Implementation of Artificial Neural Network (ANN) in C# to control the locomotor movements of a robotic ant.

Article · December 2013

CITATION
1

READS
2,769

1 author:

Jorge Párraga-Álava
University of Santiago, Chile
7 PUBLICATIONS **2** CITATIONS

SEE PROFILE

Implementación en C# de una Red Neuronal Artificial en el control locomotor de una hormiga robótica

Jorge Antonio Párraga Álava[a],MarcosCristhian Anzules Reyna[a],Cristhian Eugenio Ramírez Rodríguez[a], LincolnEduardo Santander Alcívar[a].

[a] Carrera Informática, Escuela Superior Politécnica Agropecuaria de Manabí Manuel Félix López, Campus Politécnico El Limón, Calceta, Ecuador
jbelerofonte14@hotmail.com,
marcos_car_24@hotmail.com,redramirito7@hotmail.es, liedsaal92@hotmail.es

Resumen. Las redes neuronales artificiales (RNA) son un campo de la inteligencia artificial (IA) que constituyen un modelo computacional equivalente a las redes neuronales biológicas, y permiten resolver problemas de la vida real. Este artículo tiene como objetivo demostrar la habilidad de las RNA en el control de la locomoción de las hormigas, para el efecto se creó una aplicación de software que simulo el comportamiento de este insecto. En la elaboración de la misma se utilizó el modelo de desarrollo software incremental. Se analizaron las características y movimientos posibles de la hormiga robótica, para luego, a través de programación orientada a objetos en C#, crear los sensores, codificar el espacio matricial, posición y generación de feromonas. En seguida se elaboró la interfaz gráfica mediante el IDE Visual Studio 2010. Se realizaron pruebas, bajo diferentes condiciones (ubicación de hormiga, feromonas, estado de sensores, etc.). Finalmente se corroboró la simulación correcta exitosa de la aplicación.

Palabras Clave: Redes neuronales artificiales, Perceptron unicapa, Hormigas, Inteligencia Artificial.

Abstract. Artificial neural networks (ANN) is a field of artificial intelligence (AI) that constitute a computational model equivalent to biological neural networks, and allow to solve real life problems. This article aims to demonstrate the ability of RNA in controlling locomotion of ants, to the effect created a software application that simulated the behavior of this insect. In the development of the same model was used incremental software development. It´s analyzed the characteristics and possible movements of the robotic ant, then through object-oriented programming in C #, create sensors, space coding matrix generation position and pheromones. Then the GUI was developed using the Visual Studio 2010 IDE. Tests were conducted under different conditions (location of ant pheromones, state sensors, etc.). Finally confirmed the correct simulation of the application successful.

Keywords: Artificial Neural Networks, Perceptron single layer, Ants, Artificial Intelligence.

1. Introducción

Las redes neuronales artificiales (RNAs), o artificial neural networks (ANNs) son un modelo simplificado de un sistema neural real [1]. Éstas se han desarrollado como generalizaciones de modelos matemáticos de sistemas nerviosos biológicos. Una primera oleada de interés en las redes neuronales surgió después de la introducción de las neuronas simplificadas por McCulloch y Pitts en 1943 también conocidos como modelos conexionistas.

Una RNA es una red de colecciones de procesadores muy simples ("neuronas"). Las unidades funcionan sólo en sus datos locales y en las entradas que reciben a través de las conexiones o enlaces que son unidireccionales [2].

En el pasado, las RNAs se han aplicado para modelar datos de gran tamaño con gran dimensionalidad ya que muestran el procesamiento de información de forma rápida y son capaces de elaborar un mapeo de las entradas y las salidas de tales datos, permitiendo que estos sean utilizados posteriormente para predecir resultados deseados [3].

El primero y más sencillo modelo de una neurona artificial fue el perceptron simple, compuesto por dos capas neuronales (entrada de tipo discreta y salida de tipo escalón), las cuales utilizan señales binarias y usan una función de activación [4].

Al ser un modelo inicial estuvo sujeto a limitaciones, como la imposibilidad de resolver problemas linealmente no separables [5], sin embargo su capacidad de aprender y reconocer patrones sencillos, fue la clave de su éxito.

