ARNN: Un paquete para la predicción de series de tiempo usando redes neuronales autorregresivas

ARNN: A packages for time series forecasting using autoregressive neural networks

Juan D. Velásquez. Ph.D., Cristian Zambrano. Est. & Laura Vélez. Est.
Facultad de Minas, Universidad Nacional de Colombia
jdvelasq@unal.edu.co; cozambra@unal.edu.co; lauravelez1@gmail.com

Recibido para revisión 01 de abril de 2011, aceptado 28 de junio de 2011, versión final 07 de julio de 2011

***Resumen*— En este artículo, se describe un paquete para el pronóstico de series de tiempo usando redes neuronales autoregresivas (y perceptrones multicapa imponiendo algunas restricciones al modelo) con función adaptativa de activación. El uso de los paquetes y algunas funcionalidades son ilustrados para una serie de tiempo no lineal**

***Palabras Clave*— Redes neuronales artificiales, lenguaje R, predicción.**

***Abstract*— In this article, we describe a package for nonlinear time series forecasting using autoregressive neural networks (and multilayer perceptrons by imposing some restrictions to the model) with adaptive activation function; The use of the package and some functionality are illustrated for one nonlinear time series.**

***Keywords*— Artificial neural networks, R language, prediction.**

I. INTRODUCCION

Los perceptrones multicapa (MLP, por su sigla en inglés) parecen ser la arquitectura de redes neuronales artificiales más utilizada para la predicción de series de tiempo no lineales [1]. Entre sus muchas aplicaciones se encuentran la predicción de precios de la electricidad [2] y la demanda de energía eléctrica [3].

Las redes neuronales autorregresivas (ARNN, por su sigla en inglés) se obtienen al considerar la fusión de un modelo lineal autorregresivo [4] con un MLP. Su desarrollo conceptual inicial está basado en el desarrollo de un contraste estadístico de no linealidad en el que se comparan los dos modelos anteriores [5] [6] [7]. No obstante, la ARNN es una importante alternativa al uso de los MLP en la predicción de series de tiempo debido a la incorporación de la componente lineal autorregresiva.

Una de las principales herramientas para realizar el análisis y la predicción de series de tiempo es el lenguaje de programación R [8], ya que posee un amplio repertorio de funciones para ese fin; una recopilación de las principales funciones implementadas y paquetes disponibles es presentada [9]. Adicionalmente, el lenguaje provee mecanismos para crear paquetes instalables en su propio entorno de programación; durante la creación de cada paquete, el código fuente es analizado con el fin de verificar unas condiciones mínimas de calidad, que incluyen, entre otras, que cada función implementada esté adecuada documentada en el sistema de ayuda. Aunque esta herramienta computacional está diseñada fundamentalmente para el cómputo estadístico, resulta muy apropiada para la programación de algoritmos de inteligencia computacional [10] tales como las redes neuronales artificiales.

Ya se han desarrollado varios esfuerzos encaminados a la implementación de modelos genericos de redes neuronales en el lenguaje R:

- En la versión 7.3-1 del paquete 'nnet' [11], la función del mismo nombre permite crear modelos no lineales de regresión y clasificación usando redes de propagación hacia adelante con una sola capa oculta, entre las que se incluyen los MLP y redes similares a las ARNN; la optimización es realizada usando el algoritmo BFGS del paquete 'optim'.
- El paquete AMORE [13] permite la creación de modelos de regresión basados en modelos tipo MLP; la mayor novedad de este paquete es la implementación del algoritmo TAO para la estimación de los parámetros de la red neuronal [13].
- Los modelos de redes de propagación hacia adelante con restricciones monótonas para la solución de problema de regresión [14] son implementados en el paquete 'monmlp' [15].
- El paquete 'neuralnet' permite la estimación de modelos MLP con el algoritmo RPROP [16].
- El paquete 'RSNNS' permite el uso de los modelos de redes neuronales artificiales implementados en el 'Stuttgart Neural Network Simulator' desde el entorno R [17].

