© Academia Journals 2020

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

Ing. Jacob Ezequiel Merecías Pérez[1], Dr. Marco Antonio Castro Liera[2],
Dr. Carlos Angulo[3] y MSC. Joel Artemio Morales Viscaya[4]

***Resumen*— En este trabajo se presentan los resultados que se obtuvieron prediciendo afinidades de péptidos con la molécula MHC-II. Se probó el algoritmo NN-Align que en la literatura muestra una precisión aceptable para la predicción de afinidades. La investigación se enfocó en medir el rendimiento predictivo del algoritmo e implementarlo en tres importantes herramientas de programación para medir su eficiencia computacional, ya que, los conjuntos de datos que se procesan en estas redes suelen ser muy pesados. La eficiencia computacional resultante en cada implementación empleada muestra que para procesar estos datos es necesario minimizar la cantidad de recursos de procesamiento que pueda utilizar el entorno de programación utilizado, e implementar el método en un lenguaje o herramienta de programación que trabaje sin intermediarios de código para el procesamiento eficiente del conjunto de datos.**

***Palabras clave*—MHC-II, Redes Neuronales Artificiales, Péptidos.**

Introducción

El desarrollo de una vacuna comienza con la identificación de componentes o estructuras únicas en un microorganismo capaces de generar una respuesta inmune protectora. Con las técnicas convencionales, éste puede ser un proceso largo y tedioso, ya que comúnmente se realizan pruebas de ensayo y error hasta encontrar el candidato más adecuado, sin tomar en cuenta la dificultad con la que pueda ser cultivado el microorganismo en el laboratorio (Ferreira y Porco, 2008). Con el avance de las ciencias de la computación y la bioinformática, se pueden desarrollar algoritmos que permitan un análisis profundo de la información contenida en bases de datos moleculares, que faciliten el encontrar patrones para la creación de nuevas vacunas, específicamente en microorganismos no cultivables. Actualmente existen diversos métodos que analizan la secuencia de una proteína en busca de estructuras de aminoácidos que generen una respuesta inmune en el cuerpo. Se han utilizado diferentes métodos para la predicción de estas estructuras, incluidas las redes neuronales artificiales (ANN) (Karosiene et al. 2013), los árboles de decisión (Savoie et al. 1999), los modelos ocultos de markov (HMM) (Noguchi et al. 2002), los métodos de regresión (EL-Manzalawy et al. 2011), las máquinas de vectores de soporte (SVM) (Manoj Bhasin y G. P. S. Raghava, 2004) y los métodos de consenso, que es, una combinación de varios de los métodos anteriores (Wang et al. 2008).

La afinidad de unión que tiene una molécula de MHC-II a un péptido está determinada casi por completo por la secuencia de aminoácidos del núcleo del péptido, regularmente 9 aminoácidos (*9-meros*) (Holland et al. 2013). 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 (i.e. Arnold et al. 2002).

En este trabajo se presentan avances del desarrollo de un predictor de afinidad de péptidos utilizando una red neuronal artificial capaz de obtener un porcentaje de precisión considerable respecto a los métodos que se encuentran actualmente. Este predictor se implementa en tres diferentes herramientas de programación para evaluar su eficiencia computacional mostrando resultados relevantes para implementaciones futuras, eficientes en el ámbito computacional. A continuación, se expone el método empleado y los resultados de la investigación.

Descripción del Método

Datos

El conjunto de datos se obtuvo del servidor *NetMCHIIpan*, el cual contiene afinidades de unión péptido-MHC-II recuperados de la Base de Datos de Inmunodeficiencia Epítopo y Análisis de Recursos (IEDB, por sus

[1] El Ing. Jacob Ezequiel Merecías Pérez es estudiante de la Maestría en Sistemas Computacionales del Instituto Tecnológico Nacional de México, campus La Paz. ezequiel.merecias@gmail.com
[2] El Dr. Marco Antonio Castro Liera es Maestro de la Maestría en Sistemas Computacionales del Instituto Tecnológico Nacional de México, campus La Paz. marco.cl@lapaz.tecnm.mx
[3] El Dr. Carlos Eliud Angulo Valadez es investigador del Grupo de Inmunología y Vacunología del Centro de Investigaciones Biológicas del Noroeste, S.C., en La Paz B.C.S. eangulo@cibnor.mx
[4] El MSC. Joel Artemio Morales Viscaya es Maestro de la Maestría en Sistemas Computacionales del Instituto Tecnológico Nacional de México, campus La Paz. iscviscaya@gmail.com

© Academia Journals 2020

siglas en inglés) del 2016. Para el presente método se realizaron los experimentos con los alelos descritos en el Cuadro 1, que contiene 5 conjuntos de datos de entrenamiento y 5 conjuntos de datos de validación.

