3. APLICACIONES DE LAS REDES NEURONALES A INGENIERÍA ELÉCTRICA

Las aplicaciones que se estudian en el capítulo 3, representan una serie de problemas típicos en Ingeniería Eléctrica, los cuales ya han sido resueltos por métodos tradicionales. El objetivo de emplear Redes Neuronales en su solución es ofrecer una alternativa novedosa que simplifica y permite comparar la eficiencia de este nuevo enfoque respecto a las metodológicas tradicionales, a través de conocimientos matemáticos de un nivel aceptable para comprender el fundamento y desarrollo del problema a resolver.

3.1 DETECCIÓN DE OBSTÁCULOS POR MEDIO DE UN ROBOT

3.1.1 Descripción del problema.

Un robot es un dispositivo automático que realiza acciones especificas, que dependen de las necesidades del proceso en que se encuentre involucrado, en este caso se tiene un robot que cuenta con cuatro sensores de proximidad en distintas ubicaciones que permanentemente detectan si hay objetos que se

encuentren a una distancia superior o inferior a la preestablecida, con base en esto se decide si dar marcha adelante o atrás a cada uno de los dos motores que posee; en las lecturas de los sensores podrían darse 16 posibles combinaciones ($16=2^4$) y para cada combinación cada uno de los dos motores podría dar marcha adelante o marcha atrás.

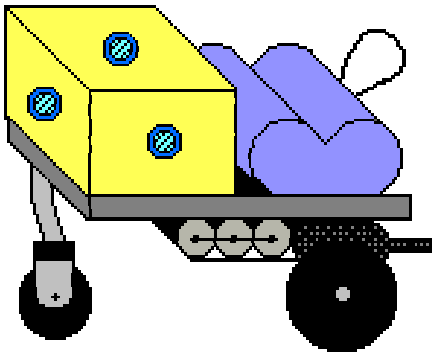

Figura 3.1.1 Robot

El comportamiento del robot lo describe la tabla 3.1.2, cuando los sensores detecten un objeto que se encuentra a una distancia inferior a la predeterminada se dirá que el objeto se encuentra cerca y esto se representa por medio de un 1 y cuando se detecte un objeto que se encuentra a una distancia mayor que la predeterminada se dirá que el objeto esta lejos lo cual se indica con un –1; dependiendo de estas lecturas los motores podrán dar marcha adelante, lo que se representara por un 1 o dar marcha atrás con un –1.

S1	S2	S3	S4	M1	M2
1	1	1	1	-1	-1
-1	1	1	1	-1	1
1	1	-1	-1	1	-1
-1	-1	-1	-1	1	1
1	-1	1	1	1	-1
1	1	-1	1	-1	1
1	1	1	-1	1	-1

Tabla 3.1.1 Comportamiento del robot

3.1.2 Justificación del tipo de red.

Este tipo de problema generalmente es resuelto suministrándole al robot una base de datos que contiene todas las posibles situaciones que se podrían presentar y sus respectivas soluciones, en este caso se necesitaría almacenar las respuestas para ambos motores ante las 16 posibles combinaciones en las lecturas de los sensores, cuando el número de variables de entrada y el número de salidas es mucho mayor, la cantidad de datos necesarios para especificar cada posible situación crece indefinidamente, debido a esto se requerirían dispositivos con gran capacidad de almacenamiento; *en contraste, una red neuronal puede entrenarse con un número representativo de patrones y aprender el comportamiento del sistema utilizando dispositivos de menos capacidad de almacenamiento y costo.*

Una red tipo Perceptrón puede ser entrenada con patrones de cualquier dimensión en la entrada y en la salida con datos binarios, por la simplicidad del problema este tipo de red es la mas adecuada.

Para garantizar que el problema puede ser resuelto por una red neuronal tipo Perceptrón se debe comprobar que los patrones de entrenamiento son linealmente separables, para esto se deben plantear las desigualdades generadas por cada patrón de entrenamiento, en este caso cada patrón de cuatro dimensiones generara dos desigualdades (una por cada salida), estas desigualdades no deben contradecirse, si esto ocurriera el problema no podría ser resuelto por una red tipo Perceptrón de una sola capa y deberá buscarse otro tipo de solución.