Hoy, el campo de la inteligencia artificial ha evolucionado, y, las redes neuronales artificiales se utilizan en aplicaciones de amplio alcance, como la predicción del tiempo, la simulación numérica de ecuaciones no lineales, la predicción financiera, las telecomunicaciones, medicina, el procesamiento de señales y en el control de funciones motoras [3]. De allí que en este artículo se muestran los resultados de la implementación de una aplicación de software que simula el uso del perceptron simple, en el control locomotor de una hormiga robótica.

2.Materiales y métodos

La implementación del perceptron simple en C# se llevó a efecto en la carrera de Informática, del campus politécnico de la Escuela Politécnica de Manabí ESPAM MFL, entre los meses de Octubre 2012 y Febrero 2013. Se hizo uso del modelo de desarrollo de software incremental[6] a través de sus cuatro fases: análisis, diseño, codificación y pruebas.

La primera fase permitió identificar que la hormiga artificial en estudio es un agente computacionalmente simple cuya finalidad es seguir un rastro de feromonas en un mundo matricial cuadriculado. La hormiga está prevista de un sensor que detecta si la celda frente a ella contiene feromona y otro sensor que determina si dicha celda ya ha sido visitada. La hormiga es capaz de realizar las siguientes acciones: girar a la derecha, avanzar una celda hacia el frente y marcar la celda como visitada. La

hormiga sólo visitará aquellas celdas que contengan feromona y no hayan sido visitadas [7].

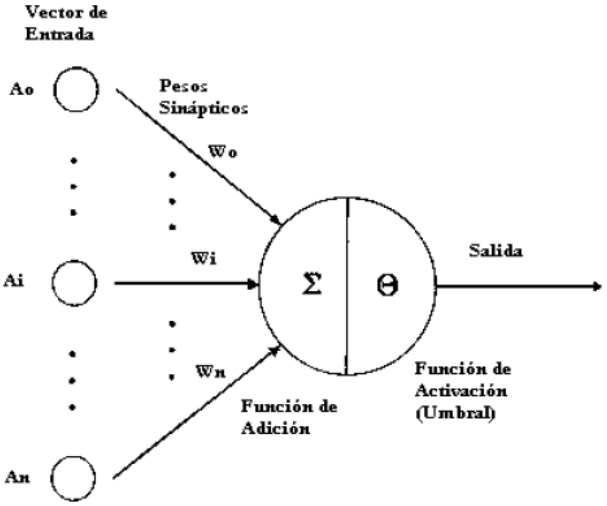


Fig. 1. Diagrama del modelo abstracto de neurona artificial

Con las características de comportamiento del agente en estudio, se continuó con la fase de diseño, apoyándose en el modelo abstracto de una neurona [8](Figura 1) y tomando en consideración el algoritmo de aprendizaje del perceptron simple[9] que se detalla a continuación:

Paso 0: Inicialización
Inicializar los pesos sinápticos con números aleatorios del intervalo [-1,1]. Ir al paso 1 con *k=1*

Paso 1: (k-ésima iteración)
Calcular

$$\mathrm{y(k)} = \mathrm{sgn} \sum_{j=1}^{n+1} w_j \, x_j \, (k)$$

Paso 2: Corrección de los pesos sinápticos
Si modificar los pesos sinápticos según la expresión:

$$w_j \, (k+1) = w_j \, (k) + \eta [z_i (k) - y_i (k)] x_j \, (k), \qquad j = 1,2, \dots, n+1$$

Paso 3: Parada
Si no se han modificado los pesos en las últimas *p* iteraciones, es decir

$$w_j(r) = w_j(k), \qquad j = 1,2,\dots,n+1, r = k+1,\dots,k+p,$$

parar. La red se ha estabilizado. En otro caso ir a **Paso 1** con *k=k+1*

Se pudo definir los sensores, y reglas de producción, (Tabla 1) [10] además de los vectores de peso, y umbrales del perceptron simple (Tabla 2)[10].

Tabla 1. Definición de sensores y reglas de producción.

Sensores	**Valores binarios**	**Significado**	**Reglas de producción**	
Sensor detector de feromona (S1)	0	No detecta feromona	$\neg S1 * S2$	Avanzar hacia el frente y marcar la celda como visitada.
	1	Detecta feromona		
Sensor de estado de celda (S2)	0	La celda no ha sido visitada		
	1	La celda ha sido visitada	$S1 + \neg S2$	Girar a la derecha.

Tabla2. Definición de vectores de peso, umbrales, acciones.

