ANEXO I

CÓDIGO DE MATLAB: FUNCIONES Y SCRIPTS

Contenido

1. INTRODUCCIÓN

En este anexo se explicará el código de Matlab que ha permitido realizar todo el trabajo anterior. Este código se fundamenta en un script principal que hace llamadas a funciones auxiliares con las cuales logrará simular la red neuronal correctamente

Antes de exponer el código de la función principal, se mostrará que hace cada función auxiliar, de tal modo que sea mucho más fácil exponer con claridad el script principal.

2. FUNCIONES Y SCRIPTS PARA MANIPULACIÓN DE LOS DATOS

2.1 Función "media_potencia"

```
% Transforma los datos cuarto-horarios en datos horarios.
% Hace la media de los 4 datos quarto horarios.
% El nuevo dato, será el dato horario utilizado.

function datos = media_potencia(datos_mes)

k=1;
n1=length(datos_mes);
for i=1:4:n1


datos(k,2)=(datos_mes(i,2)+datos_mes(i+1,2)+datos_mes(i+2,2)+datos_mes(i+3,
2))/4;
    datos(k,1)=k;
    k=k+1;

end


end
```

Con esta función se transforman los datos cuarto-horarios de temperatura en datos horarios. Para ello se cogen los 4 datos cuartos horarios y se les aplica la media, el nuevo dato, será el utilizado como potencia consumida en esa hora.

2.2 Función “spline_cubico”

```
%%%%%%%%%%%%Función spline_cubico

% Devuelve el spline cubico suavizado de los datos
% de cada mes.
% Los argumentos de entrada son el vector de potencias
% del mes a tratar y un parámetro de suavizado que determinará
% el peso relativo que queramos dar  a la aproximación.
% p puede tener valores entre 0 y 1



function spmes = spline_cubico(datos_mes,p)


    xmes = datos_mes(:,1);
    ymes = datos_mes(:,2);

    spmes = csaps(xmes,ymes,p,xmes);
```

Esta función se encarga del suavizado de la curva de potencia aplicando la interpolación de splines cúbicos. Como entrada, recibirá los datos de potencia y el parámetro p, que como ya se ha explicado es un indicador del grado de suavizado que se quiere.

El comando ‘spmes = csaps(xmes,ymes,p,xmes)’ devuelve un spline cúbico suavizado ‘para los datos ‘xmes, ymes’.

En realidad debido al cuarto argumento de la función los únicos datos que modifica son los pertenecientes a “ymes” que en nuestro caso, son los datos de potencia.

2.3 Función “extremos_temperatura”

```
%%%%%%%%%% Funcion extremos_temperatura
% Esta funcion toma la matriz de datos de la temperatura horaria
% y separa en un vector la temperatura máxima de cada día
% repetida 24 veces (24 horas al día) y en otro vector lo mismo con
% la temperatura minima.



function [Temp_Max, Temp_Min] = extremos_temperatura(Matriz_Temp)

[Filas, Columnas] = size(Matriz_Temp);

n=0;

for i=1:Filas
    for j=1:Columnas
        Temp_Max(j+Columnas*n) = max(Matriz_Temp(i,:));
        Temp_Min(j+Columnas*n) = min(Matriz_Temp(i,:));
    end
    n=n+1;
end
```

Esta función deberá identificar las temperaturas máximas y mínimas de cada día y crear dos vectores ocupando las 24 horas de cada día con las temperaturas máximas para un vector y con las temperaturas mínimas para el otro.

Con los bucles for, se van recorriendo las columnas y filas que forman la matriz de temperaturas, cada vez que se encuentre una nueva temperatura máxima o mínima, esta se actualizará al nuevo valor para las 24 horas del día.

2.4 Función " findes"

```
%%%%%%%% Función findes

% Crea un vector de una fila y el mismo número de columnas que el vector
% de entrada, en el cual se escribirá un cero (0) si el día es sábado o
% domingo, y un uno (1) si no lo es.
% La entrada o input 1 de la función será el vector de potencias ya con
% sólo los datos horarios, el input 2 será el día de la semana en el que
% comienza el mes ('l' para lunes, 'm' para martes, 'x' para miércoles,
% etc.).
% Además esta función saca por pantalla (plotea) las gráficas de los días
% según sean laborables o no, así comprobaremos si los cambios han sido o
% no efectivos.

function festivo = findes(datos, day)


if (day=='L')||(day=='l')

sab=5;

else if (day=='M')||(day=='m')

     sab=4;

     else if (day=='X')||(day=='x')

          sab=3;

          else if (day=='J')||(day=='j')

               sab=2;

               else if (day=='V')||(day=='v')

                    sab=1;

                    else if (day=='S')||(day=='s')

                         sab=0;

                         else if (day=='D')||(day=='d')

                             sab=6;

                             else

                             fprintf('\nError. Input 2 no es correcto.\n');
                             return;

                             end
                         end
                    end
               end
          end
     end
end
```

```
n2=length(datos);

festivo=ones(1,n2);


correctror=24*sab;   %cambio de días a horas para llegar al sábado

if sab==6
    horas=0;
else
    horas=sab*24;   %cambio de días a horas para llegar al sábado
end

for i=(horas+1):24*7:n2

    if sab==6           %es domingo

        for m=0:23

            festivo(i+m)=0;                 %asignamos los ceros al domingo

            if (i+m+correctror)<=n2         %comprobamos que no nos
                                            %salimos del array y asignamos
                festivo(i+m+correctror)=0;  %un 0 al sábado siguiente

            else end
        end
    else                    %resto de días

         for m=0:23

             festivo(i+m)=0;                %asignamos los ceros al sábado

             if (i+m+24)<=n2                %Comprobamos que no nos
                                            %salimos del array y asignamos
                 festivo(i+m+24)=0;         %un cero al domingo

             else end

         end
    end

end


%%%% Plotear resultados

m=0;
h=0;

for i=1:n2

    if festivo(i)==1

        m=m+1;
        dia_lab(m)=datos(i);

    else
```

```
        h=h+1;
        dia_fes(h)=datos(i);

    end

end
```

Con esta función se crea el vector “festivos” que contendrá la información que indicará a la a red neuronal si un día es laborable o no. Para ello se utiliza la siguiente codificación

1 para los días laborables

0 para los días no laborables

Las entradas de esta función serán; un vector de datos de potencia (ya sea de la semana anterior, del mes de entrenamiento o de la semana a estimar) y la letra inicial del día por el que empieza esa semana o mes (miércoles ‘x’), A partir de esta información la función rellenara con 1 los días laborables y con 0 los fines de semana.

