ANEXO

ELABORACIÓN DE UN MODELO NEURONAL ARTIFICIAL PARA LA ESTIMACIÓN DE LA DEMANDA BIOQUÍMICA DE OXÍGENO EN AGUAS MARINAS

RIAA
Revista de Investigación Agraria y Ambiental

ANEXO

El Perceptrón Multicapa.

Un *Perceptrón Multicapa* (MLP por sus siglas en inglés), es una de las redes más usadas en la estimación de variables fisicoquímicas de calidad del agua (Khalil et al., 2011; Araghinejad, 2014; Raheli et al., 2017). Al igual que otros tipos de red, un MLP se encuentra conformado por un conjunto de estructuras denominadas *neuronas* (Figura 1).

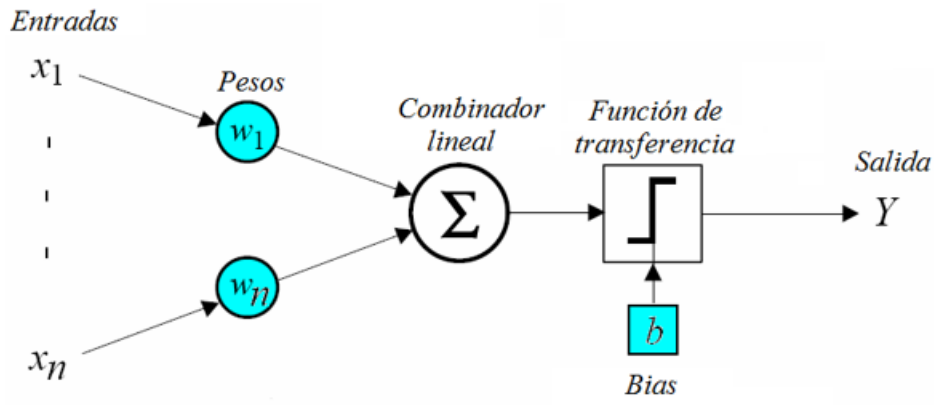

Figura 1. Representación conceptual de una neurona artificial. **Fuente:** Meza (2019)

Estas neuronas, generalmente se encuentran distribuidas en 3 niveles de procesamiento de información denominados *capas* (Figura 2): la *capa de entrada* (constituida por las variables predictoras); la *capa de salida* (donde se haya la variable a predecir); y una o más *capas ocultas* (Singh et al., 2009). Las neuronas presentes en estas últimas poseen una función matemática denominada *función de transferencia,* por medio de la cual el MLP normaliza los datos de las variables de entrada. Usualmente, en un perceptrón multicapa las funciones utilizadas son de tipo sigmoideo (Palani et al., 2008):

$$f(x) = \frac{1}{1+e^{-ax}} \qquad \text{(Ecuación 1)}$$

Donde, ***f(x)*** corresponde a la *función logística sigmoidea*, mientras que ***x*** representa la suma ponderada de los valores de entrada a la neurona oculta. "***a***" es un coeficiente que modifica la inclinación de la función de transferencia (Bai et al., 2009).

El proceso de adaptación de estas funciones, sobre el conjunto de datos de entrada y salida del modelo, es conocido como *entrenamiento* (Araghinejad, 2014). Debido a que cada neurona oculta del MLP se encuentra ligada a una función de transferencia, el entrenamiento de esta red generalmente se lleva a cabo a través de una modificación (por tanteo, o ensayo y error) en el número de sus neuronas y capas ocultas. Así, un modelo neuronal de DBO se encontrará entrenado cuando alcance una cantidad óptima de neuronas ocultas (funciones de transferencia), que represente la distribución de observaciones de las variables fisicoquímicas de entrada (pH, oxígeno disuelto, etc.) y salida (DBO).