S1	**S2**	**Pesos**	**Suma ponderada**	**Umbral**	**Función de activación**	**Acción**
0	0	-1	0	-0,5	Si	Girar a la derecha
		1				
		2	0	1,5	No	-
		-2				
0	1	-1	1	-0,5	Si	Girar a la derecha
		1				
		2	-2	1,5	No	-
		-2				
1	0	-1	-1	-0,5	No	-
		1				
		2	2	1,5	Si	Avanzar al frente y marcar como visitada
		-2				
1	1	-1	0	-0,5	Si	Girar a la derecha
		1				

		2 -2	0	1,5	No	-

Una vez definido el modelo,llegó el momento de iniciar la fase de implementación, así que se procedió a representar el mismo mediante una aplicación de software creada en C# y mediante la técnica de programación orientada a objetos [11] utilizando funciones, métodos y eventos: el método `CrearMatriz` genera el mundo cuadrático en el cual se desplaza la hormiga artificial, se modificaron las propiedades de las celdas para identificar cuales están vacías (sin feromonas). La posición inicial de la hormiga artificial y la generación de feromona podía ser creada de forma aleatoria o ingresando coordenadas/cantidad por teclado.

El movimiento de la hormiga se realiza representando las entradas binarias de los sensores, y aplicando la suma ponderada a las reglas de producción (Tabla 1)(Tabla2), todo esto mediante código, a través del uso de estructuras condicionales como se muestra en el fragmento de código:

```
op = Convert.ToInt32(matriz[x, y].Tag);
    switch (op)
{
       case 2://Cuando la hormiga mira al este
               {
              if (matriz[x + 1, y].Tag.ToString() == "0" ||
                     (matriz[x + 1, y].Image.Tag.ToString() ==
                     "1"))
                   {
                      matriz[x, y].Image = down();
                      matriz[x, y].Tag = 3;
                      acumuladorgiros += 1;
                   }
             else
                if ((matriz[x + 1, y].Tag.ToString() == "1") &&
                     (matriz[x + 1, y].Image.Tag.ToString() ==
                            "0"))
                     {
                      acumuladorgiros = 0;
                      moverEste();
                 }
                     acumulamovi += 1;
                     break;
                     }
       case 3://Cuando la hormiga mira hacia abajo.
              {
       if (matriz[x, y + 1].Tag.ToString() == "0" ||
                     (matriz[x, y + 1].Image.Tag.ToString() ==
                            "1"))
              {
                matriz[x, y].Image = oeste();
                matriz[x, y].Tag = 4;
                acumuladorgiros += 1;
}
```

```
                else
                if ((matriz[x, y + 1].Tag.ToString() == "1") &&
               (matriz[x, y + 1].Image.Tag.ToString() == "0"))
                    {
                     moverAbajo();
                     acumuladorgiros = 0;
                    }
                 acumulamovi += 1;
                   break;
                   }
       case 4:
              {//Cuando la hormiga mira al oeste.
              if ((matriz[x - 1, y].Tag.ToString() == "0") ||
              (matriz[x - 1, y].Image.Tag.ToString() == "1"))
                    {
                     matriz[x, y].Image = norte();
                     matriz[x, y].Tag = 5;
                     acumuladorgiros += 1;
}
              else
                if ((matriz[x - 1, y].Tag.ToString() == "1") &&
              (matriz[x - 1, y].Image.Tag.ToString() == "0"))
                     {
                      moverOeste();
                      acumuladorgiros = 0;
                     }

                     acumulamovi += 1;
                     break;
                     }
       case 5:
              {//Cuando la hormiga mira al norte.
                 if (matriz[x, y - 1].Tag.ToString() == "0" ||
              (matriz[x, y - 1].Image.Tag.ToString() == "1"))
                     {
                         matriz[x, y].Image = este();
                         matriz[x, y].Tag = 2;
                         acumuladorgiros += 1;
}
                 else
                    if (matriz[x, y - 1].Tag.ToString() == "1" &&
                     matriz[x, y - 1].Image.Tag.ToString() ==
                            "0")
                        {
                             moverNorte();
                             acumuladorgiros = 0;
                        }
                    acumulamovi += 1;
                    break;
              }

      }
```

En el fragmento código, cada una de las condiciones toma valores de referencia a las variables de control de la matriz, que identifican los estados de dicha celda.Las condiciones de la inteligencia artificial de la hormiga robótica están configuradas de acuerdo hacia donde esta direccionada la hormiga, mediante funciones tales como `moverEste`, `moverAbajo`, `moverOeste`, `moverNorte`, logrando el movimiento esperado del insecto.

