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

PREDICCIÓN DE AFINIDAD DE PÉPTIDOS CON LA MOLÉCULA MHC-II UTILIZANDO REDES NEURONALES ARTIFICIALES

QUE PARA OBTENER EL GRADO DE
MAESTRO EN SISTEMAS COMPUTACIONALES

PRESENTA:
JACOB EZEQUIEL MERECÍAS PÉREZ

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

LA PAZ, BAJA CALIFORNIA SUR, MÉXICO, DICIEMBRE 2020.

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Instituto Tecnológico de La Paz

"2020, Año de Leona Vicario, Benemérita Madre de la Patria"

La Paz, B.C.S., 08/diciembre/2020

DEPI_B/014/2020

ASUNTO: Autorización de impresión

C. JACOB EZEQUIEL MERECÍAS PÉREZ,
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: **"PREDICCIÓN DE AFINIDAD DE PÉPTIDOS CON LA MOLÉCULA MHC-II UTILIZANDO REDES NEURONALES ARTIFICIALES"**, 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"

JUAN PABLO MORALES ÁLVAREZ,
JEFE DE LA DIV. DE ESTUDIOS DE POSGRADO E INV.

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

c.c.p. Depto. de Servicios Escolares
c.c.p. Archivo.

JPMA/icl*

DICTAMEN DEL COMITÉ TUTORIAL
SUBDIRECCIÓN ACADÉMICA
DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN.

La Paz, B.C.S., **08/DICIEMBRE/ 2020**

C. JUAN PABLO MORALES ALVAREZ,
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 M19310001	Nombre JACOB EZEQUIEL MERECÍAS PÉREZ
Maestría en:	SISTEMAS COMPUTACIONALES
Título de la tesis: PREDICCIÓN DE AFINIDAD DE PÉPTIDOS CON LA MOLÉCULA MHC-II UTILIZANDO REDES NEURONALES ARTIFICIALES	
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

DR. CARLOS ELIUD ANGULO VALADEZ

MSC. ILIANA CASTRO LIERA

DR. MARCO ANTONIO CASTRO LIERA

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

ITLP-DEPI-RTT-08 **Rev.1**

CARTA CESIÓN DE DERECHOS

La presente se extiende en la Ciudad de La Paz, B.C.S. El día 8 del mes diciembre del año 2020 , el (la) que suscribe Jacob Ezequiel Merecías Pérez estudiante del Programa de **Maestría en Sistemas Computacionales** con número de control M19310001, manifiesta que es autor (a) intelectual del presente trabajo de Tesis bajo la dirección del Dr. Marco Antonio Castro Liera y el Dr. Carlos Eliud Angulo Valadez y cede los derechos del trabajo intitulado PREDICCIÓN DE AFINIDAD DE PÉPTIDOS CON LA MOLÉCULA MHC-II UTILIZANDO REDES NEURONALES ARTIFICIALES , en forma NO EXCLUSIVA, al Tecnológico Nacional de México/Instituto Tecnológico de la Paz para su reproducción total o parcial en cualquier medio con fines académicos, científicos y culturales, así como para su publicación electrónica del texto completo para difusión y consulta.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección ezequiel.merecias@gmail.com y mcastroliera@gmail.com. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.

Jacob E. Merecías Pérez

Nombre y firma

Dedicatoria

Dedicado a mi madre que siempre me apoyó en esta decisión y a mi padre amado que ya no está conmigo y que en vida siempre se preguntó por qué quería estudiar más. A mi amada novia que desde que le conozco me es inspiración a alcanzar la excelencia.

Agradecimientos

Primeramente agradezco al Dios por ayudarme a culminar esta etapa que de principio me parecía imposible alcanzar, sin embargo todo se acomodó para poder concluir una de las etapas más importantes de mi vida.

Gracias a mis directores de tesis, que siempre estuvieron otorgándome sus conocimientos para realizar esta investigación. Al Dr. Carlos Angulo que me otorgó sus conocimientos para conocer la gran importancia de la inmunología en la ciencia. Al Dr. Marco Antonio Castro Liera, que en los tiempos de duda siempre contestaba amablemente para orientarme. A la MSC. Iliana Castro Liera, que también durante toda la maestría nunca dudó el resolver mis dudas respecto a cualquier tema. Gracias a todos mis profesores por obsequiarme un poco de sus conocimientos en esta bonita institución.

Gracias a mi madre que siempre me apoyó en aquellos días que me veía forzado a estar todo el día frente a la computadora. Gracias a mi musa por apoyarme en aquellos días en que pensaba que no podría resolver los problemas que enfrenté en este proyecto.

Gracias a todos mis compañeros de maestría que directa o indirectamente me impulsaron a terminar.

Gracias a CONACYT por darme el privilegio de tener una beca durante toda la maestría gracias al PNPC, y al Instituto Tecnológico de La Paz por pertenecer a ello.

Resumen

El desarrollo de vacunas empleando bases de datos moleculares y el procesamiento de su información a través de la bioinformática ha implicado un gran avance para la vacunología convencional. No obstante, para estos procesos se requieren métodos que utilizan servidores web independientes. Éstos deben de ser implementados de manera secuencial, tienen dependencia total de Internet y a su vez generan resultados particulares del método empleado, mismos que deben ser comparados y unificados, con el fin de realizar pruebas de validación en laboratorio. Ante estos retos, el presente proyecto tuvo como objetivo desarrollar un algoritmo para predecir la afinidad de péptidos con el complejo mayor de histocompatibilidad de clase II (MHC-II) utilizando redes neuronales artificiales. Para ello se recopilaron datos directamente del servidor NetMHCIIPan, que son extraídos de la base de datos y recursos de análisis del epítopo inmune (IEDB), ésto para diseñar un algoritmo que se implementó en tres diferentes versiones; una primera versión utilizando Python 3.6, otra versión utilizando Python 3.6 con TensorFlow 2.1 sobre GPGPU, y una última versión en C. Para el entrenamiento y las pruebas del algoritmo se utilizaron péptidos de diferentes longitudes (n-meros) haciendo uso de la configuración propuesta por Andreatta et Al (2011) [1]. Los resultados demostraron que implementando el algoritmo en el lenguaje C es más eficiente que en Python, aun utilizando TensorFlow sobre GPGPU, además se tuvo una precisión competitiva con otros métodos, evaluando el AUC del algoritmo. En conclusión, los resultados demuestran que el modelo diseñado permite predecir afinidades de cualquier longitud de péptidos con una precisión aceptable, a su vez implementarlo en C comprobó que contribuye a mejorar la velocidad de procesamiento. Esta investigación sienta las bases para mejorar predecir epítopos de células T y por lo tanto hay campo abierto a la investigación de mejoras del algoritmo que logren una precisión más alta en los resultados.

Abstract

The development of vaccines using molecular databases and the processing of their information through bioinformatics has implied a great advance for conventional vaccinology. However, these processes require methods that use separate web servers. These must be implemented sequentially, have total dependence of Internet and in turn generate particular results of the method used, which must be compared and unified, in order to perform validation tests in the laboratory. Faced with these challenges, the present project aimed to develop an algorithm to predict the affinity of peptides with the The major histocompatibility complex class II (MHC-II) using artificial neural networks. For this, data were collected directly from the NetMHCIIPan server, which is extracted from the Immune Epitope Database Analysis Resource (IEDB), this to design an algorithm that was implemented in three different versions; a first version using Python 3.6, another version using Python 3.6 with TensorFlow 2.1 on GPGPU, and the last version using C. For the training and testing of the algorithm, peptides of different lengths (n-mers) were converted making use of the proposed configuration by Andreatta et al. (2011) [1]. The results indicated that implementing the algorithm in the C language is more efficient than in Python, even using TensorFlow on GPGPU, in addition there was a competitive precision with other methods, evaluating the AUC of the algorithm. In conclusion, the results show that the designed model allows predicting affinities of any length of peptides with an acceptable precision, in turn implementing it in C found that it contributes to improving the processing speed. This research lays the groundwork for improving predicting T cell epitopes and therefore there is an open field for the investigation of algorithm improvements that achieve higher precision in the results.

Índice general

1. Introducción **1**

1.1. Antecedentes 2

1.2. Descripción del problema 5

1.3. Objetivos 6

1.3.1. Objetivo general 6

1.3.2. Objetivos específicos 6

1.4. Justificación 6

1.5. Alcances y limitaciones 7

1.5.1. Alcances 7

1.5.2. Limitaciones 7

1.6. Hipotesis 7

1.7. Organización del documento 7

2. Marco Teórico **9**

2.1. Sistema Inmunológico 9

2.2. Vacunología Inversa 11

2.3. Complejo mayor de histocompatibilidad 12

2.4. Patógeno 13

2.5. Antígeno 14

2.6. Células B 14

2.7. Células T 14

2.8. Aminoácidos y proteínas 15

2.9. Péptido (epítope) 16

2.10. Alelo 17

2.11. Secuencias k-mero . . . 17
2.12. Concentración inhibitoria IC50 . . . 18
2.13. Alineamiento de secuencias . . . 18
2.14. Redes neuronales artificiales . . . 19
2.15. Sistema operativo Ubuntu . . . 20
2.16. Lenguaje Python . . . 20
2.17. Lenguaje C . . . 21
2.18. Plataforma CUDA . . . 22
2.19. TensorFlow . . . 22

3. Estado del Arte **24**
3.1. Técnicas de predicción . . . 24
3.2. Redes Neuronales Artificiales (RNA) . . . 24
3.3. Modelos de Regresión . . . 25
3.4. Máquinas de Vectores Soporte (MVS) . . . 25
3.5. Modelo Oculto de Markov (HMM) . . . 26
3.6. Árbol de Decisión . . . 26
3.7. Métodos de Consenso . . . 26

4. Metodología **28**
4.1. Introducción . . . 28
4.2. Instalación del ambiente de trabajo . . . 29
4.2.1. Instalación de GNU/Linux Ubuntu 18.04 . . . 29
4.2.2. Instalación de Python 3.6 . . . 30
4.2.3. Compilación e instalación de Tensowflow 2.1 . . . 31
4.3. Bases de Datos . . . 32
4.4. Matrices de sustitución . . . 34
4.4.1. Matriz identidad (one-hot encoding) . . . 34
4.4.2. Matrices de log-probabilidades . . . 35
4.4.3. Matrices PAM y BLOSUM . . . 35
4.5. Normalización de IC50 . . . 37
4.6. Medida de desempeño . . . 37

4.7. Alineamiento NN-Align 38
4.8. Método y algoritmo propuesto 40
4.9. Mejora de NN-Align 42
4.10. Implementaciones 43

5. Resultados y discusión **45**
5.1. Rendimiento predictivo AUC en cada molécula 45
5.2. Rendimiento predictivo AUC promedio en cada alelo 56
5.3. Comparaciones 56
5.3.1. Comparación del método propuesto contra otros métodos midiendo el AUC para el alelo HLA-DP 57
5.3.2. Comparación del método propuesto contra otros métodos midiendo el AUC para el alelo HLA-DQ 58
5.3.3. Comparación del método propuesto contra otros métodos midiendo el AUC para el alelo HLA-DR 60
5.3.4. Comparación del método propuesto contra otros métodos midiendo el AUC para el alelo H-2 61
5.4. Eficiencia computacional 61
5.5. Discusión 62

6. Conclusiones y trabajo futuro **64**
6.1. Conclusión 64
6.2. Trabajo futuro 65

Bibliografía **66**

Índice de figuras

1.1. Vacuna 3
1.2. Vacunología convencional e inversa 4

2.1. Sistema inmune innato y adaptativo 10
2.2. Sistema inmune adaptativo 11
2.3. Complejo mayor de histocompatibilidad clase 2 12
2.4. Estructura del SARS-CoV-2 13
2.5. Lista de aminoácidos proteicos 15
2.6. Representación gráfica de un péptido y aminoácidos 16
2.7. Núcleos de unión de un péptido 18
2.8. Composición de una red neuronal artificial 19
2.9. Bibliotecas de Aprendizaje Automático en Python. 21
2.10. Diferencia entre la arquitectura de un CPU y una GPU. 22
2.11. Diferencia entre un escalar, vector, matriz y tensor. 23

4.1. Contenido de los archivos que conforman las Bases de Datos. 33
4.2. Matriz identidad 34
4.3. Matriz BLOSUM 50 recopilada de la IEDB 36
4.4. Alineamiento NN-Align 38
4.5. Alineamiento de secuencias generado por NN-Align 39
4.6. Método propuesto 41

5.1. Predicciones para HLA-DP 46
5.2. Predicciones para HLA-DP 47
5.3. Predicciones para HLA-DQ 48

5.4. Predicciones para HLA-DQ 49
5.5. Predicciones para HLA-DQ 50
5.6. Predicciones para HLA-DQ 51
5.7. Predicciones para HLA-DR 52
5.8. Predicciones para HLA-DR 53
5.9. Predicciones para HLA-DR 54
5.10. Predicciones para HLA-DR 55
5.11. Predicción para H-2 55
5.12. Rendimiento predictivo AUC de todos los alelos evaluados 56

Índice de tablas

4.1. Conjunto de datos de Alelos MHC-II del 2016 (IEDB) 33

5.1. Comparación del método propuesto contra otros métodos midiendo el AUC para el alelo HLA-DP 57
5.2. Comparación del método propuesto contra otros métodos midiendo el AUC para el alelo HLA-DQ 58
5.3. Comparación del método propuesto contra otros métodos midiendo el AUC para el alelo HLA-DR 60
5.4. Comparación del método propuesto contra otros métodos midiendo el AUC para el alelo H-2 61
5.5. Comparación de las diferentes implementaciones prediciendo el alelo H-2-IAb. . 61

Capítulo 1

Introducción

El sistema inmunológico es la defensa natural del cuerpo contra las infecciones. Mediante una serie de procesos biológicos, las células inmunes atacan a los agentes infecciosos que causan daño en nuestro organismo. Estos agentes infecciosos son llamados patógenos. El sistema inmune identifica un patógeno como amenaza por el reconocimiento de pequeños fragmentos llamados antígenos. Algunas células del sistema inmune engullen (células presentadoras de antígenos) a los patógenos y los degradan (incluídos los antígenos), generando pequeños péptidos denominados epítopes. Los epítopes son presentados en la superficie de cada célula presentadora de antígenos mediante moléculas transportadoras. Las moléculas transportadoras responsables de la presentación de antígenos son llamadas Complejo mayor de histocompatibilidad (MHC). Se reconocen tres clases: MHC-I, MHC-II y MHC-III. Esta investigación se enfoca en el MHC-II que presentan epítopes derivados de proteínas extracelulares (patógenos extracelulares) a las células T auxiliares. [2] Muchas veces las uniones entre péptidos y moléculas MHC-II son presentados en la superficie de las células presentadoras de antígenos, pero sólo los péptidos reconocidos por el receptor de las células T (TCR), desencadenarán una respuesta inmune. A estos péptidos se le denominan epítopes de células T. La identificación de epítopes de células T es importante para la comprensión general de la inmunidad y el diseño de diagnósticos, terapias y vacunas. La molécula MHC-II es una glucoproteína heterodimérica que se conforma por una cadena de aminoácidos alfa y otra beta. Los péptidos presentados por la molécula MHC-II se unen a un surco de unión formado por residuos de las cadenas alfa y beta de MHC. El surco de unión a péptidos está abierto en ambos extremos y, por lo tanto, permite la unión de péptidos con diferentes longitudes. [3] La afinidad de unión que tiene una molécula de MHC a un péptido

está determinada casi por completo por la secuencia de aminoácidos del núcleo del péptido. Sin embargo, varias investigaciones demuestran que las regiones flanqueantes de péptidos (PFR) a ambos lados del núcleo del péptido afectan la unión a la molécula MHC-II e influyen en la inmunogenicidad del péptido. [4] [5] Hay varias circunstancias que hacen complejo el proceso de predicción de afinidades de unión de péptidos a la molécula MHC-II; la naturaleza polimórfica de las moléculas MHC-II, las variaciones en la longitud de aminoácidos del péptido, la influencia que tienen las regiones flanqueantes (PFR) sobre el núcleo del péptido y la identificación misma del núcleo de unión correcto del péptido. Debido a estas dificultades la mayoría de los métodos aún tienen un bajo rendimiento predictivo en comparación con los métodos que predicen afinidades de péptidos a moléculas MHC de clase I (MHC-I). En este proyecto se presenta un modelo de red neuronal artificial para predecir la afinidad de péptidos a la molécula MHC-II, entrenados con bases de datos de mediciones cuantitativas de unión a péptidos obtenidos del servidor NetMHCIIPan, mismos que son extraídos de la base de datos y recursos de análisis del epítopo inmune (IEDB).

1.1. Antecedentes

En junio de 1798 se publicó en Inglaterra una obra redactada por el cirujano Edward Jenner (1749-1823) que revolucionó la lucha contra la viruela. Un texto donde plasmó, después de veintiocho años de indagación metódica, una variante en la práctica inoculatoria basada en la observación empírica de que las personas infectadas por viruela desarrolladas en el ganado vacuno, denominado cowpox, se hacían refractarias a la viruela humana. Al método jenneriano se le denominó vacuna, por ello, su descubridor sería reconocido mundialmente como el padre de la vacunación. [6]

A lo largo de la historia de la humanidad se han presentado diferentes tipos de enfermedades que han sido objeto de estudio de la medicina. Estas enfermedades, en su mayoría, han sido de tipo bacterianas y virales, las cuales se han combatido con vacunas, que se desarrollan a partir de algunos fragmentos del patógeno en estudio. Estos fragmentos al mezclarlos con otros compuestos químicos provocan una reacción inmunitaria en el individuo vacunado, estimulando la producción de anticuerpos y el reclutamiento de células inmunes.

El desarrollo de una vacuna puede comenzar con la identificación de componentes o estruc-

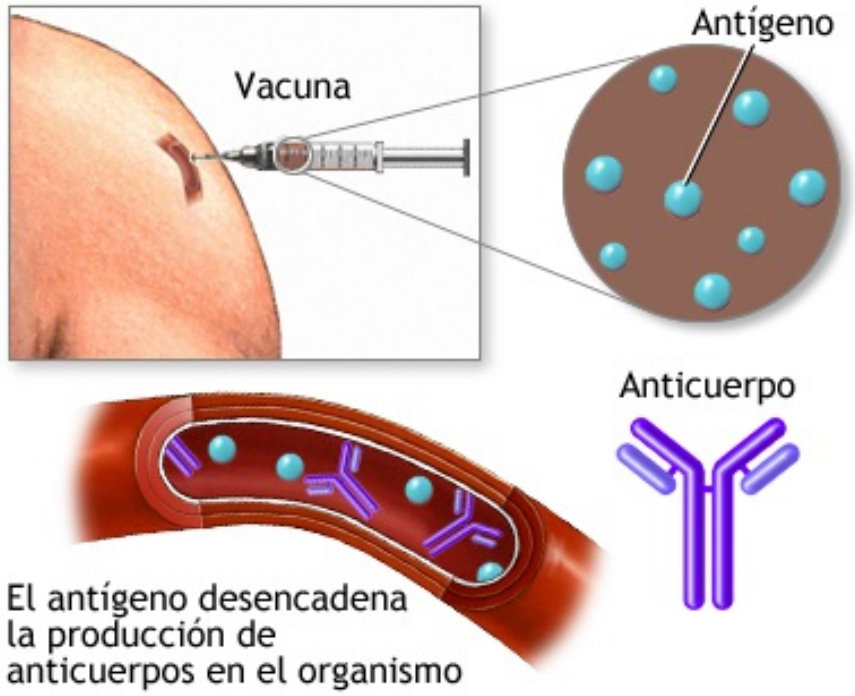


Figura 1.1: Vacuna. [7]

turas únicas en el patógeno capaces de generar una respuesta inmune protectora; Esto es, que sean capaces de activar la respuesta por las células T o B. Con las técnicas convencionales, éste puede ser un proceso largo y tedioso (Figura 1.2), ya que comúnmente se realizan pruebas de ensayo y error hasta encontrar el candidato más adecuado, a pesar de que algunos patógenos pueden ser cultivados en el laboratorio. [8]

Con el avance de las ciencias de la computación y la bioinformática, definida como el uso de técnicas y herramientas computacionales para el análisis de la información biológica, se pueden desarrollar algoritmos que permitan un análisis profundo de la información de bases de datos moleculares y que faciliten el encontrar patrones en estos datos para la creación de nuevas vacunas. No obstante, es difícil que se establezca una buena correlación patógeno-hospedero, debido al limitado conocimiento que pudiera tenerse de los candidatos antígenos. También, el que no se sea capaz de identificar antígenos tales como polisacáridos, que son componentes importantes de muchas vacunas exitosas. Finalmente, este enfoque tiene dificultad en identificar antígenos restringidos CDI (Células dendríticas I), tales como los glicolípidos, los cuales representan futuros candidatos vacunales. [8]

Actualmente existen diversos servidores en Internet que analizan la secuencia de una proteína

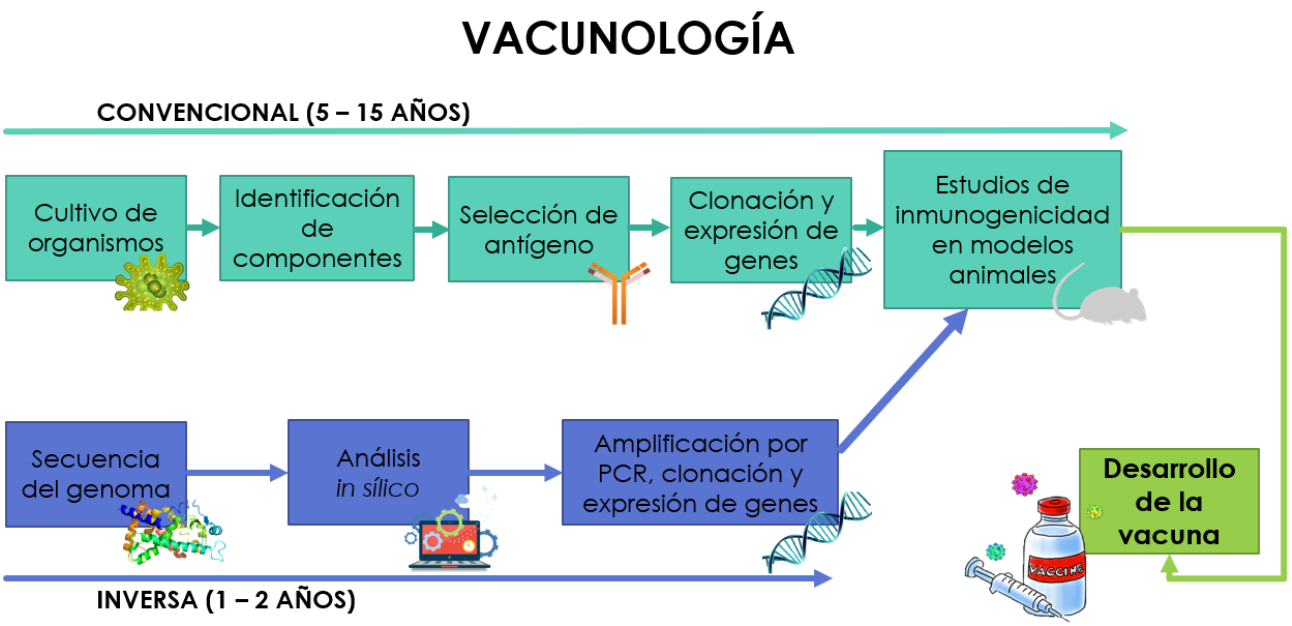


Figura 1.2: Comparación entre la vacunología convencional y la inversa. [8]

de un patógeno en busca de estructuras de aminoácidos (péptidos ó epítopes) que generen una respuesta inmune. Estos sistemas utilizan la información de la base de datos IEDB (Immune Epitope Database and Analysis Resource) para comparar pequeñas partes de la secuencia de la proteína del patógeno con péptidos (epítopes) de los que se tiene conocimiento previo que generan una respuesta inmune en el cuerpo humano. Los péptidos que tienen una fuerte reacción en el sistema inmune son probados en el laboratorio para comprobar su eficiencia (Figura 1.2). Debido a su implementación en la web el uso de estos sistemas requiere total dependencia de Internet.

