

**INSTITUTO POLITÉCNICO NACIONAL**

**ESCUELA SUPERIOR DE CÓMPUTO**

**NEURAL NETWORKS**

**PROF.:** MARCO ANTONIO MORENO ARMENDÁRIZ

**ALUMNO:** ORTEGA VICTORIANO IVAN

**No. DE LISTA:** 29

**GRUPO:** 3CM2

**PERCEPTRÓN MULTICAPA**



Índice

[INTRODUCCIÓN 3](#_Toc500444758)

[MARCO TEÓRICO 3](#_Toc500444759)

[RESULTADOS EXPERIMENTALES 3](#_Toc500444760)

[DISCUSIÓN DE LOS RESULTADOS 3](#_Toc500444761)

[CONCLUSIONES 3](#_Toc500444762)

[Referencias 3](#_Toc500444763)

[ANEXO 4](#_Toc500444764)

[mlp.m 4](#_Toc500444765)

[obtenerF.m 10](#_Toc500444766)

[obtenerConjuntoDeEntrenamiento.m 11](#_Toc500444767)

[obtenerConjuntoDePrueba.m 11](#_Toc500444768)

[obtenerConjuntoDeValidacion.m 11](#_Toc500444769)

# INTRODUCCIÓN

# MARCO TEÓRICO

# RESULTADOS EXPERIMENTALES

# DISCUSIÓN DE LOS RESULTADOS

# CONCLUSIONES

# Referencias

# ANEXO

Para una mejor visualización del código, recomiendo visitar el siguiente repositorio de Github:

<https://github.com/IvanovskyOrtega/Redes-Neuronales/blob/master/MLP/>

## mlp.m

clc

clear

% Leemos el archivo de entradas

input1 = input('Ingresa el nombre del archivo que contiene los valores de entrada "\*.txt" (sin extension): ','s');

nombreArchivo1 = strcat(input1,'.txt');

p = importdata(nombreArchivo1);

% Leemos el archivo de los valores target

input2 = input('Ingresa el nombre del archivo que contiene los valores target "\*.txt" (sin extension): ','s');

nombreArchivo2 = strcat(input2,'.txt');

targets = importdata(nombreArchivo2);

fprintf('\n');

% Se calcula el rango a trabajar

dim\_p = size(p);

lim\_inf = p(1);

lim\_sup = p(dim\_p(1));

incremento = (lim\_sup-lim\_inf)/(dim\_p(1)-1);

% Se muestra el rango y el numero de datos a trabajar

fprintf('Se trabajara el siguiente rango, de acuerdo al archivo:\n');

fprintf('[%f, %f], con un incremento de %f\n',lim\_inf,lim\_sup,incremento);

num\_datos = dim\_p(1);

fprintf('Se trabajara con %d datos, de acuerdo al archivo.\n\n',num\_datos);

% Se solicita la arquitectura del M.L.P.

fprintf('\n');

str\_arq = input('Ingresa la arquitectura del M.L.P.: ','s');

arq\_mlp = str2num(str\_arq);

num\_capas = length(arq\_mlp)-1;

R = arq\_mlp(1);

fprintf('Para las funciones de activacion se tienen las siguientes:\n');

fprintf('1) purelin()\n2) logsig()\n3) tansig()\n\n');

str\_fun = input('Ingresa las funciones de las capas de la red separadas por un espacio: ','s');

fun\_capa = str2num(str\_fun);

disp('La arquitectura del M.L.P. es:');

disp(arq\_mlp);

disp(fun\_capa);

% Se abren archivos para graficacion en modo escritura (Esto es un poco largo...)

num\_archivos\_pesos\_total = 0;

num\_archivos\_bias\_total = 0;

for i=1:num\_capas

for j=1:arq\_mlp(i+1)

for l=1:arq\_mlp(i)

num\_archivos\_pesos\_total = num\_archivos\_pesos\_total +1;

end

end

num\_archivos\_bias\_total = num\_archivos\_bias\_total +1;

end

archivos\_pesos = zeros(num\_archivos\_pesos\_total,1);

archivos\_bias = zeros(num\_archivos\_bias\_total,1);

num\_archivo = 1;

for i=1:num\_capas

path = strcat(pwd,'/Valores-de-Graficacion/Capa-',num2str(i),'/Pesos/');

if ~exist(path, 'dir')

mkdir(path);

end

for j=1:arq\_mlp(i+1)

for k=1:arq\_mlp(i)