2.5 Función " fiesta"

```
%%%%%%%% Función fiesta

% Modifica el vector 'festivo' según sea la entrada por el teclado,
% cambiando un día laborable por un día festivo. Tiene dos inputs, uno será
% el vector 'festivo' con la información de qué días son festivos o no, y
% el otro será un vector o un único número con la información del o de los
% días que queremos cambiar a festivos. Este último input es opcional, si
% no se introduce se pedirá por pantalla.

function festivo_nuevo=fiesta(festivo, day)

n=length(festivo);

festivo_nuevo=festivo;

if nargin==1

   while(1)

   disp(' ');
   d=input('Introduzca día festivo (1...31) "q" para salir: ');

   if d=='q'

      return;

   else if d>(n/24)||d<=0

       fprintf('\nError. Introduzca un día que se encuentre en el mes.\n');

       else

       indice=(d-1)*24;    %pasa de días a horas y corrige el sesgo

           for i=1:24

               festivo_nuevo(i+indice)=0;

           end



       fprintf('\n¿Algún cambio más? "s" o "n"\n');
       g=input('s/n: ');

           if g~='s'

              return

           else end

       end


    end
```

```
    end
else
    if day==0
       return;
    else if (max(day)>n/24)||(min(day)<0)
       fprintf('\nError. Input 2 es erróneo.\n');

    else
       n2=length(day);
       for k=1:n2
          indice=(day(k)-1)*24;     %pasa de días a horas y corrige el sesgo
          for i=1:24
              festivo_nuevo(i+indice)=0;
          end
       end
       end
    end

end
```

Esta función recibe como entrada el vector "festivos" creado anteriormente por la función "findes", como segundo argumento, se le introduce el día que queremos que asigne como no laboral. Es decir si introducimos un 14, asignará como no laboral al día 14 del vector festivos. Si se le introduce un 0, el vector "festivos" permanece igual.

Si en el script principal no introducimos ningún número como segundo argumento, se dará la opción de introducir un día no laborable por pantalla.

2.6 Función tiempo_bin

```
%%%%%%%% Función tiempo_bin

% Crea los inputs de la hora para un determinado vector de datos de carga
% horaria, empezando por las 00:00h hasta las 23:00. Por tanto, nuestro
% primer elemento corresponde a las 00:00, el segundo a las 01:00... el
% elemento 24 corresponde a las 23:00... y así hasta el final.
% Con esta matriz concatenada con la de la variable del día y demás inputs
% formaremos nuestra matriz de datos.

function tiempo = tiempo_bin(datos)

n2=length(datos);

for i=1:n2

    hora=rem(i,24); %resto de la hora que nos da conjuntos de 24 horas
    if hora==0
        hora_corregida=23; %% LA 24 HORA DEL DIA EMPIEZA A LAS 23
    else
        hora_corregida=hora-1; %% LA PRIMERA HORA DEL DIA EMPIEZA A LAS 0,
LA 2 A LA 1....
    end

    hora_bin=dec2bin(hora_corregida,5);

    for n=1:5               %% 5 variables binarias para el tiempo
        tiempo(n,i)=str2num(hora_bin(n));

    end
end
```

Esta función se encarga de crear 5 vectores que indicarán en binario la hora de los datos en usos.

Los datos de potencia horarios están ordenados en un vector por lo que se puede saber exactamente a la hora a la que pertenecen. Será necesario convertir esa hora a una variable binaria, por lo que harán falta 5 variables, es decir, 5 vectores.

2.7 Función “escalar”

```
%%%%%%%% Función escalar

% Escala los valores de los vectores de cargas horarias y de temperaturas
% a valores entre 0 y 1. Tiene tres inputs, el vector de datos
% correspondiente y los límites superior e inferior.
% Tiene una única salida, que es el vector de datos ya escalado.


function [datos_escalados]=escalar(datos, D_max, D_min)


datos_escalados=(datos-D_min)/(D_max-D_min);
```

Con esta función se pretende convertir las variables exógenas con valores superiores a 1, como son la potencia y la temperatura, en variables con valores acotados entre 0 y 1. Evitando de esta manera la saturación de las funciones de activación.

Para ello, esta función utiliza los valores máximos y mínimos de temperatura y potencia, junto con el dato de potencia o temperatura que será escalado.

2.8 Función "ent_sal_red"

```
%%%%%%%% Función ent_sal_red

% Crea la matriz de entradas para nuestra red neuronal (cargas, día, hora,
% temperatura).
% Constará de 5+1+5+2 filas y n2 columnas, donde n2 será la longitud del
% vector de cargas target del conjunto de entrenamiento que será el segundo
% output.
% La primera fila tendrá la carga horaria de una semana anterior a esa
% misma hora, la segunda tendrá la carga horaria del día anterior a esa
% misma hora y las tres siguientes tendrán la carga de la tercera, segunda
% y anterior hora respectivamente. Nuestro training set será de un mes,
% luego la función tendrá dos inputs, dos matrices, una con la semana
% anterior al mes de entrenamiento junto con las cargas horarias más el
% vector de fin de semana, más la matriz de horas en binario mas los
% vectores de los extremos de temperaturas, y otra con el mes de
% entrenamiento mas el resto mencionado antes
%
% Ejemplo:
%
%  inputs:    [carga;festivo;hora;extremos_temperatura]
%                1xn    1xn    5xn     2xn
%
%  Donde n es el número de horas del período de entrada.
%
%  salida 1:  [cargas; festivo; hora; extremos_temper](en una misma matriz)
%                5*n2    1*n2    5*n2    2*n2
%
%  Donde n2 es el número de horas del conjunto de entrenamiento.
%
%  En total, la salida 1 constará de n2 columnas y 13 filas
%
%  salida 2: [targets]
%               1*n2
%  Vector de targets para la red


function [entrada, targets] = ent_sal_red(datos_semana, datos_mes)

n1=length(datos_semana);
n2=length(datos_mes);

matriz=[datos_semana datos_mes];


% La primera semana se usará para poder arrancar el bucle.
% Por tanto, el bucle empezará desde la
% primera semana del mes de entrenamiento, o sea, 24*7+1=169

k=0;

for i=169:(n1+n2) %%Solo coge los datos del mes de entrenamiento

    k=k+1;

    entrada(1,k) = matriz( 1, i-24*7);
    entrada(2,k) = matriz( 1, i-24);
    entrada(3,k) = matriz( 1, i-3);
```

```
    entrada(4,k) = matriz( 1, i-2);
    entrada(5,k) = matriz( 1, i-1);
    entrada(6,k) = matriz( 2, i);
    entrada(7,k) = matriz( 3, i);
    entrada(8,k) = matriz( 4, i);
    entrada(9,k) = matriz( 5, i);
    entrada(10,k) = matriz( 6, i);
    entrada(11,k) = matriz( 7, i);
    entrada(12,k) = matriz( 8, i);
    entrada(13,k) = matriz( 9, i);

    targets(k) = matriz(1,i);

end
```