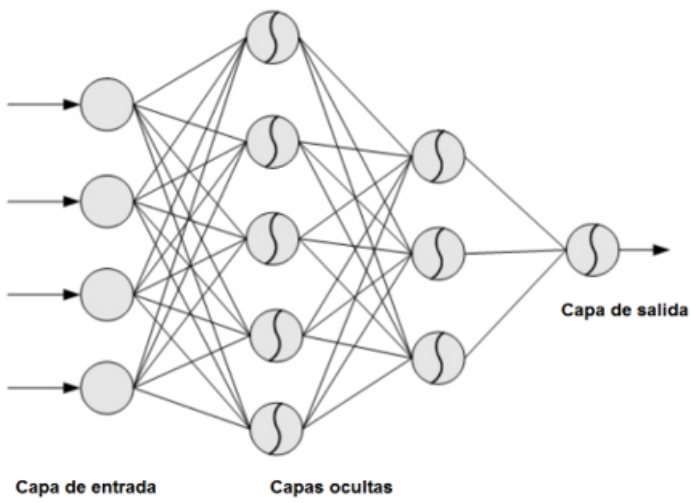

Figura 2. Estructura de un Perceptrón Multicapa. **Fuente**: Hung et al. (2009)

En un MLP el proceso de entrenamiento comúnmente se lleva a cabo a través del *algoritmo de retropropagación del error* (West & Dellana, 2011). Éste, como indica su nombre, se basa en la propagación de las entradas de la red hasta la capa de salida y en la distribución del error estimado hacia las neuronas ocultas (hacia atrás), modificando así los pesos de las conexiones (Araghinejad, 2014). El anterior procedimiento se hace iterativamente para todos los patrones hasta que la red converja a un valor de error definido (Araghinejad, 2014). En términos generales el algoritmo de retropropagación puede describirse en los siguientes pasos:

ANEXO

ELABORACIÓN DE UN MODELO NEURONAL ARTIFICIAL PARA LA ESTIMACIÓN DE LA DEMANDA BIOQUÍMICA DE OXÍGENO EN AGUAS MARINAS

- ***Inicializar los pesos con valores aleatorios:*** Estos pesos son números reales usados para la ponderación de los datos de entrada.
- ***Seleccionar un patrón de entrada:*** Este es el primer vector de la matriz de datos de entrada. En esta investigación correspondería a la secuencia [pH OD Sal SST Lat Long DBO]$_1$.
- ***Propagar el patrón hacia adelante:*** Esta etapa consiste en calcular la suma ponderada de las entradas, y utilizar el resultado obtenido en la correspondiente función de transferencia.
- ***Calcular el error en la capa de salida:*** Corresponde al cálculo de la diferencia entre la DBO estimada y la DBO observada.
- ***Propagar el error hacia las neuronas ocultas:*** Aquí el error es distribuido de manera equitativa en los pesos iniciales.
- ***Actualizar los pesos:*** Por medio de la suma de los errores a los pesos iniciales.
- ***Seleccionar un nuevo patrón de entrada y repetir los pasos anteriores hasta alcanzar el error deseado:*** El nuevo patrón es el vector con la segunda posición dentro del conjunto de datos ([pH OD Sal SST Lat Long DBO]$_2$). El entrenamiento se detiene cuando se alcanza un valor establecido de la función error, o un número definido de épocas.

El anterior algoritmo es conocido también como *algoritmo de descenso del gradiente* (DG). Para una descripción más detallada de éste puede verse a Isasi-Viñuela & Galván-León (2004), Barthakur et al. (2012), Araghinejad (2014), y a Beale et al. (2015). Matemáticamente, el algoritmo de DG se encuentra descrito por un procedimiento denominado la *regla delta generalizada*. Esta metodología es una extensión de la regla de aprendizaje usada por el perceptrón simple y es el fundamento del proceso de operación de las redes multicapa (Isasi-Viñuela & Galván-León, 2004; Zhiqiang et al., 2007). En dicha regla, la actualización de los pesos en las conexiones de las neuronas viene dada por las siguientes expresiones:

$$w^{nuevo} = w^{viejo} + \alpha\left(-\frac{\delta e}{\delta w}\right) \qquad \text{(Ecuación 2)}$$

$$e = \frac{1}{2}\sum(s-y)^2 \qquad \text{(Ecuación 3)}$$

Donde, w^{nuevo} y w^{viejo} son los pesos actuales, y anteriores a la modificación, respectivamente. "α" es la tasa de aprendizaje, una constante que toma valores entre cero y uno, y determina la velocidad de convergencia del algoritmo. El término e representa la *función error* dada por la salida observada s, y la salida estimada con la red y. La relación $\delta e/\delta w$ corresponde al gradiente o derivada parcial del error con respecto al peso (Araghinejad, 2014; Isasi-Viñuela & Galván-León, 2004).

