Modelado del músculo del brazo con redes neuronales para estimar el momento en un sistema exoesquelético

Modeling of the Arm Muscle with Neural Networks to Estimate The Moment in an Exoskeletic System

Juan Pablo González Tamayo
Ingeniero electrónico, Magíster en Automatización Industrial Universidad Nacional de Colombia. Docente auxiliar Universidad Tecnológica de Pereira.
jpgonzalez@utp.edu.co

Jason Edwin Molina Vargas
Ingeniero eléctrico, Universidad Tecnológica de Pereira. Docente auxiliar Universidad Tecnológica de Pereira.
jason@ohm.utp.edu.co

Álvaro Ángel Orozco Gutiérrez
Ingeniero eléctrico, Magíster en Ingeniería Eléctrica Universidad Tecnológica de Pereira. Profesor titular Universidad Tecnológica de Pereira.
aaog@utp.edu.co

Clasificación del artículo: investigación

Fecha de recepción: 3 de septiembre de 2007

Fechadeaceptación:9defebrerode2008

Palabras clave: exosqueleto, brazo, modelo del músculo, redes neuronales.

Key words: exoskeleton, arm, muscle model, neural networks.

Resumen

En este artículo se presenta un sistema exoesquelético en el brazo humano, que permite una interacción natural entre el operador y la máquina a través de señales electromiográficas, señales cinéticas y cinemáticas. Por medio de estas señales biológicas y mecánicas es posible estimar el momento realizado por el operador sobre una carga específica, para obtener la referencia deseada en el actuador del exoesqueleto.

Se modela el músculo del brazo a través del entrenamiento de redes neuronales, variando los parámetros del algoritmo, el número de capas ocultas y el número de neuronas. Se realiza el entrenamiento con siete variables de entrada y uno de salida, que corresponden a señales preprocesadas y posteriormente se realiza la validación del modelo con los mejores resultados.

Abstract

In this paper is showed an exoskeletic system in the human arm, that allow a natural interaction between the operator and the machine through the electromyographic signals, kinematic and kinetic signals. Through the mechanical and biological signals is possible to estimate the moment done by the operator over the specific load, to obtain the desirable reference in the exoskeleton's actuator. Arm muscle is modeled, through the neural networks training, changing the algorithm parameters, the number of occult layers and the number of neurons. Network was training with seven input neurons and one output neurons, which correspond to preprocessed signals, and finally it was taken with the best results.

* * *

1. Introducción

La integración de humanos y máquinas robóticas en una misma unidad ofrece muchas oportunidades para crear una nueva generación de tecnología de asistencia, que puede ser usada en la biomédica, la industria y en aplicaciones aeroespaciales.

Un dispositivo con las características anteriores es el exoesqueleto, que es un mecanismo o estructura externa que tiene uniones que corresponden a las del cuerpo humano. Éste es usado por el humano y el contacto físico que existe entre el operador y el exoesqueleto permite una transferencia directa de la potencia mecánica y las señales de información.

El exoesqueleto del brazo es un robot manipulador, atado en paralelo al brazo humano, con uniones y partes que corresponden a las del hombro y del codo. El torque generado por un actuador localizado en la coyuntura del codo es usado para amplificar el momento producido por los músculos del codo, de tal manera que la mayor parte del torque sea producido por el actuador y, en consecuencia, una fracción de la carga será soportada por el humano.

Muchos dispositivos de este tipo han sido diseñados e implementados sólo con señales de fuerza como en [1 y 2], pero pocos de estos han hecho uso de las bioseñales para comandar el dispositivo; se menciona esta forma de comando en [3], además de un conjunto de señales realimentadas para generar una señal de comando para el actuador del exoesqueleto.

Desde este punto de vista, la interfaz humano máquina (la unión del sistema donde el humano comparte información con la máquina) es obtenida por parte de la coyuntura neuromuscular. Las bioseñales producidas durante las contracciones del músculo del codo son utilizadas para predecir los momentos inminentes del músculo en la coyuntura del codo. Una de las ventajas importantes de establecer una interfaz en el nivel fisiológico del músculo es la capacidad de estimar las fuerzas que serán generadas por los músculos antes de que el fenómeno de la contracción se desarrolle completamente.

Durante el corto retardo de tiempo entre la aparición de las señales bioeléctricas en los músculos del codo y el movimiento previsto del brazo, el sistema de control del exoesqueleto adquiere las bioseñales y las señales de realimentación de fuerza, las procesa, y alimenta una señal de comando apropiada para el actuador de la coyuntura del codo, de tal forma que esté a tiempo para agregar su esfuerzo de torsión al momento muscular del operador para mover la carga externa.

El sistema requiere procesadores de bioseñales confiables [4], que define el módulo del sistema responsable de estimar los momentos que se aplicarán a la coyuntura, basado en las señales bioeléctricas y en la cinemática de la unión; para su posterior alimentación para el actuador. Se han utilizado dos tipos de sistemas para estimar el momento a partir de las bioseñales, como son: el modelo Hill, que evalúa expresiones analíticas que relacionan las variables bioeléctricas con las mecánicas, y modelos basados en redes neuronales, que buscan mediante entrenamiento, aprender el comportamiento de la relación entre estas variables.