Existen varios servidores de predicción documentados que hacen uso de diferentes modelos de aprendizaje automático (AA). Entre estos modelos de AA están, las redes neuronales artificiales, los modelos de regresión, máquinas de vectores de soporte, modelos ocultos de markov y árboles de decisión. Los servidores de predicción, así como las ventajas y desventajas de cada uno, se abordarán en el capítulo tercero. Sin embargo en cada uno de los servidores no se especifica si se hace uso de procesamiento en paralelo o procesamiento en alguna GPGPU para la gran cantidad de datos, por lo cual asumimos que el procesamiento es secuencial. Las ventaja que se obtienen al optar por un procesamiento de datos en paralelo es que la cantidad de tiempo de ejecución del algoritmo para generar un modelo de predicción llevará mucho menos tiempo que

si se hace de manera secuencia o en la CPU. Utilizar redes neuronales artificiales tiene como desventaja que podemos predecir solo afinidades de péptidos de longitud fija, sin embargo en la literatura se documenta una técnica para transformar los péptidos de diferentes longitudes a una longitud fija para posteriormente entrenar la red neuronal artificial y generar el modelo de predicción [1].

El presente proyecto tiene como finalidad el desarrollo de un algoritmo utilizando redes neuronales artificiales, que siendo entrenado con las bases de datos de la IEDB sea capaz de encontrar nuevos péptidos (epítopes) entre la secuencia de aminoácidos de una proteína de entrada y los péptidos ya documentados en la IEDB; péptidos que sirvan para generar una respuesta en el sistema inmune. Este algoritmo se implementará en paralelo y reducirá los tiempos en que se tengan los resultados, a su vez, no tendrá dependencia de Internet. Los péptidos que se encuentren a través de este algoritmo, servirán para el desarrollo de nuevas vacunas del Grupo de Inmunología y Vacunología del Centro de Investigaciones Biológicas del Noroeste (CIBNOR).

1.2. Descripción del problema

El desarrollo de vacunas empleando bases de datos moleculares, como la IEDB, y el procesamiento de su información a través de la bioinformática ha implicado un gran avance a los métodos convencionales de la vacunología. No obstante para estos procesos se requieren de diferentes servidores web independientes entre sí, que deben de ser utilizados de forma secuencial con dependencia total de Internet, los cuales a su vez generan resultados partículares del método empleado en cada sistema, resultados que posteriormente deben ser comparados entre sí y unificados, con el fin de realizar pruebas precisas en el laboratorio. El uso de estas herramientas de manera independiente genera una cantidad de tiempo considerable, que podría ser utilizada para otros fines. Por ello, este proyecto se enfocó en implementar un modelo de predicción propio, que con ayuda de las herramientas actuales para procesar gran cantidad de datos con la GPU, así como con los lenguajes de programación web actuales, se pueda desarrollar una aplicación que funcione como un servidor web propio dentro de la institución.

1.3. Objetivos

1.3.1. Objetivo general

Desarrollar un algoritmo para predecir la afinidad de péptidos con el complejo mayor de histocompatibilidad de clase II (MHC-II), utilizando redes neuronales artificiales.

1.3.2. Objetivos específicos

- Analizar la implementación y el funcionamiento de los sistemas predictores que se encuentran en la web.
- Modelar los conjuntos de datos correctos que sirvan para realizar el entrenamiento, validación y pruebas del algoritmo.
- Diseñar y entrenar la red neuronal para predecir la afinidad de péptidos a la molécula MHC-II.
- Comparar los resultados del modelo predictor en RNA con el estado del arte.

1.4. Justificación

Desarrollar un algoritmo que sirva para predecir con suficiente precisión péptidos (epítopes) que se unan a la molécula MHC-II ayudará a simplificar una cantidad relevante de tareas. Actualmente el Centro de Investigaciones Biólogicas del Noroeste S.C, tiene que consultar de diferentes servidores web, péptidos afines a la molécula MHC-II, los cuales varían, ya que cada servidor utiliza su propio algoritmo predictor. El algoritmo que se propone implementar facilitará seguir trabajando en el mismo, ya que se podrá alimentar localmente, es decir, que no necesitará de conexión a Internet y podrá entrenarse con datos locales que hayan sido probados con anterioridad como exitosos en el laboratorio del instituto.

Al ser un algoritmo implementado en Python utilizando TensorFlow sobre GPGPU, nos permitirá procesar una gran cantidad de información en muy poco tiempo, para obtener los mejores péptidos con los que se pueda comprobar en el laboratorio la afinidad de unión a la molécula MHC-II. Ésta última característica de nuestro algoritmo nos permitiría trabajar con

una mayor cantidad de datos para entrenar y probar nuestro algoritmo, a diferencia de como se podría trabajar secuencialmente, lo que probablemente pudiera generar mejores resultados en menor tiempo.

1.5. Alcances y limitaciones

1.5.1. Alcances

- El algoritmo será capaz de proporcionar, en un periodo corto, diferentes péptidos-MHC-II con sus núcleos (9-meros) y valores de afinidad.

1.5.2. Limitaciones

- Las pruebas y resultados solamente se harán con secuencias de aminoácidos de dominio público.
- Debido al tiempo los péptidos que se generen en la predicción serán comparados solamente con los resultados que se encuentran en el estado del arte.

1.6. Hipotesis

El diseño de una red neuronal artificial que sirva para predecir péptidos-MHC-II implementada en Python utilizando TensorFlow sobre GPGPU permitirá procesar una gran cantidad de datos de entrenamiento, obteniendo un modelo de predicción en un periodo corto, haciendo eficiente el procesamiento que conlleva el trabajo para predecir péptidos-MHC-II que puedan ser probados rápidamente *In-Vitro* en el laboratorio.

1.7. Organización del documento

El capítulo segundo de este documento comprende el marco teórico de la investigación, los cuales son fundamentales para el desarrollo y el entendimiento de este proyecto. El capítulo tercero comprende el estado del arte de la tesis siendo éste, las tecnologías, métodos y modelos en los que se ha trabajado hasta ahora para predecir uniones péptido-MHC. En el capítulo cuarto

se aborda la metodología que se usó para implementar el algoritmo, así como la definición de las posibles soluciones para la problemática. El capítulo quinto comprende la implementación del algoritmo, las implicaciones y los retos que se afrontaron teniendo como resultados lo que se sustenta en el capítulo sexto de este documento. Siendo el capítulo sexto el último, se aborda la conclusión de esta investigación y el trabajo futuro con el que se pudiera complementar.

Capítulo 2

Marco Teórico

2.1. Sistema Inmunológico

Se postula que el sistema inmune existiría porque existe la agresión endógena y exógena. Es así que sus principales funciones son la defensa contra microorganismos y la inmunovigilancia contra la emergencia de tumores y de enfermedades autoinmunes y alérgicas. Desde el punto de vista funcional podemos clasificarlo en Sistema Inmune Innato y Adaptativo. Sin embargo, esta clasificación es sólo didáctica, ya que ambos sistemas funcionan de manera integrada. Esto se evidencia de dos formas [9]:

1. El sistema inmune innato activa el sistema inmune adaptativo en respuesta a las infecciones.

2. El sistema inmune adaptativo utiliza los mecanismos efectores de la inmunidad innata para eliminar los patógenos.

El Sistema Inmune Innato (SII) es la primera línea de defensa del hospedero. Posee mecanismos preexistentes que se activan de manera rápida y que preceden a la Inmunidad Adaptativa en la respuesta defensiva. El SII responde de la misma manera frente a diferentes estímulos infecciosos y posee una especificidad limitada, es decir, distingue estructuras comunes a grupos de patógenos pudiendo no distinguir diferencias finas entre ellos. El SII es el más antiguo y está presente en todos organismos multicelulares, incluso plantas e insectos. Los principales componentes del sistema inmune innato son: Barreras físicas y químicas: epitelios, enzimas; Cé-

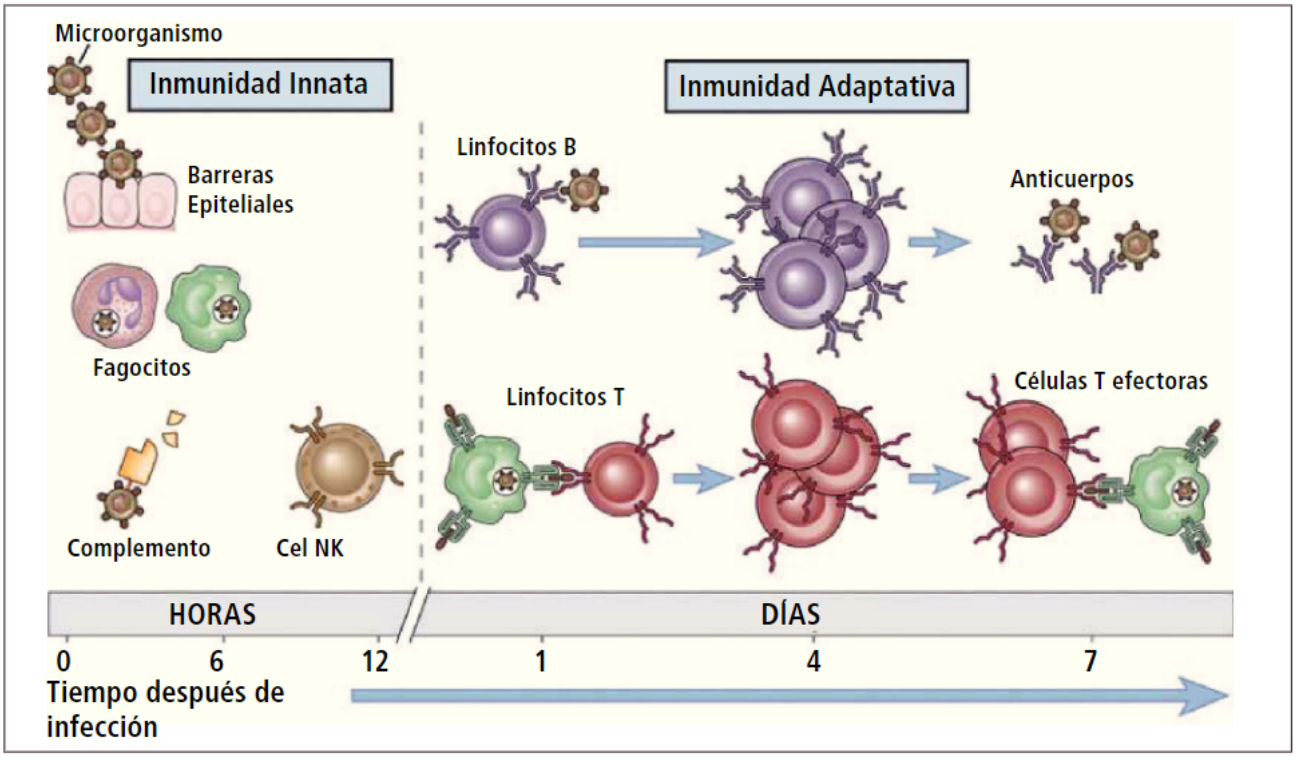


Figura 2.1: Sistema inmune innato y adaptativo [9]

lulas fagocíticas: neutrófilos macrófagos; Células NK (natural killer); Sistema del Complemento; Citoquinas; Receptores tipo Toll.[9]

El Sistema Inmune Adaptivo (SIA) (Fig. 2.2) está presente en los vertebrados, es específico para distintas moléculas y se caracteriza por mejorar la capacidad defensiva frente exposiciones sucesivas. Los principales elementos del SIA son los linfocitos B y T que se activan frente a los antígenos (sustancias que inducen respuestas inmunes específicas). El SIA posee dos tipos de respuestas inmunes: Inmunidad Humoral e Inmunidad Celular. Estas inmunidades poseen memoria, es decir, es recordada por el sistema inmune, siendo más eficaz y rápida frente a reexposiciones al mismo patógeno. Además esta respuesta es específica a cada patógeno o molécula (antígeno del patógeno), siendo capaz de distinguir de manera muy fina las características de éstas. Esta capacidad de reconocimiento específico de cada antígeno, por parte de los linfocitos, se debe a la expresión de receptores a los diferentes antígenos foráneos existentes y se denomina repertorio de linfocitos, el cual es extremadamente amplio.[9]

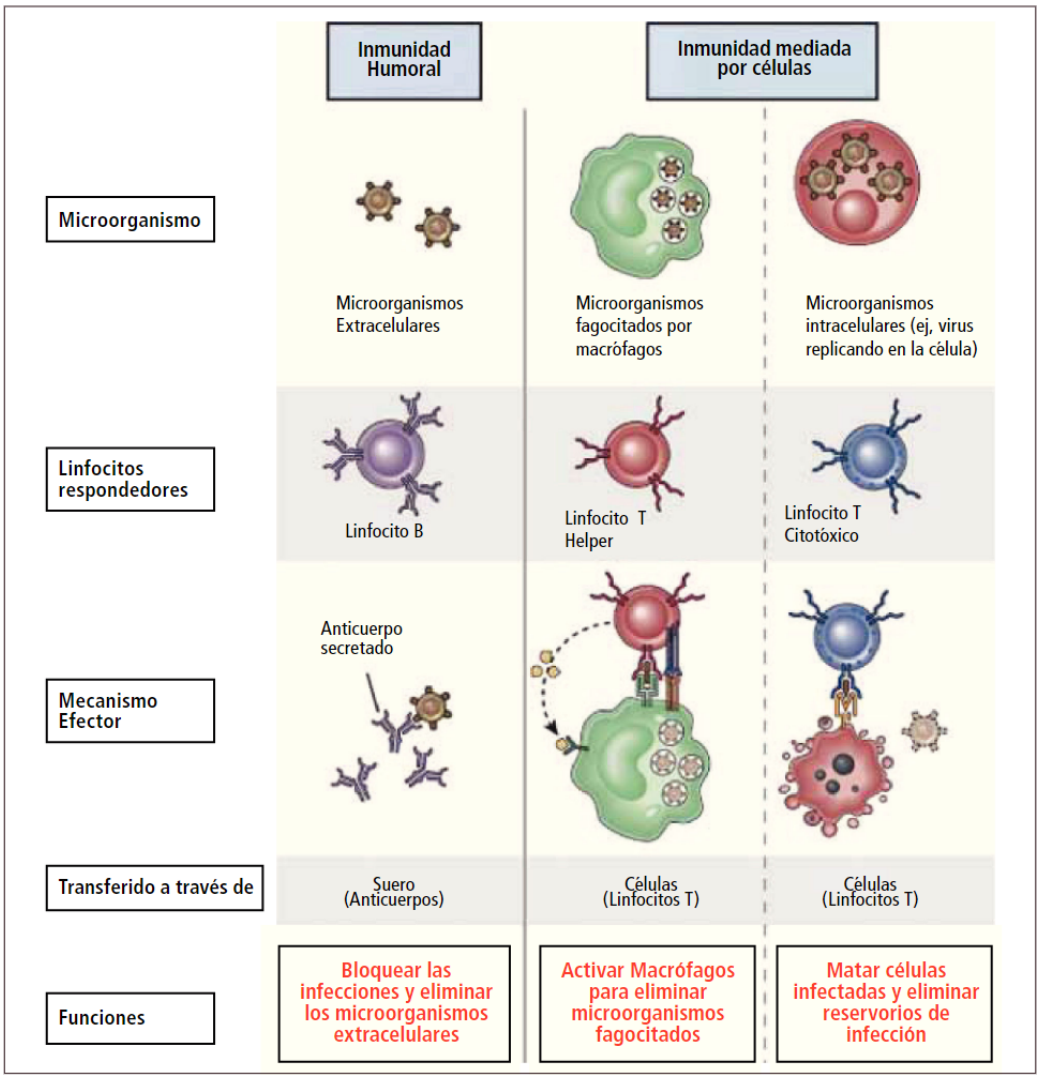


Figura 2.2: Sistema inmune adaptativo [9]

2.2. Vacunología Inversa

La vacunología es la ciencia que combina el estudio del sistema inmune con la diversidad antigénica de los patógenos y aquellas formulaciones que pueden modularlo para prevenir o curar enfermedades.[10] Gracias al desarrollo y optimización de nuevas tecnologías en el Proyecto Genoma Humano se logró la completa automatización de la técnica de secuenciación del ADN, lo que conllevó a completar el conocimiento de los genomas de numerosos microorganismos. Ello generó una gran cantidad de datos y la Bioinformática, definida como el uso de técnicas y herramientas computacionales para el análisis de la información biológica, entró a jugar un papel muy importante. El auge de la informática, junto con el establecimiento de la Internet y de las

bases de datos moleculares, han posibilitado en estos últimos años, grandes avances en el estudio de la información genética y ha permitido el desarrollo de programas que aceleran el estudio de numerosas variables, permitiendo el análisis de genomas en tiempos cortos. Así, este nuevo enfoque denominado por Rappuoli (2000)[11] como vacunología inversa, cambió completamente la forma en la que se venía realizando el diseño de vacunas.[8]

2.3. Complejo mayor de histocompatibilidad

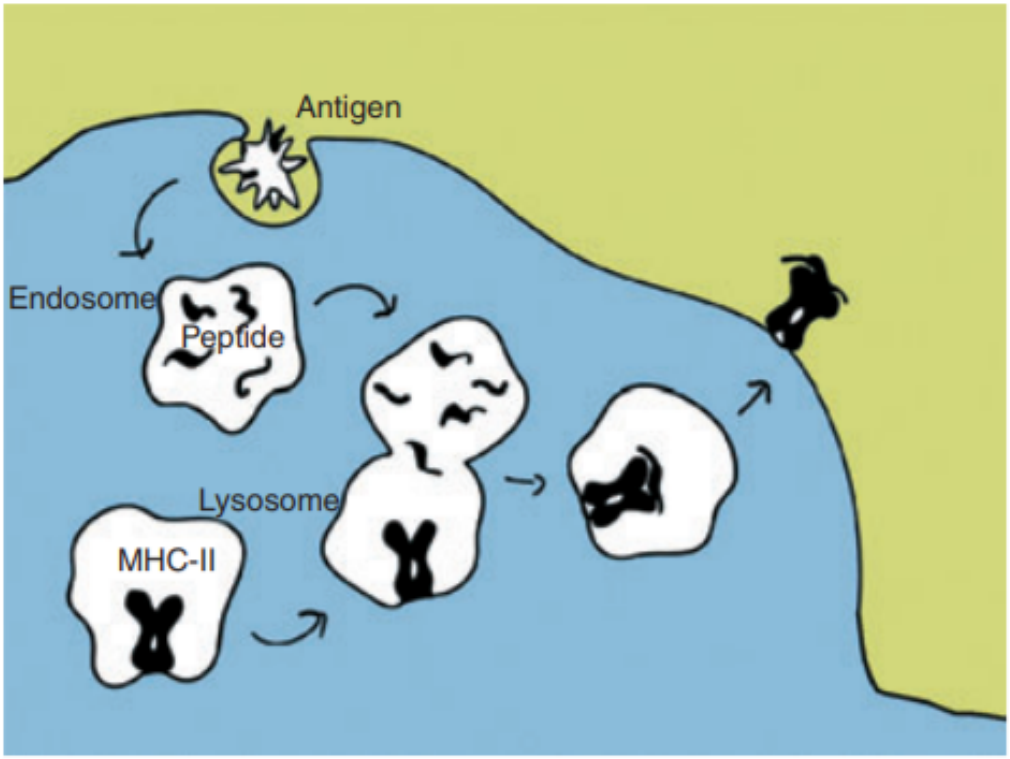


Figura 2.3: Complejo mayor de histocompatibilidad de Clase II (MHC-II)[12]

Las moléculas del complejo de histocompatibilidad principal (MHC) desempeñan un papel esencial en las interacciones hospedero-patógeno que determinan el resultado de muchas respuestas inmunitarias del hospedero. Solo una pequeña fracción de los posibles péptidos que pueden generarse a partir de proteínas de organismos patógenos en realidad generan una respuesta inmune. Las moléculas MHC de clase II presentan péptidos derivados de proteínas extraídas del entorno extracelular (Fig. 2.3). Estimulan la inmunidad celular y humoral contra microorganismos patógenos a través de las acciones de los linfocitos T auxiliares. Para que un péptido

estimule la respuesta de los linfocitos T auxiliares, debe unirse al MHC de clase II en los orgánulos endocíticos.[12]

Las células presentadoras de antígenos especializadas ingieren antígenos exógenos en los endosomas por endocitosis o fagocitosis. Los endosomas se fusionan con el MHC de clase II que contiene lisosomas. Los antígenos (y la cadena invariante de MHC de clase II) se degradan en péptidos mediante proteasas, y la liberación del péptido CLIP permite que la molécula de MHC de clase II se una a los péptidos antigénicos antes de la migración a la membrana plasmática. [12] Tal como se aprecia en la Figura 2.3.