2.9 Función "ent_sal_red_variable"

```
function [entrada, targets] = ent_sal_red_variable(datos_semana, datos_mes)

n1=length(datos_semana);
n2=length(datos_mes);

matriz=[datos_semana datos_mes];


% La primera semana se usará para poder arrancar el bucle.
% Por tanto, el bucle empezará desde la
% primera semana del mes de entrenamiento, o sea, 24*7+1=169

k=0;

for i=169:(n1+n2) %%Solo coge los datos del mes de entrenamiento

    k=k+1;


    entrada(1,k) = matriz( 1, i-3);
    entrada(2,k) = matriz( 1, i-2);
    entrada(3,k) = matriz( 1, i-1);
    entrada(4,k) = matriz( 2, i);
    entrada(5,k) = matriz( 3, i);
    entrada(6,k) = matriz( 4, i);
    entrada(7,k) = matriz( 5, i);
    entrada(8,k) = matriz( 6, i);
    entrada(9,k) = matriz( 7, i);
    entrada(10,k) = matriz( 8, i);
    entrada(11,k) = matriz( 9, i);

    targets(k) = matriz(1,i);

end
```

2.10 Función "ent_sal_red_variable1"

```
function [entrada, targets] = ent_sal_red_variable1(datos_semana,
datos_mes)

n1=length(datos_semana);
n2=length(datos_mes);

matriz=[datos_semana datos_mes];


% La primera semana se usará para poder arrancar el bucle.
% Por tanto, el bucle empezará desde la
% primera semana del mes de entrenamiento, o sea, 24*7+1=169

k=0;

for i=169:(n1+n2) %%Solo coge los datos del mes de entrenamiento

    k=k+1;

    entrada(1,k) = matriz( 1, i-24);
    entrada(2,k) = matriz( 1, i-3);
    entrada(3,k) = matriz( 1, i-2);
    entrada(4,k) = matriz( 1, i-1);
    entrada(5,k) = matriz( 2, i);
    entrada(6,k) = matriz( 3, i);
    entrada(7,k) = matriz( 4, i);
    entrada(8,k) = matriz( 5, i);
    entrada(9,k) = matriz( 6, i);
    entrada(10,k) = matriz( 7, i);
    entrada(11,k) = matriz( 8, i);
    entrada(12,k) = matriz( 9, i);

    targets(k) = matriz(1,i);

end
```

2.11 Función "ent_sal_red_variable2"

```
function [entrada, targets] = ent_sal_red_variable2(datos_semana,
datos_mes)

n1=length(datos_semana);
n2=length(datos_mes);

matriz=[datos_semana datos_mes];


% La primera semana se usará para poder arrancar el bucle.
% Por tanto, el bucle empezará desde la
% primera semana del mes de entrenamiento, o sea, 24*7+1=169

k=0;

for i=169:(n1+n2) %%Solo coge los datos del mes de entrenamiento

    k=k+1;



    entrada(1,k) = matriz( 1, i-1);
    entrada(2,k) = matriz( 2, i);
    entrada(3,k) = matriz( 3, i);
    entrada(4,k) = matriz( 4, i);
    entrada(5,k) = matriz( 5, i);
    entrada(6,k) = matriz( 6, i);
    entrada(7,k) = matriz( 7, i);
    entrada(8,k) = matriz( 8, i);
    entrada(9,k) = matriz( 9, i);

    targets(k) = matriz(1,i);

end
```

Estas funciones crean las matrices necesarias para el entrenamiento de la red neuronal. Como se puede apreciar hay 4 funciones diferentes. La diferencia entra estas funciones radica en el número de entradas que se usan para poder entrenar la red neuronal y que posteriormente, esta sea capaz de predecir.

Como entrada a la función se le pasará una matriz que contienen todos los datos necesarios (potencia, temperaturas, festivos …) tanto de la semana anterior como del mes de entrenamiento. A partir de estos datos la función creara las matrices de entrada y salida necesarias para el entrenamiento de la red neuronal.

El bucle se inicializa en 169, ya que hay que recordar que al juntar la matriz de la semana anterior y la matiz con los datos del mes de entrenamiento las primeras 168 columnas de datos no corresponden al mes de entrenamiento, sino a la semana previa.