En este trabajo se presenta una alternativa basada en redes neuronales para generar el modelo del músculo del brazo, dado que es una herramienta práctica para el modelado de diversos sistemas no lineales. Con estos resultados y con la señal obtenida de la persona, se consigue la señal de comando para el sistema de control.

Finalmente se obtienen los resultados del entrenamiento, realizando variaciones en los parámetros del algoritmo y en la estructura general de la red neuronal, para posteriormente efectuar la validación del modelo con datos diferentes a los del entrenamiento.

2. Realización del modelo del músculo

Para incluir al humano dentro del sistema de control para el exoesqueleto, debe ser incluido el modelo de la parte que desarrolla la acción del movimiento, el cual es llamado mioprocesador [5]. En este caso se realiza un modelo de los músculos que tienen mayor influencia en el movimiento tipo flexión y extensión, los cuales son: el bíceps y el tríceps [6].

Una clasificación de los modelos del músculo, basada en el nivel de la estructura por tratar, fue propuesta por Zahalak [7], según el cual los modelos del músculo pueden ser clasificados como modelos microscópicos, modelos de fibras o modelos macroscópicos. No siempre son diferentes los aspectos que caracterizan los tres grupos, de hecho, algunos de los modelos llevan características de más de un grupo. El modelo del músculo que se considera en este artículo pertenece a la categoría de los modelos macroscópicos del músculo. Este grupo puede ser subdividido como:

- Los modelos viscoso elásticos;
- Los modelos Hill, que son variaciones del modelo anterior, propuestos por A.V. Hill;
- Modelos que tratan los músculos como cajas negras, el contenido de las cuales es determinado por procedimientos de identificación formal de parámetros.

El modelo usado consiste en una red neuronal, la cual se enmarca en la categoría de los modelos macroscópicos, que determinan un modelo por ajuste de parámetros en una caja negra. Los modelos Hill son modelos semi-analíticos y son estudiados en [6, 8], en los que se puede ver que presentan resultados menos óptimos, comparados con los modelos basados en redes neuronales.

Las variables que definen el modelo del músculo para este caso son siete variables de entrada y una variable de salida. Cuatro de las variables de entrada corresponden a señales electromiográficas, dos del bíceps y dos del tríceps; ambos grupos son de signo contrario para el movimiento. Además se le aplica un preproceso a cada una de estas señales, que consiste en un filtro pasa alto, rectificación de onda completa y un filtrado pasa bajo, luego se realiza una normalización para que las señales queden todas dentro del rango [0 , 1].

El preproceso de las señales electromiográficas se resume en la figura 1.

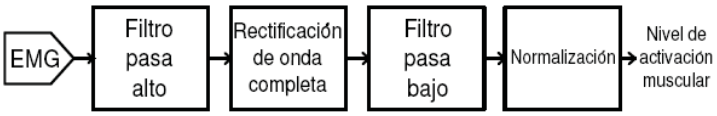


Figura 1. Proceso previo de las señales electromiográficas.

La señal resultante de este procedimiento se encuentra acotada en el intervalo [0, 1] y se denomina el nivel de activación muscular. Esta variable se tiene muestreada, para cada señal electromiográfica con 182 muestras cada una.

El modelo debe responder con una salida cinética, dadas las siete entradas. Dicha señal cinética define la única salida del modelo, la cual posteriormente será el comando de referencia del sistema de control. Para realizar el entrenamiento de la red neuronal se tomaron los datos de entrenamiento de [6], los cuales consisten en muestras recogidas experimentalmente mediante sesiones de movimiento de una persona usando el exoesqueleto con distintas cargas. Dichas señales se muestran en la figura 2.

Teniendo los datos anteriores se procede a realizar un entrenamiento tipo supervisado para la red neuronal, programada en Matlab 7 ®. La formulación que hace posible la programación del entrenamiento se da a continuación.