Concluidas las etapas aprendizaje y entrenamiento del perceptron, se procedió a ejecutar la última fase del modelo incremental, correspondiente a pruebas. Se posiciona la hormiga aleatoriamente en diferentes espacios de la matriz y generan celdas con feromonas, con lo que se verificó que efectivamente el agente inteligente realizaba los movimientos según las reglas de comportamiento esperado. [7].

3. Resultados

Al culminar con la creación de la hormiga robótica en C#, se obtuvo un producto que implementa una matriz de 15x15 como espacio límite que representa el ambiente en el que se desenvuelve el agente. Esta matriz, en cada iteración de la simulación, puede ser modificada según la cantidad de filas y columnas ingresadas por el usuario. De este modo se ejecutó la aplicación creando el espacio matricial mostrado en la figura 2.

(a) (b)

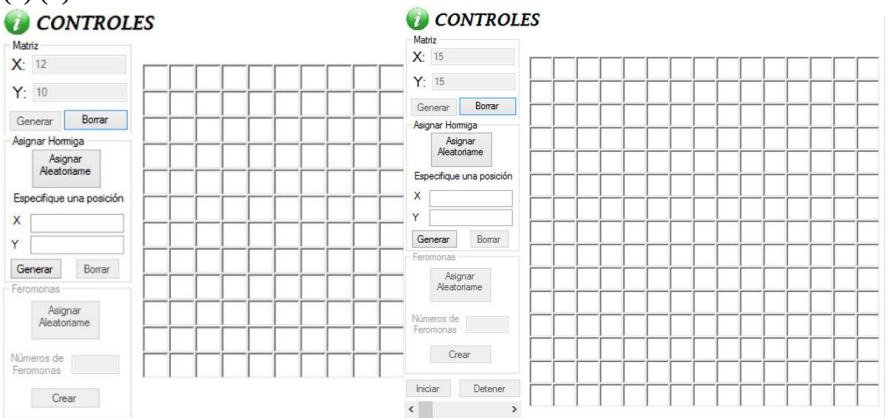
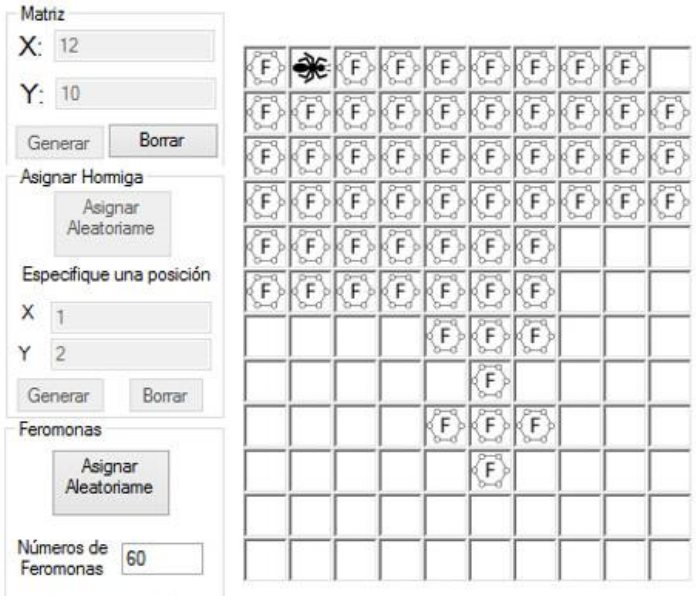


Fig. 2. Espacio matricial que representa el ambiente del agente
(a) mundo matricial 12x10
(b) mundo matricial 15x15

Con el ambiente definido, fue posible establecer la ubicación del agente en tal espacio, así como la generación aleatoria de feromona, situaciones que se visualizan en la figura 3.