2.4. Patógeno

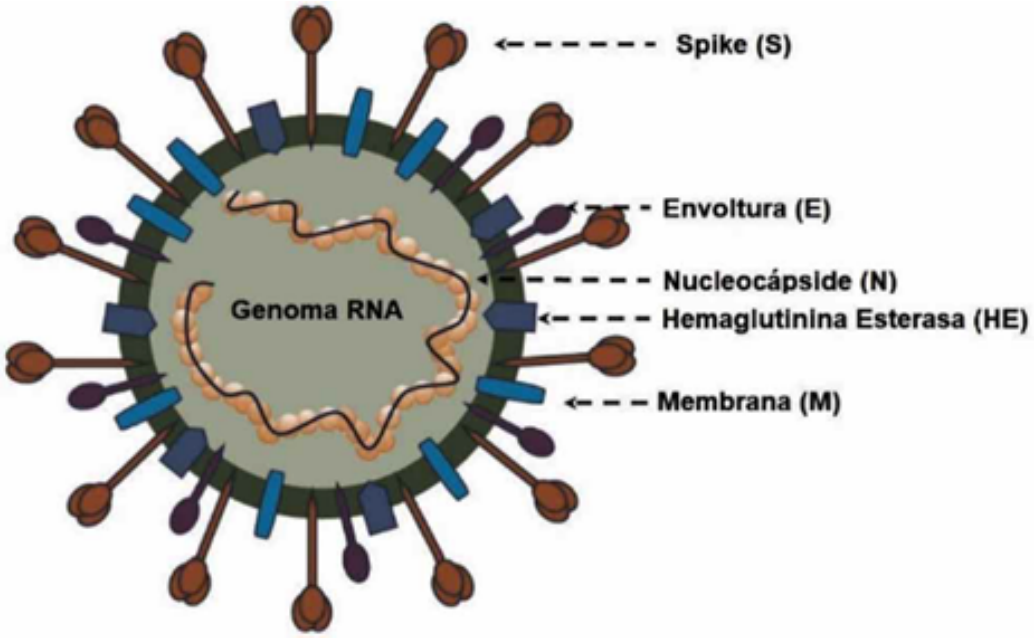


Figura 2.4: Estructura del SARS-CoV-2. [13]

Un patógeno es un agente que causa infección en un huésped vivo. Actúa como parásito dentro del huésped, ya sean individuos o células, y trastorna las actividades fisiológicas normales.[14] Los microorganismos que generalmente se catalogan como patógenos pueden ser bacterias, protozoos, hongos, virus, viroides y priones. Los patógenos, como se muestra en la figura 2.4, están compuestos por diferentes partes que varían dependiendo su tipo. Estas partes pueden estar formadas por carbohidratos, lípidos, proteínas, entre otros.

2.5. Antígeno

Cualquier sustancia capaz de provocar una respuesta inmune adaptativa se refiere como un antígeno. Los antígenos son componentes estructurales o funcionales de los patógenos. La mayor parte de lo que sabemos proviene de estudios de inmunización (p.ej. inyección) de animales de laboratorio (generalmente un ratón) o humanos para que respondan a una molécula extraña, como una proteína de un patógeno. Generalmente los antígenos se administran con inmunoestimulantes (generalmente de origen microbiano) llamados adyuvantes, que activan el sistema inmune innato.[15] En resumen, un antígeno es cualquier parte del patógeno que logre generar una respuesta en el sistema inmune. Pueden ser proteínas, carbohidratos, lípidos o cualquier componente del patógeno. En la figura 2.4 un fragmento de la membrana del SARS-CoV-2 podría convertirse en un antígeno.

2.6. Células B

Las respuestas inmunitarias adaptativas se llevan a cabo por glóbulos blancos llamados linfocitos. Existen dos clases amplias de estas respuestas: las respuestas de anticuerpos y las respuestas inmunitarias mediadas por células, y se llevan a cabo por diferentes clases de linfocitos, llamados células B y células T (Fig. 2.2). En las respuestas de anticuerpos, las células B se activan para secretar anticuerpos, que son proteínas llamadas inmunoglobulinas. Los anticuerpos circulan en el torrente sanguíneo e impregnan los demás fluidos corporales, donde se unen específicamente al antígeno extraño que estimuló su producción. La unión de anticuerpos inactiva virus y toxinas microbianas (como la toxina del tétanos o la toxina difteria) al bloquear su capacidad de unirse a los receptores de las células del hospedero. La unión de anticuerpos también marca patógenos invasores para la destrucción, principalmente facilitando que las células fagocíticas del sistema inmunitario innato las ingiera.[15]

2.7. Células T

En las respuestas inmunitarias mediadas por células,la segunda clase de respuesta inmune adaptativa, las células T activadas reaccionan directamente contra un antígeno extraño que se les presenta en la superficie de una célula hospedora. La célula T, por ejemplo, podría matar a una

célula hospedora infectada por el virus que tiene antígenos virales en su superficie, eliminando así la célula infectada antes de que el virus haya tenido la oportunidad de replicarse (Fig. 2.2). En otros casos, la célula T produce moléculas de señal que activan los macrófagos para destruir los microbios invasores que han fagocitado.[15]

2.8. Aminoácidos y proteínas

Los aminoácidos son un conjunto de 20 tipos distintos de moléculas y constituyen las piezas básicas para construir proteínas. Las proteínas constan de una o más cadenas de aminoácidos; estas cadenas se llaman polipéptidos. La secuencia de la cadena de aminoácidos determinará cómo se pliega tridimensionalmente el polipéptido, pues la forma que adquiera es muy importante para que sea biológicamente activo. De forma general, la secuencia de aminoácidos que forma una proteína está codificada en un gen. [16]

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

Figura 2.5: Lista de aminoácidos proteicos. [16]

2.9. Péptido (epítope)

Un péptido es una cadena corta de aminoácidos. Los aminoácidos de un péptido están conectados entre sí en una secuencia mediante enlaces llamados enlaces peptídicos.

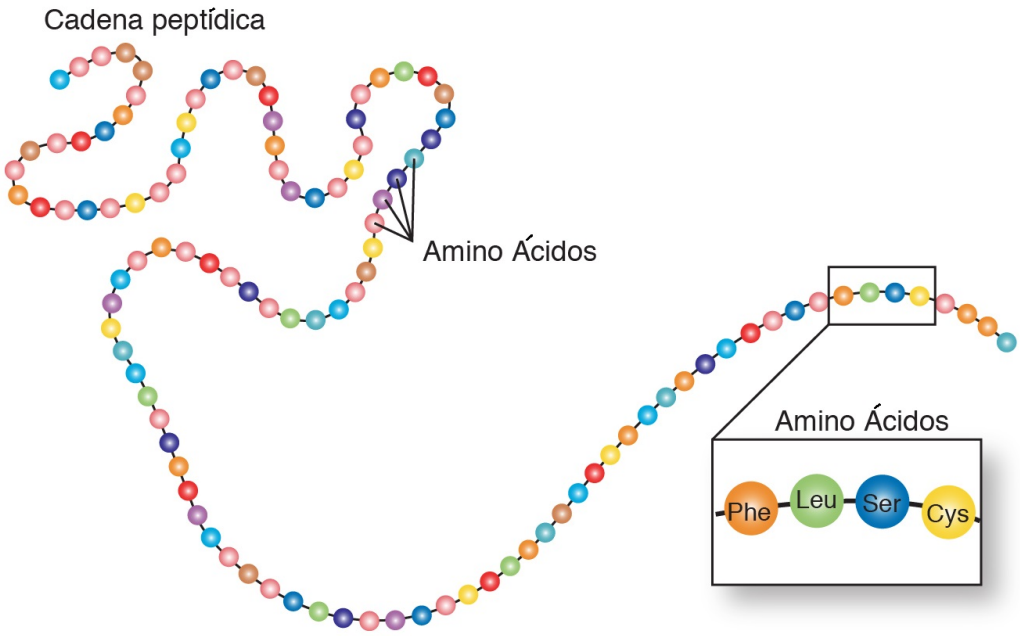


Figura 2.6: En la ilustración se visualiza una cadena larga de aminoácidos que podría ser una proteína, ésta puede contener muchos péptidos y está formada por aminoácidos [16].

Típicamente, los péptidos se distinguen de las proteínas por su longitud más corta, aunque el número de aminoácidos para definir un péptido y proteína puede ser arbitrario. Los péptidos generalmente se consideran cadenas cortas de dos o más aminoácidos. Mientras tanto, las proteínas son moléculas largas compuestas de múltiples subunidades de péptidos, y también se conocen como polipéptidos. Las proteínas pueden ser digeridas por enzimas (otras proteínas) en fragmentos de péptidos cortos. Entre las células, los péptidos pueden realizar funciones biológicas. Por ejemplo, algunos péptidos actúan como hormonas, que son moléculas que cuando se liberan de las células afectan a otras áreas del cuerpo.[17] En inmunología los péptidos son llamados epítopes. El epítope es la parte de un antígeno que es reconocido por el sistema inmunitario, específicamente por anticuerpos , células B, o células T. El epítopo es la pieza específica del antígeno al que se une un anticuerpo.

2.10. Alelo

Un alelo es cada una de las dos o más versiones de un gen. Un individuo hereda dos alelos para cada gen, uno del padre y el otro de la madre. Los alelos se encuentran en la misma posición dentro de los cromosomas homólogos. Si los dos alelos son idénticos, el individuo es homocigoto para este gen. En cambio, si los alelos son diferentes, el individuo es heterocigoto para este gen. Aunque el término alelo fue usado originariamente para describir variaciones entre los genes, ahora también se refiere a las variaciones en secuencias de ADN no codificante (es decir, que no se expresan).[18]

Las bases de datos de predicción de péptidos-MHC-II están constituidas por diferentes secuencias de péptidos de diferentes alelos. Los alelos que se manejan actualmente en el estado del arte son tres tipos de alelos de humanos HLA: HLA-DP y HLA-DQ que están constituidos por una secuencia péptido alfa y otra beta; HLA-DRB del cual solo se toma la secuencia beta del péptido para este tipo de estudio. También se estudian varios tipos de alelos de ratón H-2.

2.11. Secuencias k-mero

En bioinformática, los k -meros son subsecuencias de longitud K contenida dentro de una secuencia biológica. Se utiliza principalmente en el contexto de genómica computacional y análisis de secuencias, en las K se componen de nucleótidos (es decir: A, T, G y C). Las secuencias K-meros se utilizan para ensamblar secuencias de ADN, mejorar expresión de genes, identificar especies en muestras metagenómicas, y crear vacunas atenuadas. [19]
Para esta investigación se tratarán como *n-meros* (término en inglés *n-mer*), tal y como se utiliza en el estado del arte por ser secuencias de aminoácidos proteicos.

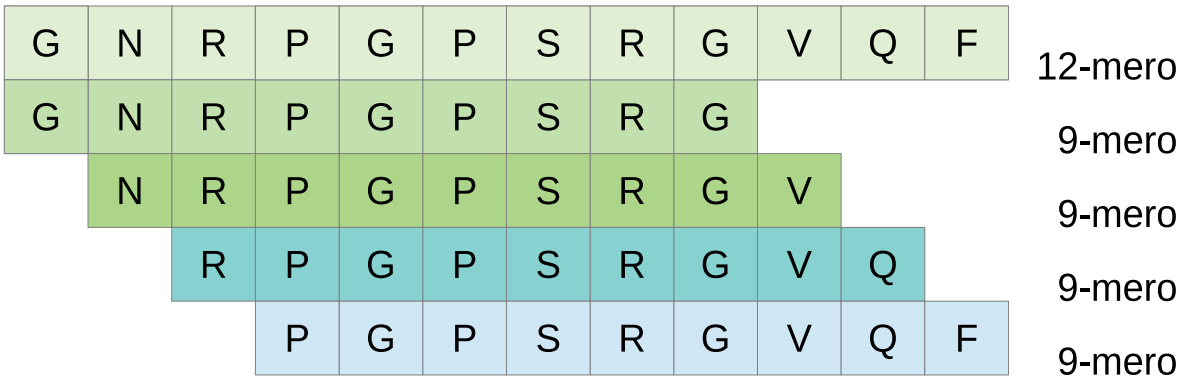


Figura 2.7: Núcleos de unión (9-mero) de un péptido 12-mero. El número previo al sufijo -mero denota el valor K de aminoácidos.

2.12. Concentración inhibitoria IC50

La concentración inhibidora semimáxima (IC50) es una medida de la potencia de una sustancia en la inhibición de una función biológica o bioquímica específica. IC50 es una medida cuantitativa que indica la cantidad de una determinada sustancia inhibidora (por ejemplo drogas) que se necesita para inhibir, in vitro, un proceso biológico dado o componente biológico en un 50 %. El componente biológico podría ser una enzima, una célula, un receptor celular o un microorganismo. Los valores de IC50 se expresan típicamente como concentración molar en unidad Nanomolar (nM). [20]

Todos los valores de afinidad de péptidos-MHC-II están medidos en IC50 y es la unidad de medida para clasificar péptidos afines y no afines.

2.13. Alineamiento de secuencias

Un alineamiento de secuencias en bioinformática es una forma de representar y comparar dos o más secuencias o cadenas de ADN, ARN, o estructuras primarias proteicas para resaltar sus zonas de similitud, que podrían indicar relaciones funcionales o evolutivas entre los genes o proteínas consultados. Las secuencias alineadas se escriben con las letras (representando aminoácidos o nucleótidos) en filas de una matriz en las que, si es necesario, se insertan espacios para que las zonas con idéntica o similar estructura se alineen. [21]

Diversos métodos de predicción de secuencias de péptidos-MHC-II a la vez de predecir los núcleos péptidos de una base de datos de entrenamiento generan un alineamiento de esas mismas

secuencias, pudiendo hacer notar la similitud entre secuencias que tiene el conjunto de datos. El método de alineamiento de secuencias por estabilización de matrices (SMM-Align) o el alineamiento por redes neuronales (NN-Align) generan este tipo de alineamientos en donde se puede estudiar más a fondo las secuencias de péptidos en estudio.

2.14. Redes neuronales artificiales

En la última década las Redes Neuronales Artificiales (RNA) han recibido un interés particular como una tecnología para minería de datos, puesto que ofrece los medios para modelar de manera efectiva y eficiente problemas grandes y complejos. Los modelos de RNA son dirigidos a partir de los datos, es decir, son capaces de encontrar relaciones (patrones) de forma inductiva por medio de los algoritmos de aprendizaje basado en los datos existentes más que requerir la ayuda de un modelador para especificar la forma funcional y sus interacciones. Las RNA son un método de resolver problemas, de forma individual o combinadas con otros métodos, para aquellas tareas de clasificación, identificación, diagnóstico, optimización o predicción en las que el balance datos/conocimiento se inclina hacia los datos y donde, adicionalmente, puede haber la necesidad de aprendizaje en tiempo de ejecución y de cierta tolerancia a fallos. En estos casos las RNAs se adaptan dinámicamente reajustando constantemente los "pesos" de sus interconexiones (Fig. 2.8).

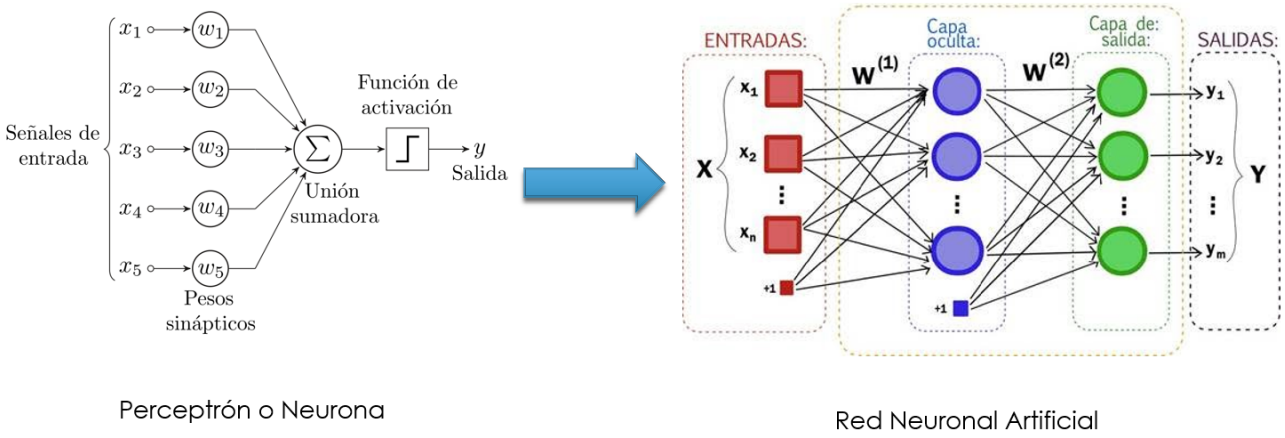


Figura 2.8: Composición de una red neuronal artificial.

Las RNA se basan en la analogía que existe en el comportamiento y función del cerebro

humano, en particular del sistema nervioso, el cual está compuesto por redes de neuronas biológicas que poseen bajas capacidades de procesamiento, sin embargo toda su capacidad cognitiva se sustenta en la conectividad de éstas. La unidad básica que compone una red neuronal artificial es un procesador elemental llamado neurona que posee la capacidad limitada de calcular, en general, una suma ponderada de sus entradas y luego le aplica una función de activación para obtener una señal que será transmitida a la próxima neurona. Estas neuronas artificiales se agrupan en capas o niveles y poseen un alto grado de conectividad entre ellas, conectividad que es ponderada por los pesos. A través de un algoritmo de aprendizaje supervisado o no supervisado, las RNA ajustan su arquitectura y parámetros de manera de poder minimizar alguna función de error que indique el grado de ajuste a los datos y la capacidad de generalización de las RNA.[22]

2.15. Sistema operativo Ubuntu

Ubuntu es un sistema operativo de software libre y código abierto, basado en la distribución GNU/Linux Debian. Es una de las distribuciones más utilizadas por usuarios comunes, ya que es fácil de usar y administrar. Ubuntu se puede instalar en computadoras de escritorio y servidores. En la comunidad científica es muy utilizado respecto a la cantidad de recursos consumidos por el sistema operativo, ya que es poca comparada con otros sistemas operativos, lo que permite que la memoria libre se asigne eficientemente para las tareas de cómputo científico. Otro punto a favor de ubuntu es la gran cantidad de repositorios que se encuentran disponibles para tareas de aprendizaje automático, machine learning, inteligencia artificial y otros tipos de computación científica. Ubuntu es un buen sistema operativo para diseñar y desarrollar sistemas de propósito científico.

2.16. Lenguaje Python

Python es un lenguaje de programación interpretado, multiparadigma, que soporta orientación a objetos, programación imperativa y programación funcional. Fue creado a finales de los ochenta, por Guido Van Rossum en el Centro para las Matemáticas y la Informática (CWI, Centrum Wiskunde & Informática) en los Países Bajos, como sucesor del lenguaje de progra-

mación ABC, capaz de manejar excepciones e interactuar con el sistema operativo Amoeba.[23] Es un lenguaje de programación conocido por el poco tiempo que se le invierte al aprenderlo,

Figura 2.9: Numpy, scipy, pandas, matplotlib y scikit learn son algunas de las bibliotecas para aprendizaje automático en Python. [24]

creciendo su uso durante los últimos años y colocándolo entre los lenguajes de programación más utilizados.

2.17. Lenguaje C

El lenguaje de programación C se ideó a principios de la década de 1970 como un lenguaje de implementación del sistema para el naciente sistema operativo Unix. Derivado del lenguaje sin tipo BCPL, desarrolló una estructura tipográfica; creado en una pequeña máquina como herramienta para mejorar un entorno de programación reducido, se ha convertido en uno de los lenguajes dominantes de la actualidad.[25]

C es un lenguaje que trabaja directamente con la memoria lo cual hace que el tiempo de ejecución sea el mínimo, sin embargo, el programador tiene que conocimiento en el manejo correcto de la memoria y las buenas prácticas utilizando C. C dispone de diferentes versiones de compiladores con gcc, para esta investigación se usó la versión 11 del compilador, abreviado c11.

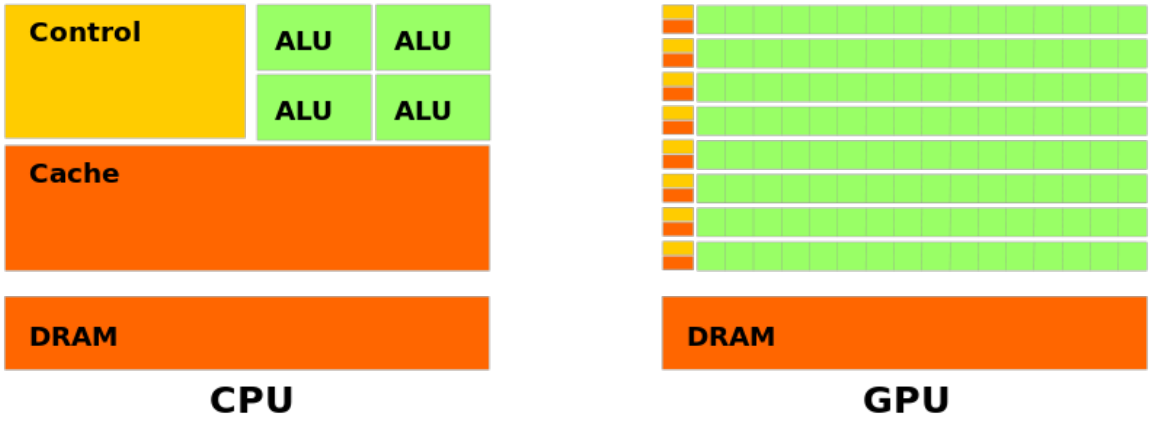


Figura 2.10: Diferencia entre la arquitectura de un CPU y una GPU. [26]

2.18. Plataforma CUDA

CUDA es un modelo de computación paralela y programación inventado por NVIDIA. Permite aumentos drásticos en el rendimiento informático al aprovechar la potencia de la unidad de procesamiento de gráficos (GPU) [27]. CUDA tiene un lenguaje de programación basado en C y C++ para codificar algoritmos en las GPU de NVIDIA, el cual se puede sustituir por lenguajes-como python fortran y Java usando wrappers. Las GPU compatibles con CUDA tienen cientos de núcleos que pueden ejecutar colectivamente miles de subprocesos informáticos. Estos núcleos tienen recursos compartidos que incluyen un archivo de registro y una memoria compartida. La memoria compartida en el chip permite que las tareas paralelas que se ejecutan en estos núcleos compartan datos sin enviarlos a través del bus de memoria del sistema. CUDA ha sido en la actualidad muy utilizado para ejecutar programas que requieren procesar una gran cantidad de datos o algoritmos que ejecutan tareas complejas, disminuyendo considerablemente el tiempo de ejecución que normalmente tardaría en una CPU.