La expresión $\Sigma(s-y)^2$ corresponde a la *suma de los errores cuadráticos*, o *SSE* (ya mencionada en la sección de *Materiales y Métodos*). En lugar de ésta pueden usarse otras expresiones como el *MSE* (*error cuadrático medio*), o el *RMSE* (*raíz del error cuadrático medio*) (Hamed et al., 2004). Estos criterios tienen como objetivo acotar o limitar el proceso de entrenamiento de la red, y no son los indicadores estadísticos usados para evaluar el desempeño de los modelos. En la evaluación de los resultados del modelo generalmente se usa el *coeficiente de correlación* o el *coeficiente de determinación*. También pueden usarse otras pruebas como el MSE, RMSE, o el mismo SSE (Šiljić et al., 2016; Raheli et al., 2017).

El algoritmo de retropropagación es un método ampliamente usado en el entrenamiento de redes multicapa, no obstante, diversas modificaciones han sido desarrolladas con el fin de incrementar su velocidad de operación (Patan, 2008). De acuerdo con Prusty et al. (2015) existen seis categorías en las que pueden clasificarse estas modificaciones del algoritmo. A continuación, se hace una breve descripción de uno de los enfoques más usados por los investigadores: el algoritmo de *Levenberg-Marquardt* (Raheli et al., 2017).

Algoritmo de Levenberg-Marquardt

El método de Newton es una técnica que utiliza una matriz denominada *Hessiana* para la optimización de la red. Las entradas de esta matriz son las derivadas de segundo orden de la función de error con respecto a los pesos (Dongardive & Abraham, 2016):

$$H = \begin{bmatrix} \frac{\partial^2 e}{\partial w_1^2} & \frac{\partial^2 e}{\partial w_1 \partial w_2} & \cdots & \frac{\partial^2 e}{\partial w_1 \partial w_n} \\ \frac{\partial^2 e}{\partial w_2 \partial w_1} & \frac{\partial^2 e}{\partial w_2^2} & \cdots & \frac{\partial^2 e}{\partial w_2 \partial w_n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial^2 e}{\partial w_n \partial w_1} & \frac{\partial^2 e}{\partial w_n \partial w_2} & \cdots & \frac{\partial^2 e}{\partial w_n^2} \end{bmatrix}$$

Los métodos *Cuasi-Newton* usan una aproximación de tipo iterativo para la matriz Hessiana o su inversa, en lugar de una evaluación de las derivadas (Du & Swamy, 2014). De manera similar a éstos, el algoritmo de *Levenberg-Marquardt* se encuentra diseñado para realizar un aprendizaje a través de las derivadas de segundo orden sin la necesidad de estimar la matriz Hessiana (Zayani et al., 2008; Payal et al., 2015). Si se considera que la función error (el término e mencionado anteriormente), es algún tipo de suma cuadrática (SSE, MSE, o RMSE, por ejemplo), entonces la matriz Hessiana puede ser aproximada como:

$$H = J^T J \qquad \text{(Ecuación 4)}$$

Mientras que el gradiente (g) puede estimarse a través de la expresión:

$$g = J^T e \qquad \text{(Ecuación 5)}$$

Donde J es la *matriz Jacobiana* que contiene las derivadas de primer orden de los errores de la red con respecto a los pesos. El término J^T representa la matriz traspuesta de J. "e" es el vector con los errores de la red. Debido a que la matriz Jacobiana es computacionalmente más fácil de estimar que la matriz Hessiana, el algoritmo de Levenberg-Marquardt realiza el ajuste de los pesos de manera más rápida que otros métodos (Zayani et al., 2008). Así, la regla de actualización de los pesos bajo este enfoque se describe como:

$$w^{nuevo} = w^{viejo} + [J^T J + \mu I]^{-1} J^T e \qquad \text{(Ecuación 6)}$$

