradas posteriormente para la construcción de la matriz H, por lo que se hace un conteo del total de péptidos obtenidos en el pre-procesamiento, de ese conteo se desprende la dimensión 1 o Altura que tendrá la matriz H, después se calcula la dimensión 2 o Anchura de la matriz, definido mediante la formula: (Dimensión del vector péptido * Dimensión del vector alfabeto) + 1.

Cabe mencionar que el “1” sumado al final del producto corresponde al valor offset añadido al inicio de lo que será cada uno de los vectores que conformarán posteriormente a la matriz H.

6.1.3 Construcción de la matriz H

Una vez concluido el pre-procesamiento de los datos de entrada, ya contamos con la información necesaria para construir la matriz H que será una representación binaria de cada uno de los péptidos correspondientes a los datos de entrada. Ya se conocen las dimensiones de altura y anchura de la matriz así como el alfabeto completo de aminoácidos presentes en cada péptido.

La matriz H se forma mediante una función que transforma cada péptido en un vector fila, cada vector se representa mediante valores binarios donde se ponen en “1” las posicións correspondientes al aminoácido presente entre todo el alfabeto disponible para esa muestra. En el caso del método como tal, primero s genera la matriz H “vacía” es decir exclusivamente en ceros, luego se van “encendiendo” cada uno de los aminoácidos del vector según corresponda, teniendo en cuenta que el primer elemento de cada fila siempre estará en 1 debido a la columna offset propia del método de bagging.

Después de tener H, se procede a generar su traspuesta pues es necesaria para las operaciones de predicción que serán ejecutadas más adelante. También debe considerarse el hecho de que la matriz H abarca a todo el conjunto de datos, por lo que es necesario generar una matriz más pequeña que hemos denominado Htrain, que sólo contiene la cantidad de datos necesaria para el entrenamiento del programa y por supuesto su correspondiente traspuesta.

6.1.4 Descenso máximo

El descenso máximo es un método del álgebra lineal que en nuestro caso nos permite obtener una lambda óptima, dicha lambda será utilizada posteriormente durante el proceso de bagging.

El método necesita de una serie de parámetros iniciales para ejecutarse, que son:

- MAXITER: el número máximo total de iteraciones que el método puede ejecutarse, independientemente de si converge al punto óptimo deseado.
- X0: el punto inicial al que le asignamos el valor 1.0, a partir de este punto avanzaremos hacia adelante o hacia atrás durante cada vuelta del descenso máximo, al final este punto debe ser nuestra lambda óptima.
- Htrain: la matriz H de entrenamiento.
- Htrain traspuesta: se utiliza junto a Htrain para obtener el vector de pesos w en un apartado específico del método.
- Input: el arreglo de péptidos que necesitamos para realizar la comparación entre los valores obtenidos previamente in vitrio contra las que se generen dentro de la aplicación, es decir, in silico.

El método de descenso máximo consiste un punto óptimo o lambda ideal, para ello se toma un punto inicial y se escoge una dirección de descenso hacia atrás y hacia adelante basado en un paso alfa dado a nuestro punto inicial. Se realiza una prueba de convergencia con un valor epsilón que sería el límite, de modo que si no converge entonces se ajusta el paso alfa hasta que se cumpla la condición de que el nuevo punto x sea menor que el anterior. Una vez que el método converga o cumpla con el número máximo de iteraiones, se obtiene una lambda ideal que se usará posteriormente.

6.1.5 Método de bagging

Una vez obtenida la lambda óptima se procede a ejecutar el método de bagging que como se mencionó anteriormente transforma todos los péptidos de entrada en una matriz H a la cual se le aplican una serie de operaciones de álgebra lineal, la lambda se usa para añadirla como valor a la diagonal de dicha matriz H para entonces partirse en subconjuntos de entrenamiento y subconjuntos de prueba.

Todos los subconjuntos se generan de forma aleatoria, los de entrenamiento se someten a una función $w=(H^t*H+\wedge)^{-1}*H^t*y_{meas}$, dicha función tiene por objeto obtener un vector w_i específico para ese caso en concreto, al final todos esos vectores w_i se promedian para obtener un vector w general, la idea es que este vector w general sea o contemple muchos casos diferentes de la matriz H original de manera que pueda ser utilizado para obtener predicciones confiables sobre los datos de prueba. Al final se obtiene entonces el vector w para nuestra matriz H, que al multiplicarlos nos dará como resultado un vector de predicciones y_{pred} que deberá ser comparado con el vector de valores y_{med} medidos in vitro para esos mismos péptidos.

Se debe puntualizar que el método de bagging con descenso máximo se describe más detalladamente en el capítulo cinco.

6.1.6 Bibliotecas

Para el funcionamiento de la aplicación se tienen tres librerías que tuvieron que ser programadas por separado:

- jo.h: contiene todas las funciones utilizadas en el programa princial, todo lo referente a operaciones matriciales, vectoriales, algebra lineal y los metodos de descenso máximo y bagging.
- alea.h: contiene métodos para generación de números aleatorios bajo diferentes criterios, además de los métodos necesarios para generar aleatoriamente semillas y el shuffle usado en el bagging. A su vez contiene la referencia a otra librería llamada “mt.h”.
- mt.h: contiene los métodos para generación de aleatorios usando el algoritmo mt19937

6.2 Proceso de paralelización

Dada la naturaleza de la aplicación, que basa su funcionamiento en operaciones entre matrices y vectores, se puede anticipar que existen muchas secciones que pueden ser sometidas al proceso de paralelización. Específicamente todo lo concerniente al proceso de entrenamiento y prueba en donde se utilizan tanto el descenso máximo como el bagging.

La operaciones que fueron paralelizadas dentro del método de predicción, el cual se basa en la formula $w=\left(H^{t}*H+\wedge\right)^{-1}*H^{t}*y_{meas}$ son los siguientes:

- Método de SeudoInversa (Matrix A, Matrix B)
- Método MatMul (Matrix A, Matrix B, Matrix D)
- Método Inversa (Matrix A, Matrix I)
- Método GaussJordanDiv(Matrix A, Matrix I)

- Método GaussJordanReng (Matrix A, Matrix I, float pivote, int k)

6.3 Acciones dentro de la GPU

Como se mencionó con anterioridad, el proceso de paralelización afecta directamente todas las operaciones entre matrices y vectores que se encuentran dentro del proceso de predicción, basado en el método de la pseudoinversa. A continuación se describen dichas operaciones, que son realizadas por parte de la GPU.

La pseudoinversa se aplica a la matriz cuadrada $H_t H$ y las operaciones involucradas serían las siguientes:

- Se define la matriz traspuesta $H_t H$ que sería $(H_t H)_t$
- Se obtiene el producto de $H_t H$ por $(H_t H)_t$, dentro de la GPU se gestiona memoria para ambas matrices mediante la operación cudaMalloc, recordando que los tamaños en la GPU se manejan linealizando cada matriz de la siguiente manera:

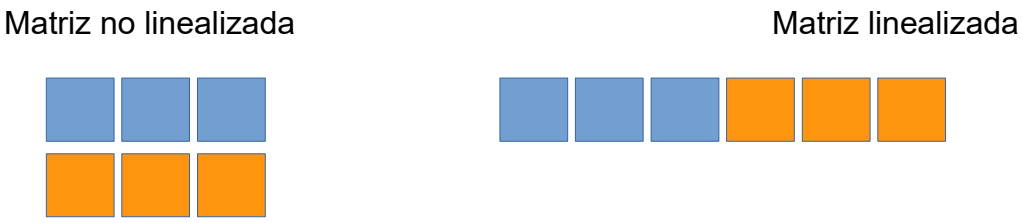

Figura 11: Linealización de una matriz

- Tomando en cuenta el criterio anterior y una vez reservado el espacio en memoria dentro de la GPU para las matrices que serán multiplicadas, la operación como tal sigue la misma lógica tradicional de multiplicar cada elemento de cada fila de la matriz $H_t H$ de forma consecutiva por cada elemento de cada columna de la matriz $(H_t H)_t$, la única diferencia es el enfoque para tomar cada elemento a multiplicar, respetando la vectorización de ambas matrices. Se debe determinar el tamaño del bloque dentro de la GPU, en este caso el bloque tiene un tamaño estándar de 1, pues es espacio suficiente para cubrir las dimensiones de cada matriz. Se debe determinar también el tamaño del grid o malla que serían la cantidad de hilos dentro del bloque que estarán disponibles para realizar cada operación. La característica principal de trabajar en paralelo es, como se ha mencionado anteriormente, el uso de estos hilos para realizar de forma concurrente tareas individuales, en este caso cada hilo se encargará de multiplicar un elemento de la matriz $H_t H$ por un elemento de la matriz $(H_t H)_t$.

