Congr. Int. Ing. Electrón. Mem. Electro 2012, vol.34, pp. 7-11 Chihuahua, Chih. México
http://depi.itchihuahua.edu.mx/display/memorias_electro/MemoriaElectro2012.zip
ISSN 1405-2172

MODELO NEURONAL DE DIAGNÓSTICO DE CÁNCER DE RIÑÓN UTILIZANDO INFORMACIÓN PROTEÓMICA DE ORINA

Orona Medina Dorian, Celis Porras Jesús, Guerrero Rivera Rubén
Instituto Tecnológico de Durango.
Blvd. Felipe Pescador # 1830, ote.
C.P. 34080 Durango, Dgo.
d.oronamed@gmail.com , jcelisp@gmail.com

RESUMEN

El cáncer de riñón es el sexto tipo más frecuente de cáncer y la décima causa más común de muerte por cáncer en los hombres. Es el octavo cáncer más frecuente en las mujeres [1]. Por medio del análisis proteómico en orina es posible identificar proteínas que permiten diagnosticar la enfermedad o pronosticar la evolución de la misma. Existen varios métodos utilizados en la clasificación de datos como son los arboles de decisiones, análisis discriminante, sistemas borrosos entre otros.

En este trabajo se propone el uso de un modelo de diagnóstico de cáncer de riñón utilizando un método alternativo como son las redes neuronales (RNA), las cuales son algoritmos de cálculo computacional que tienen la capacidad de aprender inspirados en el funcionamiento del cerebro. Los resultados obtenidos de este trabajo muestran un porcentaje de 95% de acierto en el diagnostico por medio del modelo desarrollado.

1. INTRODUCCIÓN

Una red neuronal es una red u organización de neuronas. Fisiológicamente hablando, una neurona es la célula encargada de darle forma al sistema nervioso de los seres vivos [2]. Esta organización de neuronas, por ejemplo, logra el objetivo de especializarse en ciertos y determinados procesos cognitivos como leer, hablar, sumar, reconocer la cara de las personas, etc. Esta especialización se logra incentivando o fortaleciendo algunas conexiones entre algunas neuronas y desechando otras conexiones [3].

Desde el punto de vista matemático o computacional, una red neuronal artificial es una abstracción matemática que trata de imitar el funcionamiento del cerebro y es representada a través de un código en algún procesador de computadora. Las redes neuronales en computación aprenden y se especializan en determinadas actividades, inclusive, al igual que las redes fisiológicas, ampliando el aprendizaje para incluir nueva información.

Este aprendizaje se logra mediante el entrenamiento de la red el cual consiste en exponer la red a datos para que los procese, pero indicándole cual debe ser el resultado a obtener, dentro de la lógica del programa que representa a la red. Este entrenamiento se aprecia en el ajuste de W(x,y), conocido también como peso (*weight*), y b(x), conocido como desplazamiento (*bias*), para promover las conexiones que propician la disminución del error, en la figura 1 se muestra la arquitectura de una neurona artificial. La idea central de una red neuronal artificial es que estos parámetros pueden ser ajustados de tal manera que la red exhiba el comportamiento esperando, es decir, se puede entrenar una red ajustando los valores de peso y desplazamiento para efectuar una labor en particular o la red misma puede ajustar estos parámetros para lograr la meta planteada. En este trabajo la red neuronal se programó empleando Matlab versión R2012a.

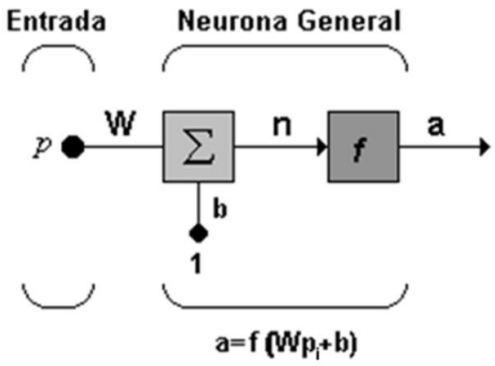

Figura 1. Arquitectura de una neurona artificial

2. RED BACK-PROPAGATION

Las redes de back-propagation (retropropagación) fueron creadas generalizando las reglas de

aprendizaje de Widrow-Hoff a redes de múltiples capas como se muestra en la figura 2 y funciones de transferencia diferenciables. Cerca del 80% de las aplicaciones prácticas de redes neuronales están basadas en este tipo de redes. Los vectores de entrada y sus correspondientes vectores de salida se emplean para entrenar una red hasta que esta pueda aproximar una función, asociar los vectores de entrada de la manera que el diseñador lo haya definido. Redes con bias, una capa con neuronas con función de transferencia sigmoidal y la capa de salida con función de transferencia lineal son capaces de aproximar cualquier función con número finito de discontinuidades [4].