archivo\_pesos = strcat(path,'/pesos',num2str(j),'\_',num2str(k),'.txt');

archivos\_pesos(num\_archivo) = fopen(archivo\_pesos,'w');

num\_archivo = num\_archivo +1;

end

end

end

num\_archivo = 1;

for i=1:num\_capas

path = strcat(pwd,'/Valores-de-Graficacion/Capa-',num2str(i),'/bias/');

if ~exist(path, 'dir')

mkdir(path);

end

for j=1:arq\_mlp(i+1)

archivo\_bias = strcat(path,'/bias',num2str(j),'.txt');

archivos\_bias(num\_archivo) = fopen(archivo\_bias,'w');

num\_archivo = num\_archivo +1;

end

end

% Se terminan de abrir los archivos

% Se solicita el valor del factor de aprendizaje

alfa = input('Ingresa el valor del factor de aprendizaje(alfa): ');

% Se solicitan los valores de usuario eit,itmax, itval, numval

itmax = input('Ingresa el numero de iteraciones maximas de la red(itmax): ');

itval = input('¿Cada cuanto se hara una iteracion de validacion? (itval): ');

numval = input('Numero maximo de incrementos consecutivos del error de validacion (numval): ');

eit = input('Ingrese l valor minimo del error en una epoca (eit): ');

fprintf('\n');

% Se solicita la configuracion para dividir en subconjuntos

fprintf('Seleccione una de las siguientes configuraciones a trabajar:\n');

fprintf('1) 80-10-10\n');

fprintf('2) 70-15-15\n');

config = input('Ingrese su seleccion: ');

% Se forman los subconjutnos de acuerdo al tipo de configuracion

switch(config)

case 1

num\_elem\_val = round(num\_datos\*.2); % Numero de elementos del conjunto de validacion

num\_elem\_prueba = num\_elem\_val; % Numero de elementos del conjunto de prueba

num\_elem\_ent = num\_datos - 2\*num\_elem\_val; % Numero de elementos del conjunto de entrenamiento

case 2

num\_elem\_val = round(num\_datos\*.15);

num\_elem\_prueba = num\_elem\_val;

num\_elem\_ent = num\_datos - 2\*num\_elem\_val;

end

disp('Se usaran los siguientes tamanios en los subconjuntos:');

fprintf('Conjunto de entrenamiento: %d elementos\n',num\_elem\_ent);

fprintf('Conjunto de validacion: %d elementos\n',num\_elem\_val);

fprintf('Conjunto de prueba: %d elementos\n',num\_elem\_prueba);

cto\_val = obtenerConjuntoDeValidacion(p,targets,num\_datos,num\_elem\_val);

cto\_prueba = obtenerConjuntoDePrueba(p,targets,num\_datos,num\_elem\_prueba);

cto\_ent = obtenerConjuntoDeEntrenamiento(p,targets,num\_datos,num\_elem\_ent,cto\_val,cto\_prueba);

disp('Conjunto de validacion:');

disp(cto\_val);

disp('Conjunto de prueba:');

disp(cto\_prueba);

disp('Conjunto de entrenamiento:');

disp(cto\_ent);

% Se inicializan la matriz de pesos y el vector bias con valores aleatorios

% entre -1 y 1

num\_archivos\_pesos = 1;

num\_archivos\_bias = 1;

W = cell(num\_capas,1);

b = cell(num\_capas,1);

disp('Valores iniciales de las matrices:');

for i=1:num\_capas

temp\_W = -1 + 2\*rand(arq\_mlp(i+1),arq\_mlp(i));

W{i} = temp\_W;

fprintf('W\_%d = \n',i);

disp(W{i});

temp\_b = -1 + (2)\*rand(arq\_mlp(i+1),1);

b{i} = temp\_b;

fprintf('b\_%d = \n',i);

disp(b{i});

% Se imprimen los valores iniciales en los archivos

for j=1:arq\_mlp(i+1)

for k=1:arq\_mlp(i)

fprintf(archivos\_pesos(num\_archivos\_pesos),'%f\r\n',temp\_W(j,k));

num\_archivos\_pesos = num\_archivos\_pesos +1;

end

end

for j=1:arq\_mlp(i+1)

fprintf(archivos\_bias(num\_archivos\_bias),'%f\r\n',temp\_b(j));

num\_archivos\_bias = num\_archivos\_bias + 1;

end

end

% Se utiliza una cell para guardar las salidas de cada capa

a = cell(num\_capas+1,1);

% Se utiliza una cell para guardas las sensitividades de cada capa y las

% matrices de derivadas.