2.19. TensorFlow

TensorFlow es una plataforma de código abierto de extremo a extremo para el aprendizaje automático. Cuenta con un ecosistema integral y flexible de herramientas, bibliotecas y recursos comunitarios que permite a los investigadores impulsar lo más avanzado en Machine Learning (ML) y a los desarrolladores crear y desplegar fácilmente aplicaciones con tecnología ML. [28]

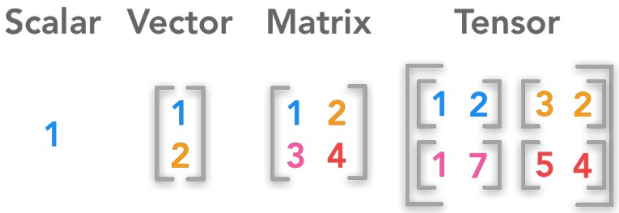


Figura 2.11: Diferencia entre un escalar, vector, matriz y tensor. [31]

TensorFlow fue desarrollado por Google para satisfacer sus necesidades de sistemas capaces de construir y entrenar redes neuronales para detectar y descifrar patrones y correlaciones, análogos al aprendizaje y razonamiento usados por los humanos [29]. TensorFlow puede correr en múltiple CPUs y GPUs (con extensiones opcionales de CUDA para informática de propósito general en unidades de procesamiento gráfico) [30]. El nombre TensorFlow deriva de las operaciones que tales redes neuronales realizan sobre arrays multidimensionales de datos. Estos arrays multidimensionales son referidos como “tensores”. En una primera versión los computos de TensorFlow se expresaban como stateful dataflow graphs, sin embargo, con la evolución de la biblioteca la forma de codificar estas redes se ha vuelto más imperativa [31].

Capítulo 3

Estado del Arte

3.1. Técnicas de predicción

Durante la investigación para encontrar nuevas técnicas de predicción de unión de péptidos más precisas se han utilizado varios modelos de aprendizaje automático, tal como, redes neuronales artificiales (RNA), modelos de regresión, máquinas de vectores de soporte (SVM), modelos ocultos de markov (HMM), arboles de decisiones y métodos de consenso. Éste último modelo combina varios de los primeros. En esta sección se abordará rapidamente la aplicación de las técnicas de aprendizaje automático utilizadas para la predicción de unión de péptidos que se han encontrado en el estado del arte.

3.2. Redes Neuronales Artificiales (RNA)

Desde la primera aplicación de este modelo para la predicción de unión de péptidos en el artículo Prediction of binding to MHC Class I Molecules [32], se ha utilizado para predecir la unión de péptidos para un gran número de alelos MHC. Para generar un modelo de RNA, las secuencias de péptidos se transforman en descriptores numéricos que luego alimentan a varias capas de neuronas artificiales. El valor de cada neurona artificial se deduce de la capa anterior mediante fórmulas matemáticas y se calcula un valor de predicción final. Los parámetros dentro de las fórmulas se determinan durante el proceso de entrenamiento mediante retropropagación o algún otro proceso de cálculo de gradiente. Varios artículos y servidores han implementado redes neuronales artificiales como IEDB [33], NetMHC/NetMHCII [1] y NetMHCpan/NetMHCIIpan

[34].

Las RNA se pueden utilizar para realizar predicciones cualitativas y cuantitativas para ambas clases de MHC. Se han logrado rendimientos confiables pero todavía se encuentra en desarrollo y mejora en muchos servidores como IEDB y NetMHCpan/NetMHCIIpan. Un problema con este modelo es que requieren un número fijo de neuronas de entrada, por lo tanto, los péptidos de varias longitudes necesitan ser procesados con procesos adicionales para obtener una secuencia de longitud fija.

3.3. Modelos de Regresión

Se han desarrollado modelos de regresión para las predicciones cuantitativas para la unión de péptidos. MHCPRED [35] implementó un módelo de regresión de relación de actividad de estructura cuantitativa (QSAR, por sus siglas en inglés, Quantitative Structure Activity Relationship regresion) escrito en Perl, para predecir la afinidad de unión de péptidos MHC. En este método, las contribuciones de aminoácidos individuales en cada posición se calculan utilizando mínimos cuadrados parciales. SVRMHC [36] utilizó la regresión de vectores de soporte (SVR) para predicciones cuantitativas.

Los modelos de regresión tienen la ventaja de realizar predicciones cuantitativas que no solo identifican si un péptido es un afín o no hacia un MHC, sino que también indican qué tan fuerte es la unión. Sin embargo, hacer una predicción cuantitativa puede tener un grado de complejidad mayor que una predicción cualitativa.

3.4. Máquinas de Vectores Soporte (MVS)

Las MVS crean un hiperplano en el espacio de alta dimensión de los datos de entrenamiento para clasificarlos en diferentes grupos. Éstas han sido utilizadas por algunos servidores para la predicción de unión de péptidos MHC, incluídos MHC2PRED [37], SVMHC [38] y POPI [39]. La alterna cuantitativa de MVS fue implementada por SVRMHC como se mencionó en el apartado de Modelos de Regresión.

Las MVS generalmente logran una alta precisión. Igualmente como las RNA, requieren de una dimensión o longitud fija de datos de entrada, lo que limita su aplicabilidad. Sin embargo,

Salomón y Flower propusieron un método de kernel derivado de SVM que puede procesar péptidos con varias longitudes usando puntajes de similitud [40].

3.5. Modelo Oculto de Markov (HMM)

Los modelos ocultos de markov son ampliamente utilizados en el reconocimiento de patrones, los cuales también ha sido probados para las predicciones de unión de péptidos MHC. En un HMM, un péptido se convierte en diferentes estados para diferentes posiciones. En cada posición o estado, se calculan las probabilidades de los aminoácidos y se da un valor final combinando todas las probabilidades. Hay diferentes formas de modelar un HMM, tal como, los HMM completamente conectados [41], los HMM optimizados con algoritmos de división de estado sucesivo [42] y los HMM de perfil [43], que pueden fusionar patrones superpuestos.

Los HMM tienen la ventaja de procesar péptidos con varias longitudes, sin embargo, se deben desarrollar diferentes HMM por separado para afines y no afines. Debido a la separación de modelos, los HMM se han desarrollado solo para un número limitado de alelos MHC.

3.6. Árbol de Decisión

Los árboles de decisión son un grupo de estructura de árbol construída a partir de muestras de entrenamiento. Las reglas de división se determinan a partir del proceso de formación. Cuando llega una nueva muestra, se somete a una estructura similar a un diagrama de flujo y finalmente alcanza una predicción de clasificación. Se utilizó inicialmente para hacer predicciones para HLA-A*02:01 en 1999 [44].

Los árboles de decisión son más fáciles de interpretar que las RNA debido a su naturaleza basada en reglas. Aunque este método se usa ampliamente en el campo del aprendizaje automático, está menos implementado para predecir la unión de péptidos MHC.

3.7. Métodos de Consenso

Un método de consenso utiliza una combinación entre las predicciones de los modelos internos que la componen, es decir, está compuesto de dos o más modelos de predicción. Cada uno de

los modelos que la componen hacen primero una predicción por separado y luego se hace la predicción final considerando todas las predicciones individuales. En los servidores de IEDB la predicción se realiza a partir de cuatro modelos basados en PSSM (Position-Specific Scoring Matrix) para MHC de clase I, mientras que para los MHC de clase II, el resultado proviene de nueve modelos, incluidos varios modelos en PSSM y modelos de aprendizaje automático [45]. Al predecir una unión MHC-péptido, no todos los modelos pueden devolver un valor, sin embargo, si tres o más modelos proporcionan predicciones, se seleccionan los tres modelos con mejor rendimiento y se utiliza el valor de la mediana como la puntuación de consenso final.

Los métodos de consenso generalmente superan a los modelos individuales, ya que preferiblemente puede seleccionar los métodos de mayor rendimiento de las pruebas de entrenamiento. Sin embargo, la aplicabilidad del método de consenso está limitada por sus componentes individuales, ya que requiere resultados simultáneamente de esos modelos.

Capítulo 4

Metodología

4.1. Introducción

Se pretende alcanzar el objetivo mediante la aplicación de las siguientes fases.

1. Instalación del ambiente de trabajo.
2. Análisis y recopilación de datos.
3. Diseño del algoritmo.
4. Implementación de la red neuronal artificial.
5. Pruebas y documentación de resultados.

La primera fase comprende la instalación de las herramientas necesarias para llevar a cabo el procesamiento de los datos y el diseño e implementación del algoritmo, en ella se procederá a instalar el sistema operativo necesario, así como el lenguaje de programación y bibliotecas de código necesarias para la implementación de la red neuronal artificial.

En la fase de análisis y recopilación de datos se buscarán y recopilarán los datos de secuencias de aminoácidos que se encuentran en las fuentes de dominio público, como el servidor NetMHCIIPan y la IEDB, para armar los conjunto de datos que se necesitarán en la fase de implementación del algoritmo. También se analizarán los algoritmos que se encuentren en el estado del arte.

La fase del diseño del algoritmo comprenderá diseñar las distintas formas en que puede desarrollarse el algoritmo para resolver de una manera eficiente el problema, optimizando código y tiempo de ejecución.

En la fase de implementación se escribirá el código fuente del algoritmo, así como probar bibliotecas de desarrollo que nos faciliten algunas operaciones que se requieren hacer en el algoritmo, como la multiplicación de matrices, obtención de funciones matemáticas, etc. En esta fase también se implementará una red neuronal artificial en el lenguaje Python haciendo uso de la biblioteca TensorFlow para posteriormente comparar la eficiencia computacional en tiempo frente al lenguaje C.

En la fase final que comprende las pruebas y documentación de resultados, se escribirán todas las pruebas que se hicieron para encontrar los péptidos con afinidad de unión a la molécula MHC-II, así como la solución de problemas que se pudieran presentar durante la fase de implementación.

4.2. Instalación del ambiente de trabajo

4.2.1. Instalación de GNU/Linux Ubuntu 18.04

Para la elección de un sistema operativo sobre el cual tener accesibilidad a las herramientas necesarias para llevar acabo el diseño e implementación de una red neuronal artificial que resuelva la problemática de predecir afinidades de unión de péptidos con la molécula MHC-II, se opta por la instalación de una distribución linux que albergue la mayor cantidad posible de bibliotecas de código que estén enfocadas en el ámbito científico para, poder procesar cantidades de datos grandes, tener la facilidad de desarrollar algoritmos de inteligencia artificial y Machine Learning, y manejar de una manera eficiente los recursos del equipo de cómputo donde se realicen las pruebas de la investigación, esto es, donde se ejecute el algoritmo que se va a implementar. El candidato más adecuado a esta serie de requerimientos es GNU/Linux Ubuntu en su versión LTS 18.04.

La instalación de GNU/Linux Ubuntu 18.04 se hace mediante una USB/CD de arranque que contenga el instalador del sistema operativo en cuestión. El archivo ISO del sistema operativo se puede obtener mediante la siguiente liga: https://releases.ubuntu.com/18.04/. Se elige la versión

Desktop Image para tener una versión de escritorio y se realiza una instalación de tipo *Normal* seleccionando que sea una instalación con las actualizaciones más recientes para poder tener acceso al interprete de Python y los compiladores de GCC que se requerirán posteriormente para ejecutar el código fuente del algoritmo.

4.2.2. Instalación de Python 3.6

El sistema operativo Ubuntu 18.04 ya viene con Python 2.7 y los repositorios para actualizar a la versión más reciente de Python. Para compilar TensorFlow para GPU en su versión más reciente y tener la paquetería numpy actualizada, se necesita tener instalado Python 3.6, por lo cual procedemos a instalarlo, realizando el siguiente proceso.

Primero se necesita actualizar la lista de paquetes, para lo cual ingresamos en consola el siguiente comando.

```
$sudo apt update
```

Con la instrucción anterior, Python 3.6 estará instalado. En caso que haya habido un error con las dependencias y no se haya instalado se procede a instalar la versión requerida a través de la siguiente linea.

```
$sudo apt install python3.6
```

Se verifica que se tenga la instalación correcta de python con la siguiente linea.

```
$python --version
```

El resultado en la consola debe ser el siguiente:

```
Python 3.6.9
```

Posteriormente se procede a instalar la paquetería pip3 para poder instalar todos los paquetes necesarios ejecutar el proyecto. Para ésto se ejecuta la siguiente instrucción.

```
$sudo apt install python3-pip
```

Habiendo seguido el procedimiento anterior, se tiene Python en su versión 3.6 junto con el gestor de dependencias pip3, para poder instalar los paquetes necesarios para la ejecución del programa. Si se requiere alguna dependencia simplemente se instala con pip3.

4.2.3. Compilación e instalación de Tensowflow 2.1

Una herramienta indispensable para trabajar con Machine Learning mediante el uso de redes neuronales artificiales es TensorFlow. Python ofrece la instalación de versiones precompiladas anteriores a la versión 2.0 con el gestor de dependencias pip. Pero cabe mencionar que estas versiones carecen de algunos parámetros de optimización que TensorFlow ofrece apartir de las versión 2.0, por lo cual es necesario seguir de ciertas instrucciones para compilar el código fuente de TensorFlow 2.1 para el equipo de cómputo en donde se realizará la implementación del algoritmo de redes neuronales artificiales.

Los prerequisitos previos a la instalación de TensorFlow 2.1, son los siguientes.

1. Descarga e instalación de los drivers de la tarjeta gráfica NVIDIA.
2. Descarga e instalación de CUDA 10.1 para Ubuntu 18.04.
3. Descarga e instalación de la librería CuDNN 7.6 para Ubuntu 18.04.

Posteriormente se procede a instalar TensorFlow, para lo cual se prepara el gestor de dependencias pip, con las siguientes instrucciones.

```
$pip3 install -U --user pip numpy wheel
$pip3 install -U --user keras_preprocessing --no-deps
```

Se descarga del repositorio de TensorFlow, el código fuente de la biblioteca en la rama a la que se refiere la versión 2.1, ésta es la rama r2.1, con las siguientes instrucciones.

```
$git clone https://github.com/tensorflow/tensorflow.git
$cd tensorflow
$git checkout r2.
```

Teniendo descargado el código fuente de TensorFlow 2.1 y ubicando sobre su directorio, se procede a configurar el instalador con la siguiente instrucción.

```
$./configure
```

Las opciones de configuración se establecen de acuerdo a las características del equipo de cómputo, por ejemplo la capacidad de cómputo de la tarjeta gráfica o la ruta de instalación de las

librerías de CUDA. Un ejemplo con el que se puede guiar para realizar una configuración correcta se encuenta en el siguiente enlace: https://www.tensorflow.org/install/source#expandable-1.

Teniendo configurado el instalador del código fuente se procede a compilar el código, para ello se requiere tener instalado Bazel, se puede descargar el instalador o seguir las instrucciones de instalación en consola con el siguiente enlace: https://docs.bazel.build/versions/master/install.html.

El comando para compilar el código fuente de TensorFlow con Bazel es el siguiente.

```
$bazel build --config=cuda //tensorflow/tools/pip_package:build_pip_package
```

Se pueden agregar más opciones de configuración en el parámetro *–config*, dependiendo el tipo de compilación que se desee, ésto se puede consultar en la página de instalación de Bazel. Habiendo compilado el código fuente de TensorFlow 2.1, se genera una carpeta en la ruta /tensorflow/tools/pip_package que contiene el paquete necesario de archivos .py para generar un instalador que reconozca el gestor de dependencias pip. Para esto se ejecuta con Bazel la siguiente instrucción.

```
$./bazel-bin/tensorflow/tools/pip_package/build_pip_package /tmp/
    ↪ tensorflow_pkg
```

El instalador pip que se genera contiene extensión .whl y se crea en la ruta que especificó en la instrucción anterior, /tmp/tensorflow_pkg. Para instalar en Python 3.6 la biblioteca TensorFlow 2.1 previamente compilada, se ejecuta la siguiente instrucción.

```
$pip3 install /tmp/tensorflow_pkg/tensorflow-2.1.whl
```

Siguiendo este procedimiento se tiene acceso a todo el contenido de la biblioteca TensorFlow 2.1, y como la compilación se genera para ejecutarse con CUDA, cualquier red neuronal artificial creada con TensorFlow en Python, se ejecutará automáticamente dentro de la GPGPU del equipo de cómputo.

4.3. Bases de Datos

Las bases de datos se extrajeron del servidor NetMHCIIPan 3.2, mismas que fueron recopiladas de la IEDB en 2016. Estas bases de datos están compuestas por 10 archivos de texto que contienen afinidades de unión de péptidos-MHC-II. Cada uno de los archivos contiene en

Alelo	Moléculas
HLA-DR	36
HLA-DQ	27
HLA-DP	9
H-2	8

Tabla 4.1: Conjunto de datos de Alelos MHC-II del 2016 (IEDB)

la primera columna la secuencia de aminoácidos del péptido en estudio, en la segunda columna el valor de afinidad y en la tercera columna el alelo al que pertenece el péptido.

Secuencia	Afinidad	Alelo
PKYVKQNTLKLAT	0	HLA-DPA10103-DPB10201
DSDVGEFRAVTELG	0.047212	HLA-DQA10501-DQB10301
AAAAGWQTLSAALDA	0.23891	HLA-DQA10201-DQB10402
AALDAQAVELTARLN	0.357937	DRB1_0301
AAPAAGYTPATPAAP	0.076722	DRB1_0901
ADAGYAPATPAAAGA	0.119299	DRB1_1201
ADLGYGPATPAAPAA	0.285795	DRB1_1501
AEVELRQHGSEEWEP	0.202669	H-2-IAb
AGSYAADLGYGPATP	0.108843	H-2-IAk

Figura 4.1: Contenido de los archivos que conforman las Bases de Datos.

Los valores de afinidad que tiene la IEDB vienen en unidad de concentración inhibitoria al 50 % (**IC50**), pero los que se extraen del servidor NetMHCIIPan ya vienen normalizados a valores entre 0 y 1 con la relación que propone Nielsen et al. (2003) [46].

El conjunto de datos tiene en total 134281 péptidos que pertenecen a 36 moléculas de tipo HLA-DR, 27 tipo HLA-DQ, 9 tipo HLA-DP y 8 tipo H-2.

De los 10 archivos de texto que componen las bases de datos, 5 se utilizan para entrenar el algoritmo y los 5 restantes para realizar las pruebas de validación cruzada. Esta configuración fue propuesta por Nielsen et al. (2007) [47].

4.4. Matrices de sustitución

Los alineamientos de secuencias de proteínas se han convertido en una herramienta importante para los biólogos moleculares. Los alineamientos locales se construyen frecuentemente con la ayuda de Matrices de Puntuación de Sustitución, que especifica una puntuación para alinear cada par de residuos de aminoácidos. A lo largo de los años, se han propuesto muchas matrices de sustitución diferentes, basadas en una amplia variedad de fundamentos. Los resultados estadísticos, sin embargo, demuestran que cualquier matriz de este tipo es implícitamente una matriz log-probabilidades, con una distribución objetivo específica para pares alineados de residuos de aminoácidos. [48]

Existen diversos tipos de matrices de sustitución, la mayoría para alineamiento de secuencias de aminoácidos de proteínas, ya que en el caso de nucleótidos, se suele utilizar un sistema de puntuación más simple, basado en los cuatro nucleótidos a sustituir posibles.

4.4.1. Matriz identidad (one-hot encoding)

La matriz identidad es la más simple posible, se utiliza en la mayoría de problemas informáticos para sustituir cadena de caracteres por secuencias numéricas. En Machine Learning se conoce como codificación One-Hot. Para problemas generales este tipo de matriz funciona

	A	R	N	D	C	...	T	W	Y	V	B	Z
A	1	0	0	0	0	0	0	0	0	0	0	0
R	0	1	0	0	0	0	0	0	0	0	0	0
N	0	0	1	0	0	0	0	0	0	0	0	0
D	0	0	0	1	0	0	0	0	0	0	0	0
C	0	0	0	0	1	0	0	0	0	0	0	0
...	0	0	0	0	0	1	0	0	0	0	0	0
T	0	0	0	0	0	0	1	0	0	0	0	0
W	0	0	0	0	0	0	0	1	0	0	0	0
Y	0	0	0	0	0	0	0	0	1	0	0	0
V	0	0	0	0	0	0	0	0	0	1	0	0
B	0	0	0	0	0	0	0	0	0	0	1	0
Z	0	0	0	0	0	0	0	0	0	0	0	1

Figura 4.2: Matriz identidad.

bien, pero en la bioinformática, especialmente para alineamientos de secuencias de aminoácidos, tiene sus desventajas, ya que, aporta información al alineamiento solamente para secuencias de aminoácidos con mucha similitud pero cuando se trata de secuencias poco similares utilizar este tipo de matriz no genera un alineamiento correcto. Para esta investigación se usó de principio este tipo de sustitución pero los resultados eran poco favorables por lo cual se descartó.

4.4.2. Matrices de log-probabilidades

En una matriz de log-probabilidades se expresan las probabilidades de mutación de un aminoácido en otro mediante lo que se denomina puntuación por log-probabilidades (logit). La matriz de puntuaciones se define como:

$$a_{i,j} = log\frac{p_{i,j}}{p_i * p_j} = log\frac{frecuencia\ observada}{frecuencia\ esperada} \tag{4.1}$$

donde $p_{i,j}$ es la probabilidad, de acuerdo a las observaciones tomadas en consideración, de que el aminoácido i se transforme en el j, p_i es la frecuencia de aparición del aminoácido i, y p_j es la frecuencia de aparición del aminoácido j. [49]

Lo anterior indica que, el denominador $p_i * p_j$ es la probabilidad de que ambos aminoácidos queden alineados al azar, y el numerador entre ambas partes puede resultar en lo siguiente.

1. Mayor que 1: que la probabilidad de mutación observada entre aminoácidos es superior a la aleatoria.

2. Igual a 1: la mutación entre uno y otro aminoácido corresponde a la que puede encontrarse aleatoriamente a causa de mutuaciones puntuales.

3. Menor que 1: la tasa de sustitución entre los aminoácidos es inferior a la que podría encontrarse aleatoriamente, por lo que con la magnitud del cociente podría asumirse, en principio, que esta sustitución no es aceptada evolutivamente de buen grado.

