UNIVERSIDAD EIA
Ser, Saber y Servir

PARCIAL 2021_2

SISTEMAS INTELIGENTES

7 de septiembre de 2021

Santiago García Arango

NOTA: En el quiz se permitirá tener programas desarrollados previamente y utilizarlos como plantilla, pero no se permitirán notas ni diapositivas, como tampoco el uso de internet para fines diferentes al de acceder a TEAMS.

PUNTO 1 (2.0)

Suponga que se tiene un cultivo de palma donde se desean clasificar 4 tipos de estados de la misma, Estado 1, Estado 2, Estado 3 y Estado 4.

Usted como experto en sistemas inteligentes ha sido contratado para desarrollar un sistema que permita clasificar entonces 4 tipos de estados genéricos de la palma según las siguientes características:

- Color de la hoja (Verde, amarilla, café, verde claro)
- Color del fruto (Rojo, amarillo, verde, café)
- Tallo (bajo, grueso, alto, corto)
- Tamaño fruto (no tiene, pequeño, mediano, grande)

Se pide lo siguiente:

- Asigne valores numéricos a cada cualidad de cada categoría(Especifique por escrito).
- Diseñe una tabla para identificar el estado genérico de la palma con mínimo 4 combinaciones para cada estado(Especifique por escrito).
- En MATLAB, randomice la base de datos para generar almenos 60 datos (15 datos que correspondan a cada categoría)

La salida de la red debe ser de la siguiente manera:

Salida de la red para cada estado genérico de palma

SALIDA	Estado 1	Estado 2	Estado 3	Estado 4
1	**1**	**0**	**0**	**0**
2	**0**	**1**	**0**	**0**
3	**0**	**0**	**1**	**0**
4	**0**	**0**	**0**	**1**

En MATLAB, entrene la red neuronal que considere conveniente para este caso, se pide lo siguiente:

- Base de datos en un espacio de trabajo .mat que se llame BASE_DATOS.mat el cual contenga entradas y deseados.
- Programa de la red neuronal.
- Coloque un Alfa, número de ocultas y número de iteraciones estáticas, para que luego el profesor pueda correr el programa con las variables que usted ha seleccionado y entrenar la red.
- Muestre el error cuadrático medio de entrenamiento en una gráfica y la matriz de confusión. (Se busca que la red sobrepase el 80% de aciertos totales)
- Entrene la red y guarde el espacio de trabajo COMPLETO después de entrenamiento como RESULTADOS_PUNTO1.mat para que el profesor pueda evaluar su programa.
- Guarde la BASE_DATOS.mat, programas de la red y RESULTADOS_PUNTO1.mat en una carpeta llamada PUNTO1_NOMBRE
- Documente en un PDF ó Word, el proceso paso a paso, gráficas y análisis con sus propias palabras.

1

- Color de la hoja (Verde, amarilla, café, verde claro)
- Color del fruto (Rojo, amarillo, verde, café)
- Tallo (bajo, grueso, alto, corto)
- Tamaño fruto (no tiene, pequeño, mediano, grande)

x_1 x_2 x_3 x_4

Color de hoja	Color fruto	Tallo	Tamaño fruto
Verde	Rojo	Bajo	No tiene
Amarillo	Amarillo	Grueso	Pequeño
Café	Verde	Alto	Mediano
Verde claro	Café	Corto	Grande

Equivalencia

x_1 x_2 x_3 x_4

Color de hoja	Color fruto	Tallo	Tamaño fruto
1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4

Teniendo en cuenta esta clasificación general y numérica, se procede a generar un dataset con patrones de palmas específicos... Es decir, con combinaciones estratégicas de datos que pertenezcan a un tipo de palma.

Este proceso se realizará siguiendo patrones en MATLAB, para un total de 60 series de datos, con entradas que siguen patrones particulares...

Por ejemplo, tipos de planta con aleatorios de 1-2, 2-3, 3-4, etc...

En la próxima página, se mostrará el algoritmo para generar esta base de datos pseudo-aleatoria para el proceso de clasificación y entrenamiento futuro de la red.

Nota. Este proceso creará un archivo “.mat” llamado “PALMS.mat”, que tiene la info de la base de datos ya creada y será el insumo para los próximos numerales...

Con este script creamos la base de datos llamada “PALMS.mat”

```
% SIMPLE SCRIPT TO CREATE "PALMS" WORKSPACE AND SAVE IT TO "PALMS.mat"
% Santiago Garcia Arango

% Important variables to fill input data
N = 60;  % Total data collected  for all palms
data_collection_size = 15;  % Amount of data per type of palm
types_of_entries = 4;  % Amount of entries per data taken
types_of_outputs = 4;  % Amount of "classifications" for the desired
 outputs

% Initialize entries and desired
entries = zeros(types_of_entries, N);
desired = zeros(types_of_outputs, N);

% Create entries matrix
current_mean_value = 1;
central_value = 1;
counter = 0;
limit_1 = 4;
for j=1:N
    if (counter == data_collection_size)
        counter = 0;
        current_mean_value = current_mean_value + 1;
    end

    if (current_mean_value == limit_1)
        current_mean_value = 1;
    end

    for i=1:types_of_entries
        entries(i, j) = randi([current_mean_value, current_mean_value
 + 1]);
    end
    counter = counter + 1;
end

% Create desired matrix
current_category = 1;
counter = 0;
for j=1:N
    if (counter == data_collection_size)
        counter = 0;
        current_category = current_category + 1;
    end
    desired(current_category, j) = 1;
    counter = counter + 1;
end

% Automatically save the data to the "PALMS.mat" workspace
save("PALMS.mat", "entries", "desired");
```

Published with MATLAB® R2020a

El paso siguiente es el desarrollo de la selección y entrenamiento de la red neuronal.