Cada renglón del conjunto de datos contiene en la primera columna la secuencia de aminoácidos del péptido; en la segunda columna los valores de IC_{50} normalizados a valores entre 0 y 1 utilizando la fórmula $1 - \frac{Log(IC_{nM})}{Log(50000)}$ tal como explica Nielsen et al. (2003); y en la última columna el alelo al que pertenece el péptido.

Se toma un umbral de valores menores a 500_{nM} como afines y los valores mayores a este umbral como no afines. Los péptidos con el valor de afinidad ya normalizado mayores a 0.426 se etiquetan con el valor de 1 (afines o aglutinantes) y los valores menores a éste se etiquetan como 0 (no afines o no aglutinantes) esto con el fin de hacer una predicción cualitativa.

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

Cuadro 1. Conjunto de datos 2016 (IEDB).

Red Neuronal Artificial

El método que se empleó consiste en una red neuronal artificial (RNA) que consta de 270 neuronas en la capa de entrada, 10 neuronas en la capa oculta y 1 neurona en la capa de salida.

```
AlgoritmoPrincipal(BD)
1: XEnt, XPrueba, YEnt, YPrueba=BD
2: NcsEnt,NcsPrueba=Nucleos(XEnt,XPrueba)
3: Codificar(NcsEnt, NcsPrueba)
4: Normalizar(YEnt, YPrueba)
5: W=NN_Align(NcsEnt, YEnt)
6: para 1 hasta NPeptidos hacer
7:     YPred = Predecir(NcsPrueba, W)
8: MSE=Error(YPred, YPrueba)
```

Figura 1. Algoritmo principal.

El número de neuronas en la capa de entrada es producto de la codificación de la secuencia de aminoácidos de los péptidos a valores numéricos obtenidos de la matriz de sustitución *BLOSUM50* del Centro Nacional para la Información Biotecnológica (NCBI). Se utilizó *BLOSUM50* por el buen desempeño que tiene frente a la codificación *One-Hot* en la fase de entrenamiento de la RNA.

El algoritmo comienza por la obtención de todos los núcleos de 9 aminoácidos (*9-meros*) que puede contener un péptido, esto es, si un péptido contiene 12 aminoácidos (*12-meros*) los núcleos posibles que debe tener son 4, o sea, el número de aminoácidos menos 8, como se muestra en la Figura 2. En el mismo proceso se obtienen las regiones flanqueantes (PFR), que son los aminoácidos que se encuentran al extremo derecho e izquierdo del núcleo, tomándose de 0 a 3 aminoácidos. Teniendo ya el núcleo y las PFR, se codifican usando la matriz de sustitución *BLOSUM50*, que tiene de longitud 25 valores de sustitución. Esto se hace mediante el siguiente proceso: se codifica el núcleo formándose un vector de 225 valores (9 x 25); los aminoácidos en cada PFR (derecho e izquierdo) se codifican y se promedian quedando un vector de 50 valores (2 x 25).

Después de codificar los aminoácidos del núcleo y los PFR se crean vectores con la información de las longitudes del péptido y de los PFR como se describe en lo siguiente: se forma un vector de 2 valores, siendo el primer valor la longitud del péptido y el segundo valor el resultado de la resta de 1 menos la longitud del péptido (LPEP, 1 - LPEP); se forma otro vector de 2 valores, siendo el primer valor la longitud del PFR derecho y el segundo valor el resultado de la resta de 1 menos la longitud del PFR derecho dividido entre 3 (LPFRD, 1 - LPFRD/3); por último otro vector de 2 valores con la relación anteriormente descrita pero con la información del PFR izquierdo.