2.12 Función "division_datos"

```
%%%%%%% Funcion division_datos

% Esta funcion realiza la division de los datos de entrada de la potencia
% consumida en el mes a dividir (mesh) segun el porcentaje en que el
% usuario quiera dividir cada conjunto de datos (train, val y set).
% Es valido para cualquier relacion de porcentajes y para cualquier numero
% de dias que tenga el mes, puesto que la division la realiza de una
% manera aproximada ajustandose lo mas posible a los valores deseados de
% porcentajes.
% La funcion realiza tambien el desplazamiento de indices de cada
% subconjunto para el estudio de su robustez.
% Al final muestra por pantalla como ha dividido los subconjuntos, así
% como si se ha llevado a cambio el desplazamiento o no.
% La salida de la función son los conjuntos de valores que necesitará la
% red en el entrenamiento y que tenga en cuenta esta division de datos.


function [trainP,valP,testP,trainInd,valInd,testInd,desplaz] =
division_datos(mesh,trainRatio,valRatio,testRatio)

horas=0;
%%%%creo un vector para las horas
for i=1:length(mesh)
    vector_horas(1,i)=horas;
    horas=horas+1;
end


%%%%%creo los vectores índice
v=1;
k=1;
l=1;
cont=0;


ntrain=trainRatio*10;
nval=valRatio*10;
ntest=testRatio*10;

%el factor multiplicador depende del porcentaje dedicado a ntrain

for i=1:round(trainRatio*length(vector_horas));

    trainInd(i) = v;

    if rem(i,ntrain)==0
        cont=cont+1;
        v=v+1;
        for j=1:(nval)
            valInd(k) = v;
            v=v+1;
            k=k+1;
        end
```

```
        for j=1:(ntest)
            testInd(l) = v;
            v=v+1;
            l=l+1;
        end
    else
        v=v+1;
    end
end


%ajuste de tamaño por las divisiones no enteras
if (length(trainInd)+length(valInd)+length(testInd))<length(vector_horas)
    for
i=((length(trainInd)+length(valInd)+length(testInd))+1):length(vector_horas
)
        if (length(trainInd)-ntrain*cont)<ntrain
            trainInd(length(trainInd)+1) = v;
            v=v+1;
        else if (length(valInd)-nval*cont)<nval
            valInd(length(valInd)+1) = v;
            v=v+1;
        else
            testInd(length(testInd)+1) = v;
            v=v+1;
            end
        end
    end
end




%%%Avanzar en línea temporal


i=1;
j=1;
k=1;

for n=1:length(mesh)
    if trainInd(i)== n
        vec_Ind(n)=1;
        if i<length(trainInd)
        i=i+1;
        end
    else if  valInd(j)== n
            vec_Ind(n)=2;
            if j<length(valInd)
            j=j+1;
            end
    else if testInd(k)== n
            vec_Ind(n)=3;
            if k<length(testInd)
            k=k+1;
            end
        end

        end

    end
```

```
end



%%despalazar


desplaz=input('\nIntroduzca desplazamiento (0~9): ');
if isempty(desplaz)
    desplaz = '1';
end


switch desplaz
    case 0, vec_Ind=vec_Ind;
    case 1, vec_Ind=[3 vec_Ind];
    case 2, vec_Ind=[3 3 vec_Ind];
    case 3, vec_Ind=[2 3 3 vec_Ind];
    case 4, vec_Ind=[2 2 3 3 vec_Ind];
    case 5, vec_Ind=[1 2 2 3 3 vec_Ind];
    case 6, vec_Ind=[1 1 2 2 3 3 vec_Ind];
    case 7, vec_Ind=[1 1 1 2 2 3 3 vec_Ind];
    case 8, vec_Ind=[1 1 1 1 2 2 3 3 vec_Ind];
    case 9, vec_Ind=[1 1 1 1 1 2 2 3 3 vec_Ind];
    otherwise vec_Ind=vec_Ind;
end

for i=1:(length(vec_Ind)-desplaz)
     vec_aux(i) = vec_Ind(i);
end
  vec_Ind = vec_aux;
  clear vec_aux

%%%%

%%%descodifico
i=1;
j=1;
k=1;

for n=1:length(vec_Ind)
    if vec_Ind(n)==1
        trainInd(i)=n;
        i=i+1;
    else if vec_Ind(n)==2
        valInd(j)=n;
        j=j+1;
    else if vec_Ind(n)==3
        testInd(k)=n;
        k=k+1;
        end
        end
    end
end
%%%



%calculo
[trainP,valP,testP] = divideind(vector_horas,trainInd,valInd,testInd);
```

```
%dibujo
figure;
plot(vector_horas,mesh,'k');
hold on
for n=1:length(trainP)
    plot(trainP(n),mesh(trainInd(n)),'rx');
end

for n=1:length(valP)
    plot(valP(n),mesh(valInd(n)),'gx');
end

for n=1:length(testP)
    plot(testP(n),mesh(testInd(n)),'bx');
end

title('Uso de divideind:  Train = rojo    Val = verde   Test = azul');
xlabel('Tiempo(horas)');
ylabel('Energia(kW)');


%limpio las variables
clear i
clear j
clear n
clear v
clear k
clear l
clear cont
clear ntrain
clear nval
clear ntest
clear vector_horas
clear horas
```

Esta función es la encargada de dividir el conjunto de datos entre los tres grupos que se van a utilizar durante el entrenamiento, training set, validation set y testing set.

Posee cuatro entradas, una de las cuales es el vector de datos, que servirá para medir la longitud del mismo. Las otras tres recogen los valores introducidos por el usuario del porcentaje dedicado a cada uno de los subgrupos.

La función pedirá por pantalla que se introduzca el valor de desplazamiento (el cual puede tener un valor de entre 0 y 9) para escoger entre uno de los 10 conjuntos de entrenamiento.

3 FUNCIONES DE LA RED NEURONAL

3.1 INTRODUCCIÓN

Como se ha comentado durante este proyecto, Matlab tiene programada una *toolbox* destinada a redes neuronales. Las funciones que dicha *toolbox* tiene implementadas son las que hemos utilizado para la creación, entrenamiento y simulación de las redes neuronales probadas. En este apartado nombraremos

3.2 Función "newff"

Esta función es la encargada de crear la red neuronal. La doble "f" se refiere a "feed-forward", lo que significa que esta red se la alimenta desde la capa de entrada y la información va avanzando hasta el final pasando por cada capa sucesiva. Este tipo de red también es del tipo perceptron multicapa "backpropagation", lo que refiere al método de aprendizaje, de delante hacia atrás.

La sintaxis de esta función es sencilla, igualando un término (nombre de la red) a la función. Los argumentos de esta función pueden ser bastantes según queramos modificar las opciones que tiene Matlab por defecto para esta función. Lo mínimo que hay que indicar en la función son el número de neuronas que queremos que tenga en la capa o capas ocultas, el vector o matriz de entradas de entrenamiento y el vector o matriz de salidas. Como la función de activación que tiene Matlab por defecto es la tangente hiperbólica, y nosotros queremos usar la función sigmoide, tenemos que incluir un cuarto argumento en la función para satisfacer nuestra demanda. Un ejemplo de creación de una red de este tipo puede ser:

$$red = newff(INPUTS,\ TARGETS,\ neuronas,\ \{'logsig'\});$$

Donde INPUTS es la matriz con los datos de entrada; TARGETS es la matriz de datos de salida; neuronas es un vector con el nº de neuronas en cada capa oculta; y el último argumento es el nombre con el que Matlab asigna la función sigmoide a las neuronas de la capa oculta. La salida de esta función es la variable objeto red neuronal.

3.3 Función "train"

Esta función, como dice su nombre, es la encargada de entrenar la red neuronal. El método de entrenamiento se fija cuando se crea la función. Como defecto Matlab entrena a este tipo de redes con el algoritmo Levenberg-Marquardt que ya se ha comentado en la memoria.