- Se obtiene la inversa de $H_t H * (H_t H)_t$, en esta sección se hace uso de la GPU específicamente en el kernel del método de Gauss-Jordan, tanto al pivotear cada renglón de la matriz $H_t H * (H_t H)_t$ y de su matriz identidad, como para la división entre la matriz identidad y la matriz $H_t H * (H_t H)_t$.

CAPÍTULO VII. RESULTADOS Y CONCLUSIONES

7.1 Resultados

Después de varias modificaciones y ajustes al programa, se procedió a realizar las pruebas finales utilizando tres conjuntos diferentes de datos pre-procesados, todos extraídos desde la base de datos IEDB que ya ha sido mencionada anteriormente en este trabajo. Los tres conjuntos son:

- TrainInput2 con 97 muestras.
- TrainInput3 con 103 muestras.
- TrainInput4 con 99 muestras.

La totalidad de los péptidos probados pertenecen a la categoría de alelos H-2-Iab 15 de ratón, con una longitud fija de 15 aminoácidos cubriendo el total de 20 aminoácidos esenciales.

Básicamente el programa se divide en dos partes con distintos enfoques, primero tenemos el método de álgebra lineal denominado Descenso Máximo, que transforma los datos en una matriz que se intenta estabilizar, se busca un valor lambda ideal y un resultado residual de este proceso es la obtención del vector de pesos w que puede utilizarse para predecir los valores reservado de prueba. El segundo método es el Bagging que sí es estocástico, transforma los péptidos a predecir en una matriz, entonces la divide aleatoriamente en subconjuntos de entrenamiento, donde cada subconjunto se usa para obtener un vector de pesos w , al final todos estos wi se suman para obtener un vector de pesos general que abarque muchos casos particulares de la matriz H . Entrenamos 70 péptidos de cada conjunto y guardamos el resto para probar dicho método.

A continuación presentamos las gráficas que reflejan el comportamiento de predicción para los tres conjuntos de datos, para ambos métodos.

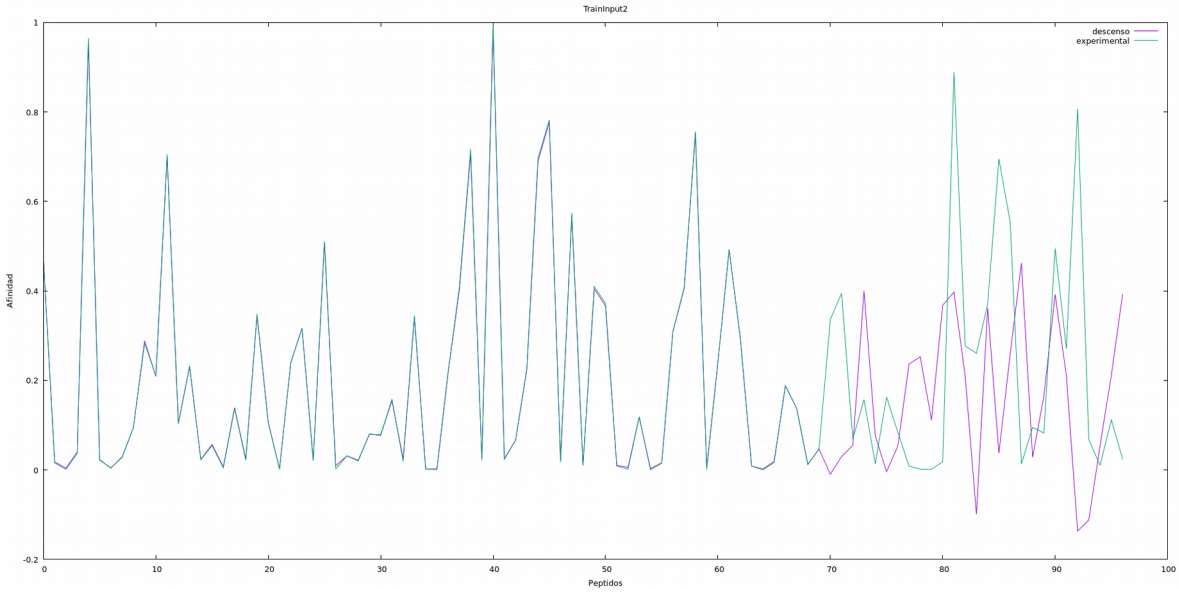

Figura 12: Resultados de afinidad utilizando método de descenso máximo para TrainInput2