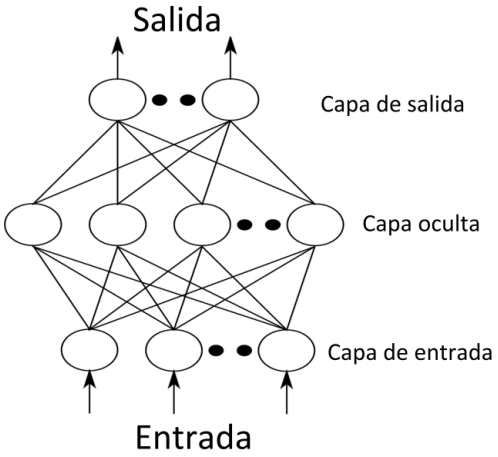

Figura 2. Arquitectura de una red back-propagation

El diseño de una red de retropropagación no está completamente restringido por la naturaleza del problema a ser resuelto. El número de entradas a la red está restringido por el número de entradas requeridas por el problema, y el número de neuronas de la capa de salida está restringido por el número de salidas requeridas por el problema; sin embargo el número de capas entre las capas de entrada y salida así como el número de neuronas en cada una de estas capas intermedias queda bajo la decisión del diseñador.

Las redes de retropropagación una vez entrenadas, usualmente generan respuestas razonables cuando se le presentan entradas que nunca han visto, esto se conoce como inferencia. Típicamente una nueva entrada conducirá a una salida similar a la salida correcta para los vectores de entrada usados en el entrenamiento similar a la nueva entrada presentada. Esta propiedad de inferencia hace posible que se entrene una red en un grupo representativo de pares de entrada/salida y se obtengan buenos resultados sin entrenar la red con todos los posibles pares de entrada/salida.

El entrenamiento de las redes de retropropagación puede conducir a un mínimo local en vez de un mínimo global. Este mínimo local podría ser satisfactorio, pero si no lo es, una red con más neuronas podría ser la solución. El problema es que el número de neuronas a añadir, así como la configuración en capas, no es algo obvio; y los valores iniciales de peso y desplazamiento podrían generar soluciones diferentes, por lo cual deberá seguirse un proceso de ensayo y error. Dicho de otra forma, aunque la red converja eso no quiere decir que el resultado que genere sea el correcto.

La retropropagación puede ser mejorada en dos formas: usando heurística o usando métodos de optimización más poderosos. Dentro de la heurística entran métodos como el uso de momento y la velocidad de aprendizaje adaptativo. El momento decrece la sensibilidad de la retropropagación a los pequeños detalles en la superficie del error. Esto ayuda que la red no de quede varada en un mínimo local que evite que llegue al mínimo global. El tiempo de entrenamiento puede disminuirse usando una velocidad de aprendizaje adaptativo la cual mantiene las velocidades de aprendizaje tan grande como sea posible mientras el aprendizaje sea estable. La velocidad de aprendizaje es responsable de la complejidad de las superficies locales del error. La optimización Levenberg-Marquardt hace los tiempos de entrenamiento más cortos, este métodos es más sofisticado que el algoritmo de disminución de gradiente.

3. METODOLOGIA

Se diseñó un modelo basado en una red neuronal back-propagation para diagnosticar el estado de salud del paciente por medio de tres diferentes clases: cáncer renal, enfermedad renal y el grupo de control (healthy control). El proceso de creación del modelo neuronal se realizó a partir de los siguientes pasos:

1. Recopilación de datos.
2. Creación del modelo neuronal.
3. Entrenar y validar el modelo.

ISSN 1405-2172

3.1. Recopilación de datos

Las muestras de orina fueron recolectadas de más de 20 centros clínicos, la información obtenida proviene de un espectrómetro de masas, el cual es un instrumento que permite analizar con gran precisión la composición de diferentes elementos químicos, separando los núcleos atómicos en función de su relación masa-carga (m/z). La base de datos es de 768 pacientes por 5010 proteínas (ver Tabla 1), para ello se utilizaron dos técnicas acopladas la CE y ESI-TOF/MS (Capillary Electrophoresis Mass Spectrometry). Por lo anteriormente mencionado no hay problema en lo que se refiere al tamaño de la muestra.