(a)



(b)

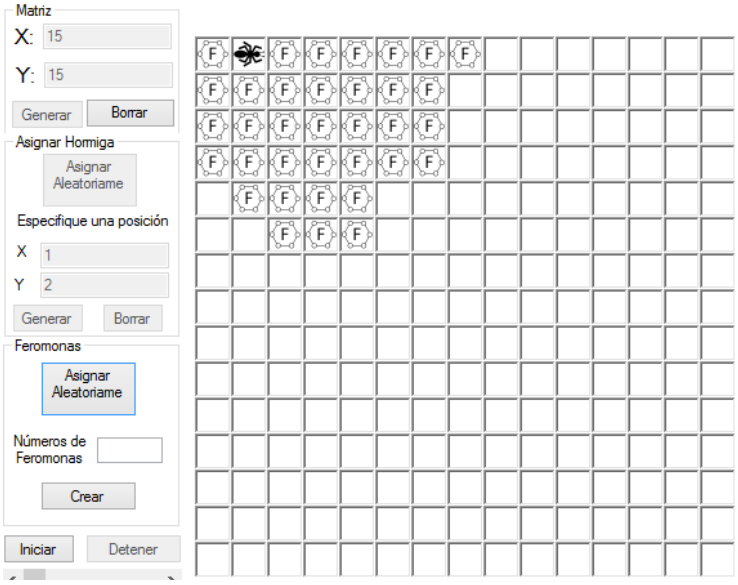


Fig. 3. Matriz que representa la ubicación de la hormiga y feromonas en el ambiente de estudio (a) agente ubicado en la celda x=1, y=2; feromonas ubicadas aleatoriamente con cantidad 60 fijada por el usuario. Espacio matricial de 12x10. (b) agente ubicado en la celda x=1, y=2 y feromonas ubicadas aleatoriamente sin cantidad establecida. Espacio matricial de 15x15.

Luego de la simulación de los movimientos de la hormiga se obtuvieron los resultados que se observan en las figura 4, 5,6.Tomando en cuenta que se estas iteraciones se efectuaron con el estado matricial correspondiente a la figura 3a

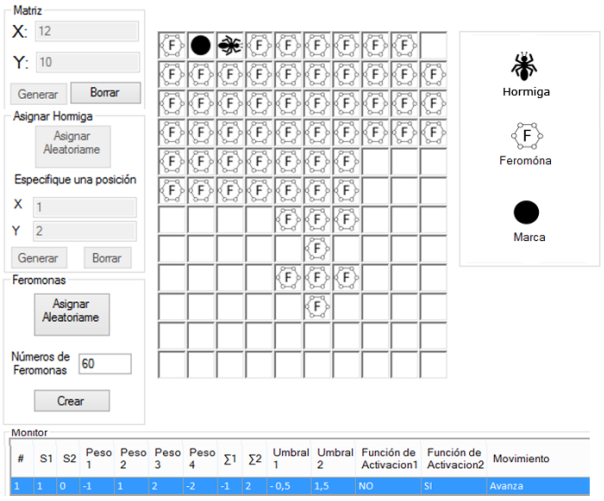


Fig. 4. Movimiento de la hormiga, según valores percibidos por los sensores. En este caso Avanzar y marcar celda como visitada

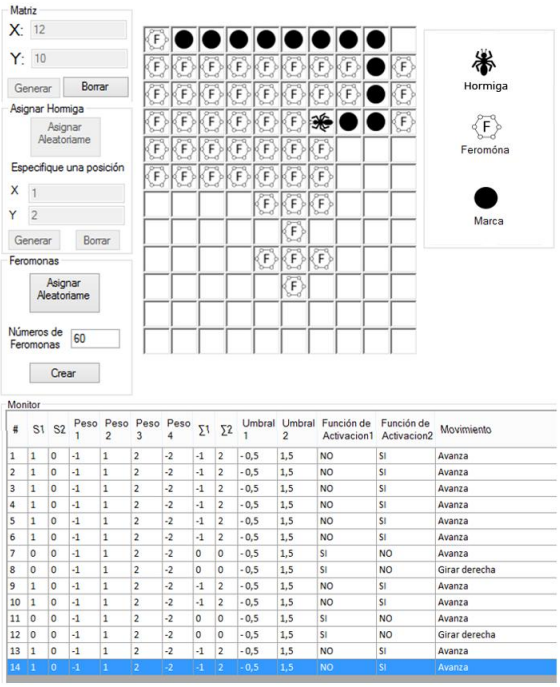
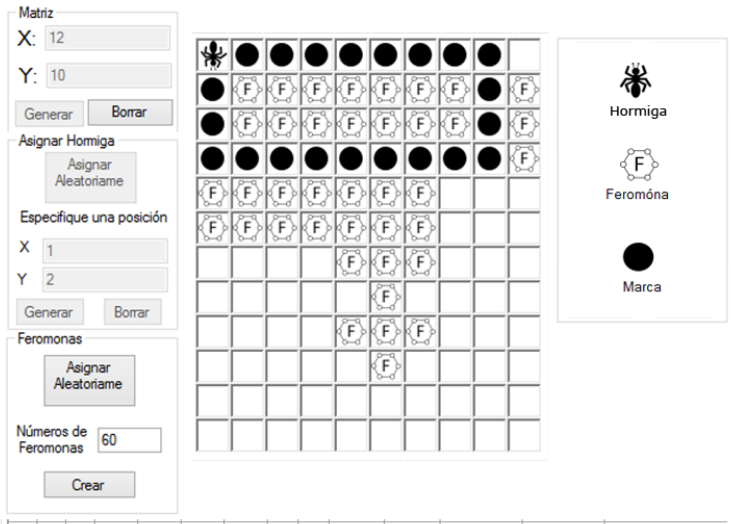