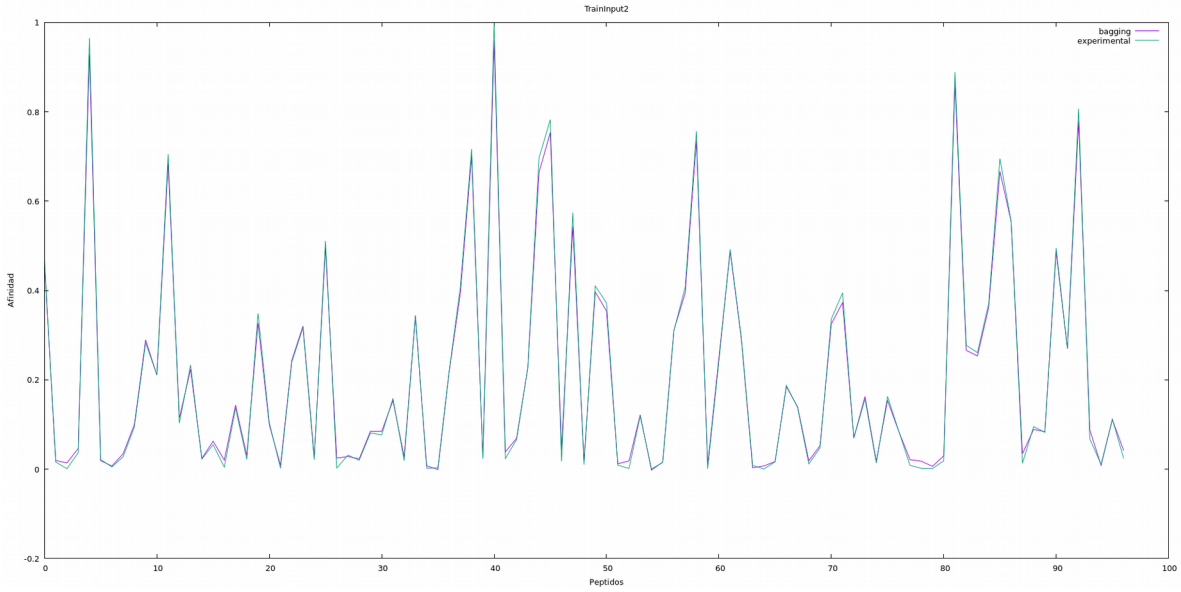

Figura 13: Resultados de afinidad utilizando método de bagging para TrainInput2

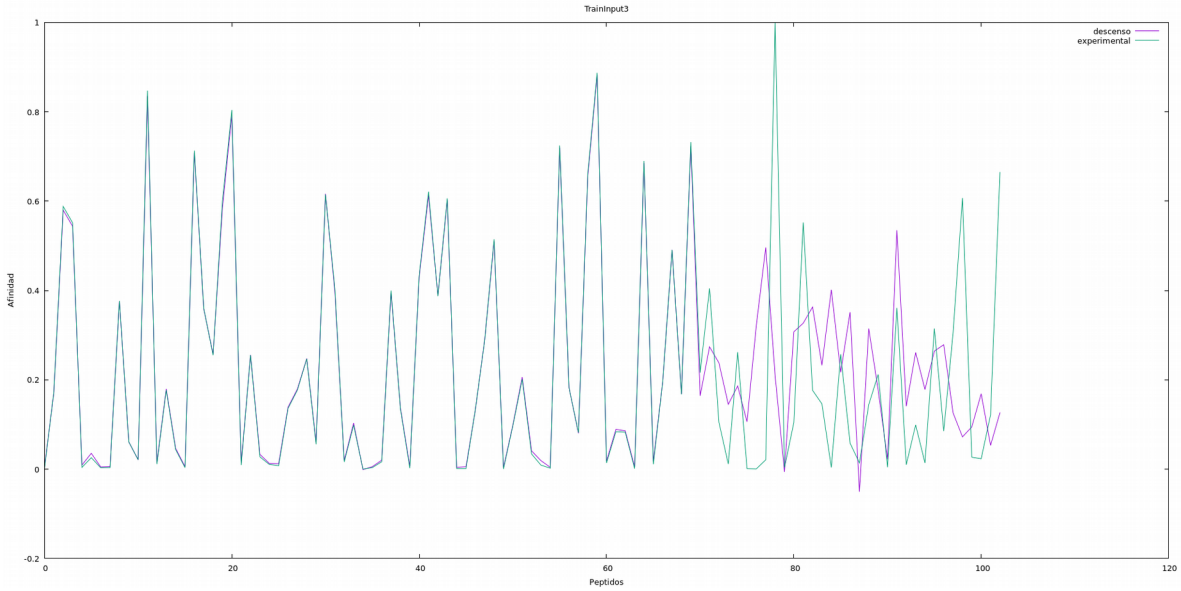

Figura 14: Resultados de afinidad utilizando método de descenso máximo para TrainInput3

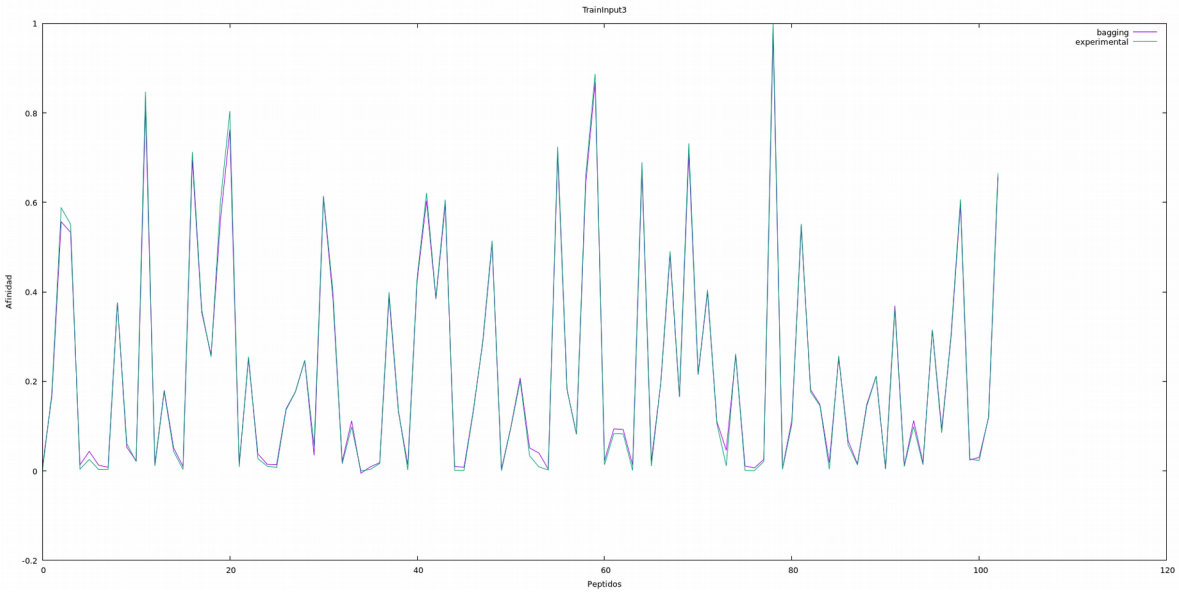

Figura 15: Resultados de afinidad utilizando método de bagging para TrainInput3

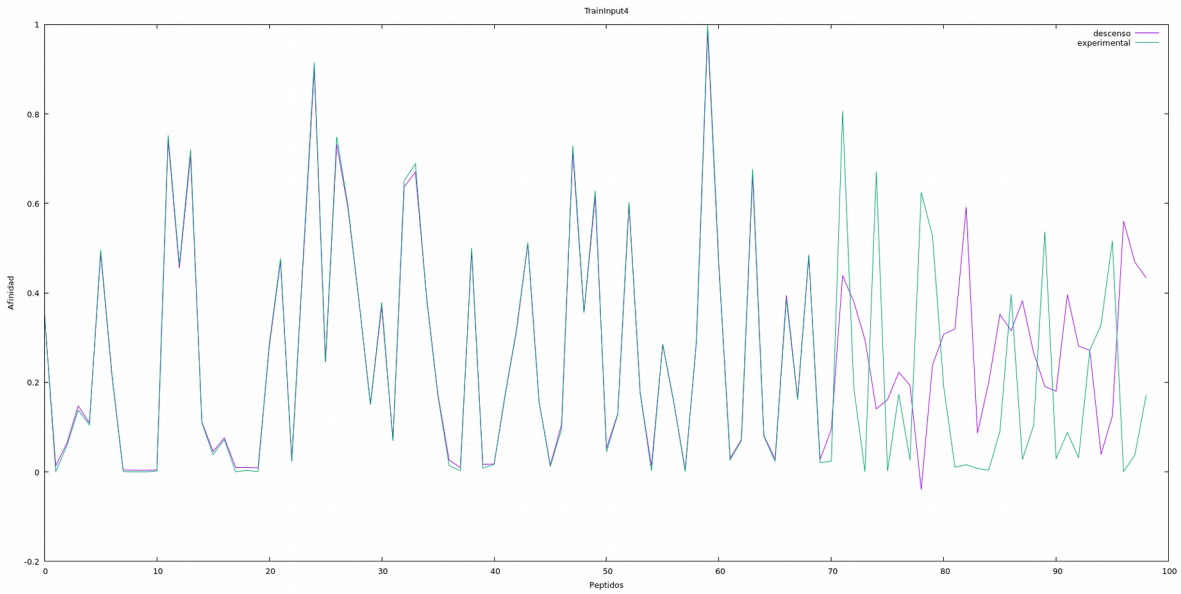

Figura 16: Resultados de afinidad utilizando método de descenso máximo para TrainInput4

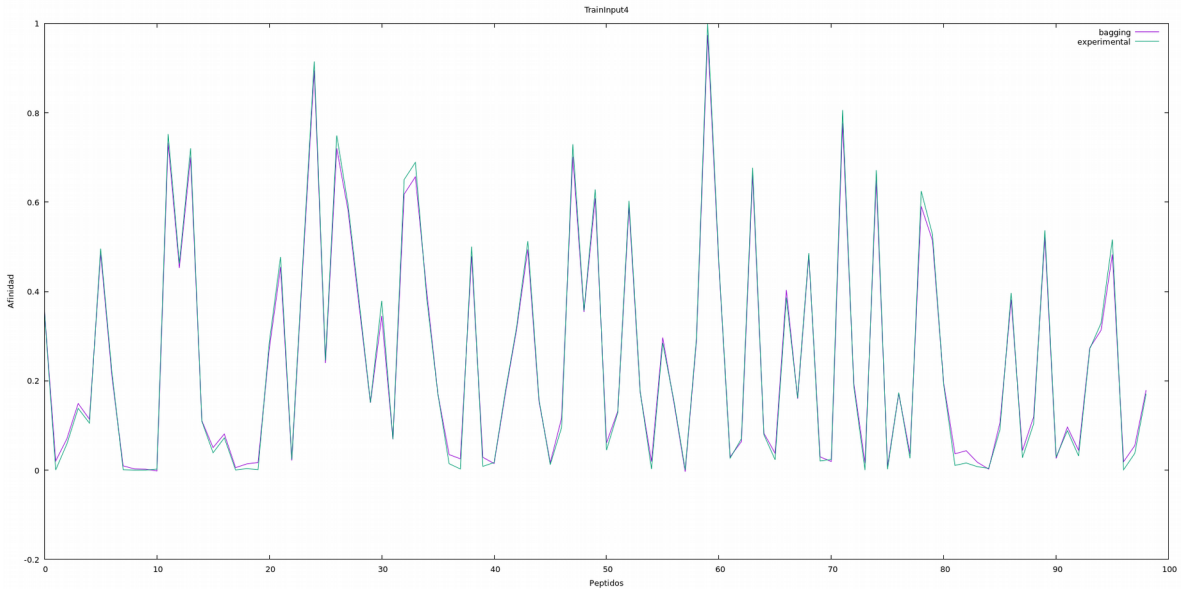

Figura 17: Resultados de afinidad utilizando método de bagging para TrainInput4

Para una correcta interpretación de los resultados existen varios puntos a considerar. Lo primero es referente a las muestras utilizadas, como se comentó anteriormente los tres conjuntos de datos utilizados en nuestras pruebas provienen de la base de datos IEDB, por lo que su principal característica es que todas las mediciones realizadas para la obtención de estos datos fueron *in vivo*, es decir se tomaron directamente sobre muestras celulares por lo que no puede haber mayor fiabilidad y exactitud respecto al origen de las muestras.