Revista Avances en Sistemas e Informática, Vol.8 No.2, julio de 2011 - Medellín. ISSN 1657-7663

- La función NNET del paquete 'tsDyn' [18] permite el uso de redes neuronales tipo MLP para la predicción de series de tiempo. La optimización es realizada usando el algoritmo BFGS implementado en la función 'optim'. En este mismo paquete se implementan otras metodologías, tales como los modelos STAR.

En este artículo se discuten los algoritmos implementados y la funcionalidad del paquete ARNN escrito en el Lenguaje R para el cómputo estadístico. El resto de este artículo está organizado como sigue: la descripción del modelo matemático es presentada en la Sección II; la funcionalidad del paquete es presentada en la Sección III; finalmente, las principales conclusiones son presentadas en la Sección IV.

II. LA RED NEURONAL AUTORREGRESIVA

En esta sección se describe la arquitectura de la red neuronal. En un modelo ARNN, la variable dependiente y_t es obtenida como una función no lineal de sus P valores pasados y_{t-p}, para $p = 1, \ldots, P$:

$$y_t^* = \eta + \sum_{p=1}^{P} \varphi_p y_{t-p} + \sum_{h=1}^{H} \beta_h G(\omega_h + \sum_{p=1}^{P} \alpha_{p,h} y_{t-p}) \quad (1)$$

donde *G()* es la función sigmoidea adaptativa [19] definida como:

$$G(u) = \left[\frac{1}{1 + \exp(-u)} \right]^M \quad (2)$$

La arquitectura del modelo es presentada en la Figura 1. Los parámetros del modelo: $\eta, \varphi_p, \beta_h, \omega_h, \alpha_{p,h}$ y M para $i = 1, \ldots, P$ y $h = 1, \ldots, H$ son estimados minimizando el error de regularización:

$$\lambda E_* \quad (3)$$

En la ecuación (3), λ es un parámetro externo definido por el usuario; e_t son los errores entre el pronóstico y_t^* y el valor deseado y_t. T es la longitud de la serie de tiempo y_t. La última cantidad es función, únicamente, de los parámetros del modelo:

$$E_* = |\eta| + \sum_{h=1}^{H} (|\beta_h| + |\omega_h|) + \sum_{p=1}^{P} |\varphi| + \sum_{p=1}^{P} \sum_{h=1}^{H} |\alpha_{p,h}| \quad (4)$$

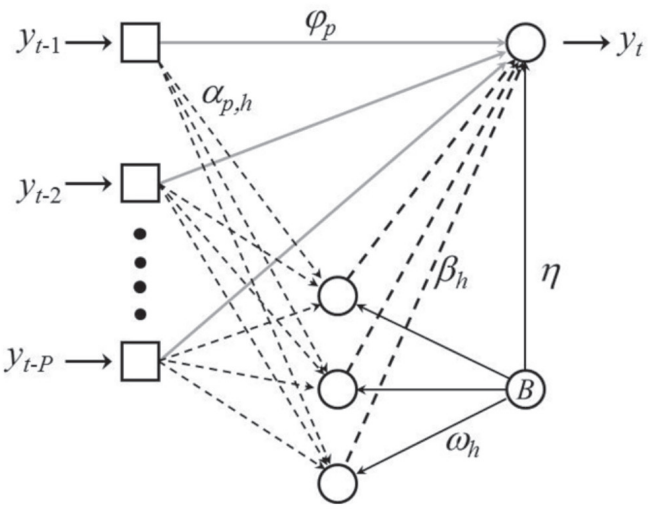

Figura 1. Arquitectura de una red neuronal autorregresiva.

El modelo escrito en la Ecuación (1) se reduce a un perceptrón multicapa imponiendo la restricción: $\varphi_1 = \varphi_2 = \cdots = \varphi_P$. Igualmente, dicha red neuronal se reduce a un modelo autorregresivo imponiendo que $H = 0$.