Lo único que se ha modificado en este aspecto son los porcentajes o porciones del training set que se van a dedicar al entrenamiento, el test y la validación. Como se ha comentado a lo largo de la memoria, dichos porcentajes los hemos fijado en 60, 20 y 20% respectivamente. Esto se ha hecho con tres sentencias simples posteriores a la creación de la red y anteriores a su entrenamiento

Por tanto, el uso por nuestra parte de esta función "*train*" es la más sencilla, igualando la red a la función teniendo ésta como argumentos la propia red, la matriz con los datos de entrada y la matriz con los datos de salida:

$$red = train(red,\ INPUTS,\ TARGETS);$$

Donde "*red*" es el nombre de la red creada anteriormente y las matrices son las comentadas también anteriormente, aunque pueden ser diferentes ya que en la creación, dichas matrices sólo sirven para determinar el número de neuronas en las capas de entrada y salida, aunque en todos los ejemplos de Matlab, dichas matrices son las mismas pues es lo más sencillo si ya están creadas.

3.4 Función "Init"

El cometido de esta función es el de inicializar (poner a cero) todos los pesos sinópticos de la red neuronal, como al crearse la red neuronal. Se ha tenido que recurrir a esta función porque las estimaciones en el estudio principal resultaban de la media de 12 estimaciones. La red es entrenada cada vez antes de cada estimación, pero para que los resultados sean diferentes en cada estimación, hay que inicializar la red, puesto que si no se hace, el primer entrenamiento fija los parámetros y los demás entrenamientos llegan a la misma solución dando como resultado la misma estimación y volviendo inútil el sistema de las 12estimaciones. Parece ser que, con los pesos sinópticos de la primera iteración, los algoritmos de optimización del error llegan rápidamente a la solución que es exactamente la misma. Por tanto, esta función es de vital importancia en nuestro trabajo.

La sintaxis es muy sencilla, sin argumentos de salida y con un único argumento de entrada que es la propia red:

$$red = init(red);$$

Donde "red" es el nombre de la red neuronal, como en las anteriores funciones.

4 SCRIPT PRINCIPAL

En total se tienen 5 scripts principales, pero cuya diferencia entre ellos es mínima, por lo que se ha decidido presentar solamente uno en el cual se explicará el funcionamiento general y en aquellos puntos donde se puedan diferenciar los scripts, comentar como cambiaría el script.

- Golay
- Spline
- Splinevariable
- Splinevariable1
- Sploinevariable2

- Lo primero que hará el script será cargar todos los ficheros necesarios con los datos correspondientes de temperaturas y potencias y crear el fichero donde se guardarán los errores.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%                                                                                 %
% Red Neuronal Para La Estimación De La Curva De Consumo Eléctrico                %
%                                                                                 %
%             Metodo De Los Extremos De Temperatura                               %
%                                                                                 %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


clear all
close all
warning off


%Pretratamiento de Datos

load mes_abril_2008.mat
load mes_mayo_2008.mat
load mes_junio_2008.mat
load mes_julio_2008.mat
load mes_agosto_2008.mat
load mes_septiembre_2008.mat
load mes_octubre_2008.mat
load mes_noviembre_2008.mat
load mes_diciembre_2008.mat
load mes_enero_2009.mat
load mes_febrero_2009.mat
load mes_marzo_2009.mat
load mes_abril_2009.mat
load mes_mayo_2009.mat
load mes_junio_2009.mat
load mes_julio_2009.mat
load mes_agosto_2009.mat
load mes_septiembre_2009.mat
load mes_octubre_2009.mat
load mes_noviembre_2009.mat
```

```
load mes_enero_2010.mat
load mes_febrero_2010.mat
load mes_marzo_2010.mat
load mes_abril_2010.mat

load temp_abril_2008.mat
load temp_mayo_2008.mat
load temp_junio_2008.mat
load temp_julio_2008.mat
load temp_agosto_2008.mat
load temp_septiembre_2008.mat
load temp_octubre_2008.mat
load temp_noviembre_2008.mat
load temp_diciembre_2008.mat
load temp_enero_2009.mat
load temp_febrero_2009.mat
load temp_marzo_2009.mat
load temp_abril_2009.mat
load temp_mayo_2009.mat
load temp_junio_2009.mat
load temp_julio_2009.mat
load temp_agosto_2009.mat
load temp_septiembre_2009.mat
load temp_octubre_2009.mat
load temp_noviembre_2009.mat
load temp_enero_2010.mat
load temp_febrero_2010.mat
load temp_marzo_2010.mat
load temp_abril_2010.mat

load Error.mat
```

- Se pasan los datos cuarto-horarios de potencia a datos horarios:

```
abril_2008media = media_potencia(mes_abril_2008);
mayo_2008media = media_potencia(mes_mayo_2008);
julio_2008media = media_potencia(mes_julio_2008);
junio_2008media = media_potencia(mes_junio_2008);
agosto_2008media = media_potencia(mes_agosto_2008);
septiembre_2008media = media_potencia(mes_septiembre_2008);
octubre_2008media = media_potencia(mes_octubre_2008);
noviembre_2008media = media_potencia(mes_noviembre_2008);
diciembre_2008media = media_potencia(mes_diciembre_2008);
enero_2009media = media_potencia(mes_enero_2009);
febrero_2009media = media_potencia(mes_febrero_2009);
marzo_2009media = media_potencia(mes_marzo_2009);
abril_2009media = media_potencia(mes_abril_2009);
mayo_2009media = media_potencia(mes_mayo_2009);
junio_2009media = media_potencia(mes_junio_2009);
julio_2009media = media_potencia(mes_julio_2009);
agosto_2009media = media_potencia(mes_agosto_2009);
septiembre_2009media = media_potencia(mes_septiembre_2009);
octubre_2009media = media_potencia(mes_octubre_2009);
noviembre_2009media = media_potencia(mes_noviembre_2009);
enero_2010media = media_potencia(mes_enero_2010);
febrero_2010media = media_potencia(mes_febrero_2010);
marzo_2010media = media_potencia(mes_marzo_2010);
abril_2010media = media_potencia(mes_abril_2010);
```

- Se suavizan las curvas mediante splines o savitzky-golay