Lo siguiente es respecto de los valores de cada dato, las entradas del programa son como se ha comentado anteriormente péptidos representados mediante una cadena de caracteres, cada uno de éstos viene asociado a un valor numérico de medición realizado en laboratorio, dichos valores van desde las decenas hasta las unidades de millar, todas con hasta seis decimales, por lo que a efectos de simplificar el entendimiento de los resultados obtenidos se procedió a normalizar todos los valores de modo que entren en un rango de 0 a 1.

En lo que refiere a las gráficas plasmadas en esta sección, tenemos dos para cada conjunto de datos, una como resultado del método de descenso máximo y otra como resultado del método de bagging, es decir, seis gráficas diferentes, de las que las lineas verdes representan los valores medidos previamente en laboratorio, las líneas púrpuras representan los valores de predicción de cada caso.

Al observar y analizar detenidamente los resultados obtenidos en todos los casos, podemos determinar lo siguiente:

Al ejecutar el método de descenso máximo observamos una “predicción casi perfecta” de los datos usados en el entrenamiento, es decir los primeros 70 péptidos de cada conjunto de datos, pero en cuanto sometemos a prueba los datos no utilizados durante el entrenamiento obtenemos resultados bastante deficientes, lo que refleja que el método no es capaz de predecir correctamente datos a menos que estén sobre-entrenados, es decir, no es un buen método de

predicción puesto que no es estocástico sino determinista. El margen de error general de este método es grande.

En el caso del método bagging podemos ver que la predicción de los datos de entrenamiento no es tan buena como en el descenso máximo, que realiza una predicción casi idéntica para los primeros 70 péptidos, pero en cambio se obtiene una mejora significativa en los datos que no han sido utilizados en el entrenamiento (el conjunto ciego), debido a que el algoritmo propio del método permite reconocer la mayoría de los datos del conjunto en la etapa de entrenamiento. Tomando en cuenta lo anterrior, es evidente que el método de bagging es balanceado y presenta una actuación predictiva satisfactoria, por encima del método de descenso máximo, por lo que al final optamos por unir los puntos fuertes ambos métodos, es decir, con el descenso máximo obtenemos una lambda que se añade a la diagonal de la matriz H para que ésta pueda someterse al proceso de estabilización permitiendo obtener la inversa de $Ht * H$ para luego ejecutar el bagging con 1000 iteraciones y así mejorar la predicción del método.

Como conclusión, se logró desarrollar una aplicación de escritorio paralela con capacidad para predecir de forma eficaz la afinidad de epítopos a la molécula MHC-II, sin la necesidad de acceso a internet y utilizando equipo hardware de bajo costo, específicamente una computadora portatil con una tarjeta gráfica Nvidia GEFORCE GT 635M de 2 GB.

A continuación se presentan las tablas de resultados para los valores ya normalizados de predicción en cada caso, así como el error absoluto total de cada valor predicho con respecto a su equivalente medido en laboratorio:

Péptido	*Valor medido*	*Bagging*	*Descenso máximo*	*Error absoluto*	
				Bagging	*Descenso*
FERVGPEWEPVPLTV	0.469835	0.462067	0.469251	0.007768	0.000584
AGYLVGRKPLAFFSW	0.01546	0.019095	0.017407	0.003635	0.001947
AMFVEDIAMGYVVSS	0.000832	0.013612	0.003623	0.01278	0.002791
AQLGLRKKTKQSITE	0.036382	0.046001	0.039431	0.009619	0.003049
ATERFRWLLIDLLRE	0.965044	0.928166	0.955032	0.036878	0.010012
CIANGVSTKIVTRIS	0.021313	0.018661	0.022417	0.002652	0.001104
CRDSLASMVKSHVAL	0.004493	0.006629	0.003857	0.002136	0.000636
CSNSHVNTLRFLVKN	0.027727	0.034426	0.029144	0.006699	0.001417
DQEYHRLIHSLSKTS	0.094132	0.099436	0.093743	0.005304	0.000389
EEALNVALAVVTLLA	0.282915	0.289255	0.288357	0.00634	0.005442
EGWPYIACRTSIVGR	0.210488	0.210367	0.208904	0.000121	0.001584
EKMFVSPTPGQRNPY	0.705257	0.683620	0.700087	0.021637	0.00517
FLIMRNLTNLLSARK	0.102639	0.114323	0.105656	0.011684	0.003017
FLVKCQLQNPGVADL	0.233019	0.223508	0.230648	0.009511	0.002371
FMRMAWGGSYIALDS	0.022192	0.023462	0.022880	0.00127	0.000688
FRILSSISLALVNSM	0.054239	0.062347	0.056797	0.008108	0.002558
GFVGLCRTLGSKCVR	0.004239	0.019855	0.006683	0.015616	0.002444
GILHNLSDLYALITE	0.136554	0.142981	0.138887	0.006427	0.002333
GRYNCKCCWFADKNL	0.021453	0.030043	0.024158	0.00859	0.002705
GSLIVNPSLNGFLSK	0.348586	0.326697	0.344346	0.021889	0.00424
GVWVLAEPTKGKNER	0.1041	0.099397	0.106095	0.004703	0.001995
IHSLRRLYPSVFEKH	0.001548	0.007940	0.001607	0.006392	5.9E-05
IISTFHLSIPNFNQY	0.23965	0.243526	0.239627	0.003876	2.3E-05
ILLLDYMTSTNTNNS	0.315893	0.320141	0.316958	0.004248	0.001065
INSMKTSFSSRLLIN	0.020862	0.025479	0.020986	0.004617	0.000124
ISGYNFSLGAAVKAG	0.510768	0.496563	0.508012	0.014205	0.002756
ITLRQMSILTHVNNV	0.002055	0.024492	0.009188	0.022437	0.007133
IYGLPWMTTQTSALS	0.03121	0.027959	0.031185	0.003251	2.5E-05
KGSPEFDWILGWTIK	0.019547	0.022643	0.021409	0.003096	0.001862
KIPTHRHIVGKPCPK	0.080609	0.084465	0.079477	0.003856	0.001132
KNKVNLLTHSINALI	0.07636	0.084223	0.078378	0.007863	0.002018
KSAFQSSIASGFVGL	0.15827	0.153166	0.154135	0.005104	0.004135
KSRFFIWSQEVPLLT	0.018916	0.027543	0.024583	0.008627	0.005667
LIRKKLMTSPKWVQM	0.344419	0.342209	0.341814	0.00221	0.002605
LKAEAQMSIQLINKA	0.001755	0.007172	0.001763	0.005417	8E-06
LLKILVLSILSSPTK	0.002577	-0.001119	0.000621	0.003696	0.001956
LLPSHSTVLTSHTLT	0.219179	0.216908	0.217282	0.002271	0.001897
LLTKFVAAALHNIKC	0.408746	0.394099	0.403150	0.014647	0.005596
LLWDYMCISLSTAIE	0.716665	0.701781	0.707913	0.014884	0.008752
LMTSPKWVQMCSRTL	0.023111	0.026690	0.022320	0.003579	0.000791
LQGLRYFIMAYVNQA	1	0.960004	0.987307	0.039996	0.012693
LQIILSGKMAHLRKV	0.023362	0.038474	0.024855	0.015112	0.001493
LQLVGIQRAGLAPTG	0.066001	0.068787	0.066348	0.002786	0.000347
LQMVGMRRPQQGASG	0.226821	0.226452	0.223201	0.000369	0.00362
LRLFDYNKNAIKTLN	0.697159	0.664630	0.689988	0.032529	0.007171
LRNVACQEAVKLKLI	0.782347	0.753440	0.777974	0.028907	0.004373
MGQFISFMQEIPTFL	0.01705	0.041993	0.020918	0.024943	0.003868