Fig. 5. Movimiento de la hormiga, según valores percibidos por los sensores. En este caso ya ha avanzado varias casillas y sigue su recorrido



14	1	0	-1	1	2	-2	-1	2	-0,5	1,5	NO	SI	Avanza
15	1	0	-1	1	2	-2	-1	2	-0,5	1,5	NO	SI	Avanza
16	1	0	-1	1	2	-2	-1	2	-0,5	1,5	NO	SI	Avanza
17	1	0	-1	1	2	-2	-1	2	-0,5	1,5	NO	SI	Avanza
18	1	0	-1	1	2	-2	-1	2	-0,5	1,5	NO	SI	Avanza
19	1	0	-1	1	2	-2	-1	2	-0,5	1,5	NO	SI	Avanza
20	0	0	-1	1	2	-2	0	0	-0,5	1,5	SI	NO	Girar Derecha
21	1	0	-1	1	2	-2	-1	2	-0,5	1,5	NO	SI	Avanza
22	1	0	-1	1	2	-2	-1	2	-0,5	1,5	NO	SI	Avanza
23	0	0	-1	1	2	-2	0	0	-0,5	1,5	SI	NO	Girar Derecha
24	1	1	-1	1	2	-2	0	0	-0,5	1,5	SI	NO	Girar derecha
25	1	1	-1	1	2	-2	0	0	-0,5	1,5	SI	NO	Girar derecha
26	0	0	-1	1	2	-2	0	0	-0,5	1,5	SI	NO	Girar Derecha
27	1	1	-1	1	2	-2	0	0	-0,5	1,5	SI	NO	Girar derecha

Fig. 6. Movimiento de la hormiga, según valores percibidos por los sensores. En este caso, luego de varios acciones gira a la derecha y finaliza.

4. Discusión

Las técnicas computacionales, desde sus inicios, marcaron un paradigma en la creación. Susaplicaciones, hoy día, van desde la industria de los juegos hasta las cadenas de producciónde varias empresas. Para ello se han desarrollado múltiples técnicas como las relacionadascon la inteligencia artificial. Las más conocidas son la lógica difusa (aprendizaje inductivo),algoritmos genéticos y sobre todolas redes neuronales artificiales [11]. Las RNAs presentan muchas facilidades para simular procesos relacionados con el comportamiento de varias especies de seres vivos, de hecho un estudio elaborado por Miranda, D. *et al.* [12]utilizó ésta técnica para modelar el comportamiento de ratas en laberinto en cruz elevadodado susignificado neurobiológico, debido a que la finalidad del estudio era comprender estados anímicos en pruebas de medicamentos ansiolíticos y ansiogénicos para determinar trastornos de ansiedad generalizada. Otro ejemplo de los usos de las RNAs se

vislumbra en un trabajorealizado en 2011 porLuera, W. y Minim, L. [13],en el cual se construyó una red de tipo recurrente dado que el estudio buscaba la modelización del tratamiento térmico de alimentos, el cual es un proceso de tiempo dependiente situación idónea para este tipo de redes.

En ambos estudios las redes neuronales fueron utilizadas únicamente como modelo matemático para el procesamiento de datos.