```
p=0.95;

abril_2008 = spline_cubico(media_potencia(mes_abril_2008),p);
mayo_2008 = spline_cubico(media_potencia(mes_mayo_2008),p);
julio_2008 = spline_cubico(media_potencia(mes_julio_2008),p);
junio_2008 = spline_cubico(media_potencia(mes_junio_2008),p);
agosto_2008 = spline_cubico(media_potencia(mes_agosto_2008),p);
septiembre_2008 = spline_cubico(media_potencia(mes_septiembre_2008),p);
octubre_2008 = spline_cubico(media_potencia(mes_octubre_2008),p);
noviembre_2008 = spline_cubico(media_potencia(mes_noviembre_2008),p);
diciembre_2008 = spline_cubico(media_potencia(mes_diciembre_2008),p);
enero_2009 = spline_cubico(media_potencia(mes_enero_2009),p);
febrero_2009 = spline_cubico(media_potencia(mes_febrero_2009),p);
marzo_2009 = spline_cubico(media_potencia(mes_marzo_2009),p);
abril_2009 = spline_cubico(media_potencia(mes_abril_2009),p);
mayo_2009 = spline_cubico(media_potencia(mes_mayo_2009),p);
junio_2009 = spline_cubico(media_potencia(mes_junio_2009),p);
julio_2009 = spline_cubico(media_potencia(mes_julio_2009),p);
agosto_2009 = spline_cubico(media_potencia(mes_agosto_2009),p);
septiembre_2009 = spline_cubico(media_potencia(mes_septiembre_2009),p);
octubre_2009 = spline_cubico(media_potencia(mes_octubre_2009),p);
noviembre_2009 = spline_cubico(media_potencia(mes_noviembre_2009),p);
enero_2010 = spline_cubico(media_potencia(mes_enero_2010),p);
febrero_2010 = spline_cubico(media_potencia(mes_febrero_2010),p);
marzo_2010 = spline_cubico(media_potencia(mes_marzo_2010),p);
abril_2010 = spline_cubico(media_potencia(mes_abril_2010),p);
```

- o

```
span=6;
degree=3;

abril_2008media = media_potencia(mes_abril_2008);
abril_2010media = media_potencia(mes_abril_2010);

abril_2008 = smooth(abril_2008media(:,2),span,'sgolay',degree);
mayo_2008 = smooth(mayo_2008media(:,2),span,'sgolay',degree);
junio_2008 = smooth(junio_2008media(:,2),span,'sgolay',degree);
julio_2008 = smooth(julio_2008media(:,2),span,'sgolay',degree);
agosto_2008 = smooth(agosto_2008media(:,2),span,'sgolay',degree);
septiembre_2008 = smooth(septiembre_2008media(:,2),span,'sgolay',degree);
octubre_2008 = smooth(octubre_2008media(:,2),span,'sgolay',degree);
noviembre_2008 = smooth(noviembre_2008media(:,2),span,'sgolay',degree);
diciembre_2008 = smooth(diciembre_2008media(:,2),span,'sgolay',degree);
enero_2009 = smooth(enero_2009media(:,2),span,'sgolay',degree);
febrero_2009 = smooth(febrero_2009media(:,2),span,'sgolay',degree);
marzo_2009 = smooth(marzo_2009media(:,2),span,'sgolay',degree);
abril_2009 = smooth(abril_2009media(:,2),span,'sgolay',degree);
mayo_2009 = smooth(mayo_2009media(:,2),span,'sgolay',degree);
junio_2009 = smooth(junio_2009media(:,2),span,'sgolay',degree);
julio_2009 = smooth(julio_2009media(:,2),span,'sgolay',degree);
agosto_2009 = smooth(agosto_2009media(:,2),span,'sgolay',degree);
septiembre_2009 = smooth(septiembre_2009media(:,2),span,'sgolay',degree);
octubre_2009 = smooth(octubre_2009media(:,2),span,'sgolay',degree);
noviembre_2009 = smooth(noviembre_2009media(:,2),span,'sgolay',degree);
```

```
enero_2010 = smooth(enero_2010media(:,2),span,'sgolay',degree);
febrero_2010 = smooth(febrero_2010media(:,2),span,'sgolay',degree);
marzo_2010 = smooth(marzo_2010media(:,2),span,'sgolay',degree);
abril_2010 = smooth(abril_2010media(:,2),span,'sgolay',degree);
```

- Se almacenan en los vectores las potencias o las temperaturas necesarias.

```
for desplaz=0:9
mes_anterior = agosto_2008;        %Semana anterior para el estudio
for i=1:168
semana_anterior(i)=mes_anterior(length(mes_anterior)-168+i);
end

Vector_Datos_Semanas=[semana_anterior septiembre_2008' octubre_2008'
noviembre_2008' diciembre_2008' enero_2009' febrero_2009' marzo_2009'
abril_2009' mayo_2009' junio_2009' julio_2009' agosto_2009'
septiembre_2009'];
Vector_Datos_Semanas_h=Vector_Datos_Semanas;
Vector_Datos_Semanas_h_real=[semana_anterior septiembre_2008media(:,2)'
octubre_2008media(:,2)' noviembre_2008media(:,2)' diciembre_2008media(:,2)'
enero_2009media(:,2)'  febrero_2009media(:,2)'  marzo_2009media(:,2)'
abril_2009media(:,2)' mayo_2009media(:,2)'  junio_2009media(:,2)'
julio_2009media(:,2)'  agosto_2009media(:,2)'  septiembre_2009media(:,2)'];

%Maximos y minimos de temperatura
temp_mes_anterior = temp_agosto_2008;        %Semana anterior para el
estudio

for i=1:7
    for j=1:24
        temp_semana_anterior(i,j)
=temp_mes_anterior(length(temp_mes_anterior)-7+i,j);
    end
end

Matriz_Temp_Semanas=[temp_semana_anterior; temp_septiembre_2008;
temp_octubre_2008; temp_noviembre_2008; temp_diciembre_2008;
temp_enero_2009; temp_febrero_2009; temp_marzo_2009; temp_abril_2009;
temp_mayo_2009; temp_junio_2009; temp_julio_2009; temp_agosto_2009;
temp_septiembre_2009];

[Vector_Temp_Max_Semanas, Vector_Temp_Min_Semanas] =
extremos_temperatura(Matriz_Temp_Semanas);


%%%%%%% Disposicion de datos
i=1;
j=1;
k=1;
primera_vez=0;
```

- Se forman los vectores correspondientes a la semana anterior al mes de entrenamiento, al mes de entrenamiento y a la semana a predecir. Además se indica el día por el que empiezan estos y si tienen algunos días no laborables.

