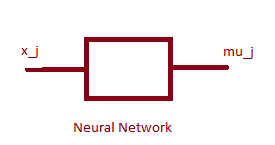
**Laboratorio Calificado N.- 3** Inteligencia Artificial (CC 441)

1. Implementar una Red Neuronal Multicapa (funcion de activacion Logistica) para construir un "membership function" según lo explicado en clase. A continuación se detalla el trabajo de cada participante:





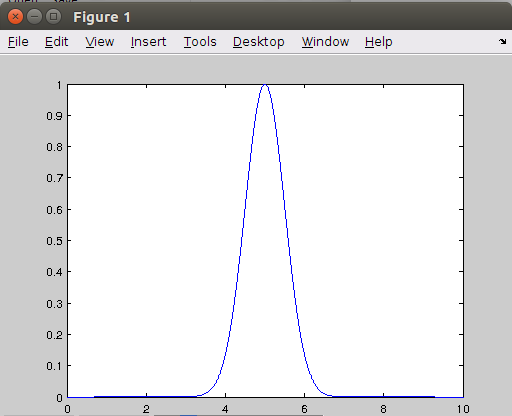
**Solución:**

Se tiene primero los datos de entrada:

train\_inp = (0:0.001:10)';

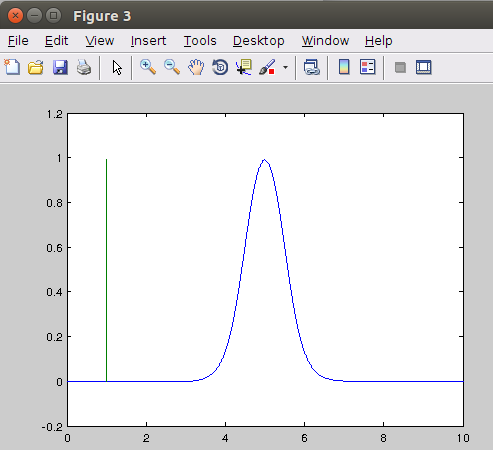
train\_out = gaussmf(train\_inp,[0.5,5]);

Se grafica la función original:

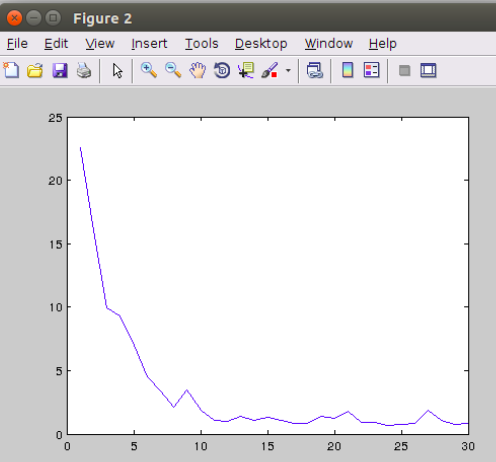


Se aplican 3 neuronas internas con epochs de 1000 y los resultados obtenidos fueron:

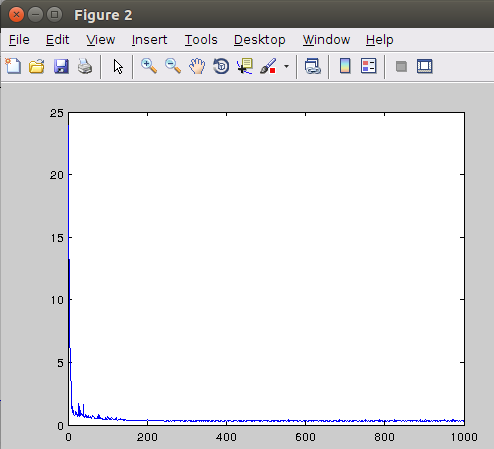
Los cuales generán la gráfica de predicción según su entrenamiento:



como se observa es muy próxima salvo algunos errores, los cuales se representan mendiante:



los cuales convergen a 0 conforme se van haciendo los epochs.