El parámetro μ es un escalar que controla el comportamiento del algoritmo. Si μ es igual a cero, el algoritmo sigue el método de Newton, usando la aproximación de la matriz Hessiana. Si μ es muy alto, el algoritmo de Levenberg-Marquardt se asemeja a la regla delta generalizada con una tasa de aprendizaje baja. El término I representa la matriz identidad (Zayani et al., 2008; Peteiro & Guijarro, 2013).

Como se describió en la sección de Materiales y Métodos, el entrenamiento de la red utilizada en este estudio es llevado a cabo sólo con el número de capas y neuronas ocultas, el número de épocas, el indicador de desempeño, el método de segmentación del conjunto de datos, y un algoritmo para optimizar el desempeño del MLP. Éste último algoritmo, realiza su proceso de operación de manera automática, ya que el software *Matlab* lo incluye como una de sus herramientas. Así, en el caso del algoritmo de Levenberg-Marquardt, no es necesario realizar los cálculos matriciales anteriormente expuestos. En el código, sólo se escribe la instrucción *"trainlm"* (*"entrenamiento con Levenberg-Marquardt"*), y éste queda ligado a la estructura del MLP.

Una vez entrenada la red, ésta permite calcular los valores de DBO para cualquier vector de entrada ([pH OD Sal SST Lat Long]), que no haya sido incluido en el proceso de entrenamiento.

Para ver los códigos de entrenamiento de la red, véase a Meza (2019).

ANEXO

ELABORACIÓN DE UN MODELO NEURONAL ARTIFICIAL PARA LA ESTIMACIÓN DE LA DEMANDA BIOQUÍMICA DE OXÍGENO EN AGUAS MARINAS

LITERATURA CITADA

Araghinejad, S. (2014). Artificial Neural Networks. In Data-Driven Modeling: Using MATLAB® in Water Resources and Environmental Engineering. Water Science and Technology Library 67. https://doi.org/10.1007/978-94-007-7506-0

Bai, Y., Zhang, H., & Hao, Y. (2009). The performance of the backpropagation algorithm with varying slope of the activation function. Chaos, Solitons and Fractals, 40(1), 69-77. https://doi.org/10.1016/j.chaos.2007.07.033

Barthakur, M., Thakuria, T., & Sarma, K. K. (2012). Artificial Neural Network (ANN) Based Object Recognition Using Multiple Feature Sets. Soft Computing Techniques in Vision Science, 127-135. https://doi.org/10.1007/978-3-642-25507-6_11

Beale, M. H., Hagan, M. T., & Demuth, H. B. (2015). Neural Network Toolbox User's Guide. MathWorks, (June), 2–3.

Dongardive, J., & Abraham, S. (2016). Reaching optimized parameter set: protein secondary structure prediction using neural network. Neural Computing and Applications, 28, 1947-1974. https://doi.org/10.1007/s00521-015-2150-2

Du, K.-L., & Swamy, M. N. S. (2014). Chapter 5 Multilayer Perceptrons: Other Learning Techniques. In Neural Networks and Statistical Learning. https://doi.org/10.1007/978-1-4471-5571-3_5

Hamed, M. M., Khalafallah, M. G., & Hassanien, E. A. (2004). Prediction of wastewater treatment plant performance using artificial neural networks. Environmental Modelling & Software, 19(10), 919-928. https://doi.org/10.1016/j.envsoft.2003.10.005

Hung, N. Q., Babel, M. S., Weesakul, S., & Tripathi, N. K. (2009). An artificial neural network model for rainfall forecasting in Bangkok, Thailand. Hydrology and Earth System Sciences, 1413-1425. https://doi.org/10.5194/hess-13-1413-2009

Isasi-Viñuela, P., & Galván-León, I. M. (2004). Redes de Neuronas Artificiales. Un Enfoque Práctico. Madrid: PEARSON EDUCACIÓN, S.A. PEARSON-Prentice Hall.