```
for n=337:504
    if n==1
        primera_vez=1;
    end
    semana_anterior_h(i)=Vector_Datos_Semanas_h(n);
    temp_max_semana_anterior_h(i)=Vector_Temp_Max_Semanas(n);
    temp_min_semana_anterior_h(i)=Vector_Temp_Min_Semanas(n);
    i=i+1;
end
clear i
                                        %Mes para datos del training set
for n=505:1176
    mes_datos_h(j)= Vector_Datos_Semanas_h(n);
    temp_max_mes_datos_h(j)=Vector_Temp_Max_Semanas(n);
    temp_min_mes_datos_h(j)=Vector_Temp_Min_Semanas(n);
    j=j+1;
end
clear j
                                        %Semana a estimar
for n=1177:1344
    semana_estimar_h(k)= Vector_Datos_Semanas_h(n);
    semana_estimar_h_real(k)= Vector_Datos_Semanas_h_real(n);
    temp_max_semana_estimar_h(k)=Vector_Temp_Max_Semanas(n);
    temp_min_semana_estimar_h(k)=Vector_Temp_Min_Semanas(n);
    k=k+1;
end
clear k

%Dia por el que empieza la semana anterior: l,m,x,j,v,s,d
d_semana= 'l';

%Dia por el que empieza el mes: l,m,x,j,v,s,d
d_datos= 'l';

%Dia por el que empieza la semana a estimar: l,m,x,j,v,s,d
d_estimar= 'l';

%Festivos distintos a fin de semana en la semana anterior, de 1 a 7
fes_semana=[1];

%Festivos distintos a fin de semana en el mes de datos, de 1 a 28
fes_datos= [0];

%Festivos distintos a fin de semana en la semana a estimar, de 1 a 7
fes_estimar= [0];
```

- Se crea el vector festivo, este vector se llenará de 1 y 0, si el día es laborable, todas las horas del vector correspondiente a ese día tendrán un valor de 1, si el día es festivo o no laborable, el valor será de 0.

```
%Crear vector festivos

fes_semana_anterior_h=findes(semana_anterior_h,d_semana);
fes_mes_datos_h=findes(mes_datos_h,d_datos);
fes_semana_estimar_h=findes(semana_estimar_h,d_estimar);

%Cambiar vector festivos

fes_semana_anterior_h=fiesta(fes_semana_anterior_h, fes_semana);
```

```
fes_mes_datos_h=fiesta(fes_mes_datos_h, fes_datos);
fes_semana_estimar_h=fiesta(fes_semana_estimar_h, fes_estimar);


%Crear la matriz de tiempo (horas en binario)

tiempo_semana_anterior_h=tiempo_bin(semana_anterior_h);
tiempo_mes_datos_h=tiempo_bin(mes_datos_h);
tiempo_semana_estimar_h=tiempo_bin(semana_estimar_h);

%Guardo el vector mes_datos_h en otro vector para su posterior uso

vec_mes_datos_h = mes_datos_h;
```

- Se escalan las variables exógenas de potencia y temperatura, para ello harán falta lo valores máximos y mínimos de la potencia y temperatura durante el periodo.

```
%Escalar los datos


Lmax=2663; %Máximo global (horario) del periodo 2008-2010 (Septiembre'09)
Lmin=1129; %Mínimo global (horario) del periodo 2008-2010 (Mayo'08)

[semana_anterior_h]=escalar(semana_anterior_h, Lmax, Lmin);
[mes_datos_h]=escalar(mes_datos_h, Lmax, Lmin);

Tmax=334; %Máximo global (horario) del periodo 2008-2010 (Julio'08)
Tmin=-96; %Mínimo global (horario) del periodo 2008-2010 (Enero'10)

[temp_max_semana_anterior_h] = escalar(temp_max_semana_anterior_h, Tmax,
Tmin);
[temp_min_semana_anterior_h] = escalar(temp_min_semana_anterior_h, Tmax,
Tmin);
[temp_max_mes_datos_h] = escalar(temp_max_mes_datos_h, Tmax, Tmin);
[temp_min_mes_datos_h] = escalar(temp_min_mes_datos_h, Tmax, Tmin);
```

- Se crean los INPUT y los TARGET necesarios para el entrenamiento de la red.

```
%Crear los inputs y targets de la red

semana_anterior_h_in = [semana_anterior_h; fes_semana_anterior_h;
tiempo_semana_anterior_h; temp_max_semana_anterior_h;
temp_min_semana_anterior_h];
mes_datos_h_in = [mes_datos_h; fes_mes_datos_h; tiempo_mes_datos_h;
temp_max_mes_datos_h; temp_min_mes_datos_h];

[INPUTS, TARGETS]=ent_sal_red(semana_anterior_h_in,mes_datos_h_in);
```

- Se dividen los datos de entrenamiento en "`trainInd`" "`ValInd`" y "`testInd`" y se crea la red, el número de neuronas variará en función del script utilizado, serán 15 para los scripts llamados spline, splinevariable y golay, 14 para splinevariable1 y 13 neuronas para splinevariable2.

```
%Crear la red

neuronas=15;    % N° de neuronas capa oculta

red=newff(INPUTS, TARGETS, neuronas, {'logsig'});


%%%division de datos manual
red.divideFcn='divideind';

trainRat=0.60;
valRat=0.20;  %Fijo
testRat=0.20;

pause(1)
[trainP,valP,testP,trainInd,valInd,testInd] =
division_datos_prueba(vec_mes_datos_h,trainRat,valRat,testRat,desplaz);

red.divideParam.trainInd = trainInd;
red.divideParam.valInd = valInd;
red.divideParam.testInd = testInd;
```

- Por último se simula y se obtienen los resultados correspondientes y las gráficas

```
%%%%SIMULAR


% Empezamos la predicción


% Hacemos coincidir la primera semana estimada con la última de nuestro
% conjunto de entrenamiento para facilitar las cuentas en la predicción.


