

# Reporte de práctica 3 - Redes neuronales

1288433 - Pedro I. López [dreilopz@gmail.com]

4 de mayo de 2012

**Maestro:** Dr. Luis M. Torres Treviño

**Materia:** 483 PERCEPCIÓN (Laboratorio)

**Dependencia:** FIME UANL

## 1. Introducción

El ejercicio práctico consiste en implementar una red neuronal basándose en teoría discutida en clase. Al igual que con la calculadora analógica y el generador de funciones, las entradas del sistema son 2 señales analógicas con un rango de de 0-5 V (potenciómetros rotacionales). La salida del sistema se mide directamente en el pin de salida analógica del USB6008 (USB6008.A00).

## 2. Descripción

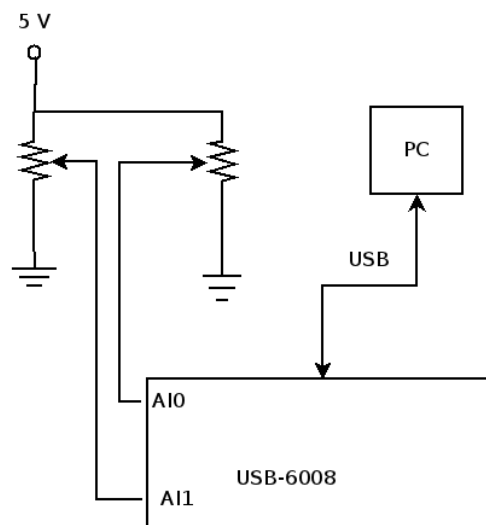


Figura 1: Diagrama de sistema.

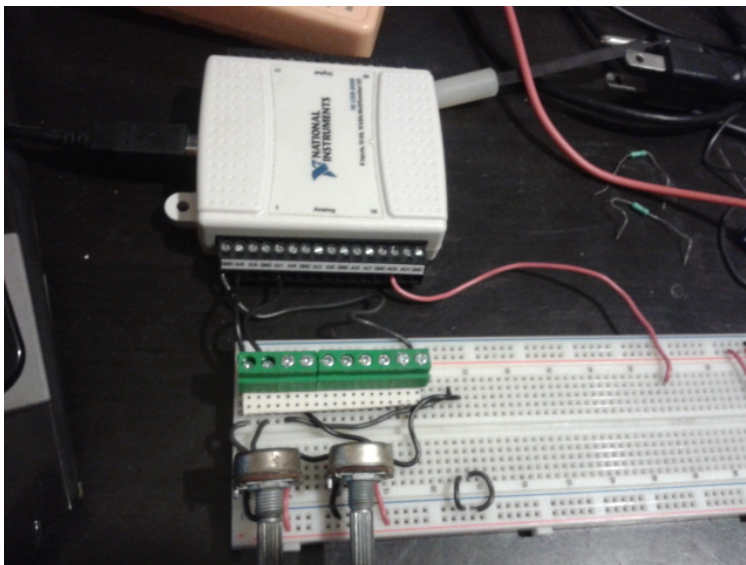


Figura 2: Fotografía de sistema.

La red neuronal se implementará con un USB6008, una PC y otros componentes eléctricos montados en un protoboard, con un programa desarrollado en lenguaje MATLAB. Los 2 potenciómetros rotacionales servirán para proporcionar 2 entradas analógicas. La fase de entrenamiento de la red neuronal se realizó en la clase, obteniendo la matriz de pesos  $w$  y  $o$ . Basándose en el código Scilab proporcionado por el Dr. Torres, la misma red neuronal fue implementada en MATLAB. La fase de entrenamiento arrojó un error de 0,011345 aproximadamente.

La NI USB-6008 (USB6008) es una interfaz de adquisición y salida de datos para PC con conexión USB. Los voltajes de los potenciómetros se envían a los canales analógicos de entrada USB6008.AI0 y USB6008.AI1, y la salida se detecta en USB6008.A00. El software se desarrolló en MATLAB, utilizando el *Data Acquisition Toolbox* para obtener acceso a la API de la USB6008: *NIDAQ-mx*. La interfaz del programa es una línea de comandos interactiva.