Debido a la naturaleza bipolar de la salida, la función de transferencia a utilizar es *hardlims* (ver sección 1.3.2), la cual se rige por las siguientes condiciones.

$$\begin{Bmatrix} n \geq 0 & a = 1 \\ n < 0 & a = -1 \end{Bmatrix} \qquad (3.1.1)$$

La salida de la red está dada por la ecuación 3.1.2, (ver sección 2.1)

$$n = (\boldsymbol{W}\boldsymbol{p} + b) \qquad (3.1.2)$$

Aplicando esta ecuación a cada patrón de entrenamiento se tienen las desigualdades de la tabla 3.1.2, las cuales se satisfacen plenamente, lo que implica que el problema es linealmente separable y puede ser resuelto por una red tipo Perceptron

$$p_1\begin{cases}W_{11}+W_{12}+W_{13}+W_{14}+b_1<0\\W_{21}+W_{22}+W_{23}+W_{24}+b_2<0\end{cases}\qquad p_2\begin{cases}-W_{11}+W_{12}+W_{13}+W_{14}+b_1<0\\-W_{21}+W_{22}+W_{23}+W_{24}+b_2\geq 0\end{cases}$$

$$p_3\begin{cases}W_{11}+W_{12}-W_{13}-W_{14}+b_1\geq 0\\W_{21}+W_{22}-W_{23}-W_{24}+b_2<0\end{cases}\qquad p_4\begin{cases}-W_{11}-W_{12}-W_{13}-W_{14}+b_1\geq 0\\-W_{21}-W_{22}-W_{23}-W_{24}+b_2\geq 0\end{cases}$$

$$p_5\begin{cases}W_{11}-W_{12}+W_{13}+W_{14}+b_1\geq 0\\W_{21}-W_{22}+W_{23}+W_{24}+b_2<0\end{cases}\qquad p_6\begin{cases}W_{11}+W_{12}-W_{13}+W_{14}+b_1<0\\W_{21}+W_{22}-W_{23}+W_{24}+b_2\geq 0\end{cases}$$

$$p_7\begin{cases}W_{11}+W_{12}+W_{13}-W_{14}+b_1\geq 0\\W_{21}+W_{22}+W_{23}-W_{24}+b_2<0\end{cases}$$

Tabla 3.1.2 Desigualdades que garantizan que el problema sea linealmente separable

3.1.3 Entrenamiento de la red.

A la red se le presentaran 7 patrones de la tabla 3.1.1, para los cuales dependiendo de las lecturas de los sensores se le dirá al robot que hacer específicamente y luego se probará la red con los casos restantes para comprobar la capacidad de generalización de la red neuronal ante un patrón nunca antes visto.

Los estados de lecturas de los sensores y de operación de los motores fueron designados con 1 y –1, puesto que para la convergencia del proceso de entrenamiento resulta más ventajosos propagar valores de 1 y –1 que de 1 y 0.

Debido a la naturaleza de la salida y de la entrada de la red, la función de transferencia apropiada es *hardlims*, la cual trabaja con valores bipolares.

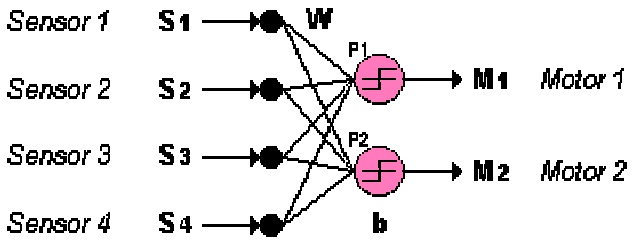

Figura 3.1.2 Red tipo Perceptrón

Se creará una red de 4 entradas con una neurona tipo Perceptrón para cada salida, teniendo así una salida bidimensional, los pesos iniciales aleatorios de la red se muestran en la tabla 3.1.3

Wi=net.IW{1,1}

	S1	S2	S3	S4
M1	0.8636	-0.1627	0.0503	0.3443
M2	-0.0680	0.6924	-0.5947	0.6762

bi=net.b{1}

M1	1
M2	1

Tabla 3.1.3 Pesos Iniciales

El siguiente código crea una red tipo Perceptrón con función de transferencia *hardlims*, dos neuronas en la salida, utiliza como patrones de entrenamiento las lecturas de los cuatro sensores almacenados en ***p*** y como patrones objetivo o salidas deseadas las acciones de ambos motores almacenados en el vector ***t***.

```
net=newp([-1 1;-1 1;-1 1;-1 1],2,'hardlims');
net.adaptParam.passes=200;
Wi;
[net,a,e]=adapt(net,P,t);
Wf=net.IW{1,1};
bf=net.b{1};
```