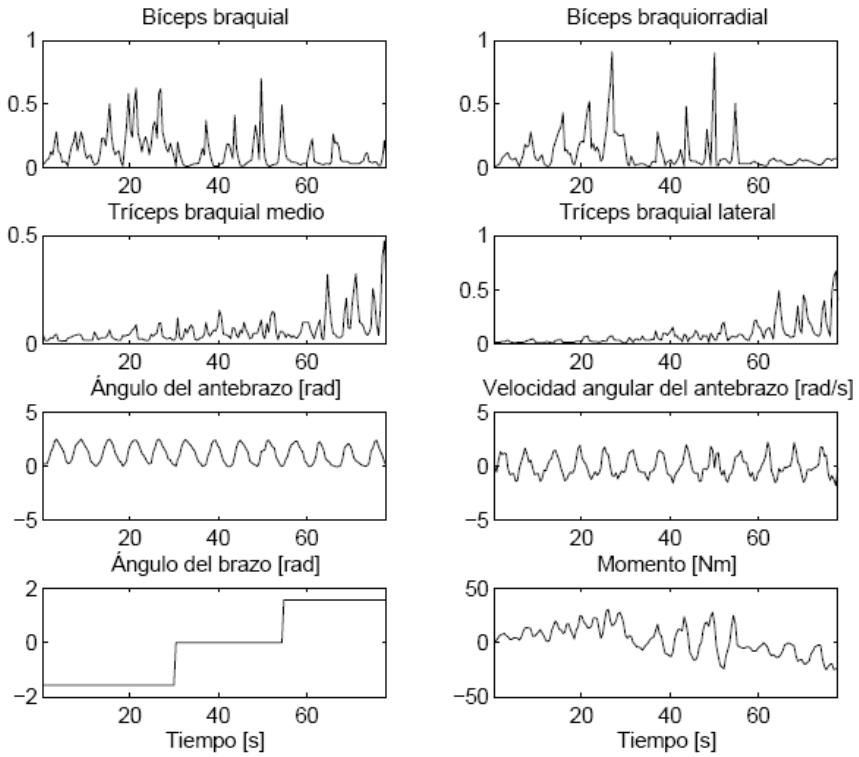


Figura 2. Datos de entrenamiento para el modelo del músculo.

3. Entrenamiento de la Red Neuronal

Si se parte del conjunto de datos se pueden obtener dos subconjuntos de datos, uno para entrenar y otro para validar. El objetivo de este procedimiento es que se pueda obtener un criterio de generalización para el modelo, en el caso en que la entrada no coincida con las entradas del conjunto de entrenamiento. Se dan entonces en este orden, los conjuntos:

$$\{x(l),\ l \in \Omega_1\} \ y \ \{x(l),\ d(l),\ l \in \Omega_2\} \quad (1)$$

en los que Ω_1 y Ω_2 son disjuntos y definen los conjuntos de entrenamiento y de validación respectivamente.

Para ilustrar el comportamiento del flujo de datos por el paradigma, se muestra la estructura de la red neuronal en la figura 3.

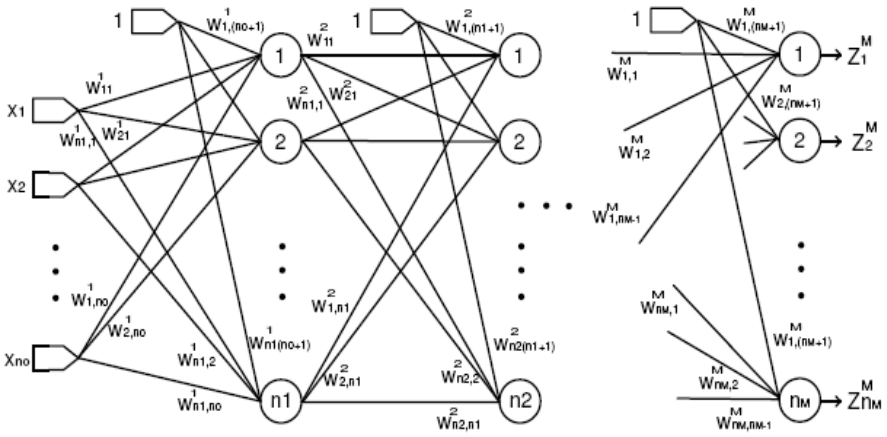


Figura 3. Estructura de la red neuronal.

En la figura 3 se distingue una capa de entrada, *L-1* capas ocultas y una capa de salida. Las capas que no son de entrada ni de salida se denominan ocultas. La capa de entrada no tiene neuronas sino que sólo distribuye los datos del vector de entrada a la primera capa oculta. Cada neurona recibe cierto número de entradas y las procesa para dar paso a una salida.

En cada neurona, la salida está dada en forma recursiva, por [9]:

$$z_q^L = f\left(w_{q,n_{L-1}}^L + \sum_{p=1}^{n_{L-1}} w_{qp}^L x_p\right) = f\left(w_q^L \begin{bmatrix} z^{L-1} \\ 1 \end{bmatrix}\right), \text{ con } x_p = z_p^{L-1} \quad (2)$$

En donde f es una función no lineal llamada activación.

Para realizar el entrenamiento de la red, se ajusta la colección de matrices W^L o pesos para producir el resultado deseado [10]. Esto se realiza minimizando la función definida como el promedio de los errores medios cuadráticos de las salidas a lo largo de todo el conjunto de entrenamiento:

$$\{x(k),\ d(k);\ 1 \le k \le K\}$$

$$E = \frac{1}{K}\sum_{k=1}^{K}\frac{1}{2}\sum_{q=1}^{n_M}\left[d_q(k) - z_q^M(k)\right]^2 \quad (3)$$

El mínimo valor que toma E es cero, cuando las salidas del conjunto de entrenamiento son iguales a las salidas de la red para todo $1 \le k \le K$. No obstante, esto no es siempre deseable dado que se puede tener el problema de sobreentrenamiento, que consiste en que el error es muy pequeño para los datos de entrenamiento, pero no para los de validación, por esto se controla el ajuste con el tamaño de la red para lograr una adecuada generalización.

El paradigma usado para ajustar las matrices de pesos es basado en el método *error Backpropagation*, que es uno de los más usados porque sus cálculos son relativamente simples. Este método consiste en hacer que el paso de entrenamiento sea un pequeño incremento de las matrices de pesos en la dirección del gradiente negativo del error respecto a W, que es la dirección del máximo descenso del error en el punto actual [11].