Componentes utilizados:

- 2 potenciómetros de  $10K\Omega$
- 1 NI USB6008
- Alambre
- Conectores
- 1 fuente de poder de 5 V.
- 1 PC corriendo Windows XP.

### 3. Resultados

La red neuronal presenta un comportamiento satisfactorio, como se puede ver en la Figura 5 el programa muestra el resultado producido por la red neuronal, el resultado real de la operación aritmética y el voltaje de salida. Nótese que el voltaje de salida corresponde a un escalamiento lineal del resultado de la red neuronal, de manera que la salida siempre sea entre 0-5 V.

1. Ejecutar MATLAB y cambiar el directorio actual al de la práctica.
2. Ejecutar p3main.
3. Variar los voltajes analógicos de entrada con los potenciómetros y observar los cambios en la pantalla.



Figura 3: Fotografía de sistema.

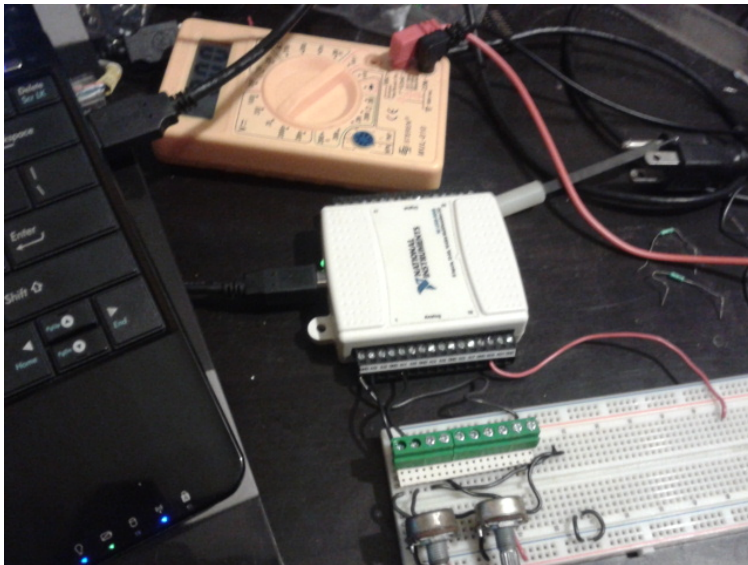
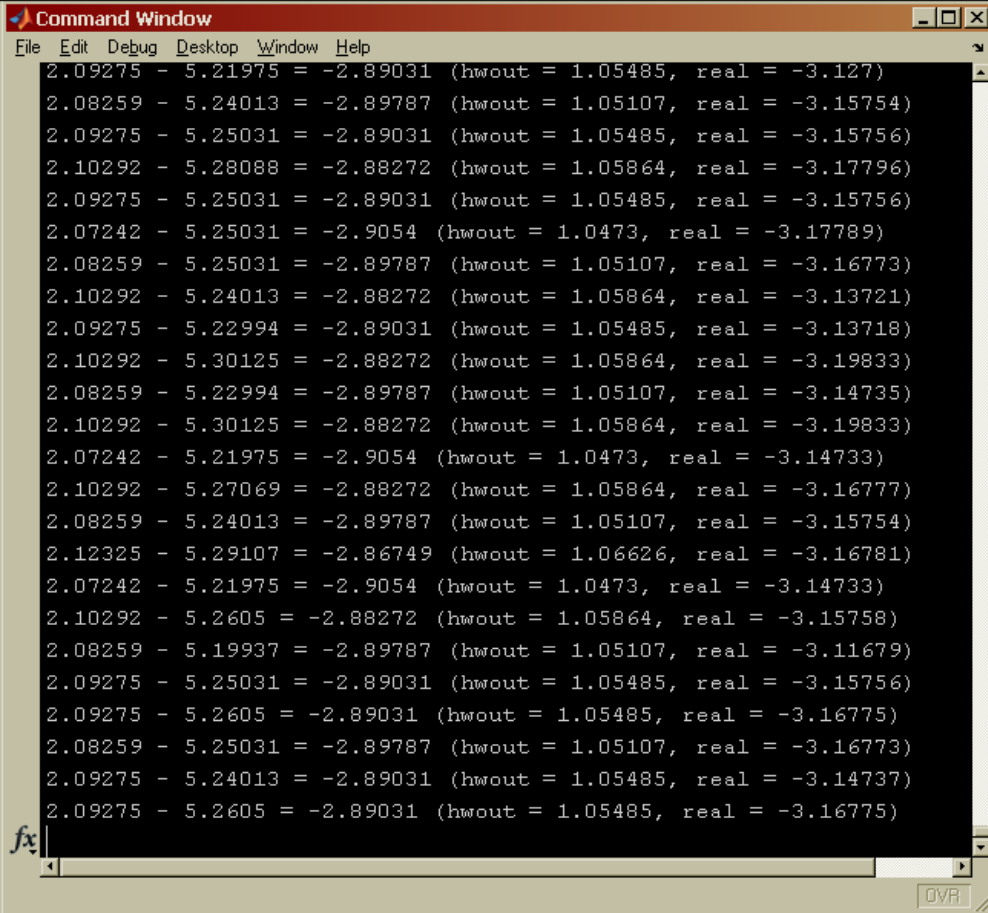