4.4.3. Matrices PAM y BLOSUM

Existen diferentes matrices log-probabilidades que se utilizan para sustituir secuencias de aminoácidos por valores numéricos. Las principales matrices de sustitución son las matrices

.	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V	B	J	Z	X	*
A	5	-2	-1	-2	-1	-1	-1	0	-2	-1	-2	-1	-1	-3	-1	1	0	-3	-2	0	-2	-2	-1	-1	-5
R	-2	7	-1	-2	-4	1	0	-3	0	-4	-3	3	-2	-3	-3	-1	-1	-3	-1	-3	-1	-3	0	-1	-5
N	-1	-1	7	2	-2	0	0	0	1	-3	-4	0	-2	-4	-2	1	0	-4	-2	-3	5	-4	0	-1	-5
D	-2	-2	2	8	-4	0	2	-1	-1	-4	-4	-1	-4	-5	-1	0	-1	-5	-3	-4	6	-4	1	-1	-5
C	-1	-4	-2	-4	13	-3	-3	-3	-3	-2	-2	-3	-2	-2	-4	-1	-1	-5	-3	-1	-3	-2	-3	-1	-5
Q	-1	1	0	0	-3	7	2	-2	1	-3	-2	2	0	-4	-1	0	-1	-1	-1	-3	0	-3	4	-1	-5
E	-1	0	0	2	-3	2	6	-3	0	-4	-3	1	-2	-3	-1	-1	-1	-3	-2	-3	1	-3	5	-1	-5
G	0	-3	0	-1	-3	-2	-3	8	-2	-4	-4	-2	-3	-4	-2	0	-2	-3	-3	-4	-1	-4	-2	-1	-5
H	-2	0	1	-1	-3	1	0	-2	10	-4	-3	0	-1	-1	-2	-1	-2	-3	2	-4	0	-3	0	-1	-5
I	-1	-4	-3	-4	-2	-3	-4	-4	-4	5	2	-3	2	0	-3	-3	-1	-3	-1	4	-4	4	-3	-1	-5
L	-2	-3	-4	-4	-2	-2	-3	-4	-3	2	5	-3	3	1	-4	-3	-1	-2	-1	1	-4	4	-3	-1	-5
K	-1	3	0	-1	-3	2	1	-2	0	-3	-3	6	-2	-4	-1	0	-1	-3	-2	-3	0	-3	1	-1	-5
M	-1	-2	-2	-4	-2	0	-2	-3	-1	2	3	-2	7	0	-3	-2	-1	-1	0	1	-3	2	-1	-1	-5
F	-3	-3	-4	-5	-2	-4	-3	-4	-1	0	1	-4	0	8	-4	-3	-2	1	4	-1	-4	1	-4	-1	-5
P	-1	-3	-2	-1	-4	-1	-1	-2	-2	-3	-4	-1	-3	-4	10	-1	-1	-4	-3	-3	-2	-3	-1	-1	-5
S	1	-1	1	0	-1	0	-1	0	-1	-3	-3	0	-2	-3	-1	5	2	-4	-2	-2	0	-3	0	-1	-5
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	2	5	-3	-2	0	0	-1	-1	-1	-5
W	-3	-3	-4	-5	-5	-1	-3	-3	-3	-3	-2	-3	-1	1	-4	-4	-3	15	2	-3	-5	-2	-2	-1	-5
Y	-2	-1	-2	-3	-3	-1	-2	-3	2	-1	-1	-2	0	4	-3	-2	-2	2	8	-1	-3	-1	-2	-1	-5
V	0	-3	-3	-4	-1	-3	-3	-4	-4	4	1	-3	1	-1	-3	-2	0	-3	-1	5	-3	2	-3	-1	-5
B	-2	-1	5	6	-3	0	1	-1	0	-4	-4	0	-3	-4	-2	0	0	-5	-3	-3	6	-4	1	-1	-5
J	-2	-3	-4	-4	-2	-3	-3	-4	-3	4	4	-3	2	1	-3	-3	-1	-2	-1	2	-4	4	-3	-1	-5
Z	-1	0	0	1	-3	4	5	-2	0	-3	-3	1	-1	-4	-1	0	-1	-2	-2	-3	1	-3	5	-1	-5
X	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-5
*	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	-5	1

Figura 4.3: Matriz BLOSUM 50 recopilada de la IEDB. Matriz tipo log-probabilidades que se utiliza para la sustitución de secuencia de aminoácidos de proteínas.

PAM (Point accepted mutation) y las BLOSUM (BLOck SUbstitution Matrix).

Una de las primeras matrices de sustitución fue desarrollada por Margaret Dayhoof en los años 70. Esta matriz se calcula observando la relación de proteínas cercanamente relacionadas, aproximadamente con un mínimo de similitud del 85 %. [49] La matriz PAM1 estima qué probabilidad de sustitución debería esperarse si el 1 % de los aminoácidos han cambiado, siguiendo esa lógica se crearon matrices como PAM30 y PAM70, elevando a una determinada potencia la matriz PAM1.
Este tipo de matrices se utiliza para seguir orígenes evolutivos de proteínas aunque actualmente no son las más utilizadas.

Las matrices BLOSUM se crearon por Henikoff y Henikoff [49], quienes a partir de múltiples alineamientos de proteínas evolutivamente divergentes (hecho que no se consideró en las

matrices PAM) construyeron este tipo de matriz. Las probabilidades que se usan en los cálculos de la matriz se computan observando grupos de secuencias conservadas encontrados en múltiples alineamientos de proteínas. Los grupos por encima de cierto umbral de similitud fueron agrupados, ponderando con un factor de 1 a cada uno de los grupos. Para la matriz BLOSUM 62, el umbral se estableció en el 62 %. Así para las matrices BLOSUM existentes el sufijo en número define el umbral de similitud entre grupos con las que fueron creadas. Existen matrices BLOSUM 45, BLOSUM 50 y hasta BLOSUM 80 con un 80 % de similitud entre grupos para esta última matriz.

La matriz BLOSUM 50 ha demostrado resultados efectivos según lo descrito por Jensen et al. (2018) [50] al utilizar este tipo de matriz en la sustitución de secuencias de aminoácidos de péptidos utilizados para el servidor NetMHCIIPan 3.2, hecho por el cual se elige la matriz BLOSUM 50 para este proyecto.

4.5. Normalización de IC50

Los valores de afinidad que se encuentran en las bases de datos de péptidos-MHC-II de la IEDB vienen en unidad de concentración inhibitoria al 50 % (**IC50**). Estas afinidades ya son normalizadas en las bases de datos que pone a disponibilidad pública Jensen et al. (2018) [50] para el servidor NetMHCIIPan. Los valores de afinidad se normalizan utilizando la relación descrita por Nielsen et al. (2003) [46].

$$1 - \frac{Log(IC50_{nM})}{Log(50,000)} \tag{4.2}$$

Usando esta fórmula se normalizan los valores de afinidad a valores entre 0 y 1. Esta normalización de valores es necesaria para que la red neuronal artificial converja a una solución correcta.

4.6. Medida de desempeño

Con la fórmula descrita por Nielsen et al. (2003) [46] los valores IC50 mayores a 46,000 convergerán a 0, ya que este valor de concentración inhibitoria es irrelevante para asegurar una

afinidad de unión de un péptido a la molécula MHC-II. Los valores más pequeños de concentración inhibitoria se acercarán a 1 y serán los candidatos a representar una afinidad de unión óptima a la molécula MHC-II.

El umbral de afinidad de unión, para clasificar péptidos afines y no afines, se siguió con base en la investigación de Jensen et al. (2018) [50], para lo cual se eligió un umbral de valores menores a 500_{nM} como afines o aglutinantes, lo que en valores normalizados, son los mayores a 0.426.

Con esta división de péptidos afines y no afines se mide el rendimiento predictivo del algoritmo utilizando el valor AUC (área bajo la curva) de la métrica de la curva ROC (Característica operativa del receptor) utilizada ampliamente para medir el desempeño de algoritmos de predicción de péptidos-MHC-II en el estado del arte.

4.7. Alineamiento NN-Align

El alineamiento NN-Align es un método basado en una red neuronal artificial que permite la identificación simultánea del núcleo de unión de péptidos-MHC-II y su afinidad de unión. [51]

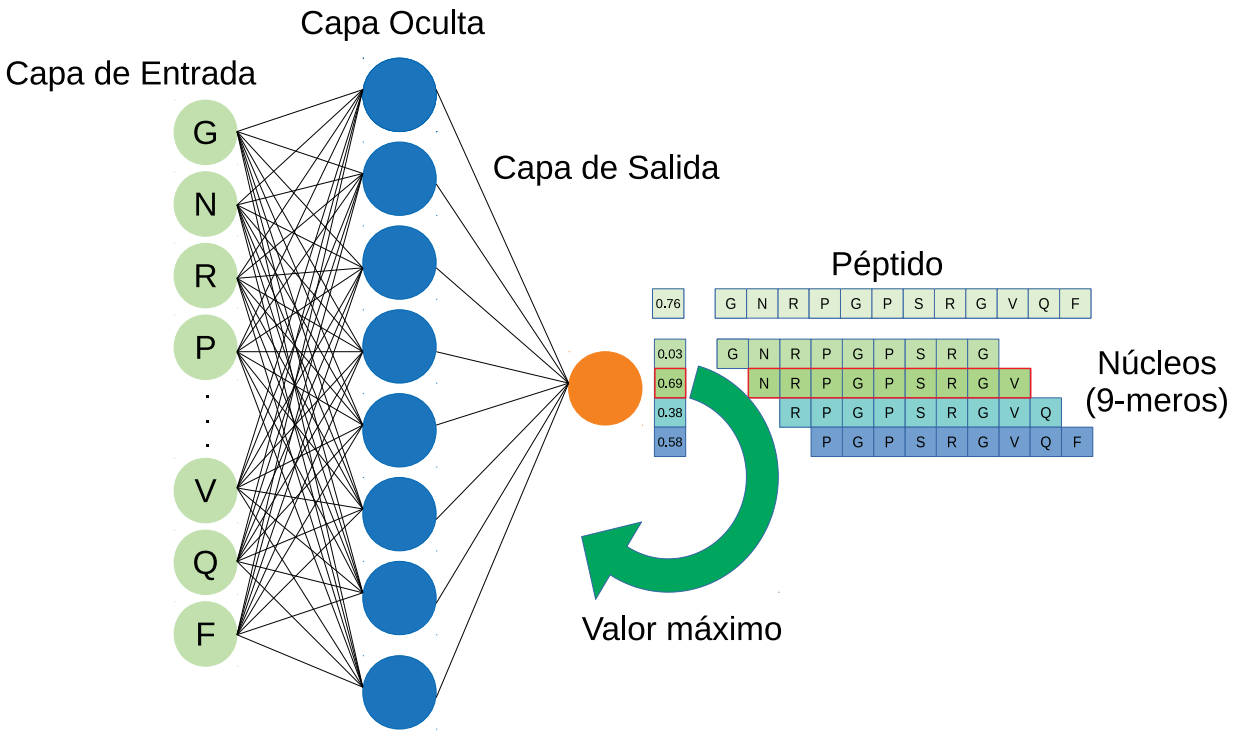


Figura 4.4: Alineamiento NN-Align basado en una red neuronal artificial que predice el núcleo de unión y la afinidad.

El método consiste en un procedimiento de dos pasos que estima simultáneamente el registro óptimo de unión (núcleo 9-mero) de un péptido y el valor de afinidad, discriminando como el núcleo óptimo por el que tenga la afinidad más alta de todos los núcleos de unión del péptido. Identificando todos los núcleos de unión y afinidades (proceso de alineamiento) se actualizan los pesos de la red mediante retropropagación de descenso de gradiente. Inicialmente los pesos de la red se asignan aleatoriamente. Los pesos se actualizan para reducir la suma de los errores al cuadrado (SSE) entre la afinidad de unión predicha dentro de la red y el valor de afinidad real del péptido.

Los datos de unión cuantitativos contienen un alto grado de redundancia, ya que se han medi-

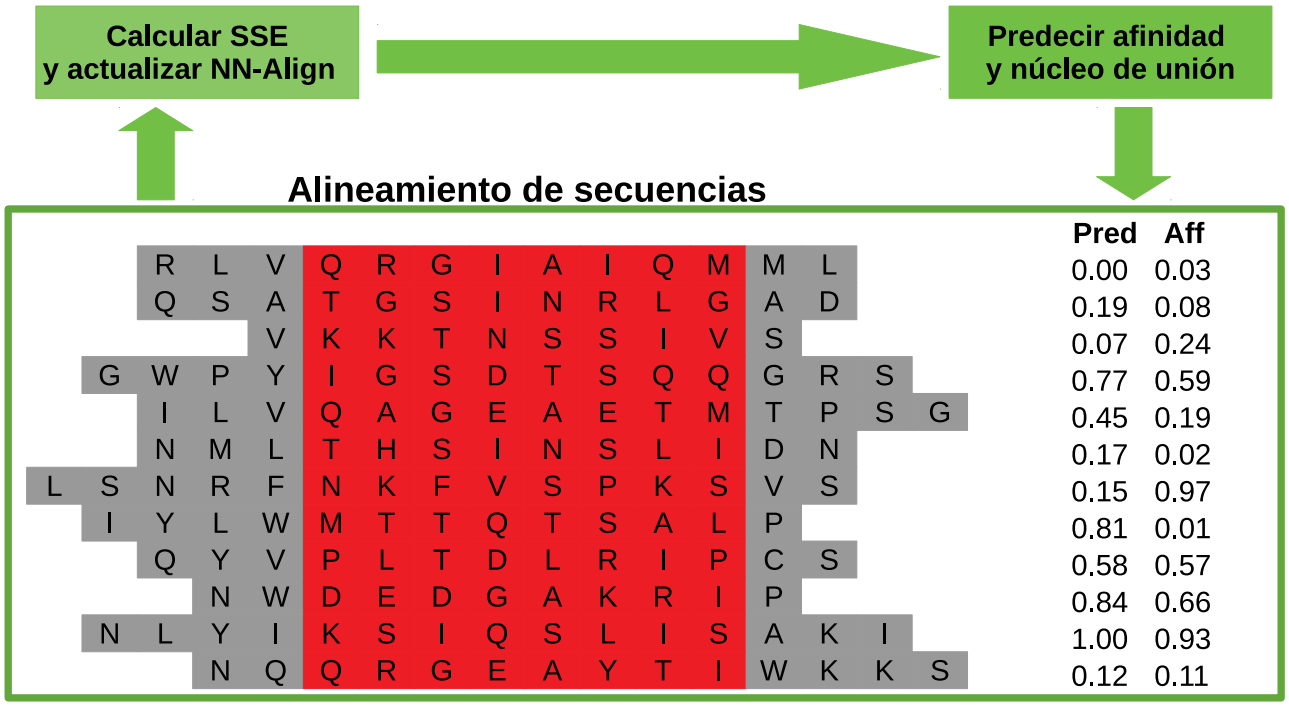


Figura 4.5: NN-Align genera un alineamiento de secuencias de péptidos-MHC-II.

do repetidamente muchos puntos de datos de péptidos con mutaciones de un sólo aminoácido para identificar, por ejemplo, el efecto de los aminoácidos que flanquean el núcleo de unión del péptido (PFR, Región flanqueante). [51]

Algoritmo 1: Algoritmo NN-Align

Entrada: Núcleos péptidos y afinidades de entrenamiento

Salida: Vector W de pesos

```
Inicializar Vector W de pesos aleatorios;
mientras 1 hasta MAX épocas hacer
    para 1 hasta N péptidos hacer
        Evaluar núcleos del péptido enésimo;
        Elegir mejor núcleo basado en la afinidad más alta;
        Calcular E^2 de la afinidad del mejor núcleo contra afinidad real del péptido;
        Acumular SSE;
    Retropropagar Error al Vector W;
```

Para limitar ese efecto Nielsen et al. (2009)[51] modificó la retropropagación de la red para que el tamaño de paso (step-size o delta) de la retropropagación se dividiera por la redundancia del núcleo de unión del péptido dado. Ésto se probó inicialmente en esta investigación pero el rendimiento predictivo no era significante con esta modificación, ya que los valores de AUC que vienen en la tabla 5.3 no mejoraban significativamente.

4.8. Método y algoritmo propuesto

El método propuesto consiste en los siguientes puntos.

1. Utilizar las bases de datos del servidor NetMHCIIPan que tienen las afinidades ya normalizadas.

2. Proponer un algoritmo para predecir péptidos-MHC-II.

3. Incorporar al algoritmo propuesto una mejora al alineamiento NN-Align para mejorar la predicción.

4. Implementar el algoritmo propuesto utilizando algún lenguaje de programación o librería de código abierto para procesar de manera eficiente y rápida la información contenida en las bases de datos.

5. Generar predicciones y comparar la eficiencia y el rendimiento predictivo del algoritmo propuesto.

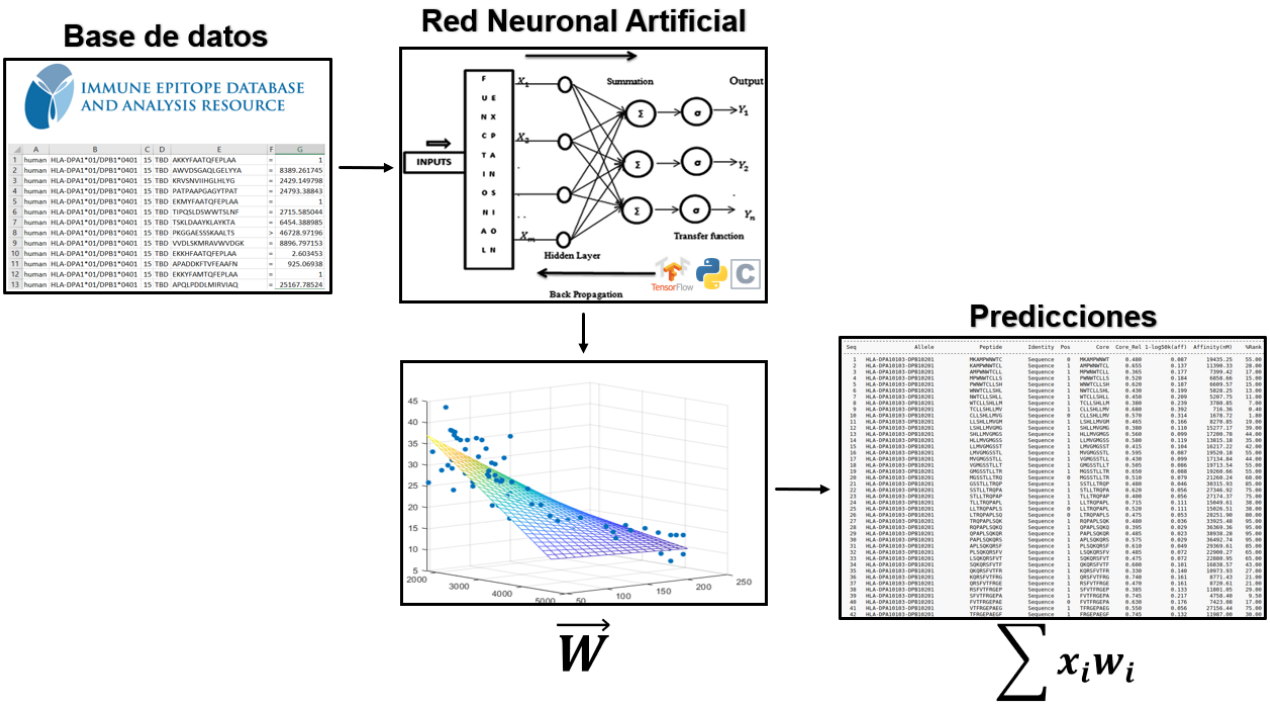


Figura 4.6: Algoritmo propuesto para predicción de péptidos-MHC-II.

El primer punto hace referencia a que los valores de afinidad de los archivos de texto que vienen en el servidor NetMHCIIPan 3.2 ya vienen normalizados, lo que facilita el procesamiento de ello, además vienen separados en cinco archivos para el entrenamiento y otros cinco archivos para pruebas, facilitando la validación y obtención del rendimiento predictivo del algoritmo propuesto.

En el algoritmo 2 se detalla cada uno de los pasos del algoritmo propuesto. En él, como primer y segundo paso se obtienen todos los núcleos posibles de cada péptido. También adicional a ésto y después de haber sido codificados con la matriz de sustitución *BLOSUM 50*, en la linea 5 se eliminan las secuencias de péptidos repetidos para reducir la redundancia de datos, redundancia generada por péptidos similares en las bases de datos de péptidos-MHC-II, tal como lo explican Nielsen et al. (2009)[51] y Noguchi et al (2002) [42].

Posteriormente en la linea 6 y 7 en un bucle de 5 repeticiones se generan 5 vectores W de pesos óptimos haciendo uso del algoritmo NN-Align mejorado (algoritmo 3).

En la linea 8 se evaluarán los péptidos de validación con cada uno de los pesos W generados

para predecir el núcleo de unión óptimo y su afinidad, basado en la selección de la afinidad más alta. Estas predicciones se persistirán en memoria para después en la linea 11 promediar la suma de las afinidades entre el número de repeticiones de este proceso (linea 6). Ésto de acuerdo a la ecuación 4.3.

$$Aff = \frac{\sum_i^N AffPred_i}{N} \tag{4.3}$$

Se realizan hasta 5 predicciones por la cantidad de mínimos locales que se generan dentro del algoritmo NN-Align, ésto ampliamente documentado por Nielsen et al. (2009) [51]. La afinidad promedio que tenga cada péptido será comparada contra la afinidad real del péptido para calcular el error cuadrático de ésta predicción.

En la linea 13 se calcula el error cuadrático generado de las predicciones en cada una de las 5 validaciones cruzadas k-fold para posteriormente en la linea 15, al finalizar el bucle, calcular el error cuadrático medio del algoritmo propuesto (algoritmo 2).

Algoritmo 2: Algoritmo del método propuesto

Entrada: Bases de datos

Salida: MSE, **Afinidades predichas**, **Núcleos predichos**

1 **para** ***1*** *hasta* ***5*** *validaciones* **hacer**
2 Asignar datos de entrenamiento y validación;
3 Obtener todos los núcleos y PFR de todos los péptidos de entrenamiento y validación;
4 Codificar secuencias de péptidos utilizando la matriz **BLOSUM 50**;
5 Eliminar secuencias de péptidos repetidos del conjunto de entrenamiento;
6 **para** ***1*** *hasta* ***5*** *repeticiones* **hacer**
7 Generar vector **W** de pesos óptimos con algoritmo **3**;
8 Con **W** predecir núcleos de péptidos de validación;
9 Seleccionar mejores núcleos y afinidadades y almacenarlos;
10 Promediar afinidades seleccionadas de cada péptido en el bucle anterior;
11 Calcular E^2 del promedio anterior de cada péptido contra su afinidad real;
12 Acumular **SSE**;
13 Calcular **MSE** del algoritmo

4.9. Mejora de NN-Align

El segundo punto del método propuesto (sección 4.8) aborda una modificación que mejora la predicción de afinidades dentro del método de retropropagación de la red neuronal artificial del algoritmo NN-Align (algoritmo 1).

