UNIVERSIDAD NACIONAL DE INGENIERIA
FACULTAD DE CIENCIAS

PROYECTO:
Laboratorio Calificado N.- 3 Inteligencia Artificial

Alumno: **Moreno Vera Felipe Adrian**

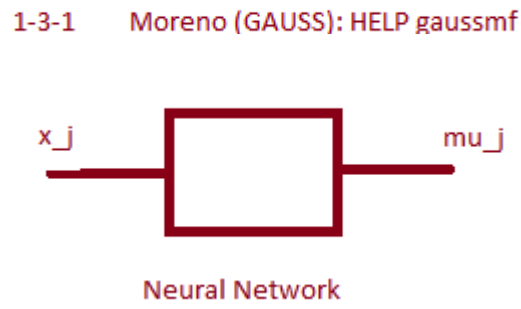
Curso: **Inteligencia Artificial**

Codigo Curso: CC441

2016-I

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:

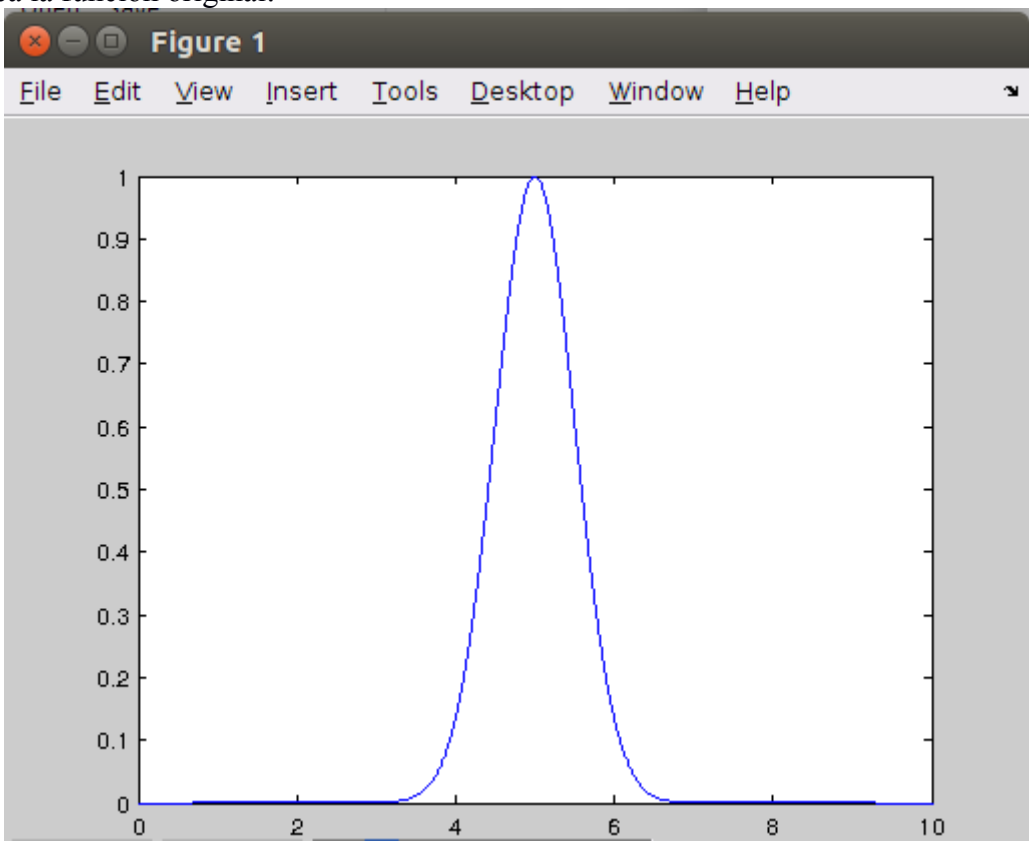


Solución:

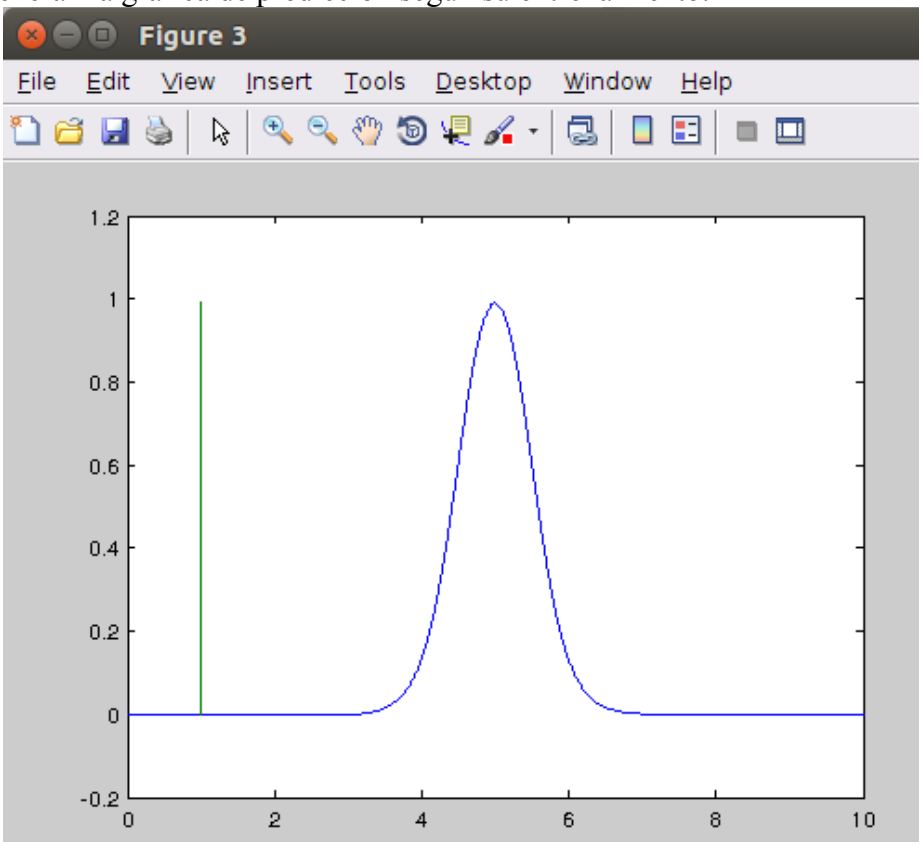
Se tiene primero los datos de entrada:

```
train_inp = (0:0.001:10)';
train_out = gaussmf(train_inp,[0.5,5]);
```