Tabla 1. Grupos y números de pacientes

Grupo	Pacientes
Cáncer renal	129
Enfermedad renal	217
Grupo de control	422
Total	768

3.2. Creación del modelo neuronal

Para crear el modelo neuronal lo recomendable es comenzar con una red de una sola capa y probar con diferente número de neuronas en esta capa oculta, si se logra obtener un modelo capaz de generalizar se debe optar por aumentar el número de capas ocultas. Utilizando una red de una capa oculta con diez neuronas se logró obtener un modelo que diagnostica de manera aceptable. En la figura 3 se muestra el diagrama de la red creada.

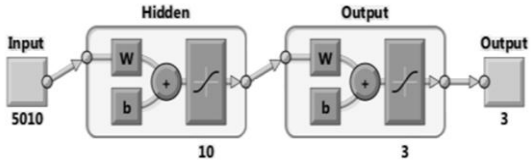

Figura 3. Red back-propagation creada

Una vez que la red ha sido creada se prosigue con el entrenamiento y validación del modelo.

3.3. Entrenar y validar el modelo

Una vez que se ha inicializado la red de manera aleatoria (aunque los pesos y desplazamientos se pueden fijar), debemos de separar nuestros datos de entrada en datos de entrenamiento, datos de validación y datos de prueba como se muestran en la Tabla 2.

Tabla 2. Base de reglas original

Datos	Muestras	Porcentaje (%)
Entrenamiento	614	80%
Validación	77	10%
Prueba	77	10%
Total	768	100%

Con la base de datos generada se entrenó una red back-propagation con una capa oculta de 10 neuronas y 3 en la capa de salida y se utilizó la función *trainscg* (Scaled Conjugate Gradient) debido a que utiliza una menor cantidad de memoria que la función *trainlm* (Levenberg-Marquardt). Para evitar el sobre- entrenamiento se utilizó el método early-stopping, en la figura 3 se muestra el mejor resultado del entrenamiento el cual se obtuvo en 107 iteraciones o épocas en un tiempo de 26 segundos.

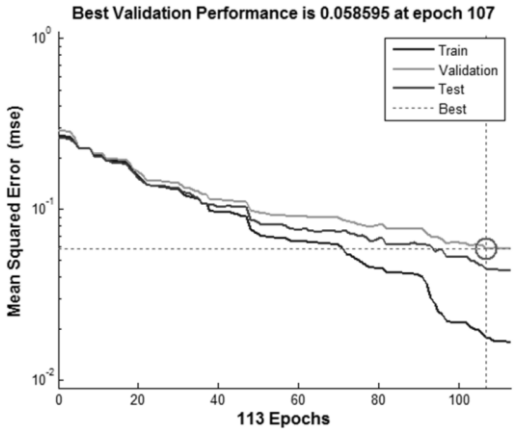

Figura 4. Grafica de rendimiento

4. RESULTADOS

En la figura 5 se muestra la matriz de confusión de prueba de la red. Una matriz de confusión es una herramienta de visualización que se emplea en aprendizaje supervisado. Cada columna de la matriz representa el número de predicciones de cada clase, mientras que cada fila representa a las instancias en la clase real.

Los resultados obtenidos de la red de 10 neuronas en la capa oculta a partir de su matriz de confusión muestra que del primer grupo (*cáncer renal*) de un total de 11 pacientes en uno de ellos se equivocó, para el segundo grupo (*enfermedad*

renal) no tuvo errores en los 14 pacientes y en el último grupo (*grupo de control*) hubo 2 errores de un total de 52 pacientes.