Esta modificación se puede visualizar en el algoritmo 3 a partir de la linea 3, donde antes del cálculo del error cuadrático derivado de la diferencia entre la afinidad predicha del mejor núcleo y la afinidad real del péptido, se realiza primeramente la evaluación de todos los núcleos de un péptido dentro del conjunto de datos de entrenamiento para obtener el mejor núcleo de cada uno y su afinidad, basado en el núcleo con la afinidad más alta.
Posteriormente en la linea 5 se eliminan los núcleos-PFR repetidos para propagar solamente el error del nuevo conjunto de datos sin la información redundante de los péptidos con núcleos-PFR repetidos.
Este modificación en el algoritmo original mejora significativamente la predicción de afinidades.

Algoritmo 3: Algoritmo NN-Align mejorado

```
   Entrada: Núcleos péptidos y afinidades de entrenamiento
   Salida: Vector W de pesos
1  Inicializar Vector W de pesos aleatorios;
2  mientras 1 hasta MAX épocas hacer
3      Evaluar todos los núcleos del conjunto de entrenamiento;
4      Elegir mejor núcleo basado en la afinidad más alta;
5      Eliminar mejores núcleos-PFR repetidos;
6      para 1 hasta N péptidos hacer
7          Calcular E^2 de la afinidad del mejor núcleo contra afinidad real del péptido;
8          Acumular SSE;
9      fin
10     Retropropagar Error al Vector W ;
11 fin
```

4.10. Implementaciones

La presente investigación se enfocó inicialmente en realizar una sola implementación utilizando el lenguaje de programación Python y la biblioteca TensorFlow, tal y como se redactó en la justificación (1.4) y la hipótesis (1.6). Se esperaba que al utilizar TensorFlow el tiempo de aprendizaje del algoritmo se estableciera en un periodo corto (menos de una hora) como actualmente los servidores de predicción presentan información de salida en sus sitios web. Se analizó implementar otras dos versiones para verificar si el tiempo de ejecución del algoritmo podía disminuirse, una versión utilizando solamente el lenguaje Python para comparar el desempeño en tiempo de ejecución contra la versión que utiliza TensorFlow, y una versión más utilizando el lenguaje C, con lo cual, en teoría, gestionando directamente memoria se podrían tener resul-

tados en un tiempo menor a los anteriores.

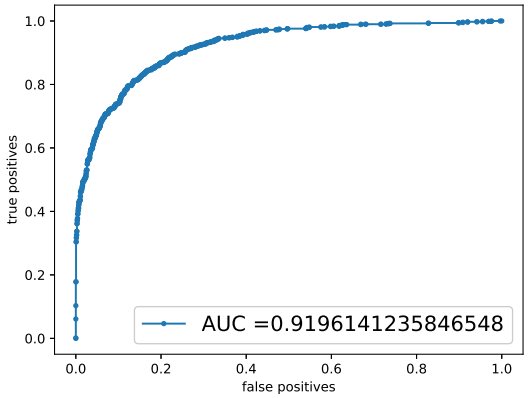
Teniendo las tres versiones programadas se evaluó el tiempo de ejecución para comparar el desempeño de las tres implementaciones. El tiempo de ejecución se evaluó midiendo únicamente el algoritmo 3.

Capítulo 5

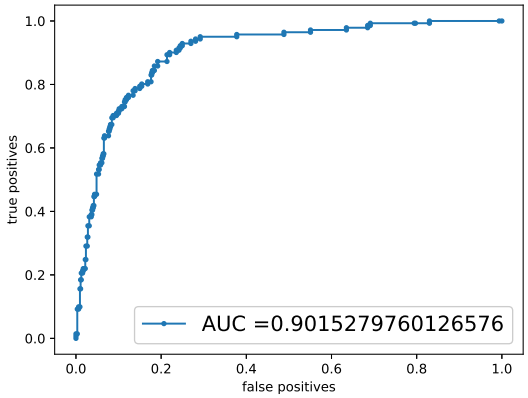
Resultados y discusión

5.1. Rendimiento predictivo AUC en cada molécula

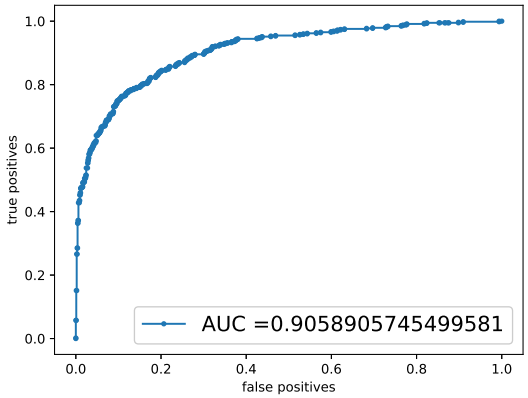
El umbral de afinidad se estableció para valores normalizados mayores a 0.426 con base en lo documentado por Jensen et al (2018) [50], ésto es valores menores a 500_{nM}. Todos los valores menores a 0.426 se etiquetan como 0 (no afin o no aglutinante) y los valores mayores o igual a este valor se etiquetan como 1 (afines o aglutinantes). Con esta clasificación se mide el valor AUC de la curva ROC para medir el desempeño del algoritmo. Se midió el rendimiento predictivo en las moléculas que solamente tenían datos suficientes para aportar una medida AUC, ya que hay moléculas en los conjuntos de entrenamiento que no disponen de datos para generar un vector **W** de pesos. A continuación se presenta el AUC para todos las moléculas de cada alelo que se evaluaron para este algoritmo (2, 3), perteneciendo las moléculas de HLA-DP, HLA-DQ y HLA-DR a alelos humanos y las moléculas de H-2 a alelos de ratón.