Figura 4: Fotografía de sistema.

## 4. Pruebas



```
Command Window
File Edit Debug Desktop Window Help
2.09275 - 5.21975 = -2.89031 (hwout = 1.05485, real = -3.127)
2.08259 - 5.24013 = -2.89787 (hwout = 1.05107, real = -3.15754)
2.09275 - 5.25031 = -2.89031 (hwout = 1.05485, real = -3.15756)
2.10292 - 5.28088 = -2.88272 (hwout = 1.05864, real = -3.17796)
2.09275 - 5.25031 = -2.89031 (hwout = 1.05485, real = -3.15756)
2.07242 - 5.25031 = -2.9054 (hwout = 1.0473, real = -3.17789)
2.08259 - 5.25031 = -2.89787 (hwout = 1.05107, real = -3.16773)
2.10292 - 5.24013 = -2.88272 (hwout = 1.05864, real = -3.13721)
2.09275 - 5.22994 = -2.89031 (hwout = 1.05485, real = -3.13718)
2.10292 - 5.30125 = -2.88272 (hwout = 1.05864, real = -3.19833)
2.08259 - 5.22994 = -2.89787 (hwout = 1.05107, real = -3.14735)
2.10292 - 5.30125 = -2.88272 (hwout = 1.05864, real = -3.19833)
2.07242 - 5.21975 = -2.9054 (hwout = 1.0473, real = -3.14733)
2.10292 - 5.27069 = -2.88272 (hwout = 1.05864, real = -3.16777)
2.08259 - 5.24013 = -2.89787 (hwout = 1.05107, real = -3.15754)
2.12325 - 5.29107 = -2.86749 (hwout = 1.06626, real = -3.16781)
2.07242 - 5.21975 = -2.9054 (hwout = 1.0473, real = -3.14733)
2.10292 - 5.2605 = -2.88272 (hwout = 1.05864, real = -3.15758)
2.08259 - 5.19937 = -2.89787 (hwout = 1.05107, real = -3.11679)
2.09275 - 5.25031 = -2.89031 (hwout = 1.05485, real = -3.15756)
2.09275 - 5.2605 = -2.89031 (hwout = 1.05485, real = -3.16775)
2.08259 - 5.25031 = -2.89787 (hwout = 1.05107, real = -3.16773)
2.09275 - 5.24013 = -2.89031 (hwout = 1.05485, real = -3.14737)
2.09275 - 5.2605 = -2.89031 (hwout = 1.05485, real = -3.16775)
```

Figura 5: Screenshot de programa en MATLAB.

Como se puede observar en la Figura 5 la red neuronal presenta un comportamiento satisfactorio.

## 5. Código

Nótese que varios fragmentos importantes del código fueron basados en el código proporcionado por el Dr. Torres en la clase para la fase de entrenamiento.