III. EL PAQUETE 'ARNN'

El paquete 'arnn' implementa la red neuronal artificial descrita en la sección anterior. Los aportes alcanzados con el desarrollo de esta herramienta son los siguientes:

- Es una herramienta de usuario final, por lo que el usuario no debe preocuparse por los elementos internos de la implementación.
- Las funciones implementadas están debidamente documentadas en el sistema de ayuda nativo del entorno. De esta forma, el paquete se integra naturalmente dentro de entorno.
- Su diseño y arquitectura computacional está basado profundamente en el paquete 'forecast' de Hyndman y Khandakar [20], por lo que sólo fue necesario implementar las funciones propias del modelo ARNN. El paquete 'forecast' tiene implementaciones de otros modelos de predicción ampliamente utilizados, por lo que resulta extremadamente fácil contrastar los resultados con los de otros modelos alternativos.
- Finalmente, no hay implementaciones de modelos ARNN con funciones adaptativas de activación.

El paquete consta de dos funciones principales: 'arnn' para crear y estimar el modelo y 'forecast' para pronosticar varios periodos adelante.

Para ejemplificar el uso de la implementación realizada se usará la serie del número de usuarios autenticados en un servidor de internet cada minuto durante cien minutos. Esta

serie de tiempo se encuentra disponible en el entorno R bajo la variable 'WWWusage'.

En este caso particular se desean usar las primeras 80 observaciones para la estimación del modelo y las 20 restantes para evaluar su capacidad de predicción. Una red neuronal que use los primeros cuatro rezagos de la serie como entrada y dos neuronas en su capa oculta puede ser creada mediante una simple llamada a la función 'arnn'; el código es presentado a continuación:

```
> ### separa la muestra de estimación
> x <- ts(WWWusage[1:80], s = 1, f = 1)
> ### crea el modelo
> fit <- arnn(x=x, lags=1:4, H=2)
> fit

Method: arnn

Call:
 arnn(x=x, lags=1:4, H = 2)

Parameters:
            M     Wio[1]     Wio[2]     Wio[3]
      3.97025    2.05758   -1.66205    0.89724
       Wio[4]   Wih[1,1]   Wih[2,1]   Wih[3,1]
     -0.32474   -0.91562   -0.91428   -0.90899
     Wih[4,1]   Wih[1,2]   Wih[2,2]   Wih[3,2]
     -0.90471   -0.90472   -0.90193   -0.89894
     Wih[4,2]     Wbh[1]     Wbh[2]     Who[1]
     -0.89309   -0.00732   -0.00677  -29.53962
       Who[2]        Wbo
     -29.4484    4.20615

Sigma^2 estimated as:  8.91769597191016
LogL:  -182.432928498831
Information Criteria:
      AK      AKc       HQ       SC
400.8659 413.7715 417.1801 441.8458
```

En este caso, M es el exponente de la función sigmoidea adaptativa (ecuación (2)); Wih equivale a $\alpha_{p,h}$, Wbh equivale a ω_h, Who es equivalente a β_h, Wio equivale a φ_p y Wbo es η.

Adicionalmente, el sistema calcula la varianza (Sigma^2) y el logaritmo de la función de verosimilitud de los errores. Finamente, se imprimen los valores de los criterios de información de Akaike, Hannan-Quinn y Schwartz.

La estimación numérica de los parámetros es realizada usando el algoritmo BFGS implementado en la función 'optim' del paquete 'stats', cuya ejecución es controlada con el argumento 'optim.control' de la función 'arnn'. La implementación realizada permite que el algoritmo sea reiniciado varias veces de forma automática, de tal manera que 'arnn' devuelve el mejor modelo encontrado.