MGQLISFFGEIPSII	0.574345	0.543454	0.569435	0.030891	0.00491
NIRYLVMAIVSDFSS	0.009813	0.017903	0.009952	0.00809	0.000139
NMLTHSINSLISDNL	0.409949	0.396674	0.404771	0.013275	0.005178
NQFGSVPAVTISCMT	0.37191	0.353753	0.366233	0.018157	0.005677
NSLLFIPDIKLAIDN	0.008563	0.011727	0.009784	0.003164	0.001221
PKGFYASPSVKTSLV	0.001298	0.017965	0.005475	0.016667	0.004177
PKIFFRPTTITANVS	0.118744	0.121264	0.117800	0.00252	0.000944
PSPIGYLGLLSQRTR	0.000209	-0.002509	0.002369	0.002718	0.00216
PSVFINPISHTSYCY	0.014609	0.015425	0.015755	0.000816	0.001146
PTHRHLKGEACPLPH	0.31008	0.311242	0.309463	0.001162	0.000617
QEYHRLIHSLAKTNN	0.407683	0.394283	0.405496	0.0134	0.002187
QGLRYFIMAYVNQAH	0.756388	0.736789	0.753255	0.019599	0.003133
QGVYMGNLSQSQLAK	0.000435	0.008213	0.003318	0.007778	0.002883
QYDVIIQHPADMSWC	0.241107	0.249513	0.240768	0.008406	0.000339
REEHYIVLSSELRLS	0.492014	0.487476	0.492797	0.004538	0.000783
RFFLPIFSEFVLLAT	0.293104	0.289890	0.292062	0.003214	0.001042
RHIVGKPCPKPHRLN	0.008227	0.002647	0.008182	0.005580	4.5E-05
RLTQSHPILNMIDTK	0	0.006796	0.001907	0.006796	0.001907
RRRQLLNLDVLCLSS	0.016063	0.016778	0.018534	0.000715	0.002471
RRRVMIQSSGGKLRL	0.187333	0.184852	0.188259	0.002481	0.000926
RVNNSYSLIRLSHNS	0.137658	0.139027	0.137872	0.001369	0.000214
RWLLIEILKASKSML	0.011433	0.018458	0.012603	0.007025	0.00117
SDDQISIMKLPLSTK	0.04728	0.052364	0.046167	0.005084	0.001113
SFDSKLHLISLLSLL	0.336022	0.324858	-0.009929	0.011164	0.345951
SFIHNLRLSSSINIK	0.394685	0.372790	0.029663	0.021895	0.365022
SGDVIVKAIGALEDI	0.069053	0.069949	0.054570	0.000896	0.014483
SGEIIRAATTSPARE	0.156935	0.162463	0.399720	0.005528	0.242785
SIPTNVTCAIHIGEV	0.013461	0.016925	0.076501	0.003464	0.06304
SKFSNRRYLCTVTTK	0.16266	0.153546	-0.004197	0.009114	0.166857
SLLNNQFGTMPSLTM	0.084777	0.084328	0.052443	0.000449	0.032334
SRVLNYDFNKLTALA	0.008283	0.020830	0.236472	0.012547	0.228189
SSKLNKFISPKSVIG	0.001612	0.017380	0.252712	0.015768	0.2511
SWLNLAAHHPLRMVL	0.001082	0.005929	0.111455	0.004847	0.110373
THRHIIGEGCPKPHR	0.018107	0.029382	0.367762	0.011275	0.349655
TPRYIPSTSISSSNI	0.88881	0.866369	0.397422	0.022441	0.491388
TQLVLSSMVNPLVLS	0.277156	0.265753	0.208249	0.011403	0.068907
TSSSQSLISSPMSKK	0.260617	0.253343	-0.099458	0.007274	0.360075
VEKSQLLNEFNNLYA	0.370684	0.361601	0.361727	0.009083	0.008957
VLSYVIGLLPPDMVV	0.695136	0.666397	0.037189	0.028739	0.657947
VPFNVAQAYCIGKLK	0.554228	0.553801	0.258852	0.000427	0.295376
VSLIAIIKGIVNLYK	0.012784	0.034144	0.462262	0.02136	0.449478
WELGLSPQQICTNFK	0.094969	0.088561	0.028228	0.006408	0.066741
WRSFLNKVKSLRILN	0.082203	0.083696	0.164933	0.001493	0.08273
YDFNKLTALAVSQLT	0.494889	0.487353	0.392357	0.007536	0.102532
YGLRNSLLVAPMPTA	0.270744	0.269519	0.209942	0.001225	0.060802
YNFATCGIFALISFL	0.806967	0.779188	-0.136933	0.027779	0.9439
YNFATCGLIGLVTFL	0.067407	0.087364	-0.112151	0.019957	0.179558
YQLFSDAKANCTAES	0.010236	0.008027	0.054319	0.002209	0.044083
YQSRNKRLISHRNSK	0.111683	0.110685	0.211600	0.000998	0.099917
YSKFRSPQPNIVAAT	0.024542	0.041924	0.391909	0.017382	0.367367
Sumatoria del error absoluto				0.948391	6.631574

Tabla 4: Comparativa entre Descenso Máximo y Bagging para TrainInput2

Péptido	*Valor medido*	*Bagging*	*Descenso máximo*	*Error absoluto*	
				Bagging	*Descenso*
ELLEFHYYLSSKLNK	0.000153	0.010123	0.004011	0.00997	0.003858
LIEDYFEALSLQLSG	0.172741	0.166973	0.171365	0.005768	0.001376
SLFSLLLVIISHNLG	0.588099	0.556836	0.579681	0.031263	0.008418
SNKFHIRLIKGELSN	0.551354	0.533295	0.543662	0.018059	0.007692
KLAFLVQTEPRMLLM	0.003622	0.013167	0.008885	0.009545	0.005263
SMHLMLANAGRSSGS	0.025333	0.043536	0.035218	0.018203	0.009885
YFIMAYVNQAHHIQL	0.002831	0.01228	0.004759	0.009449	0.001928
NFWTNVKSISPLPSP	0.003061	0.00811	0.00556	0.005049	0.002499
PSSGCYIHFFREPTD	0.376451	0.375811	0.376129	0.00064	0.000322
KVKFGHVSINPADIA	0.060754	0.052888	0.060802	0.007866	4.8E-05
NLTFSDAQSAQSQCR	0.020834	0.022334	0.020649	0.0015	0.000185
IGHLLRGRNHFIYIV	0.847591	0.815002	0.834975	0.032589	0.012616
RRIFGVFKNPCTSHG	0.011104	0.012962	0.013532	0.001858	0.002428
FLLMYEMHRESLLKS	0.176589	0.180083	0.179443	0.003494	0.002854
LKALTTKHPSLNIIT	0.043554	0.051293	0.04645	0.007739	0.002896
DIYKGVYQFKSVEFD	0.00311	0.009737	0.005409	0.006627	0.002299
FYREPVDQKQFKQDS	0.713106	0.694718	0.708373	0.018388	0.004733
HEAINIALIAVSLIA	0.360394	0.354412	0.359071	0.005982	0.001323
ELLDQSDVKEPGVSR	0.254848	0.259257	0.25754	0.004409	0.002692
KSILLIMNANTLMGR	0.600175	0.560927	0.584618	0.039248	0.015557
IVVSRCKILTVIPSQ	0.804088	0.762676	0.792897	0.041412	0.011191
KLFNDPASPVAGNPH	0.008802	0.014349	0.012479	0.005547	0.003677
VEIAIFQPQNGQFIH	0.255039	0.250648	0.255423	0.004391	0.000384
LGLLYTVKYPNLNDL	0.027185	0.03763	0.032787	0.010445	0.005602
LLSYVIGLLPQGSVI	0.010276	0.014511	0.012533	0.004235	0.002257
FIMAYVNQAHHIDLM	0.007495	0.013967	0.011774	0.006472	0.004279
WDVTHCSTICLIKRV	0.135708	0.138491	0.138536	0.002783	0.002828
FNSLISIAQHLVSDR	0.176775	0.176573	0.179146	0.000202	0.002371
WLLIEVLKGMKTTSE	0.246679	0.246998	0.247442	0.000319	0.000763
NNVVQALTSLGLLYT	0.055286	0.034859	0.058896	0.020427	0.00361
GQHTLPRCWLIRNGS	0.614726	0.611753	0.616333	0.002973	0.001607
QKFVDTILSENGVVA	0.404047	0.386931	0.397054	0.017116	0.006993
LFLHLVGFPTHRHIQ	0.015832	0.021068	0.019014	0.005236	0.003182
SLPLFTGQASFDLAA	0.09827	0.111599	0.102922	0.013329	0.004652
EFSNFKVAFSRSLND	0	-0.005294	-0.001279	0.005294	0.001279
KYRWLNLSANGDLRL	0.003068	0.008871	0.005344	0.005803	0.002276
FSLSAAVKAGASLID	0.016158	0.017995	0.019653	0.001837	0.003495
SRCYSIYLSINGVLE	0.399494	0.388757	0.394877	0.010737	0.004617
MDIKNLLTACTIFYI	0.134917	0.132257	0.137035	0.00266	0.002118
LRAAMISLAKKIDVQ	0.001875	0.013796	0.006404	0.011921	0.004529
KGLHHLQIILSGKMA	0.432677	0.424405	0.428996	0.008272	0.003681
FMPEWANFKFRDLLF	0.62107	0.603672	0.61268	0.017398	0.00839
LWTQSLRRELSGYCS	0.38696	0.384206	0.388713	0.002754	0.001753
DLGKKRFLLIRNSTW	0.606143	0.595262	0.602118	0.010881	0.004025
LNKFVSPKSVIGRFV	0.000991	0.009838	0.003818	0.008847	0.002827
FQDAYNAAGGHNAVF	0.001039	0.007929	0.005236	0.00689	0.004197
LANAGRSSGSRRPLG	0.132961	0.134919	0.132408	0.001958	0.000553
SLRKLSSVCLALTNS	0.289065	0.286864	0.290145	0.002201	0.00108
RCYSLYIAENGELTE	0.514529	0.507334	0.509706	0.007195	0.004823
QLVFNSISARALKAY	0.000171	0.003182	0.002778	0.003011	0.002607
TKKSIKEIASSISRL	0.0965	0.097591	0.098466	0.001091	0.001966
NKFVSPKSVIGTFVA	0.201201	0.20755	0.205764	0.006349	0.004563