S = cell(num\_capas,1);

F\_m = cell(num\_capas,1);

X = input('Presiona ENTER para comenzar el aprendizaje...');

% Comienza el aprendizaje

early\_stopping = 0;

Err\_val = 0;

Err\_ap = 0;

valores\_graficacion\_eap = zeros(itmax,1);

valores\_graficacion\_eval = zeros(ceil(itmax/itval),1);

count\_val = 0;

num\_it\_val = 0; % Numero de iteraciones de validacion realizadas

for it=1:itmax

num\_archivos\_pesos = 1;

num\_archivos\_bias = 1;

Eap = 0; % Error de aprendizaje

% Si no es una iteracion de validacion

if(mod(it,itval)~=0)

for dato=1:num\_elem\_ent

a{1} = cto\_ent(dato,1); % Condicion inicial

% Se propaga hacia adelante el elemento del cto. de

% entrenamiento

for k=1:num\_capas

W\_temp = cell2mat(W(k));

a\_temp = cell2mat(a(k));

b\_temp = cell2mat(b(k));

a{k+1} = funcionDeActivacion(W\_temp\*a\_temp+b\_temp,fun\_capa(k));

end

a\_temp = cell2mat(a(num\_capas+1));

ej = cto\_ent(dato,2)-a\_temp;

Eap = Eap+(ej/num\_datos);

% Se calculan las sensitividades y se propagan hacia atras,

% es decir, inicia el backpropagation.

F\_m{num\_capas} = obtenerF(fun\_capa(num\_capas),arq\_mlp(num\_capas+1),a\_temp);

F\_m\_temp = cell2mat(F\_m(num\_capas));

S{num\_capas} = -2\*F\_m\_temp\*(ej);

for m = num\_capas-1:-1:1

W\_temp = cell2mat(W(m+1));

a\_temp = cell2mat(a(m+1));

S\_temp = cell2mat(S(m+1));

F\_m{m} = obtenerF(fun\_capa(m),arq\_mlp(m+1),a\_temp);

F\_m\_temp = cell2mat(F\_m(m));

S{m} = F\_m\_temp\*(W\_temp')\*S\_temp;

end

% Se aplican las reglas de aprendizaje

for k=num\_capas:-1:1

W\_temp = cell2mat(W(k));

b\_temp = cell2mat(b(k));

a\_temp = cell2mat(a(k));

S\_temp = cell2mat(S(k));

W{k} = W\_temp-(alfa\*S\_temp\*(a\_temp'));

b{k} = b\_temp-(alfa\*S\_temp);

W\_temp = cell2mat(W(k));

b\_temp = cell2mat(b(k));

end

end

Err\_ap = Eap;

% Se guarda el valor de graficación de Eap

valores\_graficacion\_eap(it) = Eap;

% Si es una iteracion de validacion

else

E\_val = 0;

num\_it\_val = num\_it\_val + 1;

for dato=1:num\_elem\_val

a{1} = cto\_val(dato,1); % Condicion inicial

% Se propaga hacia adelante el elemento del cto. de

% validacion.

for k=1:num\_capas

W\_temp = cell2mat(W(k));

a\_temp = cell2mat(a(k));

b\_temp = cell2mat(b(k));

a{k+1} = funcionDeActivacion(W\_temp\*a\_temp+b\_temp,fun\_capa(k));

end

a\_temp = cell2mat(a(num\_capas+1));

e\_val = cto\_val(dato,2)-a\_temp;

E\_val = E\_val+(e\_val/num\_elem\_val);

end

% Se guarda el valor para graficacion

valores\_graficacion\_eval(it) = E\_val;

if count\_val == 0

Err\_val = E\_val;

count\_val = count\_val+1;

fprintf('Count val = %d\n',count\_val);

else

if E\_val > Err\_val

Err\_val = E\_val;

count\_val = count\_val+1;

fprintf('Count val = %d\n',count\_val);

if count\_val == numval

early\_stopping = 1;

fprintf('Early stopping en iteracion %d\n',it);

break;

end

else

Err\_val = 0;

count\_val = 0;

fprintf('Count val = %d\n',count\_val);

end

end

end

% Se imprimen los valores de pesos y bias modificados a archivo

num\_archivos\_pesos = 1;

num\_archivos\_bias = 1;

for k=num\_capas:-1:1

W\_temp = cell2mat(W(k));

b\_temp = cell2mat(b(k));

for j=1:arq\_mlp(k+1)