Para este caso, se implementará una red MADALINE con salida de PERCEPTRON, que tenga la posibilidad de tener una capa oculta con funciones sigmodiales y la capa de salida sea un limitador duro. Esto es porque debemos poder clasificar las salidas en rangos específicos de tipos de palma. Es por esto que esta selección de red es una muy buena alternativa..

De igual forma, es importante tener presente, que se implementará un proceso de añadir una serie de entradas adicionales, que sean variaciones sutiles de las mismas, con el fin de darle más diversidad a los datos y que los pesos puedan entrenar mejor el sistema completo...

A continuación, se muestran resultados y valores importantes para el entrenamiento de la red neuronal de 2 capas, con la capa de salida de tipo limitador duro (perceptron).

$\alpha = 0,1$

$nmax = 1000$

$no = 10$

El alfa se seleccionó en 0.1 para la primera iteración y fue un valor excelente para todos los procesos de aprendizaje.

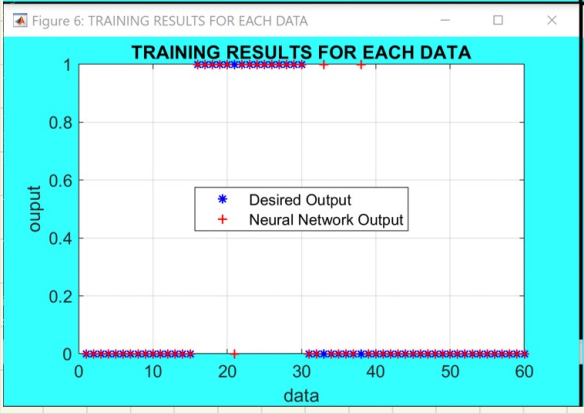
Nmax se seleccionó de 1000, para garantizar que sí se redujera al máximo el valor del ecm en el tiempo.

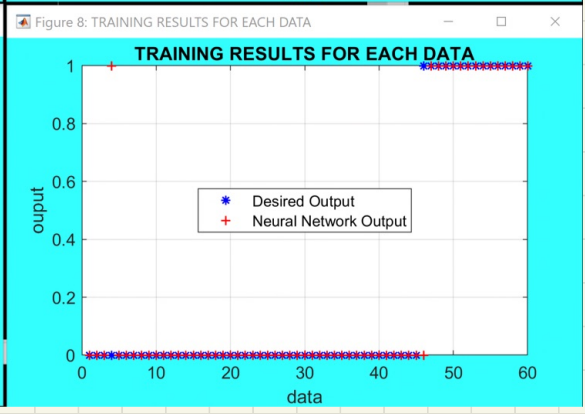
Las ocultas se seleccionaron 10, porque al ser cuatro tipo principales se palmas, es un valor coherente mayor al doble, que le permitirá a la red ser rápida, mientras almacena la información.

Los resultados de este proceso fueron EXITOSOS!!

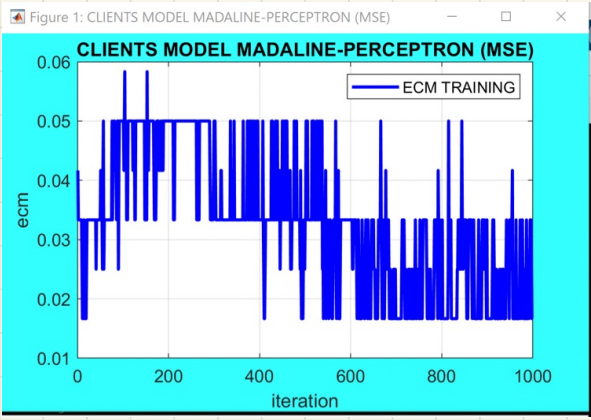
En la próxima página se explican y analizan.

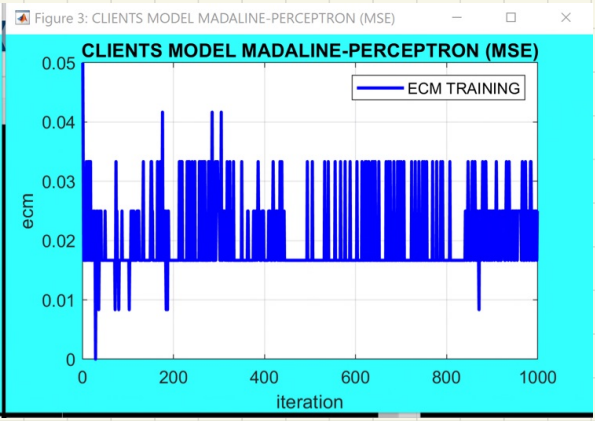
En primer lugar, se puede evidenciar las gráficas de los datos de entrenamiento, en donde se ven los resultados esperados y los de la salida de la red (Yd vs Yk). Estos se ven que son demasiado similares:

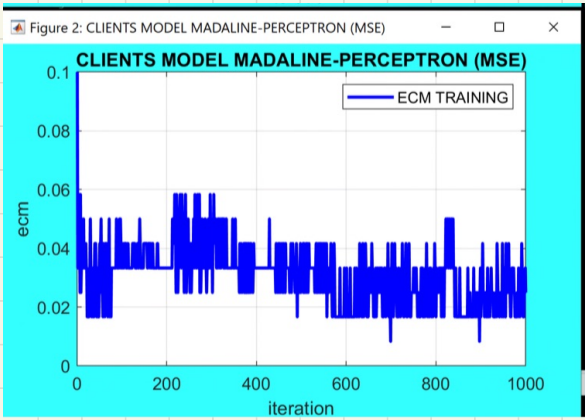


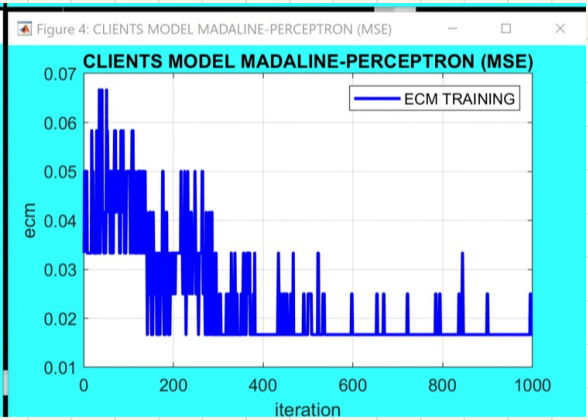


De igual forma, podemos observar los resultados del ECM, que en cada iteración es menor, y luego se mantiene estándar en rangos pequeños (lo mejor que puede dar la red).



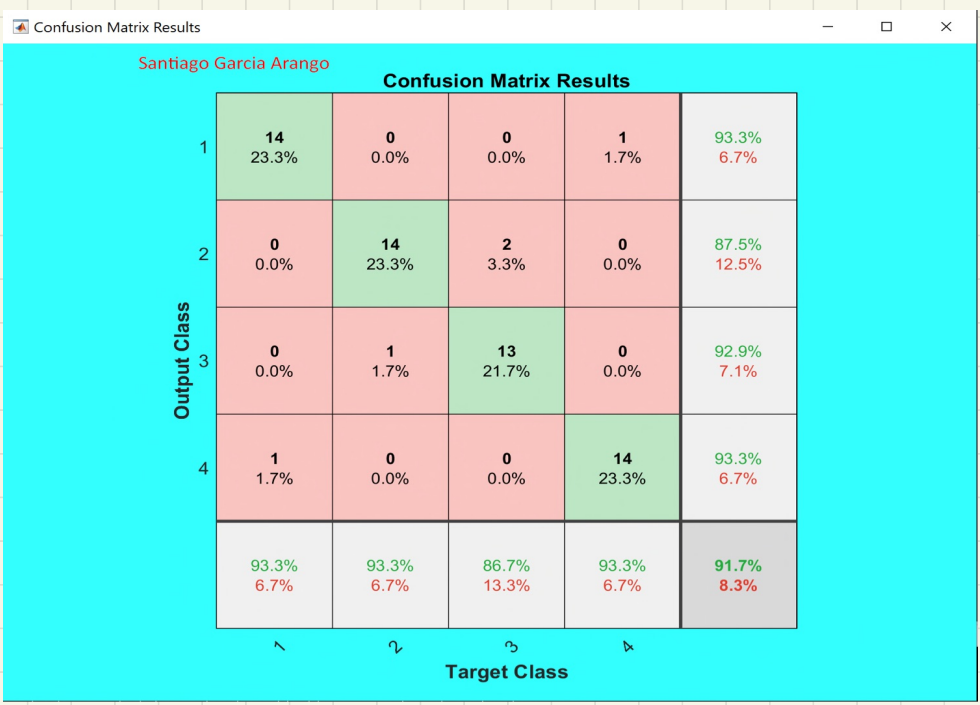






Finalmente, al ser una red clasificadora, nos interesa saber qué tan bien es capaz de clasificar los datos de entrada con base en los tipos de palma...

Este proceso se puede realizar con el toolbox de MATLAB de DeepLearning, con ayuda de la matriz de confusión...



Esta matrix nos informa que la clasificación total de las palmas es mayor al 91%, donde es un valor muy bueno y nos da indicaciones de que la red fue entrenada correctamente. También vemos que aunque comete algunos errores, fue un resultado muy exitoso, teniendo en cuents el poco tiempo de entrenamiento, los pocos datos y que apenas usamos 1 capa.

PUNTO 2 (2.0)

Se tiene una red neuronal, la cual se busca aplicar para una serie temporal, representada por la siguiente ecuación:

$$Y = \cos(X^2 + 2X^3)$$

Se busca caracterizar la serie temporal en el rango de (-5,5) con un total de 500 datos.

Se pide lo siguiente:

- Analice la serie con el diagrama de autocorrelación parcial, analice que tipo de serie temporal es y explique resultados.
- Desarrolle una base de datos a partir de la serie temporal con el tamaño de ventana adecuado.
- Ajuste y entrene una red neuronal que considere adecuada para esta clase de base de datos.(Realice experimentos con diferentes alfa, ocultas e iteraciones hasta que encuentre un sistema que se ajuste a la serie temporal de manera adecuada)
- Muestre resultados en un PDF ó Word y analice con sus propias palabras.

PUNTO 3 (1.0)

Se pide lo siguiente:

- Diseñe una red neuronal que considere adecuada para el punto anterior y dibuje el esquema de la red, con todos los componentes, número de entradas y salidas, funciones matemáticas, etc, de manera detallada.(Seleccione una función de activación gaussiana para la capa oculta y sigmoidal a la salida)
- Defina paso a paso las reglas de aprendizaje para los pesos, en base a la regla delta generalizada, detalle paso a paso las derivadas necesarias y llegue a la fórmula de aprendizaje.

NOTA: Guarde las carpetas de los Puntos 1 y 2 en una carpeta llamada PARCIAL_NOMBRE y comprímala para subirla a TEAMS.