Concatenando todos los vectores resultantes tenemos un total de 270 entradas que se obtienen en el primer y segundo paso del algoritmo principal descrito en la Figura 1.

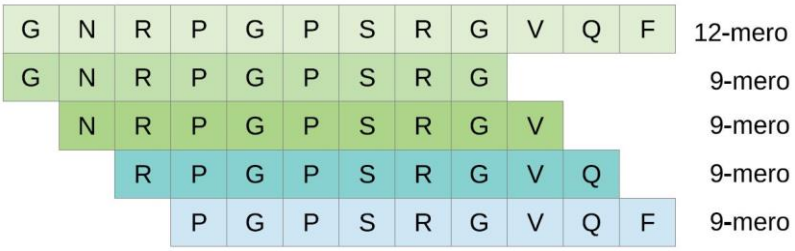

Figura 2. Núcleos (*9-meros*) posibles en un péptido.

Seguido de lo anterior se procede a normalizar las afinidades péptido-MHC-II en unidad de medida Nanomolar (nM) a valores entre 0 y 1 con la siguiente fórmula: $1 - \frac{Log(IC_{nM})}{Log(50000)}$.

Teniendo todos los núcleos posibles de cada péptido con sus afinidades normalizadas se introducen en el algoritmo descrito en la Figura 3 para su procesamiento.

NN-Align(NcsEnt, YEnt)

1: **W=Pesos Aleatorios**
2: **para** 1 hasta **Épocas hacer**
3: **para** i hasta **nNcsEnt hacer**
4: $MejorNucleo_i$=**MejorNucleo**($NcsEnt_i$)
5: $YPred_i$=**Predecir**($MejorNucleo_i$,W)
6: **Error=Error**($YPred_i$, $YEnt_i$)
7: **Retropropagación**(Error, W)
8: **Retornar** W

Figura 3. Algoritmo NN-Align.

El algoritmo NN-Align de la Figura 3 se conoce como alineamiento por redes neuronales artificiales (Neural Network Alignment, NN-Align por sus siglas en inglés), publicado por Andreatta et al. (2011), permite alinear un conjunto de secuencias de aminoácidos o nucleótidos para encontrar un modelo que permita clasificar o predecir secuencias de aminoácidos no conocidas.

Este algoritmo es un modelo de red neuronal artificial cuyo proceso comienza con la generación de una matriz *W* de pesos aleatorios, que mediante un número asignado de iteraciones (*Épocas)* y una función de error, se ajustará a los datos de entrenamiento y generará un modelo de predicción. El algoritmo tiene una particularidad en la función de error que permite evaluar previamente el error de todos los subconjuntos de datos posibles dentro del enésimo dato de entrenamiento como se muestra en la Figura 4 (linea 4 del algoritmo en la Figura 3), utilizando así, el enésimo dato con menor error para mediante algún método de cálculo de gradiente (linea 7 del algoritmo en la Figura 3) propagar el mínimo error obtenido hacia las neuronas y pesos predecesores a la neurona de salida.

El algoritmo retorna la matriz *W* de pesos óptima, que fue ajustada dentro de las iteraciones (*Épocas*) previas. Esta matriz de pesos servirá en el algoritmo principal (Figura 1) para predecir los valores de afinidad de péptidos desconocidos.