La dificultad de este proceso está en que todas las matrices de pesos están acopladas mediante operaciones matemáticas no lineales, y el cálculo del gradiente puede ser costoso. Por esto, en este trabajo se reducen las operaciones mediante un proceso de operaciones con matrices [12].

Con lo anterior, es realizado un algoritmo en Matlab 7 que calcula en cada iteración el gradiente del error y actualiza las matrices de pesos con varios factores

de entrenamiento y además se aplica en algunos intentos de entrenamiento el método del momento, para varios valores de la constante de momento. El algoritmo con momento es una modificación del algoritmo básico en el que se incluye la variación de la matriz de pesos además del gradiente [13, 9]:

La función activación *f(u)* puede tomar varias formas. Entre éstas están las formas diferenciables que son necesarias para evaluar cualquier algoritmo que utilice gradiente, y se listan a continuación las más comunes [9]:

Nombre	Fórmula	Derivada
Logística o Sigmoid	$f(x) = 1/(1+e^{-x})$	$f'(x) = f(x)[1-f(x)]$
Tangente hiperbólica o sigmoid tangencial		
Lineal	$\nabla \bullet \vec{B} = 0$	

Tabla 1. Funciones de activación diferenciables.

4. Resultados

Si se calcula el error de entrenamiento como el promedio de los errores sobre el número de muestras que son 182, el error promedio y el porcentaje de error con respecto al torque máximo, se obtienen los resultados mostrados para el entrenamiento del modelo del músculo. En las redes neuronales utilizadas se realiza una arquitectura que consta de una capa oculta con diferente número de neuronas. Todas las neuronas llevan la función de activación tangente hiperbólica, dado que, de las funciones de activación derivables mostradas en la tabla 1, la tangente hiperbólica es la única que varía en [-1, 1], lo que da un rango de variación simétrico para las salidas. En este problema la variación de la salida es igual en los valores negativos que en los valores positivos, por lo tanto, la función de activación escogida es apropiada.

Como la salida de momento varía en valores positivos y negativos, con valores superiores a uno, se utiliza en todos los casos, la función de activación lineal para la capa de salida, porque ésta permite cualquier rango de varación de la salida.

Para el modelo entrenado con 25 neuronas en una capa oculta:

Parámetros	Error de entrenamiento	Error porcentual	Error promedio	Épocas	Tiempo de entrenamiento
$\eta = 0{,}1$, $\mu = 0$	91,7732 $(Nm)^2$	0,17%	0,0526 *Nm*	150000	24 min
$\eta = 0{,}05$, $\mu = 0$	80,7407 $(Nm)^2$	0,16%	0,0494 *Nm*	150000	25 min
$\eta = 0{,}007$, $\mu = 0{,}7$	57,0083 $(Nm)^2$	0,14%	0,0415 *Nm*	150000	25 min

Tabla 2. Resultados del entrenamiento del modelo del músculo, con 25 neuronas en la capa oculta.

Para el modelo entrenado con 50 neuronas en una capa oculta:

Parámetros	Error de entrenamiento	Error porcentual	Error promedio	Épocas	Tiempo de entrenamiento
η= 0,006, μ = 0,6	23,9910 $(Nm)^2$	0,09%	0,0269 *Nm*	150000	41 min
η = 0,01, μ = 0	49,8104 $(Nm)^2$	0,13%	0,0388 *Nm*	100000	8 min
η = 0,05, μ = 0	18,6003 $(Nm)^2$	0,08%	0,0237 *Nm*	150000	25 min
η= 0,07, μ = 0	22,6619 $(Nm)^2$	0,09%	0,0262 *Nm*	150000	33 min

Tabla 3. Resultados del entrenamiento del modelo del músculo, con 50 neuronas en la capa oculta.

También se entrena un modelo con 100 neuronas en la capa oculta para observar si el error disminuye. Se tienen parámetros de entrenamiento m= 0,01, μ = 0,2; el entrenamiento tarda 1 hora 2 minutos, en 150000 épocas, obteniendo un error de entrenamiento 26,3518 $(Nm)^2$ correspondiente al 0,09% y a un promedio de 0,0282 *Nm*.

Ahora se entrena el modelo con 2 capas ocultas para algunos valores de los parámetros de entrenamiento, con una razón de aprendizaje de 0,01, un parámetro de momento de 0,2 y distintos valores de neuronas en las capas ocultas, se utiliza para ambas capas ocultas la función de activación tangente hiperbólica.

Neuronas, cada capa oculta	Error de entrenamiento	Error porcentual	Error promedio	Épocas	Tiempo de entrenamiento
Capa 1, 25; Capa 2, 25	5,1495 $(Nm)^2$	**0,04%**	0,0125 *Nm*	**150000**	**48 min**
Capa 1, 50; Capa 2, 50	3,9447 $(Nm)^2$	**0,04%**	0,0109 *Nm*	**150000**	**46 min**
Capa 1, 50; Capa 2, 25	2,0372 $(Nm)^2$	**0,03%**	0,0078 *Nm*	**150000**	**55 min**

Tabla 4. Resultados del entrenamiento del modelo del músculo, con 2 capas ocultas.