ILTYNKTSKTTILSK	0.033704	0.051121	0.040693	0.017417	0.006989
QVVLSSMINPLVMST	0.008727	0.039635	0.018779	0.030908	0.010052
FTCDQGYHSSDPNAV	0.00186	0.00369	0.003767	0.00183	0.001907
NKALELFRKDIAAKY	0.724503	0.710051	0.719746	0.014452	0.004757
IQLVFSSMINPLVIT	0.18171	0.181775	0.182028	6.5E-05	0.000318
LECFVRSTPASFEKK	0.081459	0.082395	0.080105	0.000936	0.001354
LLDNRSNHYEEVIAS	0.660928	0.642579	0.654928	0.018349	0.006
CTGMLKRRLGLMSLS	0.88726	0.868681	0.882615	0.018579	0.004645
YYGTQTILAACVDLG	0.013617	0.022649	0.017476	0.009032	0.003859
WTQSLRRGLSAWTTS	0.083413	0.09375	0.088673	0.010337	0.00526
RNNYRDSFNNFDASI	0.082441	0.092057	0.085486	0.009616	0.003045
IVYSLVTTISSLSRI	0.000976	0.012657	0.004864	0.011681	0.003888
NLMGKTLILLETFVR	0.689946	0.671574	0.685064	0.018372	0.004882
TILIKKYNLNRAMML	0.010646	0.022122	0.016922	0.011476	0.006276
VIGLLPQNMVLTTQG	0.195381	0.194747	0.195365	0.000634	1.6E-05
SIISHNFCNLTSAFN	0.490576	0.482902	0.490621	0.007674	4.5E-05
PSLIKTLQSRMSKNF	0.167207	0.165275	0.168072	0.001932	0.000865
LSLCNKIKGLKVFNT	0.732144	0.709924	0.722613	0.02222	0.009531
KVEFTGDLVVKALGA	0.215888	0.214848	0.164181	0.00104	0.051707
ITDAVGNDMPGGYCL	0.404535	0.398561	0.273742	0.005974	0.130793
KWVQMCSRTLKNSHQ	0.104715	0.110107	0.237874	0.005392	0.133159
NSVVQALTSLGLLYT	0.011248	0.045849	0.14484	0.034601	0.133592
KSRTLKSFFAWSLSD	0.26164	0.258206	0.18598	0.003434	0.07566
KLKFNSVIVNPSLNG	0.000967	0.010816	0.105902	0.009849	0.104935
VKDKYMWCYSQVNKR	0.000371	0.006342	0.322624	0.005971	0.322253
PAPGAAGPPQVGLSY	0.020544	0.025231	0.496201	0.004687	0.475657
LYKRVLHMLLLLIQT	1	0.978287	0.209205	0.021713	0.790795
YFESFVREFVATART	0.003415	0.005524	-0.006355	0.002109	0.00977
FLHLVGFPTHRHIRG	0.104202	0.113489	0.307393	0.009287	0.203191
LGMNHVLQSIRRNYP	0.552187	0.546252	0.326609	0.005935	0.225578
RQLIKTDISMSMPKF	0.176222	0.181414	0.362888	0.005192	0.186666
DKYNKQLMVSSCVTS	0.145783	0.148069	0.23235	0.002286	0.086567
LDGVNLVASQPIFTG	0.003485	0.018017	0.401461	0.014532	0.397976
TSAFNKKTFDHTLMS	0.25706	0.252351	0.216626	0.004709	0.040434
TMFEALPHIIDEVIN	0.058028	0.068022	0.351225	0.009994	0.293197
FTNFKVAYSKSLKEL	0.01363	0.015361	-0.050783	0.001731	0.064413
YRKGLGNFVQTDRKS	0.144414	0.148696	0.314467	0.004282	0.170053
RLIHSLSNVKNQSLG	0.211701	0.211148	0.177127	0.000553	0.034574
KLCLMKAQPTSWPLQ	0.003967	0.004971	0.021581	0.001004	0.017614
TMTRPILRLLVLAVL	0.360863	0.368973	0.534648	0.00811	0.173785
NNQNFFWAVKPKVVR	0.009618	0.011359	0.140706	0.001741	0.131088
TILQRLGVLFGSRIA	0.098837	0.112251	0.26106	0.013414	0.162223
VNSPRPAPGAAGPPQ	0.013648	0.01788	0.178022	0.004232	0.164374
MFISDTPGERNPYEN	0.314481	0.314818	0.264083	0.000337	0.050398
SGRLKFLDVCVALDM	0.084942	0.094007	0.278428	0.009065	0.193486
QVNTSKTGINENYAK	0.305452	0.29978	0.125525	0.005672	0.179927
RVNQLIRYSGYRETP	0.606848	0.596156	0.07196	0.010692	0.534888
LSLMLNYPNSADRYY	0.026479	0.024377	0.094338	0.002102	0.067859
FQKTILKATTALKDV	0.02289	0.030166	0.168502	0.007276	0.145612
LLNNQFGTMPSLTLA	0.121185	0.118711	0.052991	0.002474	0.068194
YQRSEEEKFPYIMGD	0.664444	0.655973	0.125965	0.008471	0.538479
Sumatoria del error absoluto				0.911003	6.633633