MUCHOS ÉXITOS!!

2
3

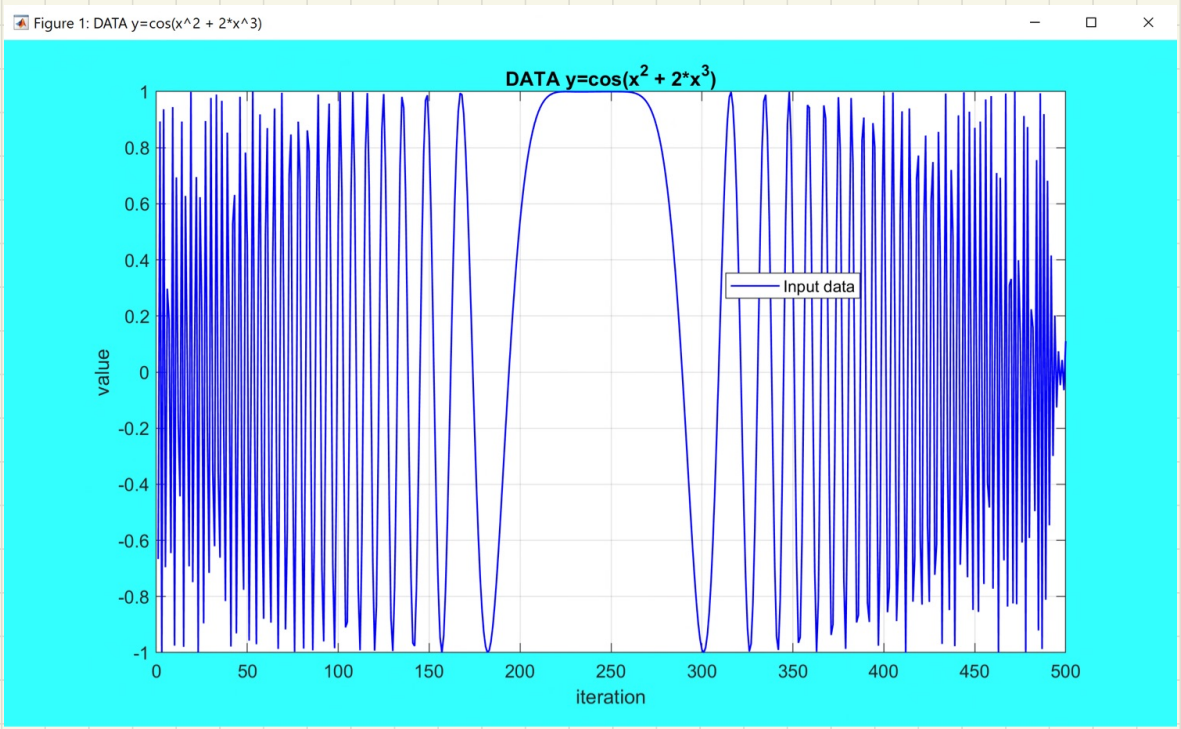
Inicialmente, se generará la gráfica de la serie de tiempo data a través de la función...

$$y = \cos\left(x^2 + 2 \cdot x^3\right)$$

Este proceso nos dicen que debe ser en el rango horizontal, para valores de “x” que pertenezcan a (-5, 5)....

Esto implica que obligatoriamente el primer valor de dicha serie temporal debe ser y(-5) y el último ser y(5). Los valores intermedios serán definidos en un rango equivalente de datos para ajustar los 498 datos faltantes.

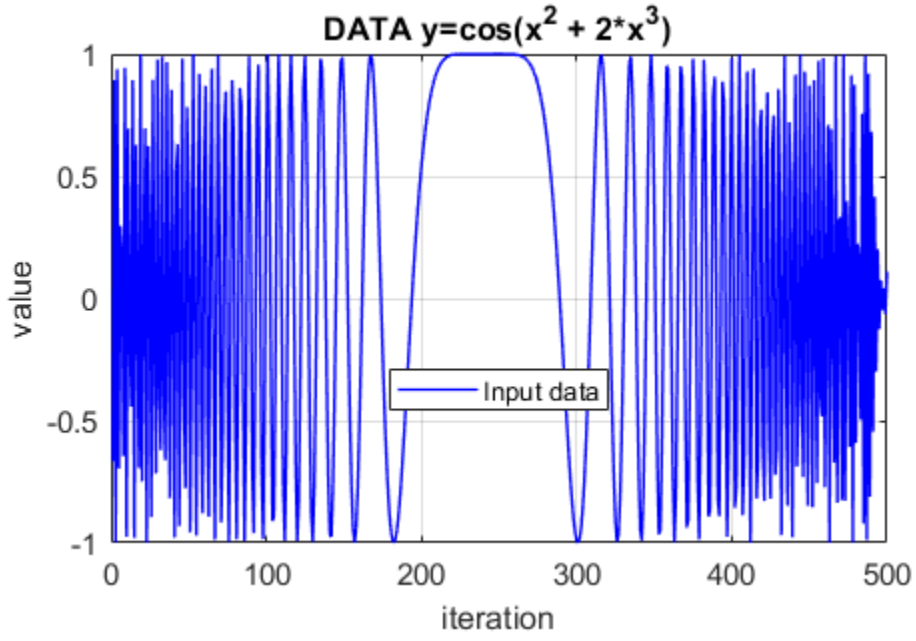
Es por esto, que se procede a realizar este procesamiento de los datos en MATLAB y el resultado en términos temporales es el siguiente:



Estos datos es importante tenerlos presente de que la iteración inicial representa un valor de “-5” y la iteración final un valor de “5”. (Simplemente fue el rango deseado para tener énfasis en él y agregar la mayor cantidad de datos al sistema).