En el presente estudio la situación es diferente pues se simuló las funciones locomotoras a través de una aplicación de software, de esto modo la complejidad aumenta pues con un paradigma convencional de programación, el objetivo es modelar matemáticamente el problema en cuestión y posteriormente formular una solución (programa) mediante un algoritmo codificado que tenga una serie de propiedades que permitan resolver dicho problema. En contraposición, la aproximación basada en las RNA parte de un conjunto de datos de entrada suficientemente significativo y el objetivo es conseguir que la red aprenda automáticamente las propiedades deseadas [14]. Es precisamente esto último lo que se logró en el presente estudio pues la hormiga logró *aprende*r desplazarse y buscar feromonas en el ambiente matricial dado, gracias a la representación en C# de sus sensores y acciones, las cuales fueron establecidas a través del algoritmo de aprendizaje de la red neuronal artificial conocida como perceptron simple.

5. Referencias

1. Karimi, H., Navid, H., Mahmoudi, A.: Detection of damaged seeds in laboratory evaluation of precision planter using impact acoustics and artificial neural networks. En: Artificial Intelligence Research, vol. 1, num. 2, pp 67 a 74. Canadá (2012)

2. Kumar, K., Mitra, G.: Advanced Applications of Neural Networks and Artificial Intelligence: A Review. En: International Journal of Information Technology and Computer Science, vol.4, num. 6, pp. 57 a 68.Estados Unidos de América (2012)

3. Mantzari, V., Mantzaris, D.: Solar radiation: Cloudiness forecasting using a soft computing approach. En: Artificial Intelligence Research, vol. 2, num. 1, pp. 69 a 80. Estados Unidos de América (2013)

4. Ngaopitakkul, A., Pothisarn, C.: Combination of discrete wavelet transform and probabilistic neural network algorithm for detecting fault location on transmission system. En: International Journal of Innovative Computing, Information and Control, vol. 7, num. 4, pp. 1861 a 1873.Estados Unidos de América (2011)

5. Tsiotas, D., Polyzos, S.: The contribution of ANN's simple perceptron pattern to inequalities measurement in Regional Science. En: Operational Research, vol. 12, num. 3, pp. 190 a 199. Polonia (2012)

6. Jensen, C., Mark, L., Roussopoulos, N.: Incremental Implementation Model for Relational Databases with Transaction Time. En: IEEE Transactions on Knowledge and Data Engineering – TKDE, vol. 3, num. 4, pp. 461 a 473.Dinamarca (1991)

7. Russell, S., Norvig, P. En: Inteligencia Artificial Un Enfoque Moderno. Segunda Edición. Pearson Education. España (2008)

8. Buendía, E., Vargas, E., Leyva, A., Terrazas, S.: Aplicación de redes neuronales artificiales y técnicas SIG para la predicción de coberturas forestales. En: Revista Chapingo Serie Ciencias Forestales y del Ambiente, vol. 8, num. 1, pp. 31 a 37. México (2002)

9. Ortiz, J.: El perceptrón simple. Tema 4. pp. 4. En: Departamento de Lenguajes y Ciencias de la Computación, Universidad de Málaga. España (2009)

10. Cedeño, L.: Redes Neuronales Artificiales. Tema 1. En: Carrera de Informática, Escuela Politécnica de Manabí. Ecuador (2008)

11.Sotolongo, G. y Guzmán, M.: Aplicaciones de las redes neuronales. El caso de la bibliometría. En: Revista Ciencias de la Información, vol. 32, num. 1, pp. 27 a 34. México (2001)

12. Miranda, D., Conde, C., Celis, C., Corzo, S.: Modelado del Comportamiento de Ratas en Laberinto en Cruz Elevado Basado en Redes Neuronales Artificiales. En: Revista Colombiana de Física, vol. 41, num. 2, pp. 406 a 408. Colombia (2009)

13. Luera, W. y Minim, L.: Aplicación de redes neuronales artificiales en la modelización del tratamiento térmico de alimentos. En: Ciencia y Tecnología Alimentaria, vol. 3, num. 2, pp. 81 a 88. México (2001)

14. Eivar, G.: Diseño de un robot cartesiano para ordenar elementos electrónicos. En: Facultad de Ingeniería en Sistemas, Electrónica e Industrial, Universidad Técnica de Ambato. Ecuador (2009)