Tabla 5 Comparativa entre Descenso Máximo y Bagging para TrainInput3

Péptido	*Valor medido*	*Bagging*	*Decenso máximo*	*Error absoluto*	
				Bagging	*Descenso*
RDIFLSQHHPSSLLL	0.355492	0.356169	0.353916	0.000677	0.001576
PKLEFGSLIVNPSLN	0.000175	0.020186	0.013170	0.020011	0.012995
DDDYGEPIIITSYLQ	0.058982	0.070352	0.064265	0.01137	0.005283
GLVSQLSVLSSITNI	0.138034	0.148924	0.147383	0.01089	0.009349
LHRVVLLESIAQFGD	0.105048	0.113828	0.109019	0.00878	0.003971
RSVQRNTVFKAGDLG	0.495941	0.485916	0.489426	0.010025	0.006515
IIKYNRRLAKSIICE	0.219128	0.210038	0.215763	0.00909	0.003365
TSKYRSPQPNIVAAT	0.000725	0.009266	0.004322	0.008541	0.003597
RYFLMAFANQIHHID	0.000531	0.002515	0.003415	0.001984	0.002884
GWPYIGSRSQILGRS	0.000539	0.001929	0.003506	0.00139	0.002967
KNLTGLVSAGPKAKS	0.001988	-0.002157	0.003882	0.004145	0.001894
LLNEFNNLYADKVSV	0.752412	0.732801	0.742868	0.019611	0.009544
RFKYLLNVSYLCHLV	0.462414	0.452423	0.455116	0.009991	0.007298
FFVFLALAGRSCTEE	0.720589	0.700661	0.707811	0.019928	0.012778
RESLESLWAPFGVLR	0.109156	0.110482	0.111309	0.001326	0.002153
SCGMYGLKGPDIYKG	0.03857	0.050242	0.045628	0.011672	0.007058
ETVEKIVDQYREPVK	0.071994	0.080666	0.075788	0.008672	0.003794
RGPWRYIRSFPILAS	0.00021	0.005347	0.009394	0.005137	0.009184
LAVFQPSSGNYVHCF	0.003059	0.013612	0.009411	0.010553	0.006352
KLNNQFGSMPALTIA	0.00092	0.016851	0.008954	0.015931	0.008034
PEIWHHLSTLIKQPD	0.285588	0.270606	0.279367	0.014982	0.006221
VLKWHLHKAVEVPIS	0.476974	0.454396	0.471753	0.022578	0.005221
NFSLGAAVKAGAALL	0.02378	0.021293	0.024576	0.002487	0.000796
TGLWPFIRINNLKVK	0.473718	0.461959	0.463818	0.011759	0.0099
EKPGNRNPYENLLYK	0.91505	0.894803	0.901184	0.020247	0.013866
TFTNDSIISHNFCNL	0.245585	0.239611	0.245365	0.005974	0.00022
GPDNPGEPLVLKEGI	0.749226	0.720534	0.732511	0.028692	0.016715
RLCFSKSKNTLMYEI	0.59538	0.582359	0.589377	0.013021	0.006003
NVKCKTPTQLAETID	0.37726	0.365705	0.377983	0.011555	0.000723
MDLADLFNAQPGLTS	0.150929	0.150927	0.150150	2E-06	0.000779
LLESLSSLGAHLDSD	0.378934	0.345127	0.369856	0.033807	0.009078
RYHAPIYGAVHPRWL	0.069082	0.068863	0.071957	0.000219	0.002875
LDMIITAVNSLISDN	0.650589	0.617927	0.637191	0.032662	0.013398
SGGNHMLLDGVSVVA	0.689029	0.656734	0.670241	0.032295	0.018788
NLTNLLSARKLDSSK	0.386508	0.400827	0.388335	0.014319	0.001827
SGFIGFCKSMGSKCV	0.170307	0.170136	0.173590	0.000171	0.003283
PEKFNARMATMLEYV	0.01411	0.034503	0.025971	0.020393	0.011861
EFENFMKAGAHPIMH	0.002326	0.025181	0.009180	0.022855	0.006854
TYSQLMTLKDAKMLQ	0.500256	0.478659	0.493249	0.021597	0.007007
NYSLSAAVKAGATLL	0.008145	0.028496	0.016422	0.020351	0.008277
GLLSYVIGLLPQNMV	0.01661	0.014489	0.017543	0.002121	0.000933
GQFIHFYREPVDQKQ	0.175408	0.170629	0.173712	0.004779	0.001696
SPLLTEGFKLLSSLV	0.319587	0.314733	0.316589	0.004854	0.002998
SVCNKVKGLKVFNTR	0.512556	0.493998	0.508769	0.018558	0.003787
ILVLILAHPSKRSQK	0.156959	0.153313	0.156867	0.003646	9.2E-05
EIPSFRWTQSLRRGL	0.012026	0.016456	0.014743	0.00443	0.002717
IPSIIHEALNIALIA	0.094976	0.116027	0.105746	0.021051	0.01077
KVGEVCSFYADPKRY	0.729692	0.701392	0.715372	0.0283	0.01432
LKGIQSLRKLSSVCL	0.357184	0.353971	0.355772	0.003213	0.001412
NHVIQSVRRLYPKIF	0.628315	0.608348	0.614927	0.019967	0.013388

GFIGFCKSMGSKCVR	0.044562	0.060683	0.052947	0.016121	0.008385
HKGIVIKSKKKGSTP	0.128189	0.131699	0.130036	0.00351	0.001847
SCLDGKLCLMKAQPT	0.602789	0.588432	0.596051	0.014357	0.006738
IPIQLLPNTLVFQAK	0.17524	0.174156	0.176608	0.001084	0.001368
DYLILKNLTGLVSAG	0.002063	0.018751	0.012144	0.016688	0.010081
ENMLRSMPVKGKRKD	0.284491	0.296559	0.285586	0.012068	0.001095
RNLKNAGLIVGQMIL	0.154906	0.150556	0.153318	0.00435	0.001588
HFLLRGPFEASWAIK	0.000528	-0.003798	0.004284	0.004326	0.003756
VLSYVIGLLPQDMVI	0.293249	0.288806	0.291604	0.004443	0.001645
LYKGVYELQTLELNM	1	0.974271	0.983934	0.025729	0.016066
FINSLLKSLLLLNTR	0.46343	0.457313	0.455754	0.006117	0.007676
EKDSPFKLSSSEPHC	0.026643	0.029236	0.030241	0.002593	0.003598
PALFFTFLANLNLTE	0.070078	0.063818	0.072106	0.00626	0.002028
NDAYNLVHLIMIYFF	0.676971	0.660001	0.662866	0.01697	0.014105
DEFFECFKYLLIQGH	0.079086	0.082521	0.080948	0.003435	0.001862
KTISVVTLLCVLPAV	0.023013	0.037376	0.027942	0.014363	0.004929
SIAQHLVSDRPIMRY	0.386243	0.403168	0.394480	0.016925	0.008237
LQFNQMMNPSHVKFL	0.161036	0.159643	0.164503	0.001393	0.003467
YLGLLSQRTRDIYIS	0.485629	0.480896	0.481700	0.004733	0.003929
DKLSVRVVYSTMDVN	0.020451	0.029507	0.027254	0.009056	0.006803
ASLFLHLVGIPTHRH	0.023668	0.019044	0.094686	0.004624	0.071018
MLIESNLAGSNDNFL	0.806168	0.776668	0.438885	0.0295	0.367283
LMIAHRVLLSSILES	0.187199	0.194983	0.380215	0.007784	0.193016
DITYKVHLATPINSR	0	0.015673	0.294692	0.015673	0.294692
IVQNAYKQMIKSRTL	0.671453	0.649685	0.140667	0.021768	0.530786
KRFFLPVFSDEVLAG	0.001765	0.007904	0.161289	0.006139	0.159524
NKNFFWAVKPKAVRQ	0.173278	0.170600	0.222547	0.002678	0.049269
VSLIAALKGMINLWK	0.02624	0.036311	0.193645	0.010071	0.167405
NDSKLLKMVTSVIKN	0.624629	0.590473	-0.039528	0.034156	0.664157
CTNFKTQLVLSSMVN	0.527466	0.513739	0.238450	0.013727	0.289016
ILKGVINIWGSGLLQ	0.189903	0.194044	0.307453	0.004141	0.11755
LSSKFNKFVSPKSVS	0.01032	0.036385	0.319497	0.026065	0.309177
RQKIIYSGAVNLDDE	0.015645	0.043202	0.591787	0.027557	0.576142
HYLVNHPEVLVEASQ	0.007401	0.017202	0.086320	0.009801	0.078919
YREEIYRKGLGNFVQ	0.003869	0.002072	0.198934	0.001797	0.195065
DGNYPLHIASKINNN	0.090374	0.105404	0.351923	0.01503	0.261549
RLLDILEAIKLIRKK	0.396455	0.381528	0.315687	0.014927	0.080768
MGQIVTMFEALPHII	0.02751	0.043380	0.382207	0.01587	0.354697
VLDILTANKLIRQKL	0.103425	0.118900	0.266130	0.015475	0.162705
GLDFSEVSNVQRLMR	0.536905	0.521182	0.190868	0.015723	0.346037
YEYKVQQAMSNLVLG	0.029094	0.025952	0.180013	0.003142	0.150919
TVYCKNILALSMTKK	0.088117	0.096221	0.396180	0.008104	0.308063
LECFVRSSPASFEKK	0.031376	0.043930	0.281127	0.012554	0.249751
IYEPEDLGNCLNKSD	0.271967	0.273786	0.271325	0.001819	0.000642
SIGSNLTIACRVSLR	0.328949	0.313187	0.039044	0.015762	0.289905
DSTVIRNLKNAGLIV	0.516126	0.482812	0.124922	0.033314	0.391204
YVYDSLTTPTASVCQ	0.000503	0.018608	0.560567	0.018105	0.560064
SVEFDMSHLNLTMPN	0.037436	0.054812	0.469279	0.017376	0.431843
VVPDGYKLTGNVLIL	0.170359	0.178224	0.434040	0.007865	0.263681
Sumatoria del error absoluto				1.225677	8.340446

Tabla 6: Comparativa entre Descenso Máximo y Bagging para TrainInput4

BIBLIOGRAFÍA

[1] "VACUNOLOGIA INVERSA | Biotecnología." [Online]. Available: http://blogs.creamoselfuturo.com/bio-tecnologia/2011/07/11/vacunologia-inversa/. [Accessed: 24-Jun-2015].