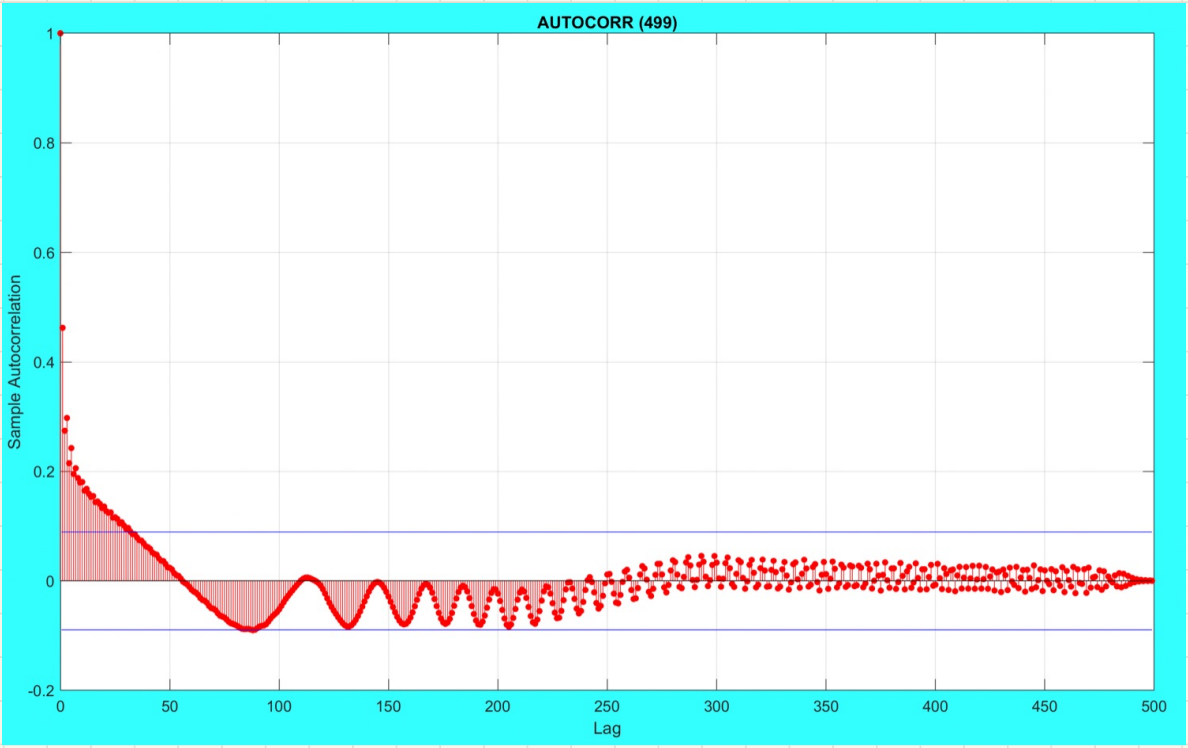
Explicación de proceso de creación de timeseries...

```
% SIMPLE SCRIPT TO CREATE "TIMESERIESDATA" WORKSPACE AND SAVE IT TO
 "TIMESERIESDATA.mat"
% Santiago Garcia Arango

N = 500;  % Total data
entries = zeros(1, N);
desired = zeros(1, N);

for i=1:N
    entries(1, i) = i;
    % function is y = cos(x^2 + 2*x^3) ... from (x=-5 to x=5)
    value = i/50 - 5;
    desired(1, i) = cos(value^2 + 2*value^3);
end

data = desired;

% Automatically save the data to the "TIMESERIESDATA.mat" workspace
save("TIMESERIESDATA.mat", "entries", "desired", "data");

figure;
plot(desired,'-b',"Linewidth",1);
xlabel('iteration');
ylabel('value');
legend('Input data', "Location", "best");
prettygraph("DATA y=cos(x^2 + 2*x^3)");

figure;
autocorr(desired, 499);
prettygraph("AUTOCORR (499)");
```



El proceso que sigue, es analizar el diagrama de autocorrelación. Este diagrama proveniente de econometría nos permite analizar dos factores importantes:
1. La ventana adecuada (aproximada) para el correcto procesamiento de datos.
2. El tipo de serie temporal al que nos estamos enfrentando.

El diagrama de autocorrelación resultante para este problema es el siguiente:



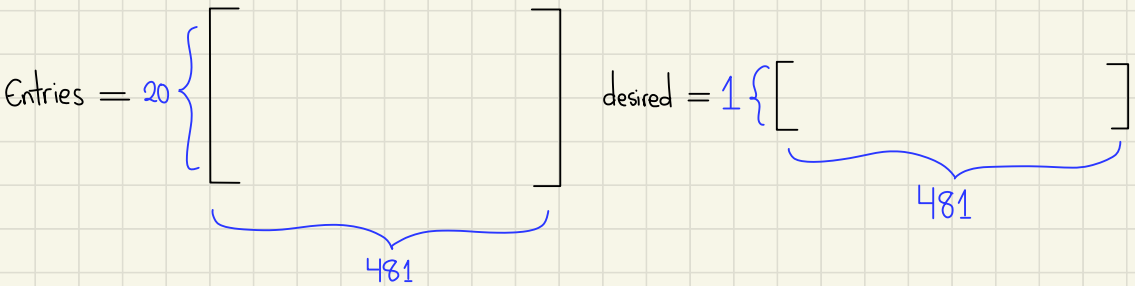
Este diagrama nos da a entender entonces información muy importante.