Khalil, B., Ouarda, T. B. M. J., & St-Hilaire, A. (2011). Estimation of water quality characteristics at ungauged sites using artificial neural networks and canonical correlation analysis. Journal of Hydrology, 405(3-4), 277-287. https://doi.org/10.1016/j.jhydrol.2011.05.024

Meza, R. (2019). Elaboración y uso de un modelo neuronal para la estimación de la DBO5. Caso de estudio: Costa Caribe del Departamento de Bolívar (Colombia). Universidad Nacional de Colombia.

Palani, S., Liong, S. Y., & Tkalich, P. (2008). An ANN application for water quality forecasting. Marine Pollution Bulletin, 56(9), 1586-1597. https://doi.org/10.1016/j.marpolbul.2008.05.021

Patan, K. (2008). Artificial Neural Networks for the Modelling and Fault Diagnosis of Technical Processes. Lecture Notes in Control and Information Sciences.

Payal, A., Rai, C. S., & Reddy, B. V. R. (2015). Analysis of Some Feedforward Artificial Neural Network Training Algorithms for Developing Localization Framework in Wireless Sensor Networks. Wireless Personal Communications, 82(4), 2519-2536. https://doi.org/10.1007/s11277-015-2362-x

Peteiro-Barral, D., & Guijarro-Berdiñas, B. (2013). A study on the scalability of artificial neural networks training algorithms using multiple-criteria decision-making methods. In ICAISC 2013 (pp. 162-173). Berlin Heidelberg: Springer-Verlag. https://doi.org/10.1007/978-3-642-38658-9_15

Prusty, M. R., Chakraborty, J., Jayanthi, T., & Velusamy, K. (2015). Performance Comparison of Supervised Machine Learning Algorithms for Multiclass Transient Classification in a Nuclear Power Plant. Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 8947, 111-122. https://doi.org/10.1007/978-3-319-20294-5_10

Raheli, B., Aalami, M. T., El-Shafie, A., Ghorbani, M. A., & Deo, R. C. (2017). Uncertainty assessment of the multilayer perceptron (MLP) neural network model with implementation of the novel hybrid MLP-FFA method for prediction of biochemical oxygen demand and dissolved oxygen: a case study of Langat River. Environmental Earth Sciences, 76. https://doi.org/10.1007/s12665-017-6842-z

Šiljić, A., Antanasijević, D., Perić-Grujić, A., Ristić, M., & Pocajt, V. (2016). Artificial neural network modelling of biological oxygen demand in rivers at the national level with input selection based on Monte Carlo simulations. Environmental Science and Pollution Research, 23(4), 3978-3979. https://doi.org/10.1007/s11356-015-5978-1

Singh, K. P., Basant, A., Malik, A., & Jain, G. (2009). Artificial neural network modeling of the river water quality-A case study. Ecological Modelling, 220, 888-895. https://doi.org/10.1016/j.ecolmodel.2009.01.004

West, D., & Dellana, S. (2011). An empirical analysis of neural network memory structures for basin water quality forecasting. International Journal of Forecasting, 27(3), 777-803. https://doi.org/10.1016/j.ijforecast.2010.09.003

Zayani, R., Bouallegue, R., & Roviras, D. (2008). Adaptive predistortions based on neural networks associated with Levenberg-Marquardt algorithm for satellite down links. Eurasip Journal on Wireless Communications and Networking. https://doi.org/10.1155/2008/132729

Zhiqiang, Z., Zheng, T., Guofeng, T., Vairappan, C., XuGang, W., & RunQun, X. (2007). An Improved Algorithm for Eleman Neural Network by Adding a Modified Error Function. In ISNN 2007 (pp. 465–473). Berlin Heidelberg: Springer-Verlag