**Código:**

function fuzzyFunction()

hidden\_neurons = 3;

epochs = 1000;

% se carga la información

train\_inp = (0:0.001:10)';

train\_out = gaussmf(train\_inp,[0.5,5]);

figure(1);

plot(train\_inp,train\_out);

% verifica la longitud de entrada y salida

if size(train\_inp,1) ~= size(train\_out,1)

disp('ERROR: data mismatch')

return

end

% calcula la media y el sigma de las salidas (0 y 1)

train\_out = train\_out';

mu\_out = mean(train\_out);

sigma\_out = std(train\_out);

train\_out = train\_out';

% lee numero de patrones

patterns = size(train\_inp,1);

% añade un bias

bias = ones(patterns,1);

train\_inp = [train\_inp bias];

% lee el número de entradas

inputs = size(train\_inp,2);

%add button for early stopping

hstop = uicontrol('Style','PushButton','String','Stop', 'Position', [5 5 70 20],'callback','earlystop = 1;');

earlystop = 0;

%add button for resetting weights

hreset = uicontrol('Style','PushButton','String','Reset Wts', 'Position', get(hstop,'position')+[75 0 0 0],'callback','reset = 1;');

reset = 0;

%add slider to adjust the learning rate

hlr = uicontrol('Style','slider','value',.1,'Min',.01,'Max',1,'SliderStep',[0.01 0.1],'Position', get(hreset,'position')+[75 0 100 0]);

% ---------- escoge pesos -----------------

% pesos random iniciales

weight\_input\_hidden = (randn(inputs,hidden\_neurons) - 0.5)/10;

weight\_hidden\_output = (randn(1,hidden\_neurons) - 0.5)/10;

%-----------------------------------

%--- Learning Starts Here! ---------

%-----------------------------------

for iter = 1:epochs

%g obtiene la taza de aprendizaje

alr = get(hlr,'value');

blr = alr / 10;

% selecciona de manera random el patrón a evaluar

for j = 1:patterns

% selecciona

patnum = round((rand \* patterns) + 0.5);

if patnum > patterns

patnum = patterns;

elseif patnum < 1

patnum = 1;

end

% añade el patrón actual

this\_pat = train\_inp(patnum,:);

act = train\_out(patnum,1);

% calcula el error del patrón actual

hval = (tanh(this\_pat\*weight\_input\_hidden))';

pred = hval'\*weight\_hidden\_output';

error = pred - act;

% ajusta el peso de salida con el error

delta\_HO = error.\*blr .\*hval;

weight\_hidden\_output = weight\_hidden\_output - delta\_HO';

% ajusta el peso de entrada con el error

delta\_IH= alr.\*error.\*weight\_hidden\_output'.\*(1-(hval.^2))\*this\_pat;

weight\_input\_hidden = weight\_input\_hidden - delta\_IH';

end

% plot el error de la red

pred = weight\_hidden\_output\*tanh(train\_inp\*weight\_input\_hidden)';

error = pred' - train\_out;

err(iter) = (sum(error.^2))^0.5;

figure(2);

plot(err)

% reiniciamos pesos si es necesario

if reset

weight\_input\_hidden = (randn(inputs,hidden\_neurons) - 0.5)/10;

weight\_hidden\_output = (randn(1,hidden\_neurons) - 0.5)/10;

fprintf('weights reaset after %d epochs\n',iter);

reset = 0;

end

% detenemos

if earlystop

fprintf('stopped at epoch: %d\n',iter);

break

end

%stop if error is small

if err(iter) < 0.001

fprintf('converged at epoch: %d\n',iter);

break

end

end

%-----FINISHED---------

%display actual,predicted & error

fprintf('state after %d epochs\n',iter);

a = (train\_out\* sigma\_out(:,1)) + mu\_out(:,1);

b = (pred'\* sigma\_out(:,1)) + mu\_out(:,1);

act\_pred\_err = [a b b-a]

figure(3);

plot(train\_inp,pred);

end