Viene al caso considerar la capacidad de generalización del modelo usado, dado que se puede presentar sobreentrenamiento. Para verificar la generalidad del modelo, se retiran aleatoriamente muestras de entrenamiento de la base de datos, tomando para el conjunto de validación, el 20% del total de los 182 datos, correspondiente a 36 muestras, y con el restante 80% se realiza el entrenamiento del modelo, para la arquitectura de 100 neuronas en una capa oculta. Se muestran a continuación los parámetros del entrenamiento para éste caso, y los resultados en cuanto a error de entrenamiento y validación:

Parámetro	Valor
H	0,01
M	0,2
Tiempo de entrenamiento	2 horas, 22 minutos
Épocas	300000

Tabla 5. Parámetros de la red neuronal para verificar generalidad.

Error	Total	Porcentaje	Promedio
Entrenamiento	3,5657 $(Nm)^2$	0,04%	0,0129 Nm
Validación		2,37%	0,7223 Nm

Tabla 6. Resultados del entrenamiento del modelo del músculo, para verificación de generalidad.

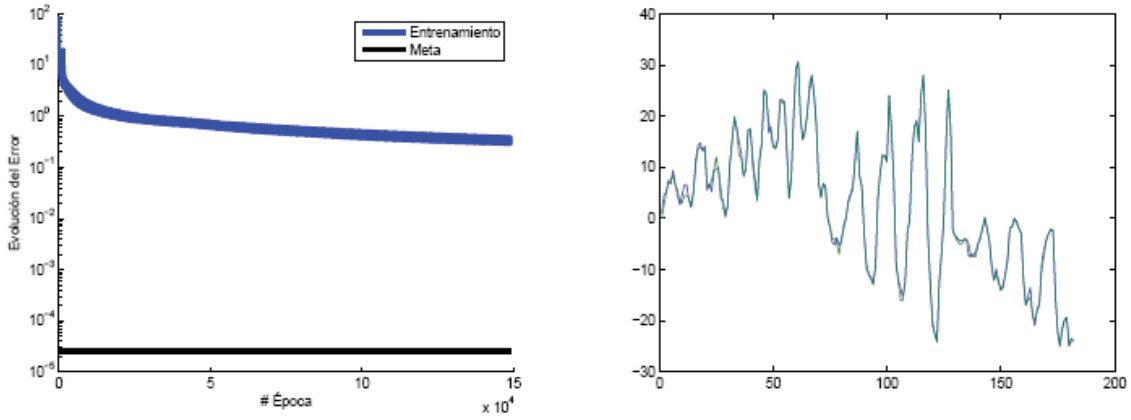
A pesar de todo, la generalidad del entrenamiento se logra verificar cuando a partir de una prueba en línea con las siete entradas se logra conseguir un comportamiento de predicción de torque aceptable. Esto se da debido a que la cantidad de datos de entrenamiento no es exagerada, considerando que se tomaron los datos para cinco valores de la carga del exoesqueleto (de 1 a 5 *kg*), y el modelo deberá servir para otros valores de carga.

En cualquier caso, la velocidad en la que el paradigma de la red neuronal evalúa la salida de momento para entrarlo a la referencia del sistema de control, es crítico pues esto limita el tiempo de muestreo del sistema en general, por esta razón se calcula el tiempo promedio que tarda el paradigma en evaluar la salida, a partir de un conjunto de entradas, para verificar la cota mínima en el tiempo de muestreo.

A continuación se muestran las figuras obtenidas del desempeño de las redes neuronales para cada arquitectura, en el mejor caso. Según la tabla 2, para veinticinco neuronas se tiene el mejor desempeño con $\eta = 0{,}007$ y $\mu = 0{,}7$, donde la evolución del error con el número de épocas es la que muestra la gráfica 1(a). Las curvas que comparan el momento real o medido y el que predice la red neuronal se muestran en la gráfica 1(b).

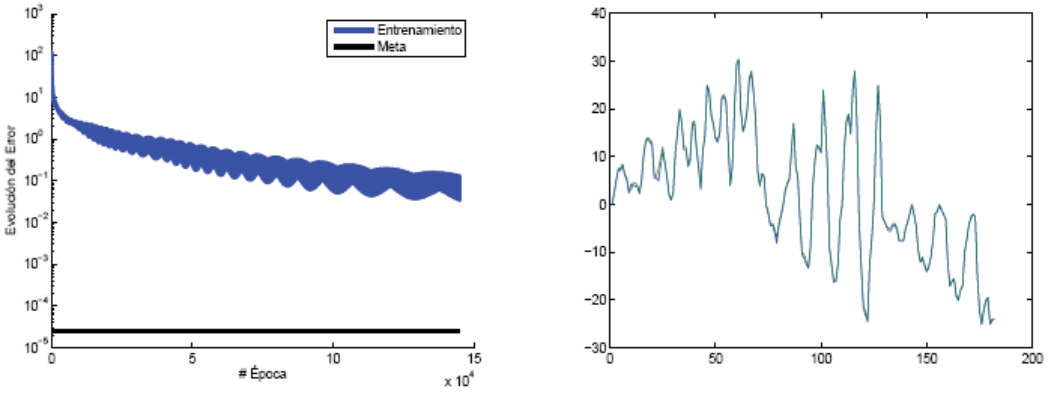
Se realiza el mismo procedimiento para la red neuronal de 50 neuronas, que según la tabla 3, el mejor desempeño se obtiene con $\eta = 0{,}05$, $\mu = 0$ y se muestra en la gráfica 2.