En primer lugar, el número de ventana aproximado para comenzar a entrenar el sistema dinámico, sería alrededor de 30. Esto quiere decir que tenemos que preparar los datos para tomar alrededor de 30 valores previos como entradas del sistema en el momento de querer predecir el instante actual. (Esta ventana puede modificarse sutilmente según se vaya probando en la práctica).

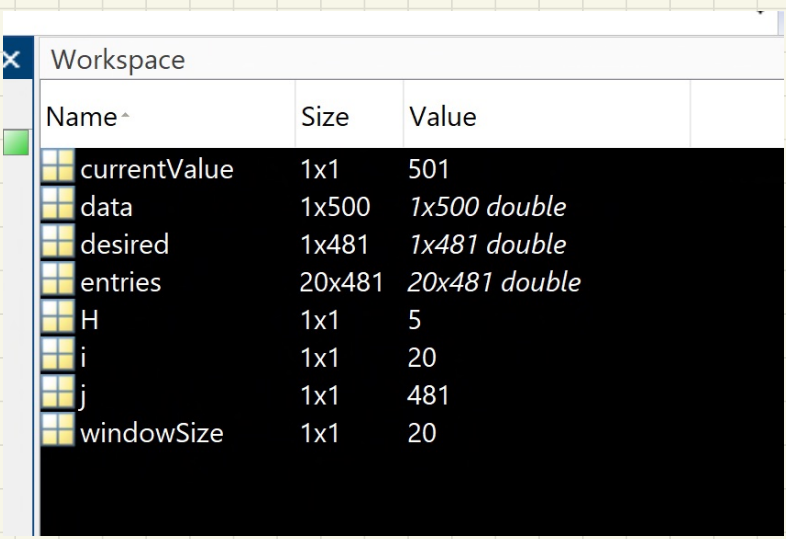
El segundo elemento interesante, es que en primera instancia el sistema tiene una tendencia hacia abajo, y luego se queda entre los tresholds superiores e inferiores, lo que nos dice que no existe una representación o patrón significativo de los datos luego de que el sistema lleva alrededor de 80 iteraciones.

Para realizar la base de datos de entrenamiento, se tomará un tamaño de ventana de exactamente 20. Esto quiere decir que requerimos preparar una base de datos que considere los últimos 20 valores de entradas basado en el comportamiento temporal.

En términos generales, es entrenar al sistema con una matriz de entradas y salidas así:



Estas matrices están desarrolladas teniendo en cuenta el número de ventana explicado de 20 (nótese las filas de las entradas) y con 481 datos (pues para cada uno de los elementos de los datos, requerimos al menos que de esos 500 datos, unos 20 previos se tomen antes de comenzar a entrenar la red. Este procesamiento se puede evidenciar en mejor detalle en la siguiente página.



Se puede visualizar cómo estos datos preparados de 500 iteraciones, ahora se transformaron en los insumos de entrada a nuestro sistema a entrenar. Esto quiere decir que ya hicimos la preparación y depuración de los mismos. En un futuro, sería ideal implementar modelos más robustos como Box-Jenkins para garantizar eliminar el ruido de las señales y que quede como un sistema más ideal.

Explicación de algoritmo para crear base de datos de preparar entradas y desired...

```
% SAMPLE SCRIPT TO LOAD AND PLAY WITH A TIME SERIES
% SANTIAGO GARCIA ARANGO

% Clean workspace data
clear; close all; clc;

% Access upper folder functions
addpath(genpath("../"));

% Load data
fprintf("...Loading database...\n");
load("TIMESERIESDATA.mat");

% Important variables for time-series
windowSize = 20;
H = 5;


% Create entries based on data
entries = zeros(windowSize, size(data, 2) - windowSize + 1);

for i=1:windowSize
    currentValue=i;
    for j=1:size(data, 2)-windowSize+1
        entries(i, j) = currentValue;
        currentValue = currentValue + 1;
    end
end

desired = zeros(1, size(data, 2)-windowSize+1);
for j=1:size(data, 2)-windowSize+1
    desired(1, j) = data(1, j+windowSize-1);
end

...Loading database...
```

Published with MATLAB® R2020a

Ahora se procede a explicar la red que va a ser implementada...

→ Con funciones de activación:

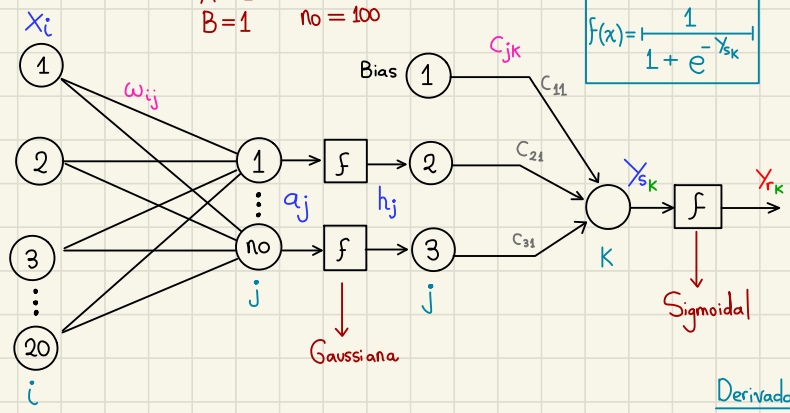
$A = 1$ $\quad ne = 20$

$B = 1$ $\quad no = 100$

$$f(x) = A \cdot e^{-B \cdot x^2}$$

y

$$f(x) = \frac{1}{1 + e^{-Y_{sK}}}$$



Reglas de entrenamiento:

$$C_{jk}{}_t = C_{jk}{}_{t-1} - \alpha \cdot \frac{\partial (ecm)}{\partial C_{jk}{}_{t-1}}$$

$$\omega_{ij}{}_t = \omega_{ij}{}_{t-1} - \alpha \cdot \frac{\partial (ecm)}{\partial \omega_{ij}{}_{t-1}}$$