La implementación del paquete 'arnn' basada en el paquete 'fo El paquete también permite calcular el pronóstico varios periodos adelante. Para modelos no lineales, este tipo de pronóstico debe realizarse mediante simulación numérica ya que no existen expresiones exactas analíticas. Este proceso es descrito a continuación.recast' permite que se usen muchas de las funciones de este último. Por ejemplo, los estadísticos del error de ajuste pueden calcularse directamente usando la función 'accuracy' del paquete 'forecast':

```
> accuracy(fit)
              ME            RMSE             MAE
-3.149810e-08  2.966540e+00  2.303534e+00
             MPE            MAPE            MASE
-7.280844e-02  1.930014e+00  5.651527e-01
```

El pronóstico un periodo adelante puede calcularse para toda la serie con el fin de evaluar el ajuste del modelo en la región de predicción extrapolativa por fuera de la muestra de calibración de los parámetros. De esta forma, el siguiente código permite realizar esta tarea sin recalcular los parámetros:

```
fit1 = arnn(x=WWWusage, model = fit)
> accuracy(fitted(fit1)[76:96], WWWusage[81:100])
      ME     RMSE      MAE      MPE     MAPE
7.529124 9.895523 8.236415 4.458222 4.776821
```

Igualmente, graficar el valor real de la serie y el pronóstico un periodo adelante es directo; por ejemplo, el gráfico presentado en la Figura 2 se obtuvo con los siguientes comandos:

```
> plot(WWWusage,lwd=2)
> lines(fitted(fit1),col='red',lwd=2)
> grid()
```

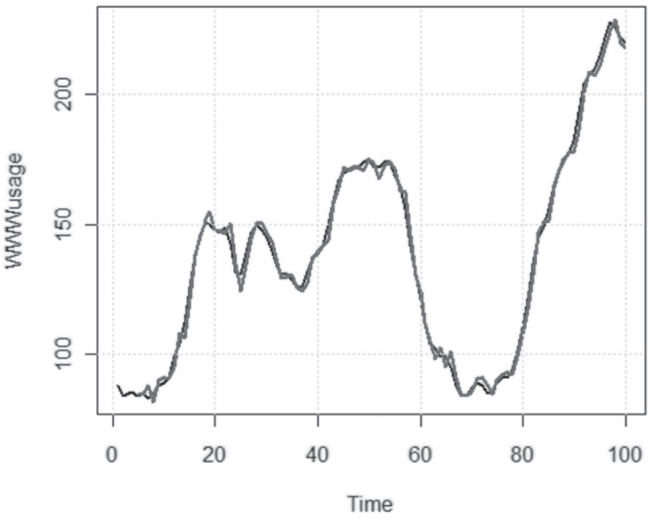

Figura 2. Pronóstico un paso adelante usando el modelo ARNN.

El pronóstico para el instante $T+1$ puede realizare directamente usando la ecuación (2) ya que todas las entradas a la red neuronal son conocidas. Para el instante $T+2$ surge el problema al tener en cuenta que para calcular el pronóstico del

periodo actual ($T+2$), una de las entradas de la red neuronal es el valor del periodo anterior ($T+1$) que es desconocido (no ha ocurrido) y es representado por el pronóstico para $T+1$ que es una variable aleatoria.

Una metodología aceptada es generar series simuladas a partir del último valor conocido y calcular el pronóstico y sus intervalos de confianza a partir de ellas. La serie simulada se genera de la siguiente forma:

- Para el periodo $T+1$, se calcula el pronóstico como el valor calculado con la ecuación (1) más un error aleatorio; se obtiene un posible valor para el periodo $T+1$.
- El valor para el periodo $T+2$, se calcula de igual manera que para el periodo $T+1$, esto es aplicando la ecuación (1) y sumando un error aleatorio. Aquí se asume que el valor simulado para el periodo anterior fue el real.
- Se continúa hasta cubrir todo el horizonte de pronóstico.

El proceso anterior se repite para un número grande de series, de tal forma, que para cada periodo del horizonte de predicción se tiene una muestra numérica de posibles valores que es igual a la cantidad de series simuladas. A partir de cada muestra de cada periodo se obtiene el valor esperado del pronóstico y sus intervalos de confianza. Este algoritmo es discutido en [21].