for l=1:arq\_mlp(k)

fprintf(archivos\_pesos(num\_archivos\_pesos),'%f\r\n',W\_temp(j,l));

num\_archivos\_pesos = num\_archivos\_pesos +1;

end

end

for j=1:arq\_mlp(k+1)

fprintf(archivos\_bias(num\_archivos\_bias),'%f\r\n',b\_temp(j));

num\_archivos\_bias = num\_archivos\_bias + 1;

end

end

% Se comprueban las condiciones de finalizacion

if Eap < eit && Eap > 0

fprintf('Aprendizaje exitoso en la iteracion %d\n',it);

break;

end

end

% Se imprimen a archivo los ultimos valores de pesos y bias de cada capa

if early\_stopping == 1

% Se imprimen los valores de pesos y bias modificados a archivo

num\_archivos\_pesos = 1;

num\_archivos\_bias = 1;

for k=num\_capas:-1:1

W\_temp = cell2mat(W(k));

b\_temp = cell2mat(b(k));

for j=1:arq\_mlp(k+1)

for l=1:arq\_mlp(k)

fprintf(archivos\_pesos(num\_archivos\_pesos),'%f\r\n',W\_temp(j,l));

num\_archivos\_pesos = num\_archivos\_pesos +1;

end

end

for j=1:arq\_mlp(k+1)

fprintf(archivos\_bias(num\_archivos\_bias),'%f\r\n',b\_temp(j));

num\_archivos\_bias = num\_archivos\_bias + 1;

end

end

end

% Se cierran los archivos de valores de graficacion de pesos y bias

for i=1:num\_archivos\_pesos\_total

fclose(archivos\_pesos(i));

end

for i=1:num\_archivos\_bias\_total

fclose(archivos\_bias(i));

end

% Se propaga el conjunto de prueba

Ep = 0; % Error de prueba

salida\_red = zeros(num\_elem\_prueba,1);

for i=1:num\_elem\_prueba

a{1} = cto\_prueba(i,1); % Condicion inicial

% Se propaga hacia adelante el elemento del cto. de

% entrenamiento

for k=1:num\_capas

W\_temp = cell2mat(W(k));

a\_temp = cell2mat(a(k));

b\_temp = cell2mat(b(k));

a{k+1} = funcionDeActivacion(W\_temp\*a\_temp+b\_temp,fun\_capa(k));

end

dato\_entrada = cell2mat(a(1));

a\_temp = cell2mat(a(num\_capas+1));

Ep = Ep+(1/num\_elem\_prueba)\*(cto\_prueba(i,2)-a\_temp);

salida\_red(i) = a\_temp;

end

% Se imprimen los valores finales de Eap, Ep y Eval

fprintf('Eap = %f\n',Err\_ap);

fprintf('Eval = %f\n',Err\_val);

fprintf('Ep = %f\n',Ep);

% Graficacion del conjunto de prueba, se muestran los targets contra los

% resultados de la red.

figure

rango = cto\_prueba(:,1);

s1 = scatter(rango,salida\_red,'d');

s1.MarkerFaceColor = [0 0 1];

s1.MarkerEdgeColor = 'b';

grid on

hold on

s2 = scatter(rango,cto\_prueba(:,2));

s2.MarkerFaceColor = [1 0 1];

s2.MarkerEdgeColor = 'm';

title('Target v.s. Salida de la red');

ylabel('f(p)');

xlabel('p');

lgd = legend('Salida de la red','Target','Location','northeastoutside');

title(lgd,'Simbología');

hold off

% Se grafica la evolucion de los errores de aprendizaje y validacion por

% epoca.

figure

rango = 1:1:it;

rango2 = itval:itval:num\_it\_val\*itval;

s1 = scatter(rango2,valores\_graficacion\_eval(itval:itval:num\_it\_val\*itval,1),'d');

s1.MarkerFaceColor = [1 0 0];

s1.MarkerEdgeColor = 'r';

grid on

hold on

s2 = scatter(rango,valores\_graficacion\_eap(1:it,1));

s2.MarkerFaceColor = [0 1 0];

s2.MarkerEdgeColor = 'g';

title('Error de aprendizaje y Error de validacion');

ylabel('Valor del error');

xlabel('Iteracion');

lgd = legend('Eval','Eap','Location','northeastoutside');

title(lgd,'Simbología');

hold off

%Se grafica la evolucion de los pesos

rango = 0:1:it;