Para la red neuronal entrenada con 100 neuronas, se muestran las curvas de desempeño obtenidas en el entrenamiento (gráfica 3a), y posterior a él, la comparación del momento real y el obtenido por evaluación de la red neuronal (gráfica 3b).



(a) Error de entrenamiento. (b) Comparación de las curvas de torque.

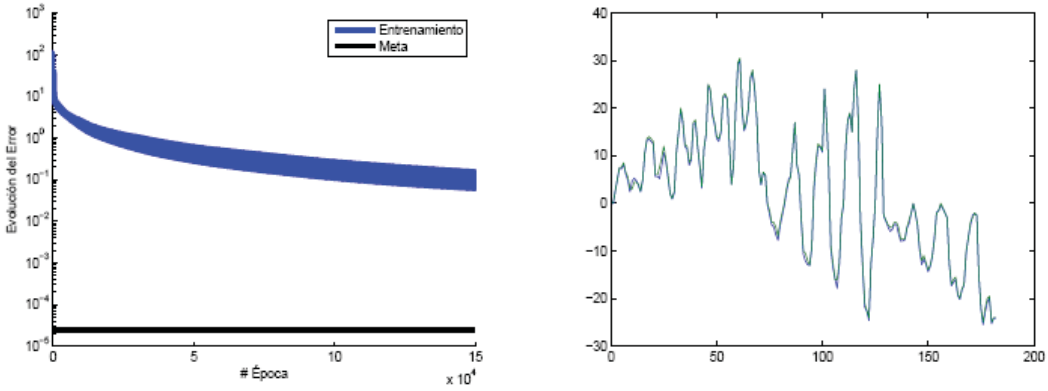
Gráfica 1. Desempeño de la red neuronal con arquitectura de 25 neuronas.



(a) Error de entrenamiento. (b) Comparación de las curvas de torque.

Gráfica 2. Desempeño de la red neuronal con arquitectura de 50 neuronas.

En el caso del entrenamiento con una arquitectura de 2 capas, se tiene el comportamiento del error y la curva de desempeño que se muestra en la gráfica 4 para la arquitectura de veinticinco neuronas en la primera capa oculta y veinticinco neuronas en la segunda capa oculta, en la gráfica 5 para la arquitectura de cincuenta neuronas en la primera capa oculta y cincuenta neuronas en la segunda capa oculta, y en la gráfica 6 para la arquitectura de cincuenta neuronas en la primera capa oculta y veinticinco neuronas en la segunda capa oculta.

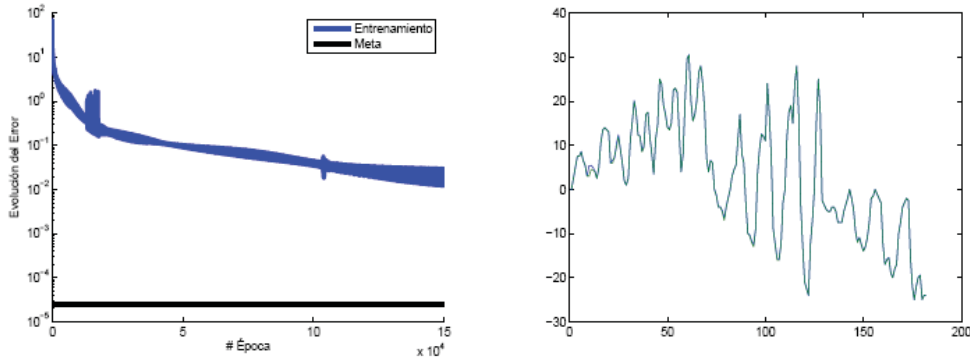


(a) Error de entrenamiento. (b) Comparación de las curvas de torque.

Gráfica 3. Desempeño de la red neuronal con arquitectura de 100 neuronas.

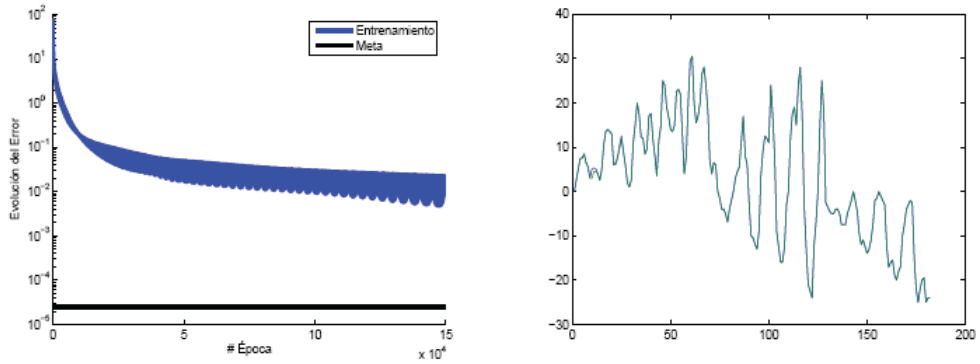
Como se puede observar en las tablas 2 y 3, el error para la arquitectura de veinticinco neuronas en una capa oculta es bastante alto, en la arquitectura de cincuenta neuronas en una capa oculta disminuye de forma notable a un promedio de 0,0237 *Nm*, para la arquitectura de 100 neuronas en una capa oculta, no disminuye. Es notable de la tabla 4, que el error disminuye significativamente, por lo cual se puede decir que la arquitectura de dos capas ocultas funciona mejor que la de una capa oculta para realizar la predicción del momento, incluso realizando igual numero de épocas y demandando tiempos de entrenamientos no muy superiores. Además se puede ver que cuando el número de neuronas en las capas ocultas disminuye en la última capa, el error disminuye de forma más estable