El algoritmo anterior está implementado en la función 'forecast' del paquete 'arnn'. Para su uso, se requiere que un modelo ya haya sido estimado. Por ejemplo, los siguientes comandos permiten obtener el pronóstico 20 periodos adelante a partir de 1000 series simuladas así como los intervalos de confianza del 90%:

```
> forecast(fit, h=20, level=90, fan=FALSE,
+ bootstrap=FALSE, npaths=1000)
    Point Forecast     Lo 90    Hi 90
 81      114.98590  98.89054 127.0504
 82      118.42760  69.85927 128.8134
 83       92.71186  61.16071 129.9980
 84      176.62253 147.49404 192.7575
 85      141.15179 121.55409 153.9245
 86      126.99396 121.00549 131.8614
 87      150.67106 116.28974 166.3873
 88      178.28462 135.11785 190.1690
 89      160.97707 118.77872 175.6652
 90      149.41385 109.59008 167.5691
 91      167.90675 136.00044 193.6261
 92      179.02418 119.71898 211.2436
 93      169.73791 127.69303 186.9587
 94      134.31042 126.69449 179.5668
 95      125.24069 114.87057 131.6202
 96      147.62555 117.38553 167.3945
 97      129.62512 124.08876 138.5866
 98      126.55297  97.46695 163.7914
 99      157.15015 128.51600 172.9813
100      134.03885 103.41638 145.5448
```

Igualmente, es posible graficar directamente el pronóstico con (ver Figura 3):

```
> plot(forecast(fit, h=20, level=90, fan=FALSE,
+ bootstrap=FALSE, npaths=1000))
```

Tal como ya se indicó, un perceptrón multicapa puede obtenerse directamente a partir de un ARNN. En este caso, esto puede realizarse con haciendo que el parámetro 'isMLP' de la función 'arnn' sea igual a TRUE. Por ejemplo:

```
> fit <- arnn(x=x, lags=1:4, H=2, isMLP=TRUE)
```

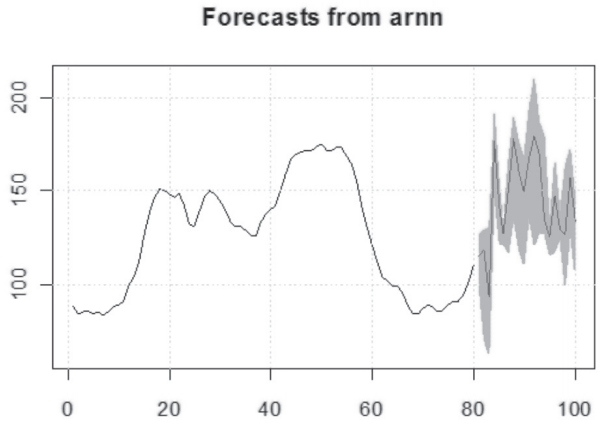

Figura 3. Pronóstico varios periodos adelante obtenidos usando la función forecast del paquete arnn.

IV. CONCLUSIONES

En este artículo se describió una implementación para estimar redes neuronales autorregresivas y perceptrones multicapa en el lenguaje R en el contexto de las series de tiempo no lineales. En este artículo sólo se han descrito las principales funciones y más información puede ser obtenida del sistema de ayudas del paquete implementado.

REFERENCIAS

[1] Zhang, G., B. Patuwo and M. HU (1998), "Forecasting with artificial neural networks: the state of the art", International Journal of Forecasting 14: 35–62.

[2] Velásquez, J.D.; Dyner, I. y Souza, R.C. (2008), "Modelado del precio spot de la electricidad en Brasil usando una red neuronal autorregresiva", Ingeniare 16 (3): 394-403.