© Academia Journals 2020

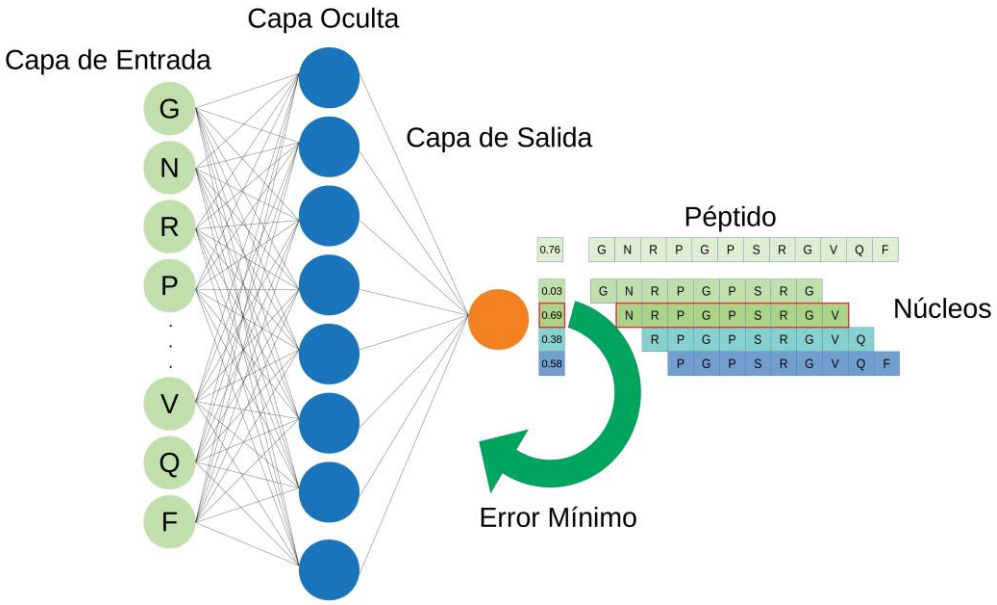

Figura 4. Funcionamiento de NN-Align.

Herramientas de programación

El algoritmo se implementó en tres versiones diferentes para evaluar su eficiencia computacional. En el lenguaje de programación *Python* versión 3.6.9; utilizando la librería *Keras* de *TensorFlow* versión 2.1.1 compilado para su ejecución con *CUDA*; y en el lenguaje de programación C.

Todas las implementaciones se probaron con solamente el proceso de ejecución del programa que contiene el algoritmo NN-Align y en una computadora con las siguientes características:

- Procesador Intel Core i7-4790, CPU 3.60GHz, 8 núcleos.
- Memoria RAM 8GB.
- GPU GeForce GTX TITAN X/PCIe/SSE2.

Resultados

La prueba se realizó como se describe en la Figura 5, utilizando 5 conjuntos de datos de entrenamiento y 5 conjuntos de datos de validación donde se incluyen todos los alelos, para posteriormente procesarlos antes de ingresarlos a la red neuronal artificial. Después de entrenar el modelo haciendo uso de cada conjunto de datos de entrenamiento en una iteración k-fold se procedió a realizar la prueba con cada una de las diferentes implementaciones. Se utilizó el cálculo del error cuadrático medio (*MSE*) para evaluar el algoritmo.

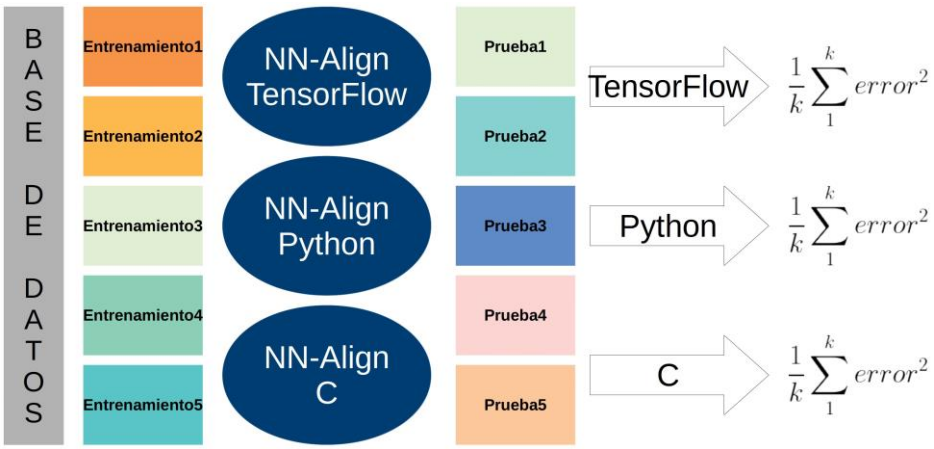

Figura 5. K-fold Cross-Validation del algoritmo principal con NN-Align en tres diferentes implementaciones, python, tensorflow y C.

© Academia Journals 2020

En la Figura 6 se puede visualizar que el modelo en el caso del alelo HLA-DP se ajusta a los datos de prueba, teniendo un error cuadrático medio de 0.007148. Un resultado similar en las demás pruebas que se realizaron.

