Archivo principal

Inicio:

//Pedimos los valores necesarios para el MLP al usuario

archivoEntradas = entradaUsuario;

archivoTargets = entradaUsuario;

archivoArquitectura = entradaUsuario;

archivoFunciones = entradaUsuario;

epochMax = entradaUsuario;

errorEpoca = entradaUsuario;

alpha = entradaUsuario;

numval = entradaUsuario;

epocaEvaluacion = entradaUsuario;

aux=0

Para cada capa en archivoArquitectura:

//Inicializamos los pesos y bias

//R en la primera capa es el valor de la entrada y posteriormente es el número de neuronas de la capa anterior

Pesos(capa) = NuevaMatriz( valor x R );

bias(capa) = NuevaMatriz(valor x 1);

//Separamos el DataSet en tres conjuntos

tamano = archivoEntrada

entrenamiento = tamano\*.8

Nentrenamiento=tamano/entrenamiento

validacion = tamano\*.1

Nvalidacion=tamano/validacion

prueba = tamano \* .1

Nprueba = tamano/prueba

Para i=1 :Nentrenamiento hasta tamano

conjuntoEntrenamiento=[conjuntoEntrenamiento archivoEntrada(i)]

targetEntrenamiento=[targetEntrenamiento archivoTargets(i)]

End

Para i=1 :Nvalidacion hasta tamano

conjuntovalidacion=[conjuntoValidacion archivoEntrada(i)]

targetValidacion = [targetValidacion archivoTargets(i)]

End

Para i=1 :Nprueba hasta tamano

conjuntoprueba=[conjuntoprueba archivoEntrada(i)]

targetPrueba = [tagertPrueba archivoTargets(i)]

end

/\*\*\*\*\* Esto ya se borraria

[conjuntoEntrenamiento, targetEntrenamiento] = separaDatos(archivoEntradas, archivoTargets);

[conjuntoValidacion, targetEntrenamiento] = separaDatos(archivoEntradas, archivoTargets);

[conjuntoPrueba, targetPrueba] = separaDatos(archivoEntradas, archivoTargets);

\*\*\*\*\*\*\*\*\*/

//iniciamos con el aprendizaje

contador = 1;

Mientras contador<epochMax:

Si ( contador%epocaEvalucacion==0 ):

/////Validacion

Naquitectura = size(archivoArquitectura)-1

//Propagamos hacia adelante los datos sin modificar pesos

P[1] = dataSet

Para i=1 Narquitectura

P[i+1] = activacion(archvioFunciones(i),wp[i]+b)

end

Salida = p[i+1]

//verificamos el error y aseguramos que no supere los errores consecutivos de lo contrario terminamos el programa

Errorv[aux]= Target- salida;

If(Errorv[aux] > Errorv[aux-1])

Aux++

End

Else

Aux=0

If(aux>errorval)

Terminarprograma();

end

Sino:

error =0;

//Propagamos hacia adelante los datos del conjunto de entrenamiento

Para cada dato en conjuntoEntrenamiento:

Para cada capa en 1:|archivoArquitectura|

if funcion(capa)==1:

//salidas{capa}=purelin(pesos(capa)\*entradas{capa} + bias{capa})

elseif funcion(capa)==2:

//salidas{capa}=logsig(pesos(capa)\*entradas{capa} + bias{capa})

else

//salidas{capa}=tansig(pesos(capa)\*entradas{capa} + bias{capa})

error+=| (target(dato)-salidas(ultimo) |;

///////Backpropation()

M = Narquitectura

S(M) = -2F(M)(nM)(t-a)

Para M = Narquitectura-1 hasta 1

S(m)=F(m)(nm)(W(m+1))\*S(m+1)

MatrizF()//aun no determinada

End

Pesos(k+1) = w(k) – alfa\*(S(m))(am-1)

B(k+1) = bm(k)-alfa\*S(m)

guardarDatos(pesos,bias);

Si error< errorEpoca:

termina;

contador+=1;

GraficarPesos();

GraficarBias();

GraficarPrueba();

FIN