[3] Velásquez, J.D; Franco, C.J. y García, H.A. "Un modelo no lineal para la predicción de la demanda mensual de electricidad en Colombia", Estudios Gerenciales, 25 (112): 37-54.

[4] Box, G. E. P.; Jenkins, G. M. (1970). "Time Series Analysis: Forecasting and Control". San Francisco: Holden–Day Inc.

[5] H. White. (1989), "An additional hidden unit test for neglected nonlinearity in multilayer feedforward networks". Proceedings of the International Joint Conference on Neural Networks, 2: 451-455. IEEE Press. Washington DC., New York.

[6] Lee, T.H.; White, H.: Granger, C.W.J.. (1993), "Testing for neglected nonlinearity in time series models". Journal of

Econometrics. Vol. 56, pp. 269-290.
[7] Teräsvirta, T.; Lin, C.F. and Granger, C.W.J.. (1993) "Power of the neural network linearity test". Journal of Time Series Analysis, 14: 209-220.
[8] Ihaka, R.; Gentleman, R. (1996), "R: A language for data analysis and graphics". Journal of Computational and Graphical Statistics 5: 299–314.
[9] Velásquez, J.D.; Olaya, Y. y Franco, C.J. (2011), "Análisis y predicción de series de tiempo en mercados de energía usando el lenguaje R". DYNA, 78 (165): 287-296.
[10] Velásquez, J.D.; Montoya, O.; Castaño, N. (2010), "Es el proyecto R para la computación estadística apropiado para la inteligencia artificial". Ingeniería y Competitividad, 12(2): 81-94.
[11] Ripley, B. "Package 'nnet': Feed-forward Neural Networks and Multinomial Log-Linear Models", Disponible en http://cran.r-project.org/web/packages/nnet/ *(última consulta en 06/07/2011)*
[12] Castejón, M.; Ordieres, J. ; González, A.; Pernía, A.V.; Martínez, F.J.; Alba, F. (2010), "Package AMORE: A MORE flexible neural networks package". Disponible en http://cran.r-project.org/web/packages/amore/ *(última consulta en 06/07/2011)*
[13] Pernía Espinoza, A.V., Ordieres Meré, J.B., Martínez de Pisón, F.J., González Marcos. A. (2005), "TAO-robust backpropagation learning algorithm". Neural Networks. 18 (2): 191–204
[14] Zhang, H.; Zhang, Z. (1999), "Feedforward networks with monotone constraints". In: International Joint Conference on Neural Networks, 3: 1820-1823.
[15] Cannon, A.J. (2010), "Package 'monmlp': Monotone multi-layer perceptron neural network". Disponible en http://cran.r-project.org/web/packages/monmlp/ *(última consulta en 06/07/2011)*
[16] Fritsch, S.; Guenther, F. (2010), "neuralnet: Training of neural networks". Disponible en http://cran.r-project.org/web/packages/neuralnet/ *(última consulta en 06/07/2011)*
[17] Bergmeir, C. (2011), "RSNNS: Neural Networks in R using the Stuttgart Neural Network Simulator". Disponible en http://cran.r-project.org/web/packages/rsnns/ *(última consulta en 06/07/2011)*.
[18 Di Narzo, F.A.; Aznarte, J.L.; Stigler, M. (2011), 'tsDyn: Nonlinear time series models with regime switching'. Disponible en http://cran.r-project.org/web/packages/tsDyn/ *(última consulta en 06/07/2011)*
[19] Chandra, P.; Singh, Y. (2004), "An activation function adapting training algorithm for sigmoidal feedforward networks". Neurocomputing, 61: 429-437.
[20] Hyndman, R. J.; Khandakar, Y. (2008), "Automatic time series forecasting: The forecast Package for R". Journal of Statistical Software 27 (3):1-24.
[21] Velásquez, J.D.; Dyner, I.; Souza, R.C. (2006), "Tendencias en la predicción y estimación de los intervalos de confianza usando modelos de redes neuronales aplicados a series temporales". DYNA 73 (149): 141-147.