Error Cuadrático medio (ecm):

$$\frac{e_k^2}{2}{}_t = ecm_t = \frac{1}{2}\left(Yd_{t-1} - Yr_{t-1}\right)^2$$

$$a_j = \sum_{i=1}^{ne} \omega_{ij} \cdot X_i \qquad W_{ij} \cdot X_i$$

$$Y_{sK} = \sum_{j=1}^{no+1} C_{jK} \cdot h_j \qquad C_{jK} \cdot h_j$$

$$Y_{rK} = f_{act}(Y_{sK})$$

$$Y_{rK} = A \cdot e^{-B \cdot (Y_{sK})^2}$$

$$h_j = f_{act}(a_j)$$

$$h_j = C \cdot e^{-D \cdot (a_j)^2}$$

Derivada del "ecm" con respecto a "Y_{rK}":

$$ecm = \frac{1}{2}\left(Yd - Y_{r_K}\right)^2$$

$$\frac{\partial (ecm)}{\partial Y_{rK}} = \frac{2}{2}\left(Yd - Y_{r_K}\right) \cdot (-1) = Y_{rK} - Yd$$

Ejemplo de función activación:

$$Y_{rK} = f(Y_{sK}) = \frac{1}{1 + e^{-Y_{sK}}}$$ Sigmoidal

$$\frac{\partial Y_{rK}}{\partial Y_{sK}} = Y_{r_K} \cdot (1 - Y_{r_K}) = \heartsuit$$

Paso 2: Encontremos $\frac{\partial (ecm)}{\partial C_{jK}}$, $\frac{\partial (ecm)}{\partial \omega_{ji}}$

$$\frac{\partial (ecm)}{\partial C_{jK}} = \frac{\partial (ecm)}{\partial Y_{rK}} \cdot \frac{\partial Y_{rK}}{\partial Y_{sK}} \cdot \frac{\partial Y_{sK}}{\partial C_{jK}}$$

$$\frac{\partial (ecm)}{\partial C_{jK}} = (Y_{r_K} - Yd) \cdot \heartsuit \cdot h_j$$

"+" por el "-" "-"

$$C_{jk}{}_t = C_{jk}{}_{t-1} + \alpha \cdot (Yd - Y_{rK})_{t-1} \cdot \heartsuit \cdot h_j$$

Derivada de "Y_{sK}" con respecto a "C_{jK}" y "h_j":

$$Y_{sK} = \sum_{j=1}^{no+1} C_{jK} \cdot h_j = C_{jK} \cdot h_j$$

$$\frac{\partial Y_{sK}}{\partial C_{jK}} = h_j \qquad \frac{\partial Y_{sK}}{\partial h_j} = C_{jK}$$

Derivada de "h_j" con respecto a "a_j":

$$h_j = f(a_j) = C \cdot e^{-D \cdot (a_j)^2}$$ Gaussiana

$$\frac{\partial h_j}{\partial a_j} = -2 \cdot C \cdot D \cdot e^{-D \cdot (a_j)^2} = \bigstar$$

Derivada de "a_j" con respecto a "ω_{ij}":

$$h_j = \sum_{i=1}^{ne} \omega_{ij} \cdot x_i = W_{ij} \cdot X_i$$

$$\frac{\partial h_j}{\partial \omega_{ij}} = X_i$$

$$\frac{\partial (ecm)}{\partial \omega_{ij}} = \frac{\partial (ecm)}{\partial Y_{rK}} \cdot \frac{\partial Y_{rK}}{\partial Y_{sK}} \cdot \frac{\partial Y_{sK}}{\partial h_j} \cdot \frac{\partial h_j}{\partial a_j} \cdot \frac{\partial a_j}{\partial \omega_{ij}}$$

$$\frac{\partial (ecm)}{\partial \omega_{ij}} = (Y_{r_K} - Yd) \cdot \heartsuit \cdot C_{jK} \cdot \bigstar \cdot X_i$$

$$\omega_{ij}{}_t = \omega_{ij}{}_{t-1} + \alpha \cdot (Yd - Y_{rK})_{t-1} \cdot \heartsuit \cdot C_{jK} \cdot \bigstar \cdot X_i$$

Al ejecutar dicha red neuronal, con su respectivo entrenamiento, haciéndolo con las siguientes funciones de activación:

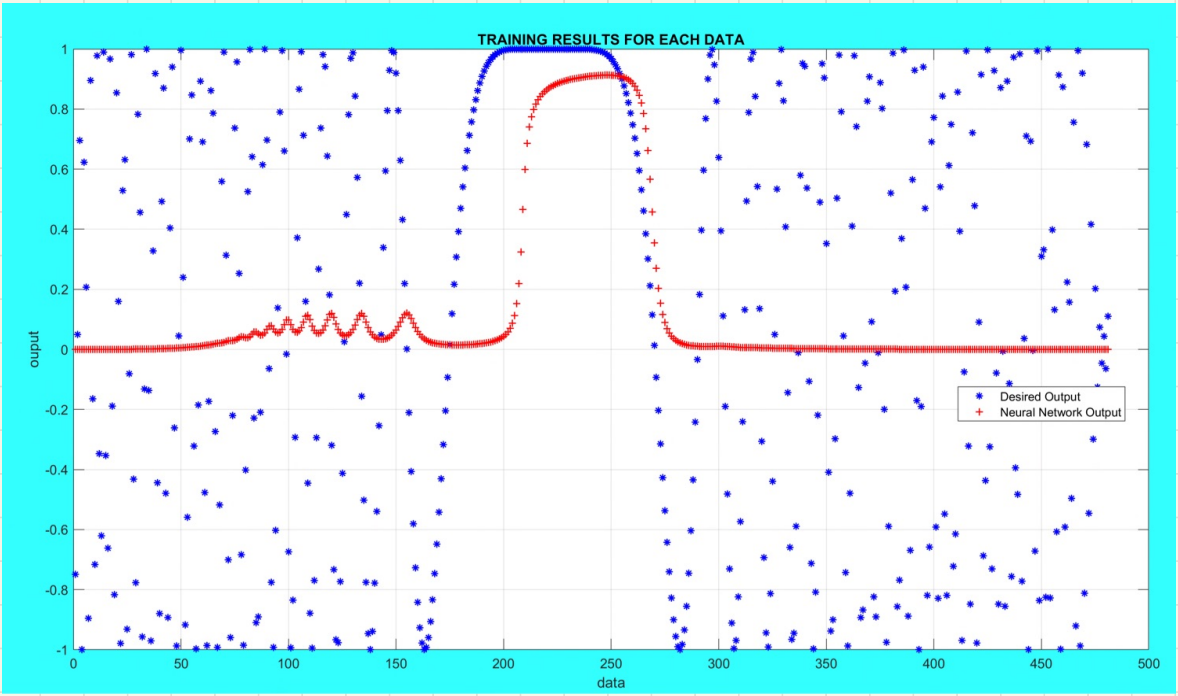
$$f(x) = A \cdot e^{-B \cdot x^2}$$

Capa 1

$$f(x) = \frac{1}{1 + e^{-\gamma_k}}$$

Capa 2

Se obtuvo:



NO se obtuvieron los resultsdos esperados... digamos que fue bueno, pero no tan bueno como pensaba...

Es por esto que quise mejorar el sistema e implementé una nueva red con las funciones de activación así:

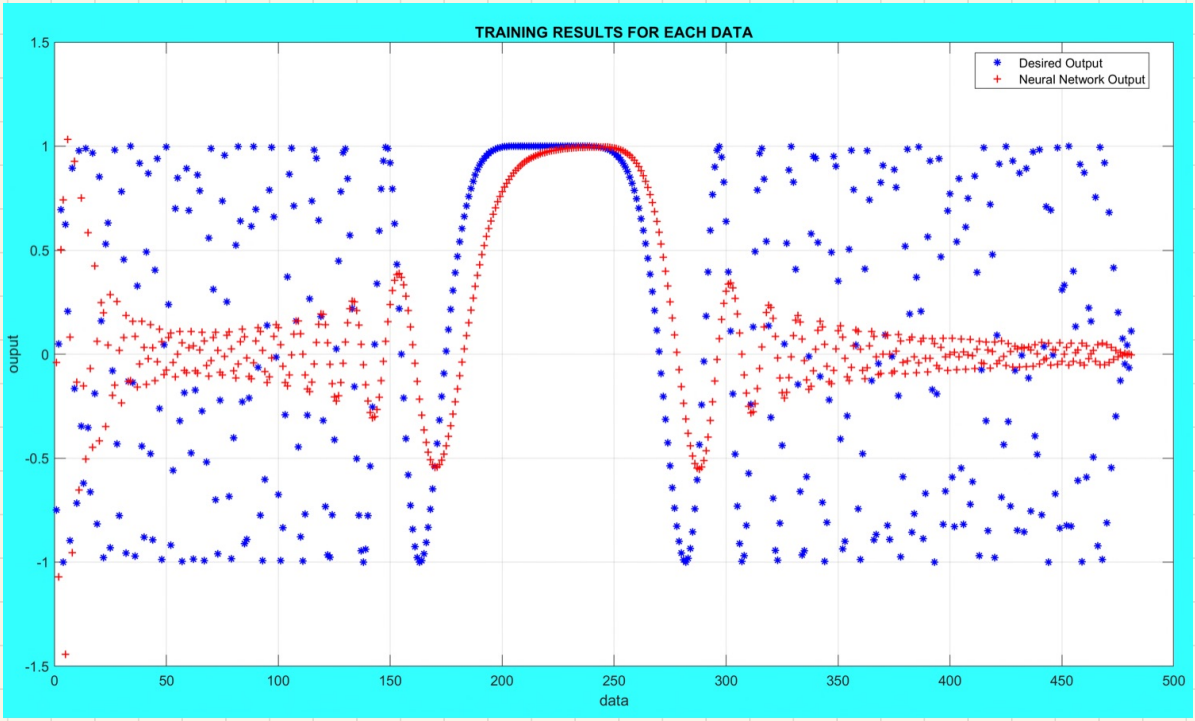
$$f(x) = \frac{1}{1 + e^{-\gamma_k}}$$

Capa 1

$$f(x) = x$$

Capa 2

Luego de realizar este nuevo esquema, se lograron resultados más geniales y más similares al resultado esperado del sistema:



Con este resultado, el sistema quedó entrenado de forma exitosa y las matrices C y W pueden ser guardadas para implementar en futuros algoritmos de control o lo que se requiera. Esta red se considera un buen resultado.

Oportunidades de mejora hay muchas. Una sería agregar filtros digitales como “Box-Jenkins”, para mejorar la adquisición de datos y los procesos del mismo.