Los pesos finales de la red entrada que satisface todos los patrones de entrada y salida son:

Wf=net.IW{1,1}

	S1	S2	S3	S4
M1	0.8636	-4.1627	0.0503	-3.6557
M2	-4.0680	0.6924	-4.5947	4.6762

bf=net.b{1}

M1	1
M2	1

Tabla 3.1.4 Pesos finales red entrenada

La red fue simulada para la totalidad de combinaciones posibles de entrada para comprobar que no exista error en el aprendizaje de los patrones de entrenamiento y para observar su capacidad de generalización en los casos restantes

```
S1=[1 -1]; S2=[1 -1]; S3=[1 -1]; S4=[1 -1];
P=combvec(S1,S2,S3,S4)
net=newp([-1 1;-1 1;-1 1;-1 1],2,'hardlims');
Wf;
t=sim(net,P);
```

La respuesta de la red a todos los patrones de entrenamiento fue exitosa, como puede observarse en la tabla 3.1.5

	S1	S2	S3	S4	M1	M2
P1	**1**	**1**	**1**	**1**	**-1**	**-1**
P2	**-1**	**1**	**1**	**1**	**-1**	**1**
P3	**1**	**1**	**-1**	**-1**	**1**	**-1**
P4	**-1**	**-1**	**-1**	**-1**	**1**	**1**
P5	**1**	**-1**	**1**	**1**	**1**	**-1**
P6	**1**	**1**	**-1**	**1**	**-1**	**1**
P7	**1**	**1**	**1**	**-1**	**1**	**-1**

Tabla 3.1.5 Simulación de la red para los Patrones de Entrenamiento

La red fue simulada para las posibles combinaciones restantes obteniéndose los siguientes resultados:

	S1	S2	S3	S4	M1	M2
C1	-1	-1	1	1	1	1
C2	-1	1	-1	1	-1	1
C3	1	-1	-1	1	1	1
C4	-1	-1	-1	1	1	1
C5	-1	1	1	-1	-1	-1
C6	1	-1	1	-1	1	-1
C7	-1	-1	1	-1	1	-1
C8	-1	1	-1	-1	-1	1
C9	1	-1	-1	-1	1	-1

Tabla 3.1.6 Simulación de la red para las nuevas combinaciones

Las combinaciones que no hacían parte del set de entrenamiento, al ser presentadas a la red fueron aproximadas al patrón del set de entrenamiento aprendido con menor distancia euclidiana.

Para las combinaciones C1, C2 y C4 la red decidió dar marcha adelante a ambos motores; para la combinación C5 la red decidió dar marcha atrás a ambos motores, para las combinaciones C6, C7 y C9 la red decidió dar marcha adelante

a M1 y marcha atrás a M2 y para las combinaciones C2 y C8 la red decidió dar marcha atrás a M1 y marcha adelante a M2

Una red tipo Perceptrón de una sola capa es una buena solución a un problema que involucre patrones linealmente separables, en el caso de contar con patrones que no son linealmente separables se tiene la alternativa de utilizar una red Perceptrón multicapa o cambiar definitivamente de red, nótese que una red Perceptrón multicapa puede solucionar el problema de separabilidad lineal a medida que aumenta el número de capas de la red.

La capacidad de generalización de las redes neuronales juega un papel importante cuando las posibles combinaciones de patrones de entrada son tantas que resultaría imposible especificarle a un dispositivo que hacer en cada caso, puesto que la red se entrena con un número de patrones representativo y no con la totalidad de ellos, ahorrando tiempo de cómputo en la solución del problema.

En las aplicaciones desarrolladas con redes neuronales juega un papel importante la tolerancia a fallas que las caracteriza, pues en caso de fallar uno o varios sensores (como en este caso) la red siempre producirá una salida que en la mayoría de los casos es la mas acertada, debido a que la red después de un proceso de aprendizaje exitoso esta en capacidad de generalizar el comportamiento del sistema.