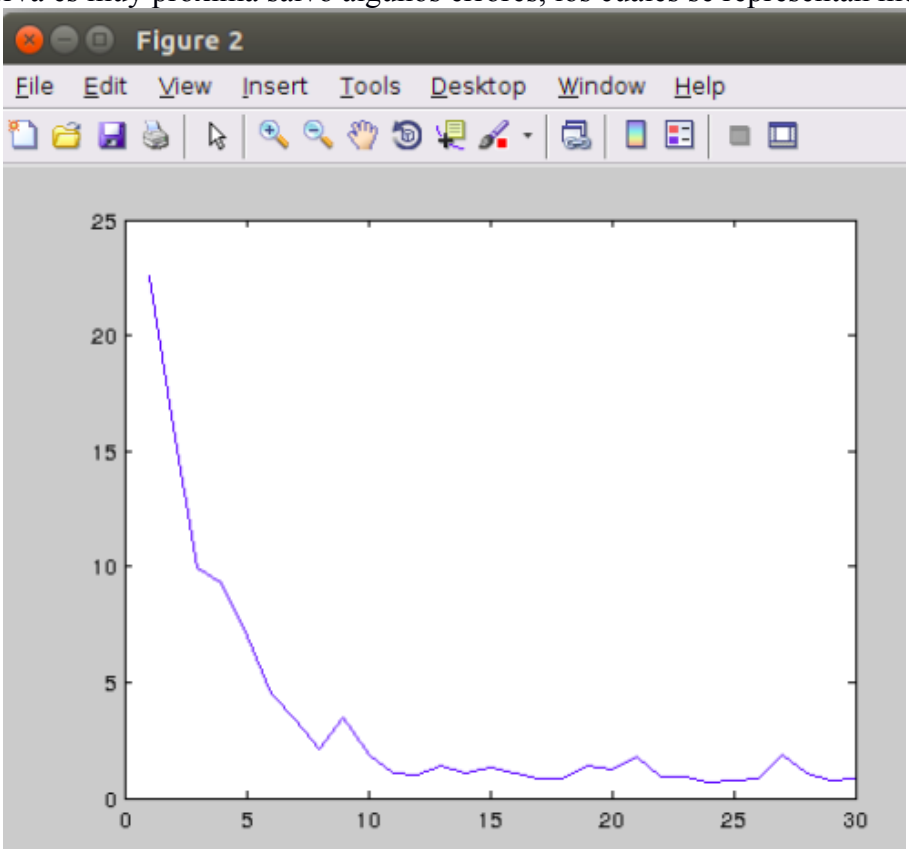
Se grafica la función original:



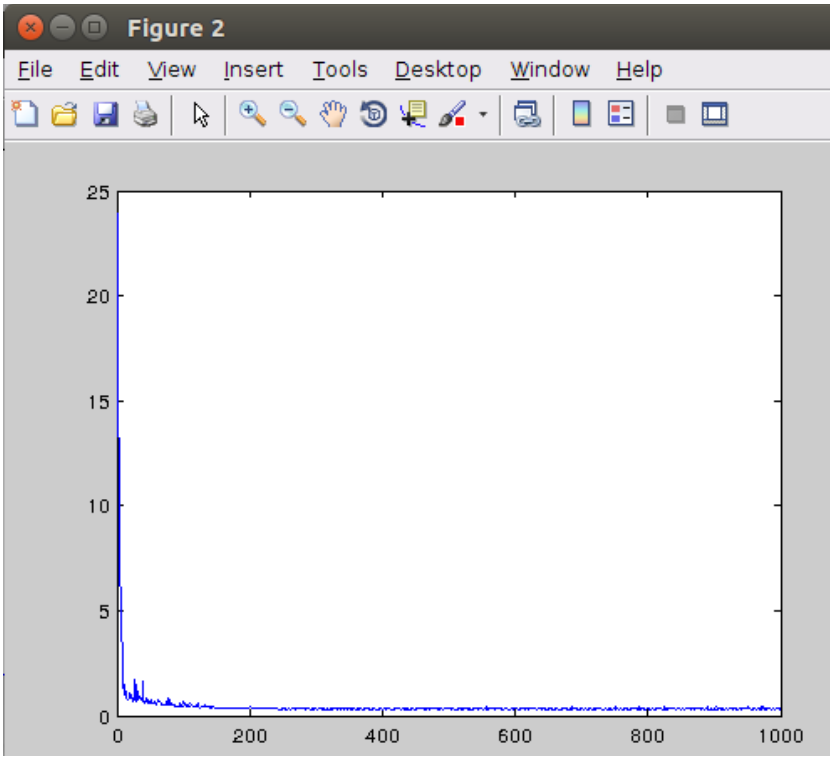
Se aplican 3 neuronas internas con epochs de 1000 y los resultados obtenidos fueron:
Los cuales generán la gráfica de predicción según su entrenamiento:



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
```