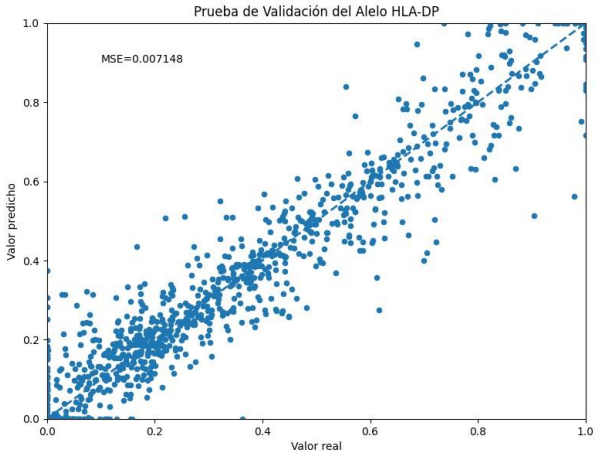

Figura 6. 1000 valores de afinidad predichos seleccionados aleatoriamente del total de afinidades predichas del alelo HLA-DP.

El tiempo de ejecución de las diferentes implementaciones se presenta en el Cuadro 2, para lo cual se tomó el tiempo de ejecución únicamente del algoritmo NN-Align para medir su eficiencia en cada una de las diferentes herramientas donde 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, 10 neuronas en la capa oculta y el cálculo del error cuadrático medio para el ajuste de los pesos.

Herramienta/Lenguaje	Tiempo (segundos)
Python 3.6.9	4136.13
TensorFlow 2.1.1	3092.24
Lenguaje C	5.37

Cuadro 2. Comparación de las diferentes implementaciones prediciendo el alelo H2-IAb.

Comentarios Finales

Resumen de resultados

En esta investigación se estudió el funcionamiento del algoritmo NN-Align para predecir afinidades de péptidos. Como tal, es una red neuronal artificial con una adaptación en la función de error para evaluar cada uno de los subconjuntos de una secuencia de aminoácidos (péptidos) antes de propagar el mínimo error a las neuronas predecesoras a la capa de salida y sus pesos.

Se reprodujeron tres implementaciones para comprobar la eficiencia computacional del algoritmo, ya que, las bases de datos de péptidos albergan una gran cantidad de datos, lo cual, genera un lento entrenamiento de la red neuronal artificial en un lenguaje de programación convencional. Se implementó el algoritmo en el lenguaje de programación *Python* y otra versión en el lenguaje de programación *C*, a su vez se implementó otra versión en Python utilizando la biblioteca *TensorFlow*, que permite la creación de redes neuronales artificiales y su ejecución en una GPGPU.

Conclusiones

Los resultados demuestran que efectivamente, el algoritmo NN-Align tiene una precisión alta frente a los métodos actuales de predicción de afinidades de péptidos-MHC-II. También se pudo comprobar que para el óptimo desempeño del entrenamiento de este algoritmo es necesario llevarlo acabo en una herramienta de programación que no requiera de intermediarios de código tal como lo es el lenguaje *C* o alguna herramienta similar que se ejecute a bajo nivel, ya que, por procesar una gran cantidad de datos, el algoritmo requiere de mayor poder computacional, el cual no se aprovecha si se utilizan lenguajes de programación interpretados o bibliotecas de código que requieren de un intérprete.

Recomendaciones

Esta investigación da campo abierto para proseguir con la predicción de epítopos de células T utilizando el enfoque de optimización de código que aplicamos, con el fin de reducir los tiempos de ejecución. A su vez, implementar el código en el lenguaje *C* da la posibilidad para migrarlo a *CUDA C/C++* y evaluar la eficiencia frente a otras bibliotecas de generación de redes neuronales artificiales con ejecución en una GPGPU, tales como *TensorFlow* y su variante *PyTorch*.

Referencias

Ferreira Joilyneth y Porco Antonietta. "Vacunas derivadas del análisis de los genomas: vacunología inversa," *Interciencia*, Vol. 33, No. 5, 2008.