[2] "inmunologiaenlinea.es - 01. Introducción." [Online]. Available: http://inmunologiaenlinea.es/index.php/01-introduccion-a-la-inmunologia-njevo. [Accessed: 13-Aug-2015].

[3] "Vacunología Inversa." [Online]. Available: http://www.sap.org.ar/docs/congresos/2011/infectologia/plotkin_estrategias.pdf. [Accessed: 13-Aug-2015].

[4] "Tema 01. Introducción a la Inmunología."

[5] R. Naves P. and M. R. Bono M., *Citoquinas*. 2002.

[6] "Community Nursing Yukon Immunization Program Section 14 -Principles of Immunology SECTION 14 -PRINCIPLES OF IMMUNOLOGY," 2011.

[7] A. López–Martínez, C. Chávez–Muñoz, J. Granados, and A. López–Martínez, "Función biológica del complejo principal de histocompatibilidad," *Rev. Investig. clínica*, vol. 57, no. 2, pp. 132–141, 2005.

[8] A. C. Goldberg, L. V. Rizzo, − Avenida, and A. Einstein, "ReVIeWING BASIC SCIeNCeS Estrutura do MHC e função – apresentação de antígenos. Parte 1," *einstein*, vol. 13, no. 1, pp. 153–6, 2015.

[9] "Complejo mayor de histocompatibilidad." [Online]. Available: https://es.wikipedia.org/wiki/Complejo_mayor_de_histocompatibilidad#CMH_de_clase_II.

[10] K. (Kenneth M. . Murphy, P. Travers, M. Walport, and C. Janeway, *Inmunobiología de Janeway*, 7th ed. México: McGraw-Hill Interamericana, 2009.

[11] “Proteína.” [Online]. Available: https://es.wikipedia.org/wiki/Proteína.

[12] “MedicinePlus enciclopedia médica - Antígeno.” [Online]. Available: https://medlineplus.gov/spanish/ency/article/002224.htm.

[13] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, and P. Walter, “The Adaptive Immune System,” 2002.

[14] G. W. Litman *et al.*, “Phylogenetic diversification of immunoglobulin genes and the antibody repertoire.,” *Mol. Biol. Evol.*, vol. 10, no. 1, pp. 60–72, Jan. 1993.

[15] “La estructura de las proteínas.” [Online]. Available: http://depa.fquim.unam.mx/proteinas/estructura/index.html.

[16] “Linfocito B.” [Online]. Available: https://es.wikipedia.org/wiki/Linfocito_B.

[17] “Síntesis de péptidos.” [Online]. Available: http://www.ibt.unam.mx/computo/pdfs/met/sintesis_de_peptidos.pdf.

[18] D. Male, J. Brostoff, D. B. Roth, and I. Roitt, *Inmunología*, 7th ed. Elsevier Ltd, 2007.

[19] “Alelo.” [Online]. Available: https://es.wikipedia.org/wiki/Alelo.

[20] J. L. Salazar Mendoza, “Herramienta web para la integración de sistemas de predicción de regiones de unión al complejo mayor de histocompatibilidad clase II,” 2014.

[21] “Cliente-servidor.” [Online]. Available: https://es.wikipedia.org/wiki/Cliente-servidor.

[22] “C (lenguaje de programación).” [Online]. Available: https://es.wikipedia.org/wiki/C_(lenguaje_de_programación).

[23] “Using the GNU Compiler Collection (GCC).” [Online]. Available: https://gcc.gnu.org/onlinedocs/gcc-6.3.0/gcc/G_002b_002b-and-GCC.html#G_002b_002b-and-GCC.

[24] “GNU Compiler Collection.” [Online]. Available: https://es.wikipedia.org/wiki/GNU_Compiler_Collection.

[25] “Introducción a Linux Mint.” [Online]. Available: https://www.linuxmint.com/documentation/user-guide/Cinnamon/spanish_17.3.pdf.

[26] “Linux Mint.” [Online]. Available: https://es.wikipedia.org/wiki/Linux_Mint.

[27] “GNU Scientific Library - Introducción.” [Online]. Available: https://www.gnu.org/software/gsl/manual/gsl-ref.pdf.

[28] M. Nielsen *et al.*, “Improved prediction of MHC class I and class II epitopes using a novel Gibbs sampling approach.,” *Bioinformatics*, vol. 20, no. 9, pp. 1388–97, Jun. 2004.

[29] M. Nielsen, C. Lundegaard, and O. Lund, “Prediction of MHC class II binding affinity using SMM-align, a novel stabilization matrix alignment method.,” *BMC Bioinformatics*, vol. 8, no. 1, p. 238, Jan. 2007.

[30] M. Nielsen, S. Justesen, O. Lund, C. Lundegaard, and S. Buus, “NetMHCIIpan-2.0 - Improved pan-specific HLA-DR predictions using a novel concurrent alignment and weight optimization training procedure.,” *Immunome Res.*, vol. 6, p. 9, Jan. 2010.

[31] E. Karosiene, M. Rasmussen, T. Blicher, O. Lund, S. Buus, and M. Nielsen, “NetMHCIIpan-3.0, a common pan-specific MHC class II prediction method including all three human MHC class II isotypes, HLA-DR, HLA-DP and HLA-DQ.,” *Immunogenetics*, vol. 65, no. 10, pp. 711–24, Oct. 2013.

[32] W. Zhang, J. Liu, and Y. Niu, “Quantitative prediction of MHC-II binding affinity using particle swarm optimization.,” *Artif. Intell. Med.*, vol. 50, no. 2, pp. 127–32, Oct. 2010.

[33] Y. EL-Manzalawy, D. Dobbs, and V. Honavar, “Predicting MHC-II binding affinity using multiple instance regression.,” *IEEE/ACM Trans. Comput. Biol. Bioinform.*, vol. 8, no. 4, pp. 1067–79, 2011.

[34] S. P. Singh and B. N. Mishra, “Prediction Model of MHC Class-II Binding Peptide Motifs Using Sequence Weighting Method for Vaccine Design,” *2012 Int. Conf. Adv. Comput. Commun.*, pp. 234–237, Aug. 2012.

[35] “IEDB Introduction.” [Online]. Available: http://help.iedb.org/hc/en-us/articles/114094147031.

[36] “Net-MHCII 2.2 Server.” [Online]. Available: http://www.cbs.dtu.dk/services/NetMHCII/.

[37] “Computación paralela.” [Online]. Available: https://es.wikipedia.org/wiki/Computación_paralela.

[38] J. Sanders and E. Kandrot, *CUDA BY EXAMPLE - An Introduction to General -Purpose GPU Programming*. Addison-Wesley, 2010.

[39] T. Rauber and G. Rünger, *Parallel Programming for Multicore and Cluster Systems*. Springer, 2010.

[40] “GPGPU.” [Online]. Available: https://es.wikipedia.org/wiki/GPGPU.

[41] D. B. Kirk and W. W. Hwu, *Programming Masively Parallel Processors - A Hands-on Approach*. Elsevier Ltd, 2010.

[42] B. Peters and A. Sette, “Generating quantitative models describing the sequence specifity of biological processes with the SMM method.”