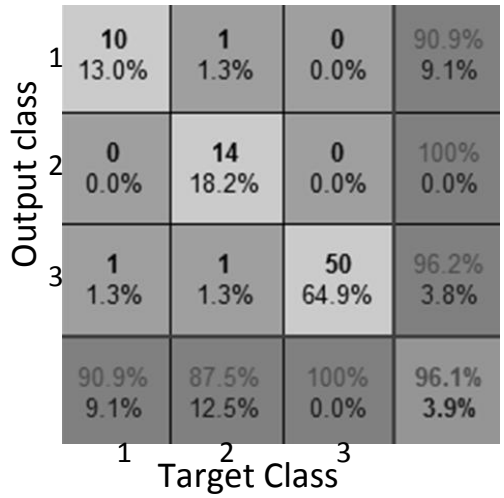

Figura 5. Matriz de confusión de prueba de la red

El modelo neuronal clasificó los 3 grupos (cáncer, enfermedad de riñón, grupo de control) acertadamente al 96.1% y tuvo un error del 3.9%. En la tabla 3 se muestra el diagnóstico del modelo neuronal.

Tabla 3. Diagnóstico del modelo neuronal

Grupo	Pacientes	Diagnostico	Error
1	11	13%	1.3%
2	14	18.2%	0%
3	52	64.9%	2.6%
Total	77	96.1%	3.9%

Como se puede apreciar en los resultados el modelo desarrollado es un buen clasificador en el diagnóstico de cáncer de riñón. Para validar lo anterior se crearon varios modelos para comparar los resultados, modelos con menos de 10 neuronas en la capa oculta y el otro modelo con más de 10 neuronas en su capa oculta.

Se obtuvo un promedio del diagnóstico de los modelos, el primero modelo es el promedio de los resultados de las redes con 10 neuronas en su capa oculta, el segundo con menos de 10 neuronas y el tercero indica los resultados de las redes con más de 10 neuronas en su capa oculta. Se puede ver que los mejores resultados se obtienen con una red que tiene 10 neuronas en su capa oculta como se muestra en la Tabla 4.

Tabla 4. Promedio de diagnóstico de los diferentes modelos neuronales

Modelo	Pacientes	Diagnostico	Error
1	77	94.1%	5.9%
2	77	89.8%	10.2%
3	77	92.9%	7.1%

5. CONCLUSIONES

Después de analizar los datos obtenidos de la red se puede concluir que:

Es posible desarrollar un modelo de clasificación para el diagnóstico de cáncer en riñón en base a una red neuronal multicapa con entrenamiento *backpropagation* como los resultados lo muestran.

La red diseñada puede utilizarse para predecir las tres clases. El error que presenta es menor al 5%, lo cual es aceptable en términos médicos.

De acuerdo a los resultados obtenidos y la alta resolución de los equipos utilizados en el análisis se abre la posibilidad de detectar el cáncer de riñón en etapas tempranas.

6. BIBLIOGRAFÍA

[1] G. Anderson, ***Cancer: Essential things to do***, USAF Cambridge Res. Lab., Cambridge, MA Rep. ARCRL-66-234 (II), 2009, vol. 2.

[2] A. Blum, ***Neural Networks in C++***, New York, NY: John Wiley & Sons, Inx, 1992, pp. 100-145.

[3] J. Freeman, ***Redes Neuronales. Algoritmos, aplicaciones y técnicas de Programación***, Reading, MA: Addison-Wesley Iberoamericana, S.A., 1993, pp. 33-105.

[4] E.Magrab, et al, ***An Engineer´s Guide to Matlab***, Upper Saddle River, NJ: Prentice-Hall, Inc., 2000, pp.87-95.

[5] H.Demuth, ***Matlab User´s Guide. For Microsoft Windows***. Natick, MA: The Math Works Inc., 2000. pp.23-183.

[6] E. R., Schwartz, J. H., & Jessel Kandel, *T. M*. ***Principles of neural science***, 4th ed. New York: McGraw-Hill, 2000.

[7] G. W. Juette and L. E. Zeffanella, ***Radio noise currents in short sections on bundle***

conductors (Presented Conference Paper style), presented at the IEEE Summer power Meeting, Dallas, TX, June 22–27, 1990, Paper 90 SM 690-0 PWRS.

[8] Van Der Zwaag, B. J., Slump, C. H., Spaaneburg, L. Process ***Identification Through Modular Neural Networks And Rule Extraction***. Ed. World Scientific. 2002. ISBN 981-238-066-3.

[9] Sharkey, A. J. ***Improving the expert networks for pattern recognition***. Ed. Springer, 2006. ISBN 978-3-540-38625-4.

[10] M. Hudson, M. T. Hagan, H. Demuth, ***Matlab Neural Network User´s Guide.*** Natick, MA: The Math Works Inc., 2012. pp.18-85