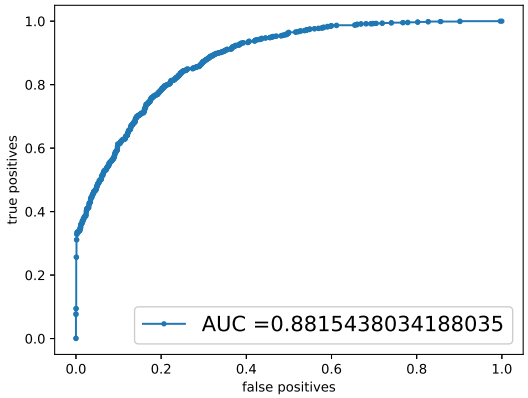
(a) HLA-DPA10103-DPB10401



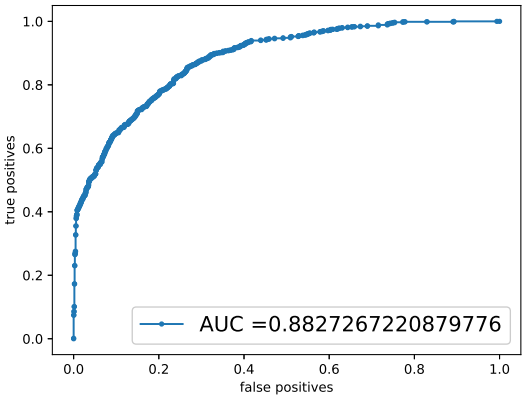
(b) HLA-DPA10103-DPB10201



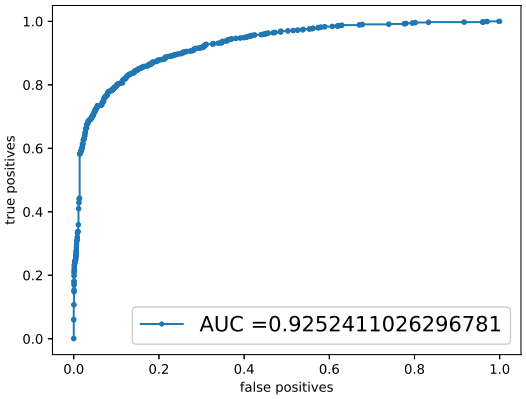
(c) HLA-DPA10103-DPB10301



(d) HLA-DPA10201-DPB10101

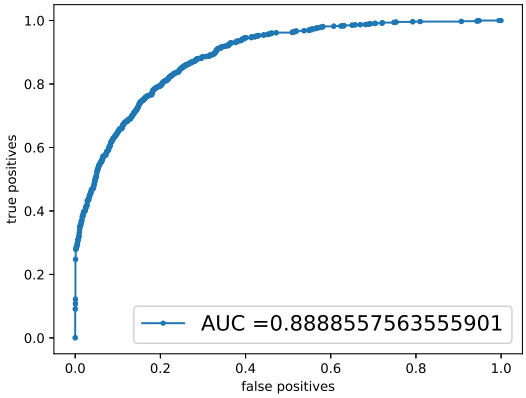


(e) HLA-DPA10201-DPB10501



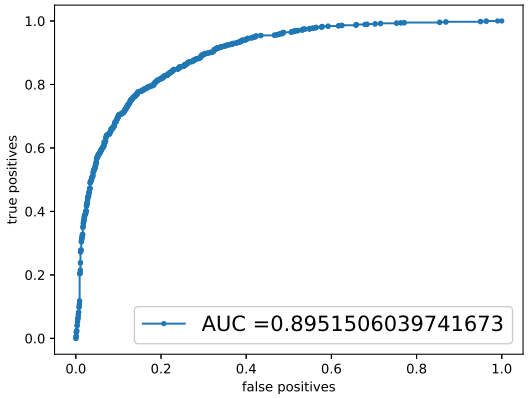
(f) HLA-DPA10201-DPB11401

Figura 5.1: Predicciones para HLA-DP

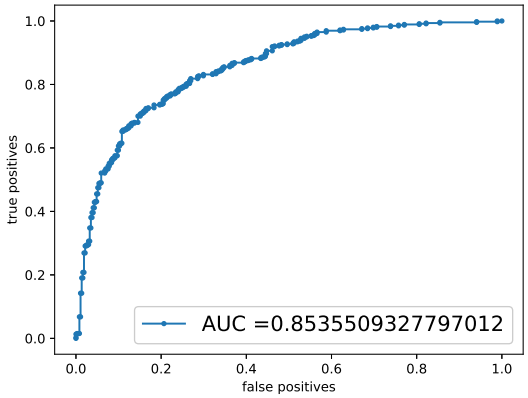


(a) HLA-DPA10301-DPB10402

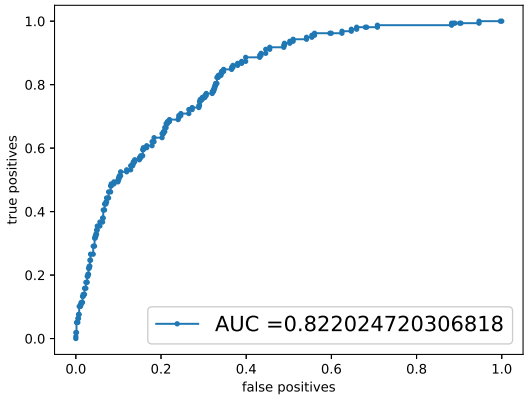
Figura 5.2: Predicciones para HLA-DP



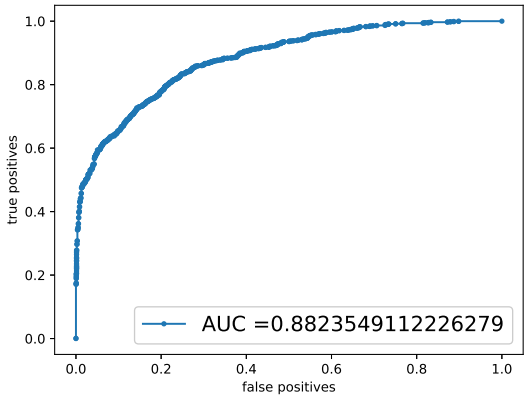
(a) HLA-DQA10101-DQB10501



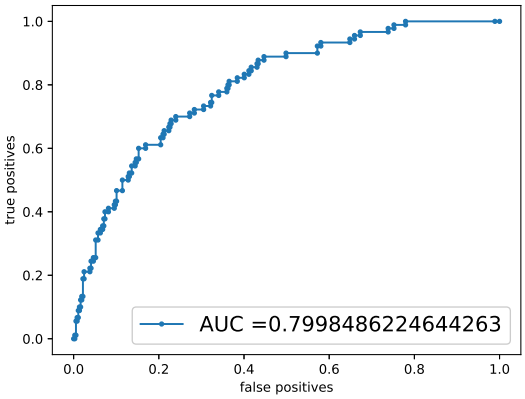
(b) HLA-DQA10102-DQB10501



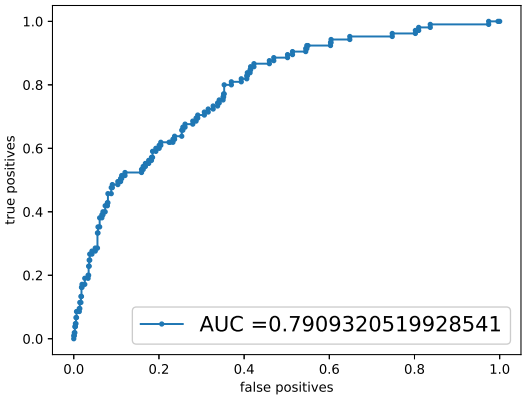
(c) HLA-DQA10102-DQB10502



(d) HLA-DQA10102-DQB10602

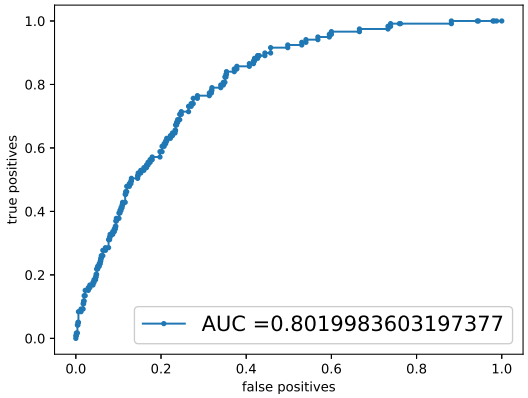


(e) HLA-DQA10103-DQB10603

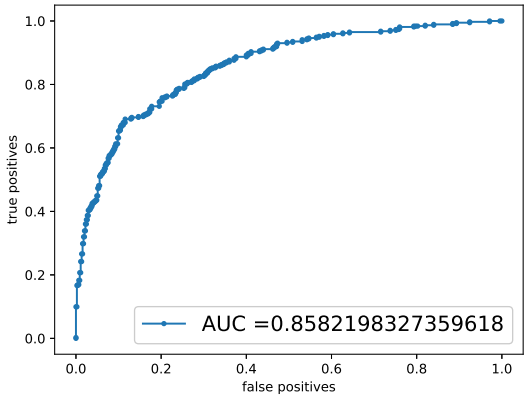


(f) HLA-DQA10104-DQB10503

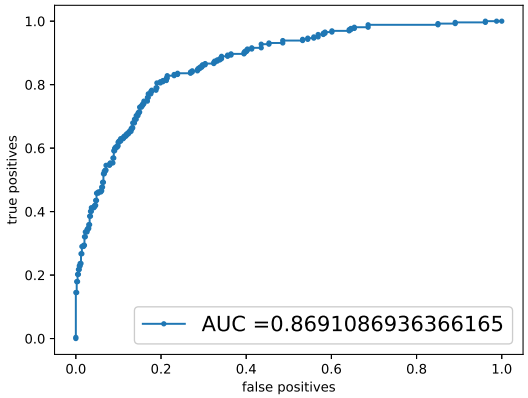
Figura 5.3: Predicciones para HLA-DQ



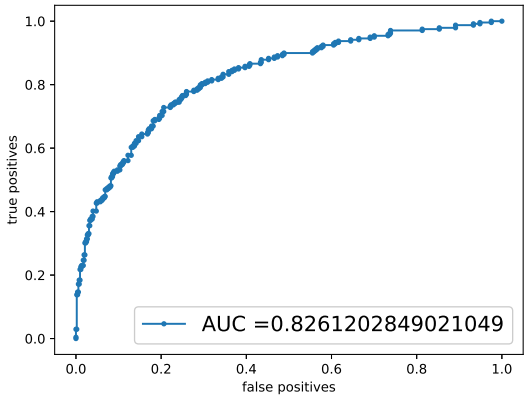
(a) HLA-DQA10201-DQB10202



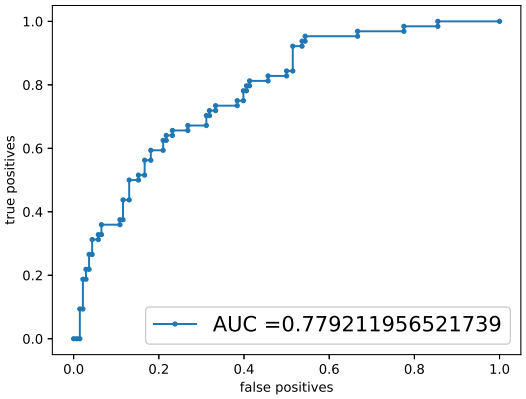
(b) HLA-DQA10201-DQB10301



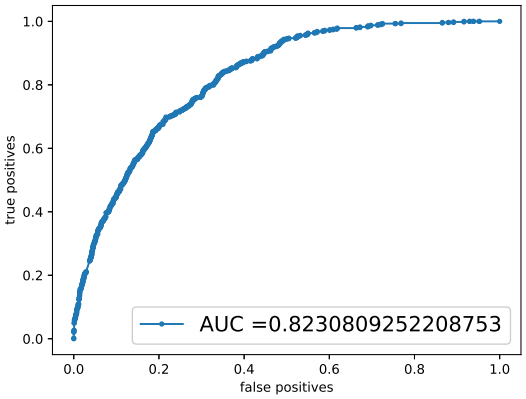
(c) HLA-DQA10201-DQB10303



(d) HLA-DQA10201-DQB10402

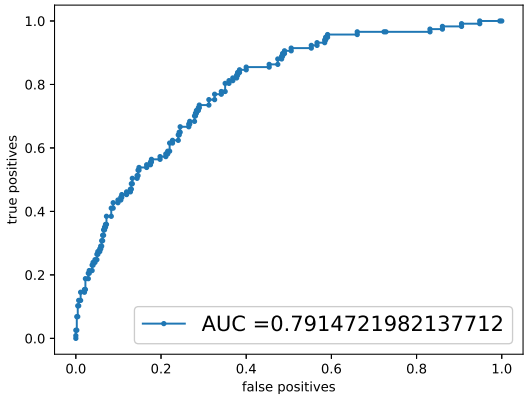


(e) HLA-DQA10301-DQB10301

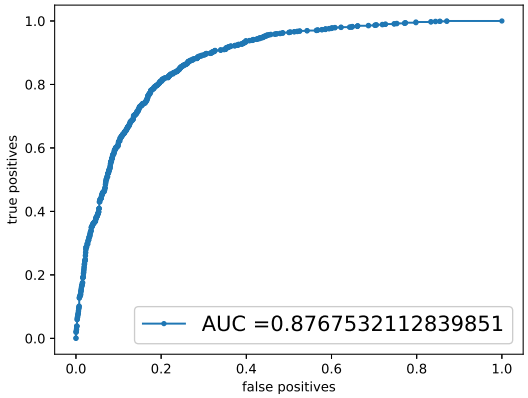


(f) HLA-DQA10301-DQB10302

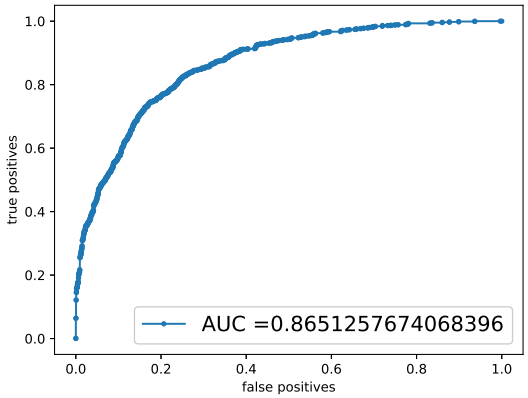
Figura 5.4: Predicciones para HLA-DQ



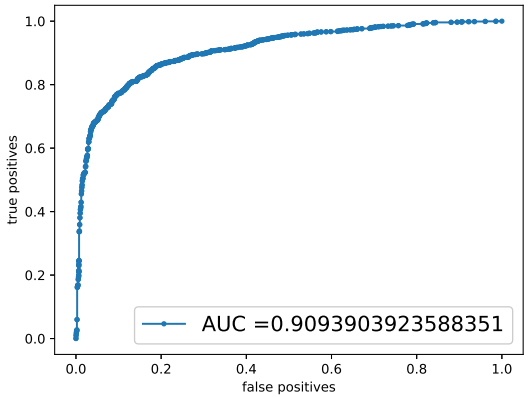
(a) HLA-DQA10303-DQB10402



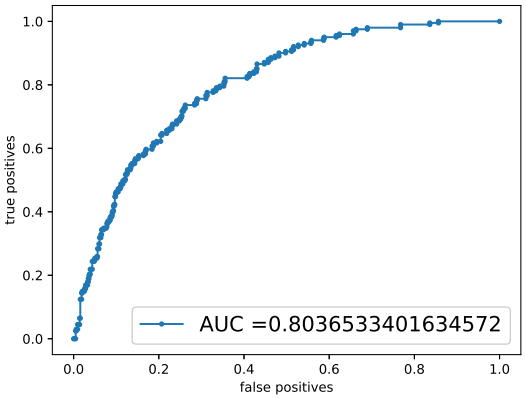
(b) HLA-DQA10401-DQB10402



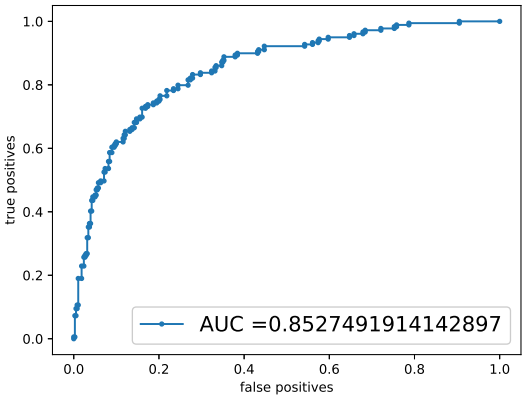
(c) HLA-DQA10501-DQB10201



(d) HLA-DQA10501-DQB10301

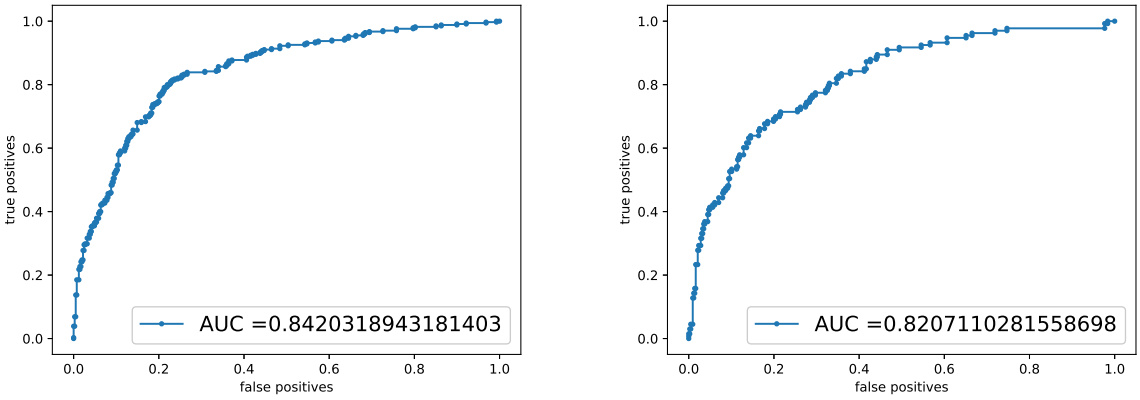


(e) HLA-DQA10501-DQB10302



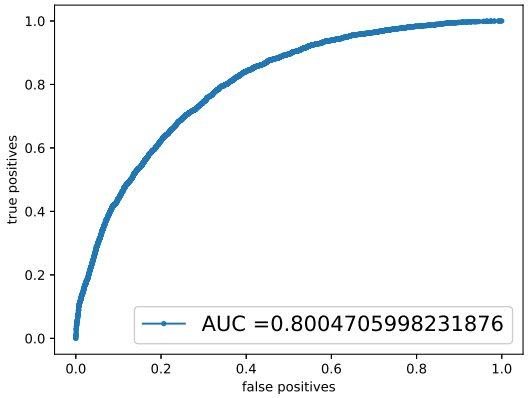
(f) HLA-DQA10501-DQB10303

Figura 5.5: Predicciones para HLA-DQ

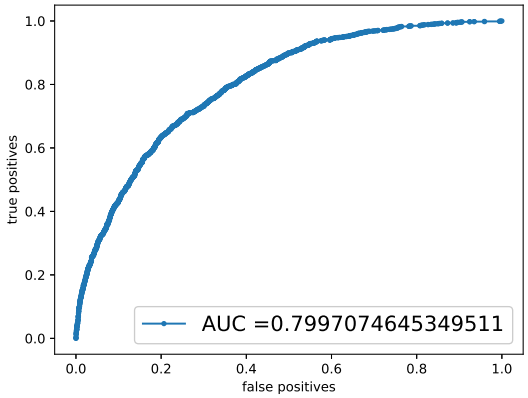


(a) HLA-DQA10501-DQB10402 (b) HLA-DQA10601-DQB10402

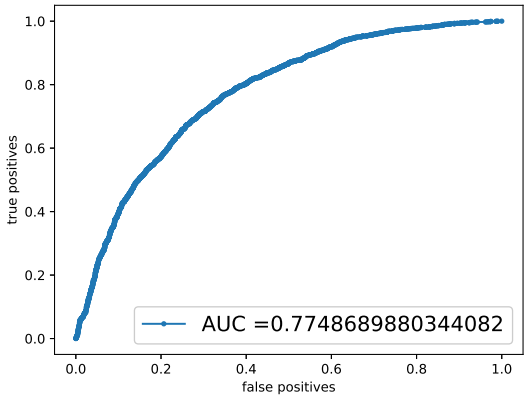
Figura 5.6: Predicciones para HLA-DQ



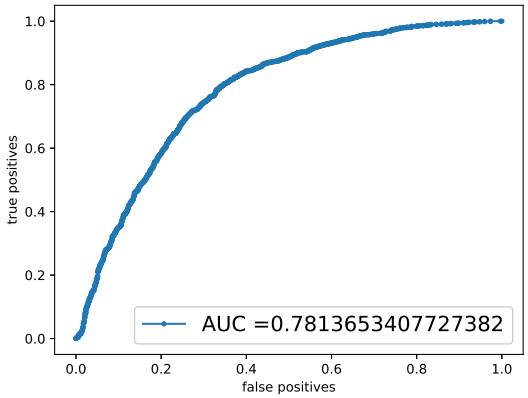
(a) DRB1*0101



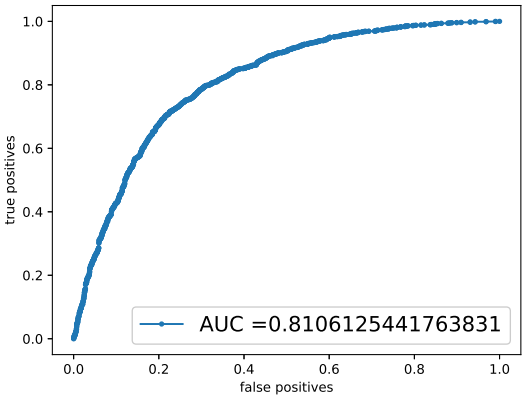
(b) DRB1*0301



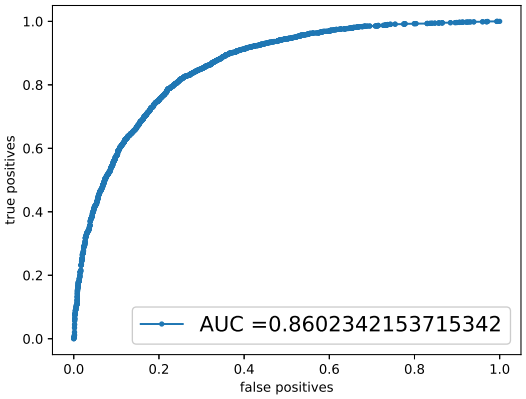
(c) DRB1*0401



(d) DRB1*0404

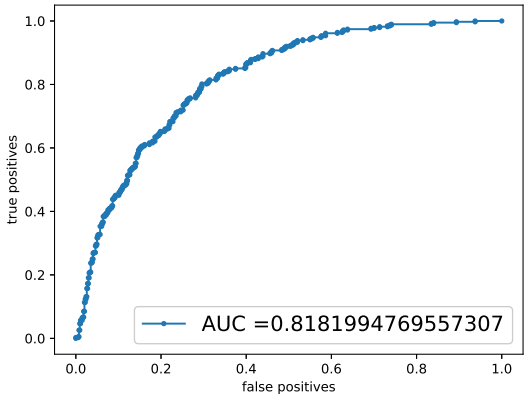


(e) DRB1*0405

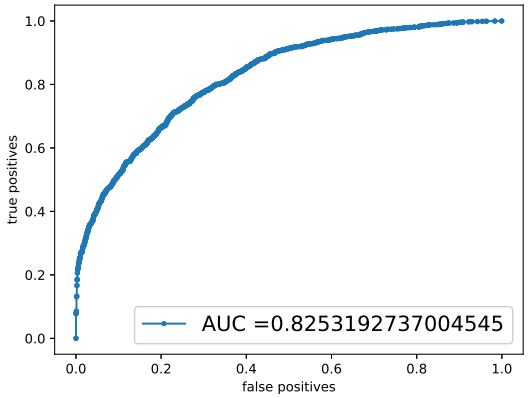


(f) DRB1*0701

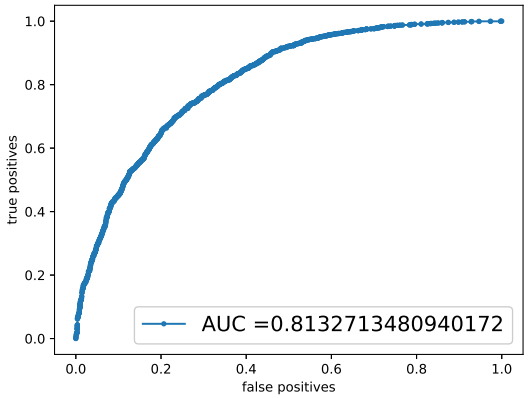
Figura 5.7: Predicciones para HLA-DR



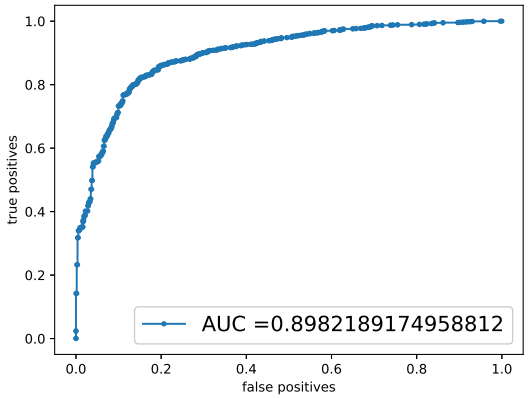
(a) DRB1*0801



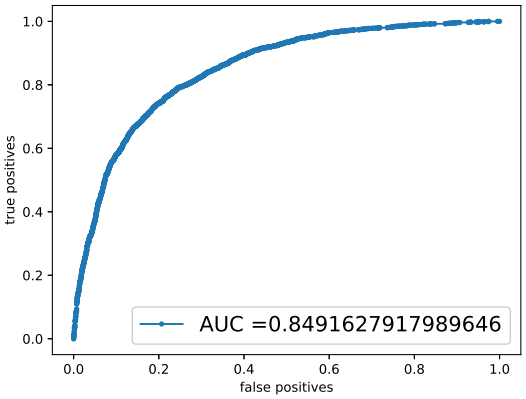
(b) DRB1*0802



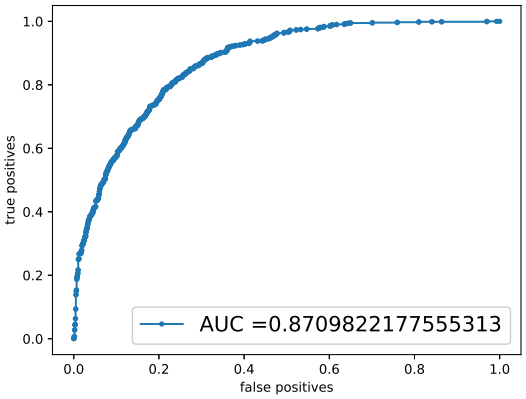
(c) DRB1*0901



(d) DRB1*1001

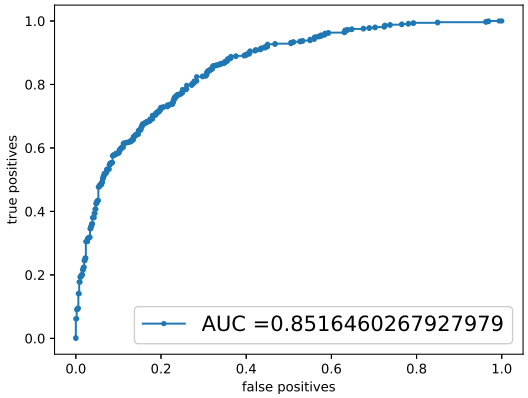


(e) DRB1*1101

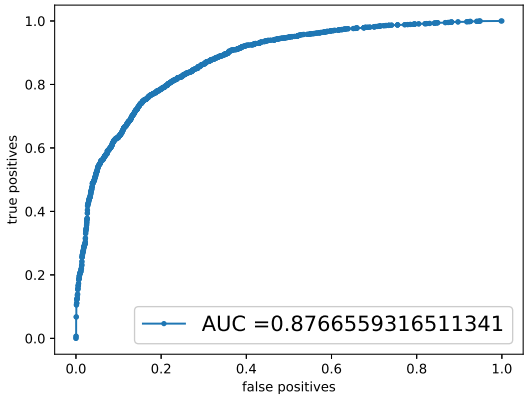


(f) DRB1*1201

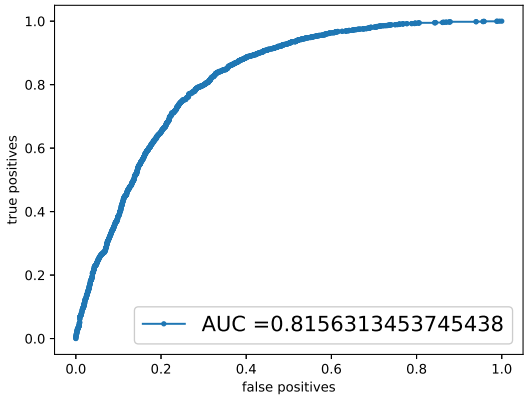
Figura 5.8: Predicciones para HLA-DR



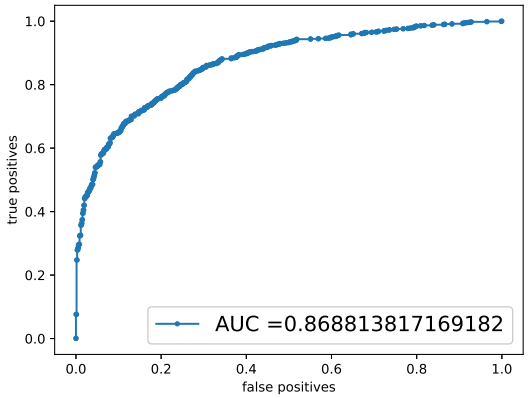
(a) DRB1*1301



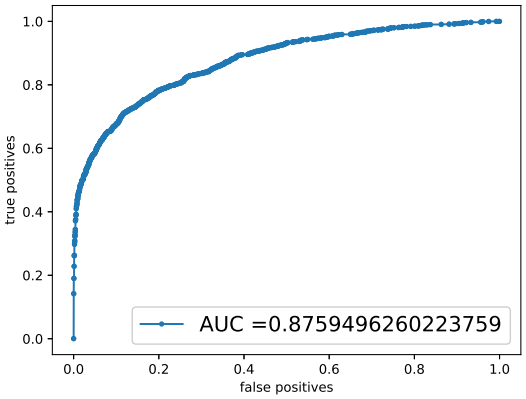
(b) DRB1*1302



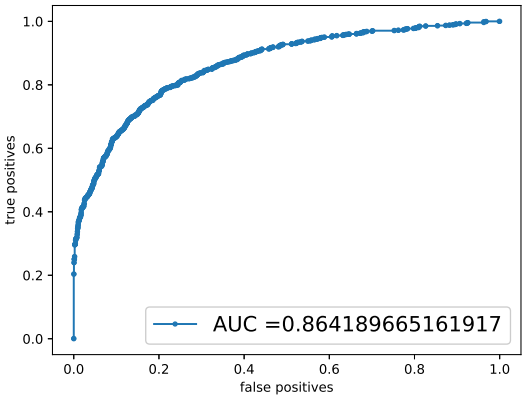
(c) DRB1*1501



(d) DRB1*1602

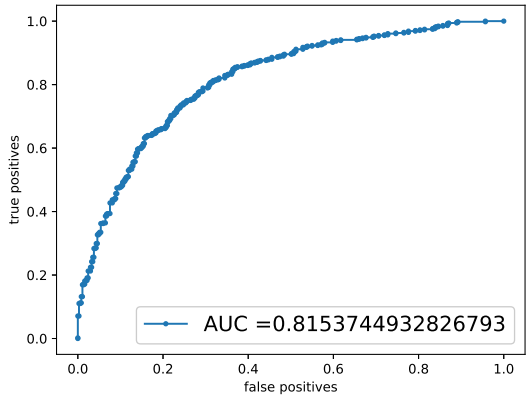


(e) DRB3*0101

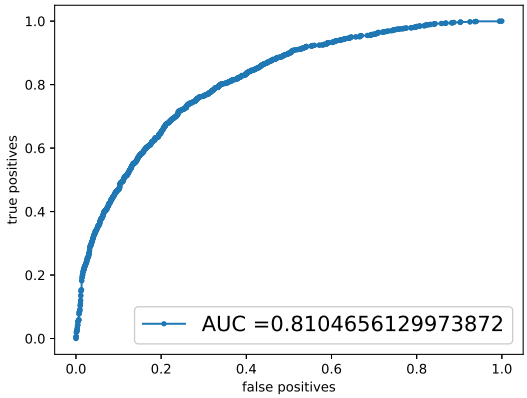


(f) DRB3*0202

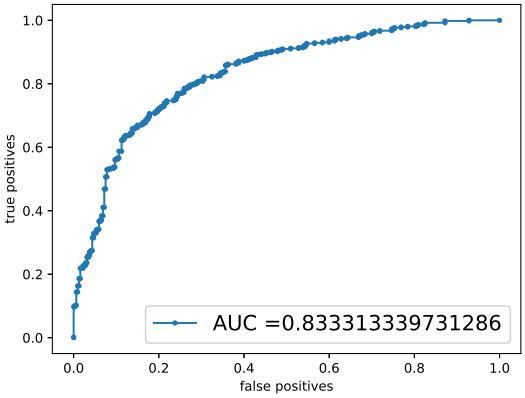
Figura 5.9: Predicciones para HLA-DR



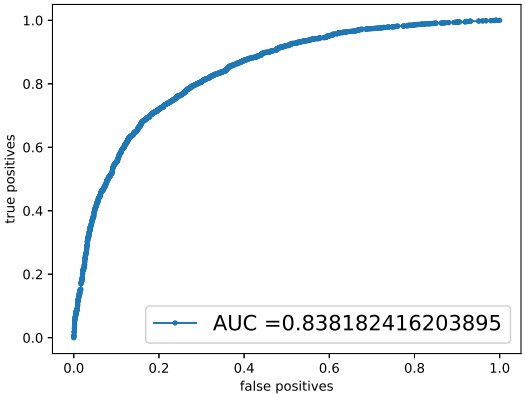
(a) DRB3*0301



(b) DRB4*0101

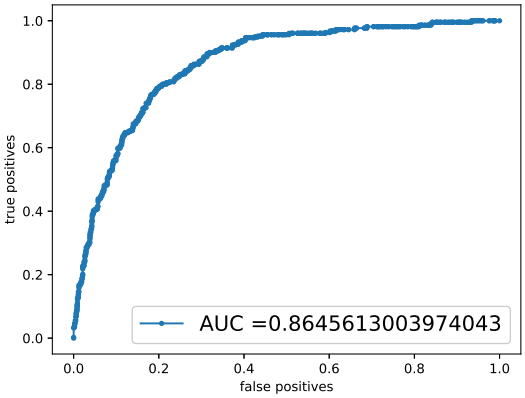


(c) DRB4*0103

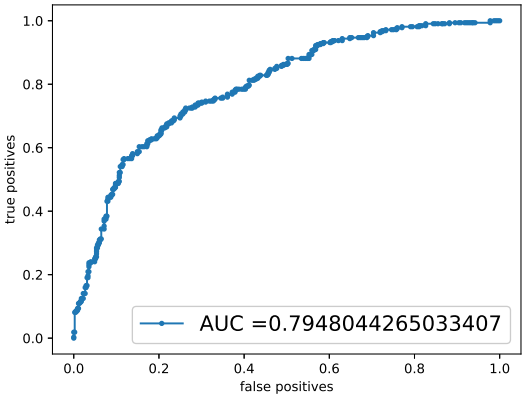


(d) DRB5*0101

Figura 5.10: Predicciones para HLA-DR



(a) H-2-IAb



(b) H-2-IAd

Figura 5.11: Predicción para H-2

5.2. Rendimiento predictivo AUC promedio en cada alelo

Al igual que el rendimiento predictivo AUC para cada molécula se mide el rendimiento que se tuvo para todo el alelo, promediando el valor AUC de todas las moléculas de cada alelo, HLA-DP, HLA-DQ, HLA-DR y H-2 de ratón. .

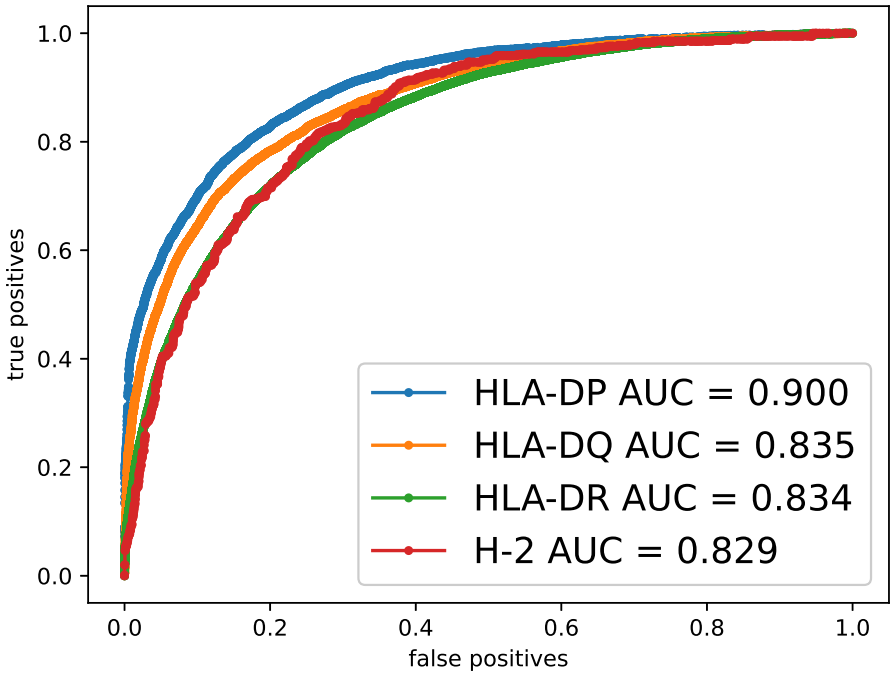


Figura 5.12: Rendimiento predictivo midiendo el AUC promedio de todas los alelos HLA y H-2 evaluados.

5.3. Comparaciones

Se comparó el método propuesto (algoritmo 2 y 3) contra algunos métodos documentados en el estado del arte, dos de ellos de gran relevancia en el ámbito de la predicción de péptidos-MHC-II, NetMHCIIPan 3.2 de Jensen et al. (2018) [50], el método de Degoot et al. (2018) [52] y NN-Align [51].

Las bases de datos moleculares utilizadas en todos los métodos de predicción presentados en las

tablas siguientes proceden de la IEDB. Cada método es entrenado con las mismas secuencias péptidos y los mismos valores de afinidad, y con el mismo umbral de afinidad (0.426) para medir el rendimiento predictivo. Jensen et al. (2018) [50] emplea una pseudosecuencia en el entrenamiento de la red neuronal artificial la cual se enfatiza mejora la precisión en la predicción. Esta pseudosecuencia no es de dominio público, para construirla utilizaron el método de Karosiene et al. (2013) [34]. A continuación se presentan las comparaciones de los alelos HLA-DP, HLA-DQ, HLA-DR y H-2 en las siguientes tablas.