Listing 1: p3main: Script principal de la red neuronal.

```
1 delete_daq_objects;
2 io_init;
3
4 m = [-1.2356573    0.9077403   -0.0741589;
5      -0.1231165   0.6287125    0.0598237  ;
6      -1.5654024   0.8574440   -0.0061877  ;
7      0.8678764    -0.4978647   -0.1730943  ;
8      0.8729172    -1.6574365   -0.0719177];
9
10 o = [-1.7722135  -0.3331932  -2.4102099  1.2988263  3.4213232];
11
12 mrange = [0 5; 0 5; -5 5];
13
14 while true
15     usenn(operands, resulthw, m, o, mrange);
16 end
```

Listing 2: delete\_daq\_objects: Script que elimina los objetos DAQ en caso de que existan.

```
1 try
2     delete(operands);
3     delete(resulthw);
4     clear('operands' , 'resulthw');
5 end
```

Listing 3: io\_init: Inicialización de objetos de hardware.

```
1 DAQ_DEVICE_NAME = 'usb6008';
2 ADAPTOR = daqhwinfo('nidaq');
3
4 delete_daq_objects
5
6 operands = analoginput(ADAPTOR.AdaptorName, DAQ_DEVICE_NAME);
7 operands.SampleRate = 1000;
8 operands.InputType = 'SingleEnded';
9 operands.TriggerType = 'Immediate';
10 operands.SamplesPerTrigger = 2;
11 addchannel(operands, 0:1);
12 resulthw = analogoutput(ADAPTOR.AdaptorName, DAQ_DEVICE_NAME);
13 addchannel(resulthw, 0);
```

Listing 4: usenn: Obtiene el resultado de la aplicación utilizando los pesos obtenidos en la fase de entrenamiento.

```

1 function result = usenn(operands, resulthw, m, o, mrange)
2
3 start(operands);
4 pause(0.15);
5 data = getdata(operands);
6 X = data(1,:);
7
8 x1n = normalization(X(1), mrange(1,1), mrange(1,2));
9 x2n = normalization(X(2), mrange(2,1), mrange(2,2));
10
11 Xn = [x1n x2n 1];
12
13 [sm so neto netm] = ForwardBKG(Xn, m, o);
14 result = desnormalization(so, mrange(3,1), mrange(3,2));
15
16 resulthwval = (result/2) + 2.5;
17 if resulthwval > 5.0
18     resulthwval = 5.0;
19 elseif resulthwval < 0.0
20     resulthwval = 0.0;
21 end
22 result_real = X(1) - X(2);
23
24 putsample(resulthw, resulthwval);
25
26 report_str = sprintf('%g - %g = %g (hwout = %g, real = %g)', X(1), X(2), ...
27 result, resulthwval, result_real);
28 disp(report_str);

```

Listing 5: normalization: Normalización de valores, basado en el código proporcionado por Dr. Torres

```

1 function a = normalization(Van,LMIN,LMAX)
2
3 a=(Van-LMIN)/((LMAX-LMIN)+0.000001);
4
5 if a>1
6     a=1;
7 end
8 if a<0
9     a=0;
10 end

```

Listing 6: ForwardBKG: Implementación de la red neuronal, basado en el código proporcionado por Dr. Torres

```

1 function [sm,so,neto,netm] = ForwardBKG(VI,m,o)

```

```

2
3 [TMID TINP] = size(m);
4 [TOUT TMID] = size(o);
5
6 neto=zeros(1,TOUT);
7 netm=zeros(1,TMID);
8
9 so = zeros(1,TOUT);
10 sm = zeros(1,TMID);
11
12 for y=1:TMID
13     netm(y)=0;
14 end
15
16 for y=1:TMID
17     for x=1:TINP
18         netm(y) = netm(y) + m(y,x) * VI(x);
19     end
20     sm(y) = fa(netm(y));
21 end
22
23 for z=1:TOUT
24     neto(z)=0;
25 end
26
27 for z=1:TOUT
28     for y=1:TMID
29         neto(z) = neto(z) + o(z,y) * sm(y);
30     end
31     so(z) = fa(neto(z));
32 end

```

Listing 7: fa: basado en el código proporcionado por Dr. Torres

```

1 function y = fa(x)
2
3 if x>20
4     x=20;
5 end
6
7 if x<-20
8     x=-20;
9 end
10
11 x = exp(-x);
12 y = 1 / ( 1 + x );

```



Listing 8: desnormalization: Desnormalización de los valores. basado en el código proporcionado por Dr. Torres

```
1 function a = desnormalization(Vn,LMIN,LMAX)
2
3 a=Vn*LMAX+LMIN*(1-Vn);
4
5 if a>LMAX
6     a=LMAX;
7 end
8
9 if a<LMIN
10    a=LMIN;
11 end
```