y con mejor aproximación al modelo, esto se puede además ver en la gráfica 3, en la cual la arquitectura es de cincuenta neuronas en la primera capa oculta y veinticinco neuronas en la segunda.



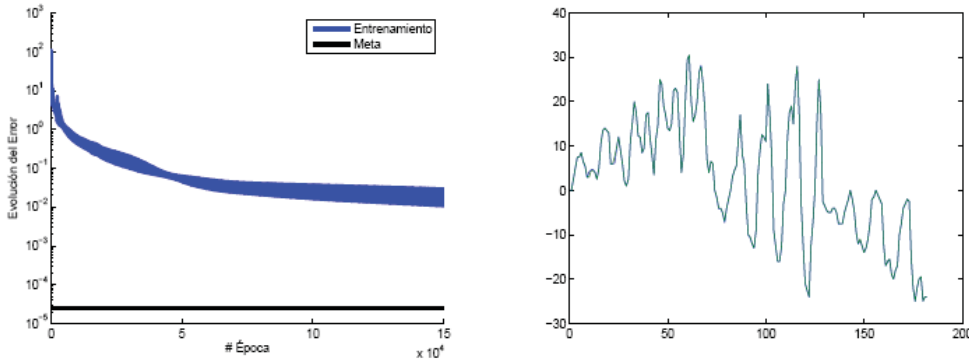
(a) Error de entrenamiento. (b) Comparación de las curvas de torque.

Gráfica 4. Desempeño de la red neuronal con 25 neuronas en las 2 capas ocultas.



(a) Error de entrenamiento. (b) Comparación de las curvas de torque.

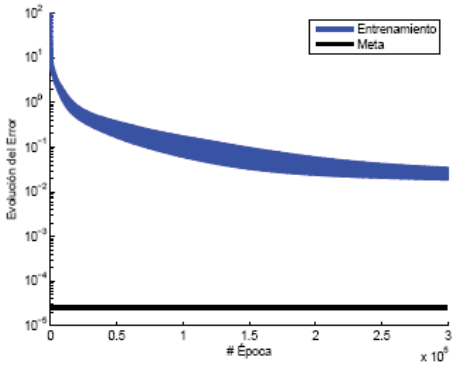
Gráfica 5. Desempeño de la red neuronal con arquitectura de 50 neuronas en las 2 capas ocultas.



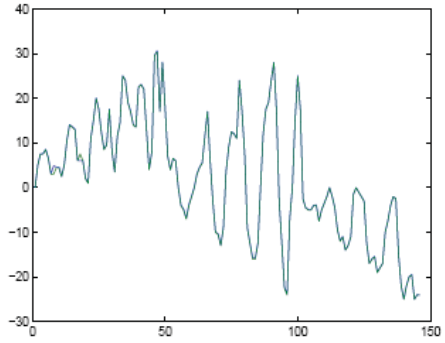
(a) Error de entrenamiento. (b) Comparación de las curvas de torque.

Gráfica 6. Desempeño de la red neuronal con arquitectura de 50 neuronas en la primera capa oculta y 25 en la segunda.

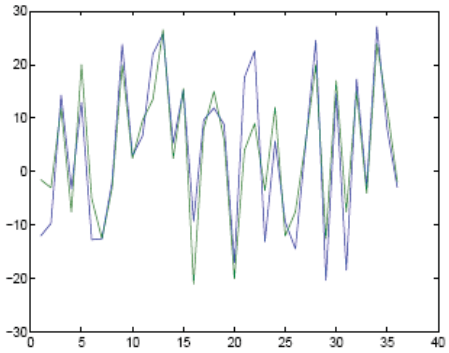
Las curvas que muestran el desempeño de la red neuronal entrenada, reduciendo los datos de entrenamiento en un 20%, son las de la gráfica 7, en la cual además se muestra la curva de validación de los datos no incluidos en el entrenamiento, cuyo error fue especificado en la tabla 6.



(a) Error de entrenamiento.



(b) Comparación de las curvas de torque, para los datos de entrenamiento.



(c) Comparación de las curvas de torque, para los datos de validación.

Gráfica 7. Desempeño de la red neuronal con arquitectura de 50 neuronas en la primera capa oculta y 25 en la segunda.