Edita Karosiene, Michael Rasmussen, Thomas Blicher, Ole Lund, Søren Buus, y 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,* Vol. 65, No. 10, 2013.

CJ. Savoie, N. Kamikawaji, T. Sasazuki y S. Kuhara. "Use of BONSAI decision trees for the identification of potential MHC class I peptide epitope motifs," *Pacific Symposium on Biocomputing '99*, páginas 182-189, 1999.

Hideki Noguchi, Ryuji Kato, Taizo Hanai, Yukari Matsubara, Hiroyuki Honda, Vladimir Brusic y Takeshi Kobayashi. "Hidden Markov model-based prediction of antigenic peptides that interact with MHC class II molecules," *Journal of Bioscience and Bioengineering,* Vol. 94, No. 3, 2002.

Yasser EL-Manzalawy, Drena Dobbs y Vasant Honavar. "Predicting MHC-II Binding Affinity Using Multiple Instance Regression," *IEEE/ACM Transactions on Computational Biology and Bioinformatics,* Vol. 8, No. 4, 2011.

Manoj Bhasin y G. P. S. Raghava. " SVM based method for predicting HLA-DRB1 * 0401 binding peptides in an antigen sequence," *Bioinformatics by Oxford Academic,* Vol. 20, No. 3, 2004.

Peng Wang, John Sidney, Courtney Dow, Bianca Mothé, Alessandro Sette, y Bjoern Peters. "A systematic assessment of MHC class II peptide binding predictions and evaluation of a consensus approach," *PLoS Computational Biology,* Vol. 4, No. 4, 2008.

Christopher J Holland, David K Cole y Andrew Godkin. "Re-directing CD4+ T cell responses with the flanking residues of MHC class II-bound peptides: The core is not enough," *Frontiers in Immunology,* Vol. 4, 2013.

Paula Y. Arnold, Nicole L. La Gruta, Tim Miller, Kate M. Vignali, P. Scott Adams, David L. Woodland y 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,* Vol. 169, No. 2, 2002.

Morten Nielsen, Claus Lundegaard, Peder Worning, Sanne Lise Lauemøller, Kasper Lamberth, Søren Buus, Søren Brunak y Ole Lund. "Reliable prediction of T-cell epitopes using neural networks with novel sequence representations," *Protein Science,* Vol. 12, No. 5, 2003.

Massimo Andreatta, Claus Schafer-Nielsen, Ole Lund, Søren Buus y Morten Nielsen. "NNAlign: A web-based prediction method allowing non-expert end-user discovery of sequence motifs in quantitative peptide data," *PLoS ONE,* Vol. 6, No. 11, 2011.

Notas Biográficas

El **Ing. Jacob Ezequiel Merecías Pérez** es estudiante de la Maestría en Sistemas Computacionales del Instituto Tecnológico Nacional de México, campus La Paz, en el cual también, realizó sus estudios de licenciatura. Presentó en el Congreso Internacional de Supercómputo en México (ISUM 2014) un artículo titulado Parallelization of filter BSSWSS on GPGPU for classifying cancer subtypes with SVM del cual fue coautor.

El **Dr. Marco Antonio Castro Liera** obtuvo su doctorado en la Universidad Central "Marta Abreu" de Las Villas en Santa Clara, Cuba. Su experiencia profesional incluye puestos en el Centro SCT de BCS, la Universidad Autónoma de Baja California Sur y actualmente tiene una plaza como profesor titular en la División de Estudios de Posgrado e Investigación del IT de La Paz donde ha dirigido 11 tesis de maestría. El profesor Castro ha publicado su trabajo sobre Algoritmos Genéticos, Cómputo Distribuido e Identificación de Sistemas en varios eventos internacionales, revistas arbitradas e indexadas y capítulos de libros. Es coautor del libro de texto "Programación Paralela Aplicada en Optimización".

El **Dr. Carlos Eliud Angulo Valadez** es Investigador Titular del Centro de Investigaciones Biológicas del Noroese, S.C. SNI 2. Autor de 83 artículos JCR sobre biotecnología, inmunología y vacunología.

El **MSC. Joel Artemio Morales Viscaya** es Maestro del Instituto Tecnológico Nacional de México, campus La Paz.