%close all % Cerramos gráficas de los históricos

m=1;

carga_estimada=[];

for i=length(mes_datos_h)-24*7:length(mes_datos_h)

    carga_estimada(m)=mes_datos_h(i);
    m=m+1;

end

i=m-1; %Marcamos el inicio para el siguiente bucle (24*7+1)
m=1;   %Reiniciamos el contador m
k=1;   %Iniciamos contador de las estimaciones


estimacion=[];

num_dias=7; %Número de días de predicción.
```

```
% Repetimos 20 veces entrenar y estimar

num_iteraciones=12;

for h=1:num_iteraciones

    %Inicializamos la red

    red=init(red);

    %Entrenar la red

    red=train(red, INPUTS, TARGETS);

    prediccion=[];


    while (m<24*num_dias+1)

        prediccion(1,m)=carga_estimada(i-24*7);
        prediccion(2,m)=carga_estimada(i-24);
        prediccion(3,m)=carga_estimada(i-3);
        prediccion(4,m)=carga_estimada(i-2);
        prediccion(5,m)=carga_estimada(i-1);
        prediccion(6,m)=fes_semana_estimar_h(m);
        prediccion(7,m)=tiempo_semana_estimar_h(1,m);
        prediccion(8,m)=tiempo_semana_estimar_h(2,m);
        prediccion(9,m)=tiempo_semana_estimar_h(3,m);
        prediccion(10,m)=tiempo_semana_estimar_h(4,m);
        prediccion(11,m)=tiempo_semana_estimar_h(5,m);
        prediccion(12,m)=temp_max_mes_datos_h(m);
        prediccion(13,m)=temp_min_mes_datos_h(m);

        carga_estimada(m+24*7)=sim(red,prediccion(:,m));

        estimacion(k,m)=carga_estimada(m+24*7)*(Lmax-Lmin)+ Lmin;


        m=m+1;
        i=i+1;
    end

    k=k+1;
    i=24*7+1;
    m=1;

end

% Obtenemos la media de las estimaciones y calculamos el error

estimacion_media=[];
error_relativo=[];
error_absoluto=[];
var=[];

for i=1:24*num_dias
```

```
    estimacion_media(i)=sum(estimacion(:,i))/num_iteraciones;

    error_relativo(i)=abs((estimacion_media(i) -
semana_estimar_h(i))/semana_estimar_h(i)*100);
    error_relativo_real(i)=abs((estimacion_media(i) -
semana_estimar_h_real(i))/semana_estimar_h_real(i)*100);

    error_absoluto(i)=abs((estimacion_media(i) - semana_estimar_h(i)));
    error_absoluto_real(i)=abs((estimacion_media(i) -
semana_estimar_h_real(i)));

    var(i)=error_absoluto(i)^2;

end
%Cálculo de la dispersión

desviacion=sqrt(sum(var)/length(var));


%Plot de resultados

limites=[estimacion_media semana_estimar_h]; %para delimitar el plot

% figure
% bar(error_relativo, 0.5, 'r');
% axis([0 length(error_relativo) 0 max(error_relativo)+1]);
% title('Error relativo');
% ylabel('Error relativo en %');
% xlabel('N° de horas');
% figure
% plot(estimacion_media, 'g');
% axis([0 length(estimacion) min(limites)-180 max(limites)+200]);
% title('Comparación de curvas. Verde=Estimada');
% ylabel('Potencia demandada en kW');
% xlabel('N° de horas');
% hold

r=24*num_dias;

%plot(semana_estimar_h(1:r), 'b');

%error medio y desviación:
disp(' ');
%%fprintf('Error medio:
%1.4f\n\n',sum(error_relativo)/length(error_relativo));
%%fprintf('Error medio real:
%1.4f\n\n',sum(error_relativo_real)/length(error_relativo_real));
err_medio(desplaz+1)=sum(error_relativo)/length(error_relativo);
err_medio_real(desplaz+1)=sum(error_relativo_real)/length(error_relativo_re
al);
hold off
%%fprintf('Desviación típica de la estimación: %1.4f kW\n\n',desviacion);
des_tipica(desplaz+1)=desviacion;

if desplaz==0
    Error_Fila=error_relativo;
    Error_Columna=error_relativo;
else
    Error_Fila=[Error_Fila error_relativo];
    Error_Columna=[Error_Columna;error_relativo];
end
```

```
if desplaz==9
figure
boxplot(Error_Columna');
title('Errores de estimación semanal');
xlabel('Grupo de entrenamiento')
ylabel('Error relativo (%)')
end

savefile='Error.mat';
save(savefile, 'Error_Fila', 'Error_Columna', 'Error_Semana');
end
```

- La parte en rojo (spline y golay) del programa también variará en función del script, para spline variable:

```
prediccion(1,m)=carga_estimada(i-24);
prediccion(2,m)=carga_estimada(i-3);
prediccion(3,m)=carga_estimada(i-2);
prediccion(4,m)=carga_estimada(i-1);
prediccion(5,m)=fes_semana_estimar_h(m);
prediccion(6,m)=tiempo_semana_estimar_h(1,m);
prediccion(7,m)=tiempo_semana_estimar_h(2,m);
prediccion(8,m)=tiempo_semana_estimar_h(3,m);
prediccion(9,m)=tiempo_semana_estimar_h(4,m);
prediccion(10,m)=tiempo_semana_estimar_h(5,m);
prediccion(11,m)=temp_max_mes_datos_h(m);
prediccion(12,m)=temp_min_mes_datos_h(m);
```

- Para splinevariable1

```
prediccion(1,m)=carga_estimada(i-3);
prediccion(2,m)=carga_estimada(i-2);
prediccion(3,m)=carga_estimada(i-1);
prediccion(4,m)=fes_semana_estimar_h(m);
prediccion(5,m)=tiempo_semana_estimar_h(1,m);
prediccion(6,m)=tiempo_semana_estimar_h(2,m);
prediccion(7,m)=tiempo_semana_estimar_h(3,m);
prediccion(8,m)=tiempo_semana_estimar_h(4,m);
prediccion(9,m)=tiempo_semana_estimar_h(5,m);
prediccion(10,m)=temp_max_mes_datos_h(m);
prediccion(11,m)=temp_min_mes_datos_h(m);
```

- Para splinevariable2.

```
prediccion(1,m)=carga_estimada(i-1);
prediccion(2,m)=fes_semana_estimar_h(m);
prediccion(3,m)=tiempo_semana_estimar_h(1,m);
prediccion(4,m)=tiempo_semana_estimar_h(2,m);
prediccion(5,m)=tiempo_semana_estimar_h(3,m);
prediccion(6,m)=tiempo_semana_estimar_h(4,m);
prediccion(7,m)=tiempo_semana_estimar_h(5,m);
prediccion(8,m)=temp_max_mes_datos_h(m);
prediccion(9,m)=temp_min_mes_datos_h(m);
```