for i=1:num\_capas

figure

path = strcat(pwd,'/Valores-de-Graficacion/Capa-',num2str(i),'/Pesos/');

for j=1:arq\_mlp(i+1)

for k=1:arq\_mlp(i)

archivo\_pesos = strcat(path,'/pesos',num2str(j),'\_',num2str(k),'.txt');

simb = strcat('W(',num2str(j),',',num2str(k),')');

evolucion\_pesos = importdata(archivo\_pesos); % Identificador para la grafica

plot(rango,evolucion\_pesos','DisplayName',simb);

hold on

grid on

end

end

titulo = strcat('Evolucion de los pesos de la capa',{' '},num2str(i));

title(titulo);

ylabel('Valor de los pesos');

xlabel('Iteracion');

lgd = legend('show','Location','northeastoutside');

title(lgd,'Simbología');

hold off

end

%Se grafica la evolucion de los bias

rango = 0:1:it;

for i=1:num\_capas

figure

path = strcat(pwd,'/Valores-de-Graficacion/Capa-',num2str(i),'/bias/');

for j=1:arq\_mlp(i+1)

archivo\_bias = strcat(path,'/bias',num2str(j),'.txt');

simb = strcat('b(',num2str(j),')');

evolucion\_bias = importdata(archivo\_bias); % Identificador para la grafica

plot(rango,evolucion\_bias','DisplayName',simb);

hold on

grid on

end

titulo = strcat('Evolucion de los bias de la capa',{' '},num2str(i));

title(titulo);

ylabel('Valor de los bias');

xlabel('Iteracion');

lgd = legend('show','Location','northeastoutside');

title(lgd,'Simbología');

hold off

end

for i=1:num\_capas

path = strcat(pwd,'/Resultados-finales/Capa-',num2str(i),'/');

if ~exist(path, 'dir')

mkdir(path);

end

W\_temp = cell2mat(W(i));

res\_pesos = strcat(path,'/pesos.txt');

dlmwrite(res\_pesos,W\_temp,';');

end

for i=1:num\_capas

path = strcat(pwd,'/Resultados-finales/Capa-',num2str(i),'/');

if ~exist(path, 'dir')

mkdir(path);

end

b\_temp = cell2mat(b(i));

res\_bias = strcat(path,'/bias.txt');

dlmwrite(res\_bias,b\_temp,';');

end

## obtenerF.m

function F = obtenerF(tipo\_funcion,num\_neuronas,a)

switch tipo\_funcion

case 1

F = diag(ones(1,num\_neuronas));

case 2

F = diag(logsig('dn',a,a));

case 3

F = diag(tansig('dn',a,a));

end

end

## obtenerConjuntoDeEntrenamiento.m

function M = obtenerConjuntoDeEntrenamiento(entradas,targets,num\_datos,num\_elem\_ent,cto\_val,cto\_prueba)

M = zeros(num\_elem\_ent,2);

j = 1;

for i=1:num\_datos

if ismember(entradas(i),cto\_val(:,1)) || ismember(entradas(i),cto\_prueba(:,1))

else

M(j,1) = entradas(i);

M(j,2) = targets(i);

j = j+1;

end

if j == num\_elem\_ent+1

break;

end

end

end

## obtenerConjuntoDePrueba.m

function M = obtenerConjuntoDePrueba(entradas,targets,num\_datos,num\_elem)

M = zeros(num\_elem,2);

inc = ceil(num\_datos/(num\_elem+1));

j = 1;

for i=inc:inc:num\_datos

M(j,1) = entradas(i-1);

M(j,2) = targets(i-1);

j = j+1;

if j == num\_elem+1

break;

end

end

if j ~= num\_elem+1

for i=inc+1:inc:num\_datos

M(j,1) = entradas(i-1);

M(j,2) = targets(i-1);

j = j+1;

if j == num\_elem+1

break;

end

end

end

end

## obtenerConjuntoDeValidacion.m

function M = obtenerConjuntoDeValidacion(entradas,targets,num\_datos,num\_elem)

M = zeros(num\_elem,2);

inc = ceil(num\_datos/(num\_elem+1));

j = 1;

for i=inc:inc:num\_datos

M(j,1) = entradas(i);

M(j,2) = targets(i);

j = j+1;

if j == num\_elem+1

break;

end

end

if j ~= num\_elem+1

for i=inc+1:inc:num\_datos

M(j,1) = entradas(i-1);

M(j,2) = targets(i-1);

j = j+1;

if j == num\_elem+1

break;

end

end

end

end