Se puede destacar que en [6] se trabajó este modelado con una arquitectura en la red neuronal de una sola capa de 25, 50 y 100 neuronas, obteniendo un error promedio mínimo de 0,015 *Nm* para la arquitectura de cincuenta neuronas, que es un poco menor al obtenido aquí, pero se obtuvo con mayor esfuerzo de entrenamiento, utilizando 200000 épocas. En el presente trabajo se entrenó el modelo utilizando 150.000 épocas, lo cual implica menor esfuerzo de entrenamiento, no obstante en el entrenamiento con arquitectura de dos capas ocultas se supera en todos los casos el resultado obtenido en [6], con menor número de épocas, obteniendo un error promedio mínimo de 0,0078 *Nm*.

5. Conclusiones

Resultan apropiadas las señales electromiográficas para determinar el comando de control, ya que éstas están correlacionadas con el torque que la persona ejerce mediante sus músculos. En el presente trabajo se ha involucrado al operador humano en

el sistema de control, usando un modelo de los músculos que intervienen en el movimiento, el cual determina el torque a partir de la cinemática del movimiento y las señales electromiográficas, representadas por el nivel de activación muscular. El torque es función no lineal de la cinemática y del nivel de activación muscular; para tal fin se ha dispuesto una red neuronal, entrenada de tal manera que determine el torque para cualquier posición del brazo en el plano perpendicular al de la persona, y en un tiempo menor a un intervalo de muestreo.

Se han entrenado en este trabajo un total de 11 redes neuronales cada una con diferente arquitectura y una para verificar generalidad. De estos modelos entrenados se destacan, aquellos que cuentan con dos capas ocultas dado que su error disminuye más rápidamente. El desempeño de la red neuronal que supera los resultados de [6], se obtiene con una arquitectura de dos capas ocultas, con un error promedio mínimo de 0,0078 *Nm* y con 100000 épocas menos en el entrenamiento.

Referencias bibliográficas

[1] Rosheim, M. E. (1994) *Robot Revolution-The Development of Antrhobotics.* New York: Wiley.

[2] Kazerooni, H. (1996) The Human Amplifier Technology at the University of California, Berkeley. *Robotics Autonomous Syst, 19*.

[3] Rosen, J. (1997) *Natural Integration of a Human Arm/Powered Exoskeleton System*. Ph.D. dissertation. Israel: Faculty of Engineering, Tel-Aviv University.

[4] Prutchi, D. (1992) *Relationships Between Mechanical and Electrical Activity of the Muscle*. Ph.D.dissertation. Israel: Tel-Aviv University.

[5] Rosen, J., Brand, M., Fuchs, M. B, and Arcan, M. (May 2001) A Myosignal-Based Powered Exoskeleton System. *IEEE Transactions on Systems, Man and Cybernetics*, *31* (3).

[6] Rosen, J., Fuchs, M. B, and Arcan, M. (September 1999) Performances of Hill-Type and Neural Network Muscle Models-Toward a Myosignal-Based Exoskeleton. *Comput. Biomed. Res*, *32* (5), 415-439.

[7] Zahalak, G. I. (1992) An Overview of Muscle Modeling. In: *Neural Prostheses (R. B. Stein, P. H. Peckham, and D. B. Popovic, Eds.).* New York: Oxford Univ. Press.

[8] *Hill-Based Model as a Myoprocessor for a Neural Controlled Powered Exoskeleton Arm -Parameters Optimization*. (Abril 2005).

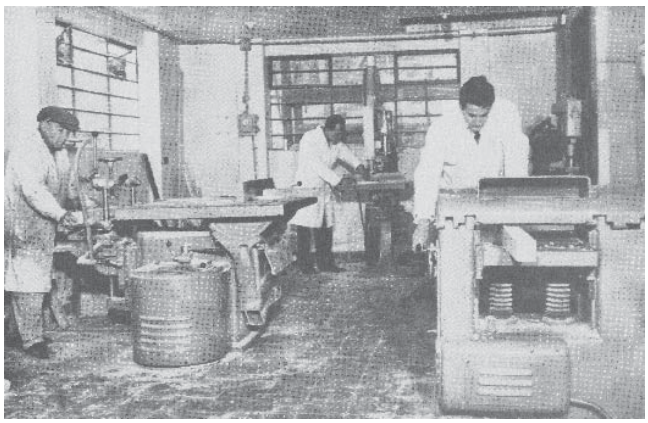
[9] Hu, Y. H, and Hwang, J.-N. (2002) *Handbook of Neural Network Signal Processing*. CRC Press LLC.

[10] Widrow, B. and M. H. Jr. (1960) Adaptive Switching Circuit. *IRE WESCON Convention Record, Pt. 4*, 96-104.

[11] Apostol, T. M. (1967) *Calculus* (2nd ed., Vol. 2).

[12] Restrepo C. y Molina, J. (Mayo de 2007) Notación matricial en el entrenamiento de redes neuronales. *Scientia et Technica, 13* (37).

[13] MathWorks. (2000) *Neural Network Toolbox: User's Guide - Version 4. For Use with MATLAB*, The MathWorks, Inc., Natick, MA.



Dpto. de Tecnología de Maderas, 1964. Fuente:
Universidad Distrital Sesenta años de memoria y vida, Bogotá. IEIE, 2008.