5.3.1. Comparación del método propuesto contra otros métodos midiendo el AUC para el alelo HLA-DP

Alelo HLA-DP			
Alelo	**Propuesto 2**	**NetMHCIIPan 3.2 [50]**	**Trans-Allelic [52]**
HLA-DPA10103-DPB10401	0.919	0.961	0.939
HLA-DPA10103-DPB10201	0.901	0.916	0.933
HLA-DPA10103-DPB10301	0.905	0.902	—
HLA-DPA10201-DPB10101	0.881	0.948	0.922
HLA-DPA10201-DPB10501	0.882	0.948	0.926
HLA-DPA10201-DPB11401	0.925	0.930	—
HLA-DPA10301-DPB10402	0.888	0.844	0.927
Promedio	**0.900**	**0.921**	**0.929**

Tabla 5.1: Comparación del método propuesto contra otros métodos midiendo el AUC para el alelo HLA-DP

5.3.2. Comparación del método propuesto contra otros métodos midiendo el AUC para el alelo HLA-DQ

Alelo HLA-DQ			
Alelo	**Propuesto 2**	**NetMHCIIPan 3.2 [50]**	**Trans-Allelic [52]**
HLA-DQA10101-DQB10501	0.895	0.857	0.833
HLA-DQA10102-DQB10501	0.853	0.839	—
HLA-DQA10102-DQB10502	0.822	0.835	—
HLA-DQA10102-DQB10602	0.882	0.887	0.815
HLA-DQA10103-DQB10603	0.799	0.861	—
HLA-DQA10104-DQB10503	0.790	0.805	—
HLA-DQA10201-DQB10202	0.801	0.814	—
HLA-DQA10201-DQB10301	0.858	0.849	—
HLA-DQA10201-DQB10303	0.869	0.894	—
HLA-DQA10201-DQB10402	0.826	0.860	—
HLA-DQA10301-DQB10301	0.779	0.839	—
HLA-DQA10301-DQB10302	0.823	0.832	0.728
HLA-DQA10303-DQB10402	0.791	0.820	—
HLA-DQA10401-DQB10402	0.876	0.903	0.873
HLA-DQA10501-DQB10201	0.865	0.883	0.871
HLA-DQA10501-DQB10301	0.909	0.943	0.799
HLA-DQA10501-DQB10302	0.803	0.822	—
HLA-DQA10501-DQB10303	0.852	0.876	—
HLA-DQA10501-DQB10402	0.842	0.868	—
HLA-DQA10601-DQB10402	0.820	0.848	—
Promedio	**0.835**	**0.857**	**0.819**

Tabla 5.2: Comparación del método propuesto contra otros métodos midiendo el AUC para el alelo HLA-DQ

5.3.3. Comparación del método propuesto contra otros métodos midiendo el AUC para el alelo HLA-DR

Alelo HLA-DR				
Alelo	**Propuesto 2**	**NN-Align [51]**	**NetMHCIIPan 3.2 [50]**	**Trans-Allelic [52]**
DRB1*0101	0.800	0.837	0.823	0.786
DRB1*0301	0.799	0.808	0.835	0.725
DRB1*0401	0.774	0.768	0.798	0.756
DRB1*0404	0.781	0.815	0.810	0.744
DRB1*0405	0.810	0.771	0.844	0.794
DRB1*0701	0.860	0.844	0.857	0.825
DRB1*0801	0.818	—	0.844	—
DRB1*0802	0.825	0.826	0.749	0.716
DRB1*0901	0.813	0.623	0.779	0.757
DRB1*1001	0.898	—	0.923	—
DRB1*1101	0.849	0.822	0.876	0.833
DRB1*1201	0.870	—	0.800	0.824
DRB1*1301	0.851	—	0.857	—
DRB1*1302	0.876	0.822	0.810	0.718
DRB1*1501	0.815	0.754	0.831	0.784
DRB1*1602	0.868	—	0.883	—
DRB3*0101	0.875	0.855	0.906	0.674
DRB3*0202	0.864	—	0.869	—
DRB3*0301	0.815	—	0.734	0.765
DRB4*0101	0.810	0.811	0.817	0.802
DRB4*0103	0.833	—	0.841	—
DRB5*0101	0.838	0.789	0.846	0.820
Promedio	**0.834**	**0.796**	**0.833**	**0.770**

Tabla 5.3: Comparación del método propuesto contra otros métodos midiendo el AUC para el alelo HLA-DR

5.3.4. Comparación del método propuesto contra otros métodos midiendo el AUC para el alelo H-2

Alelo H-2		
Alelo	**Propuesto 2**	**NetMHCIIPan 3.2 [50]**
H-2-IAb	0.864	0.868
H-2-IAd	0.794	0.808
Promedio	**0.829**	**0.838**

Tabla 5.4: Comparación del método propuesto contra otros métodos midiendo el AUC para el alelo H-2

5.4. Eficiencia computacional

El tiempo de ejecución de las diferentes implementaciones se presenta en la tabla 5.5, para lo cual se tomó el tiempo de ejecución únicamente del algoritmo NN-Align para medir su eficiencia con cada una de las diferentes herramientas con las que se implementó. El tiempo de ejecución del preprocesamiento de datos no se tomó en cuenta para estas pruebas. Esta prueba se realizó para cada una de las redes con los siguientes parámetros: 100 épocas de entrenamiento y 10 neuronas en la capa oculta. Las características del equipo de cómputo donde se realizaron las pruebas son:

1. Procesador Intel Core i7-4790 con una CPU de 3.60GHz y 8 núcleos.
2. Memoria RAM de 8GB.
3. GPU GeForce GTX Titan X.

Implementación	**Tiempo (segundos)**
Python 3.6.9	4136.13
TensorFlow 2.1.1	3092.24
Lenguaje C	5.37

Tabla 5.5: Comparación de las diferentes implementaciones prediciendo el alelo H-2-IAb.

5.5. Discusión

Debido a la interacción extremadamente polimórfica que hay entre la unión péptido-MHC-II es difícil generar un modelo lo suficientemente preciso para predecir afinidades de unión cercanas a las reales, por ello, es que en el estado del arte el rendimiento predictivo o *performance* del método empleado se mide con base en la clasificación de péptidos utilizando la métrica Curva ROC, misma que genera un valor de precisión similar a la matriz de confusión. Para la medición del rendimiento predictivo las afinidades predichas se tienen que etiquetar a partir de cierto umbral en 0 y 1. El valor de precisión AUC (área bajo la curva) es lo que define la precisión de los métodos de predicción de péptidos-MHC-II. Algunos otros como el método de Shen et al. (2018) [53] utilizan el RMSE como medida de desempeño.

En el servidor NetMHCIIPan 3.2, Jensen et al. (2018) [50] utiliza una pseudosecuencia de 34 aminoácidos para el entrenamiento de la red neuronal artificial, lo cual enfatiza mejora la precisión del algoritmo para los alelos HLA. Esta pseudosecuencia se construye tomando 15 aminoácidos de la parte alfa y 19 aminoácidos de la parte beta de los alelos HLA, la pseudosecuencia no esta disponible para el dominio público pero se cita que se logró construir con el método de Karosiene et al. (2013) [34]. Desarrollar una pseudosecuencia que se genere a partir de los alelos HLA y H-2 podría mejorar la precisión del algoritmo 2, aunque se debe de cuidar el aspecto de que se genere a partir de un conjunto de datos estrictamente para el entrenamiento del algoritmo, que no se utilice para las pruebas y validaciones, ya que podría converger a un sobreentrenamiento.

Una técnica ampliamente utilizada en bioinformática para analizar conjuntos de datos de secuencias de aminoácidos, son los motivos o *motif* que permiten, a partir de un conjunto de secuencias de la misma longitud (k-mero), ver la frecuencia con la que los aminoácidos participan en una región específica k de todo el conjunto de datos. Esta técnica tiene que ser evaluada visiblemente, ya que la letra de los aminoácidos se presentan graficamente en un plano. Evaluar los núcleos péptidos (9-meros) del conjunto de entrenamiento podría servir para escoger los aminoácidos que tienen mayor relevancia y darle cierto peso en la generación del modelo de predicción. Reynisson et al. (2020) [54] analiza los motivos de los núcleos de unión para mejorar

la predicción en la versión 4.0 del servidor NetMHCIIPan, estudio que no se alcanzó a analizar a fondo por el tiempo de esta investigación.

En general, existen varias técnicas que podrían mejorar aún más la precisión del algoritmo, ya que en sí, el método NN-Align es lo suficientemente bueno, permitiendo procesar la información de secuencias péptidos-MHC-II de diferentes longitudes, lo que contribuye al análisis de todo el conjunto de datos disponible y no una sola longitud de datos de entrada como lo requiere normalmente una red neuronal artificial. Ésto último también requiere que la información más relevante sea la que se encuentre dentro del conjunto de datos de entrenamiento, ésto es, teniendo codificado cada aminoácido por un vector de una matriz de sustitución (PAM, BLOSUM, etc.), reducir la longitud final de los datos de entrada, de modo que se procese dentro de la RNA los datos más relevantes o que tengan mejor correlación entre sí.

Capítulo 6

Conclusiones y trabajo futuro

6.1. Conclusión

La predicción de péptidos-MHC-II es un tópico que aún está en constante estudio por la dificultad que se tiene para procesar los diferentes alelos HLA y producir un modelo de predicción lo suficientemente preciso para realizar pruebas *in-vitro* de péptidos que generen una respuesta inmune satisfactoria. La problemática que se tiene con este tipo de péptidos es el gran poliformismo que hay entre la unión a la molécula MHC-II.

Para procesar estos conjuntos de datos se necesitan de diversas técnicas que permitan disminuir la redundancia de datos contenida en los aminoácidos de las secuencias de entrenamiento, ya que la similitud de diversas secuencias péptidos generan de muchos óptimos locales en el entrenamiento de la red neuronal artificial. Para disminuir el efecto de la redundancia de datos se efectuaron diferentes técnicas documentadas en el estado del arte, pero esto no fue suficiente para converger a una solución lo suficientemente buena para compararse con otros métodos documentados.

Con base en lo documentado acerca de la redundancia de datos en el entrenamiento de la red neuronal artificial se llegó a la conclusión de que podría disminuirse el conjunto de entrenamiento a solamente los núcleos-péptidos que no favorecieran el sobreentrenamiento interno de la red en su fase de entrenamiento, esto es, generar en cada época nuevos conjuntos de entrenamiento para con ellos, propagar el error de los núcleos-péptidos no redundantes. Esto mejoró significativamente el error cuadrático medio del algoritmo, sin embargo aún no se tenía una predicción competitiva por los óptimos locales que generaba el modelo, lo cual se corrigió midiendo la media

aritmética de las predicciones generadas por el modelo en cinco iteraciones de entrenamiento. Esto se describe en los algoritmos 2 y 3.
Habiendo hecho lo anteriormente descrito se logró una predicción aceptable frente a otros métodos documentados y altamente reconocidos.
La hipótesis de esta investigación formulaba conseguir resultados lo suficientemente rápidos (tiempo menor a una hora), a través de TensorFlow, una herramienta muy utilizada para el procesamiento de grandes cantidades de datos utilizando redes neuronales artificiales sobre la GPGPU, hecho que sorprendentemente pudo mejorarse en tiempos utilizando solamente el lenguaje C.

Con lo anterior se concluye, en que se desarrolló un algoritmo lo suficientemente competitivo en la predicción de afinidades y péptidos-MHC-II que pudiera utilizarse para realizar pruebas *in-vitro* de péptidos HLA y H-2 con una precisión del 80 % al 90 %. Vale la pena mencionar que faltaría este tipo de pruebas para confirmar este grado de precisión en las pruebas *in-silico* realizadas en esta investigación.

6.2. Trabajo futuro

Se puede aplicar al método propuesto como trabajo futuro otras técnicas de predicción que se combinen y puedan generar resultados más precisos, éstas técnicas se conocen como métodos de consenso. Además se podrían disminuir los datos de entrenamiento aplicando técnicas de correlación para conseguir un conjunto de datos de entrenamientos nivelado entre péptidos muy similares y otros que no tengan mucha similitud, con el fin de generar un conjunto de datos que no dote de redundancia el entrenamiento de la red.

Otro trabajo futuro es la predicción de epítopos de células T. El servidor NetMHCIIPan 3.2 dispone en su página un conjunto de datos de proteínas en la que viene la secuencia de proteína a procesar, el tipo de alelo y la secuencia del epítopo a predecir. Esta prueba es necesaria para comprender la respuesta inmunitaria mediada por células y para generar mejores métodos de detección para la identificación de péptidos inmunogénicos y con ello mejorar el diseño de vacunas.

Por último sería llevar a cabo la implementación de un servidor web con infraestructura propia utilizando el método propuesto y alimentándolo con secuencias de pruebas *in-vitro* desarrolladas en el instituto.

Bibliografía

[1] Massimo Andreatta, Claus Schafer-Nielsen, Ole Lund, Søren Buus, and Morten Nielsen. NNAlign: A web-based prediction method allowing non-expert end-user discovery of sequence motifs in quantitative peptide data. *PLoS ONE*, 6(11), nov 2011.

[2] Flora Castellino, Guangming Zhong, and Ronald N. Germain. Antigen presentation by MHC class II molecules: Invariant chain function, protein trafficking, and the molecular basis of diverse determinant capture. 54(2):159–169, may 1997.

[3] Jerry H Brown, Theodore S Jardetzky, Joan C Gorga, Lawrence J Stern, Robert G Urban, Jack L Strominger, and Don C Wiley. Pillars article: three-dimensional structure of the human class II histocompatibility antigen HLA-DR1. Nature. 1993. 364: 33-39. *Journal of immunology (Baltimore, Md. : 1950)*, 194(1):5–11, jan 2015.

[4] Christopher J. Holland, David K. Cole, and Andrew Godkin. Re-directing CD4+ T cell responses with the flanking residues of MHC class II-bound peptides: The core is not enough, 2013.

[5] Paula Y. Arnold, Nicole L. La Gruta, Tim Miller, Kate M. Vignali, P. Scott Adams, David L. Woodland, and Dario A. A. Vignali. The Majority of Immunogenic Epitopes Generate CD4 + T Cells That Are Dependent on MHC Class II-Bound Peptide-Flanking Residues . *The Journal of Immunology*, 169(2):739–749, jul 2002.

[6] http://proyectoavatar.enfermeriacomunitaria.org/vacunas/historia-de-las vacunas.

[7] https://medlineplus.gov/spanish/ency/esp_imagepages/9394.htm.

[8] Joilyneth Ferreira and Antonietta Porco. Vacunas derivadas del análisis de los genomas: Vacunología inversa. *Interciencia*, 2008.

[9] Paola Toche. Panoramic Vision of the Inmune System. *Inmunología, Unidad De Medicina, Departamento De Clínica, Interna*, 2012.

[10] D. Yero Corona. De la secuencia de un genoma bacteriano a la identificación de candidatos vacunales. *Vaccimonitor*, 2006.

[11] Rino Rappuoli. Reverse vaccinology, a genome-based approach to vaccine development. In *Vaccine*, 2001.

[12] Morten Nielsen, Ole Lund, Søren Buus, and Claus Lundegaard. MHC Class II epitope predictive algorithms, 2010.

[13] https://scielo.conicyt.cl/scielo.php?script=sci_arttext&pid=s0718-381x2020000300331.

[14] J. Glynn Henry, Gary W. Heinke, and Héctor Javier Escalona y García. *Ingeniería ambiental*. Prentice Hall, 1999.

[15] Bruce Alberts, Alexander Johnson, Julian Lewis, Martin Raff, Keith Roberts, and Peter Walter. The Adaptive Immune System. 2002.

[16] https://www.genome.gov/es/genetics glossary/Aminoacido.

[17] https://www.nature.com/scitable/definition/peptide-317.

[18] https://www.genome.gov/es/genetics glossary/Alelo.

[19] https://en.wikipedia.org/wiki/K mer.

[20] https://en.wikipedia.org/wiki/IC50.

[21] https://es.wikipedia.org/wiki/Alineamiento_de_secuencias.

[22] Rodrigo Salas. Redes Neuronales Artificiales. *ReserachGate*, 2004.

[23] https://docs.python.org/3/faq/general.html#why-was-python-created-in-the-first-place.

[24] https://www.kdnuggets.com/2019/02/setup-python-environment-machine learning.html.

[25] http://www.bell labs.com/usr/dmr/www/chist.html.

[26] https://es.m.wikibooks.org/wiki/Archivo:Cpu gpu.svg.

[27] https://docs.nvidia.com/cuda/cuda-installation-guide linux/index.html.

[28] https://www.tensorflow.org/.

[29] https://es.wikipedia.org/wiki/TensorFlow.

[30] https://www.wired.com/2015/11/googles-open-source-ai-tensorflow-signals-fast-changing-hardware world/.

[31] https://modeliza.me/blog/ejecucion-eager-con-keras-y tensorflow/.

[32] Hans Peter Adams and James A. Koziol. Prediction of binding to MHC class I molecules. *Journal of Immunological Methods*, 185(2):181–190, 1995.

[33] Randi Vita, Laura Zarebski, Jason A. Greenbaum, Hussein Emami, Ilka Hoof, Nima Salimi, Rohini Damle, Alessandro Sette, and Bjoern Peters. The Immune Epitope Database 2.0. *Nucleic Acids Research*, 38(SUPPL.1), nov 2009.

[34] Edita Karosiene, Michael Rasmussen, Thomas Blicher, Ole Lund, Søren Buus, and Morten 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*, 65(10):711–724, oct 2013.

[35] Pingping Guan, Irini A. Doytchinova, Christianna Zygouri, and Darren R. Flower. MHCPred: A server for quantitative prediction of peptide-MHC binding. *Nucleic Acids Research*, 31(13):3621–3624, jul 2003.

[36] Ji Wan, Wen Liu, Qiqi Xu, Yongliang Ren, Darren R. Flower, and Tongbin Li. SVRMHC prediction server for MHC-binding peptides. *BMC Bioinformatics*, 7, oct 2006.

[37] G. P.S. Bhasin, Manoj Raghava. SVM based method for predicting HLA-DRB1*0401 binding peptides in an antigen sequence. *Bioinformatics*, 20:421–4230, febrero 2004.

[38] Pierre Dönnes and Arne Elofsson. Prediction of MHC class I binding peptides, using SVMHC. *BMC Bioinformatics*, 3, sep 2002.

[39] Chun Wei Tung and Shinn Ying Ho. POPI: Predicting immunogenicity of MHC class I binding peptides by mining informative physicochemical properties. *Bioinformatics*, 23(8):942–949, apr 2007.

[40] Jesper Salomon and Darren R. Flower. Predicting Class II MHC-Peptide binding: A kernel based approach using similarity scores. *BMC Bioinformatics*, 7, nov 2006.

[41] Hiroshi Mamitsuka. Predicting peptides that bind to MHC molecules using supervised learning of hidden Markov models. *Proteins: Structure, Function and Genetics*, 33(4):460–474, dec 1998.

[42] Hideki Noguchi, Ryuji Kato, Taizo Hanai, Yukari Matsubara, Hiroyuki Honda, Vladimir Brusic, and Takeshi Kobayashi. Hidden Markov Model-Based Prediction of Antigenic Peptides That Interact with MHC Class II Molecules. *Journal of Bioscience and Bioengineering*, 94(3):264–270, 2002.

[43] Chenhong Zhang, Mikelis G. Bickis, Fang Xiang Wu, and Anthoy J. Kusalik. Optimally-connected Hidden Markov models for predicting MHC-binding peptides. *Journal of Bioinformatics and Computational Biology*, 4(5):959–980, oct 2006.

[44] C. J. Savoie, N. Kamikawaji, T. Sasazuki, and S. Kuhara. Use of BONSAI decision trees for the identification of potential MHC class I peptide epitope motifs. *Pacific Symposium on Biocomputing. Pacific Symposium on Biocomputing*, pages 182–189, 1999.

[45] Yohan Kim, Julia Ponomarenko, Zhanyang Zhu, Dorjee Tamang, Peng Wang, Jason Greenbaum, Claus Lundegaard, Alessandro Sette, Ole Lund, Philip E. Bourne, Morten Nielsen, and Bjoern Peters. Immune epitope database analysis resource. *Nucleic Acids Research*, 40(W1):W525, jul 2012.

[46] Morten Nielsen, Claus Lundegaard, Peder Worning, Sanne Lise Lauemøller, Kasper Lamberth, Søren Buus, Søren Brunak, and Ole Lund. Reliable prediction of T-cell epitopes using neural networks with novel sequence representations. *Protein Science*, 12(5):1007–1017, may 2003.

[47] Morten Nielsen, Claus Lundegaard, and Ole Lund. Prediction of MHC class II binding

affinity using SMM-align, a novel stabilization matrix alignment method. *BMC Bioinformatics*, 8, apr 2007.

[48] Stephen F. Altschul. Amino acid substitution matrices from an information theoretic perspective. *Journal of Molecular Biology*, 219(3):555–565, jun 1991.

[49] https://web.archive.org/web/20060903191751/http://informatics.umdnj.edu/bioinformatics/courses/

[50] Kamilla Kjærgaard Jensen, Massimo Andreatta, Paolo Marcatili, Søren Buus, Jason A. Greenbaum, Zhen Yan, Alessandro Sette, Bjoern Peters, and Morten Nielsen. Improved methods for predicting peptide binding affinity to MHC class II molecules. *Immunology*, 154(3):394–406, jul 2018.

[51] Morten Nielsen and Ole Lund. NN-align. An artificial neural network-based alignment algorithm for MHC class II peptide binding prediction. *BMC Bioinformatics*, 10(1):296, sep 2009.

[52] Abdoelnaser M. Degoot, Faraimunashe Chirove, and Wilfred Ndifon. Trans-allelic model for prediction of peptide:MHC-II interactions. *Frontiers in Immunology*, 9(JUN), jun 2018.

[53] Wen-Jun Shen, Xun Zhang, Shaohong Zhang, Cheng Liu, and Wenjuan Cui. The Utility of Supertype Clustering in Prediction for Class II MHC-Peptide Binding. *Molecules*, 23(11):3034, nov 2018.

[54] Birkir Reynisson, Carolina Barra, Saghar Kaabinejadian, William H. Hildebrand, Bjoern Peters, Bjoern Peters, Morten Nielsen, and Morten Nielsen. Improved Prediction of MHC II Antigen Presentation through Integration and Motif Deconvolution of Mass Spectrometry MHC Eluted Ligand Data. *Journal of Proteome Research*, 19(6):2304–2315, jun 2020.