



Progetto Intelligent Systems

Bike sharing dataset

Chiara Caiazza, Martina Troscia

Sommario

Introduzione	2
Prima parte.....	2
Neural Networks.....	2
Scelta del sottoinsieme di caratteristiche in <i>Day</i>	2
Calcolo del numero di nodi nell'hidden layer in <i>Day</i>	7
Rete neurale finale per <i>Day</i>	7
Scelta del sottoinsieme di caratteristiche in <i>Hour</i>	9
Calcolo del numero di nodi nell'hidden layer in <i>Hour</i>	14
Rete neurale finale per <i>Hour</i>	15
Rete RBF per <i>Day</i>	16
Rete RBF per <i>Hour</i>	23
Modello di fitting Fuzzy	28
Modello Fuzzy-Mamdani per <i>Day</i>	28
Modello Fuzzy-Mamdani per <i>Hour</i>	37
ANFIS per <i>Day</i>	46
ANFIS per <i>Hour</i>	54
Seconda parte.....	59
NN Time Series Tool su <i>Day</i> , anno 2011.....	59
Strategia ad anello chiuso su <i>Day</i> , anno 2012.....	64

Introduzione

Con questo lavoro intendiamo stimare il numero di noleggi di biciclette a Washington D.C., basandoci su informazioni stagionali e del tempo. I data set utilizzati sono quelli prelevabili dal sito <http://archive.ics.uci.edu/ml/datasets/Bike+Sharing+Dataset>, suddivisi nei due file *day* and *hourly* contenenti informazioni del noleggio tra il 1 Gennaio 2011 e il 31 Dicembre 2012.

Prima parte

La prima parte del progetto consiste nell'effettuare il fitting del numero di noleggi di biciclette basandosi su un sottoinsieme delle informazioni presenti nei due file. Come prima cosa è necessario scegliere quale sia il subset di caratteristiche che consente di allenare la rete in maniera ottimale e di ottenere le migliori performance.

Neural Networks

Scelta del sottoinsieme di caratteristiche in *Day*

Innanzitutto importiamo il dataset nel Working Space.

```
>>data = xlsread('D:\Documents\Bike-Sharing-Dataset\day.csv')
```

Come possiamo vedere, abbiamo ottenuto una matrice di 731 righe che costituiscono i diversi campioni e di 16 colonne rappresentanti:

- instant: indice di riga;
- dteday: data;
- season: 1=primavera, 2=estate, 3=autunno, 4=inverno;
- yr: 0=2011, 1=2012;
- mnth: mesi da 1 a 12;
- holiday: 1=vacanza nazionale, 0=no;
- weekday: giorno della settimana da 0=domenica a 6=sabato;
- workingday: 0=weekend/festa, 1=feriale;
- weathersit: 1=sereno, poche nuvole, 2=nebbia+nuvole, 3=neve lieve, pioggia lieve, temporali+nubi sparse, 4=pioggia pesante, grandine, temporali+nebbia, neve+nebbia;
- temp: temperatura in Celsius normalizzata;
- atemp: temperatura percepita in Celsius normalizzata;
- hum: umidità normalizzata;
- windspeed: velocità del vento normalizzata;
- casual: numero di utenti occasionali;
- registered: numero di utenti registrati;
- cnt: numero totale di noleggi di biciclette, considerando utenti registrati e non;

Per questa parte facciamo riferimento allo script **DAY_Optimal_Features_script**.

Creiamo una prima matrice degli input che include tutte le colonne eccetto instant e dteday che costituiscono l'”intestazione” del record e casual, registered e cnt che rappresentano l'output del sistema.

```
input01 = data(:,3:13);
target = data(:,16);
```

Dando il comando nftool si accede all'app interna a nnstart che consente di risolvere il problema di fitting con una rete neurale a due livelli feed-forward (con una MLP).

```
>>nftool
```

Innanzitutto abbiamo testato le performance della rete (Mean Square Error sul test set) che riceve input01 come input e target come obiettivo, per sapere quali fossero le performance del sistema considerando tutte le features e quindi con informazioni ridondanti e avere un'idea del valore dell'errore di partenza.

	input01
Season	X
Yr	X
Mnt	X
Holiday	X
Weekday	X
Workingday	X
Weathersit	X
Temp	X
Atemp	X
Hum	X
Windspeed	X
MSE (medio)	5.00E+05

Le prime features che sembrano legate tra loro sono temp e atemp; partendo da input01, creiamo input02 eliminando la feature atemp e input03 eliminando la feature temp.

```
input02 = input01;
input02(:,9) = [];
input03 = input01;
input03(:,8) = [];
```

	input02	input03
Season	X	X
Yr	X	X
Mnt	X	X
Holiday	X	X
Weekday	X	X
Workingday	X	X
Weathersit	X	X
Temp	X	
Atemp		X

Hum	X	X
Windspeed	X	X
MSE (medio)	4.85E+05	5.57E+05

Tra input02 e input03 scegliamo input02 poiché ha l'errore più basso e più stabile.

Il secondo gruppo di features che andiamo a analizzare è holiday, weekday e workingday; partendo da input02, creiamo input04 eliminando la feature workingday, input05 eliminando la feature holiday e input06 eliminando la feature weekday. Vogliamo poi vedere cosa accade considerando solo una delle variabili del gruppo, pertanto creiamo input07 in cui manteniamo solo holiday, input08 in cui manteniamo solo weekday e input09 in cui manteniamo solo workingday.

```
input04 = input02;
input04(:,6) = [];
input05 = input02;
input05(:,4) = [];
input06 = input02;
input06(:,5) = [];
input07 = input02;
input07(:,5:6) = [];
input08 = input02;
input08(:,6) = [];
input08(:,4) = [];
input09 = input02;
input09(:,4:5) = [];
```

	input04	input05	input06	input07	input08	input09
Season	X	X	X	X	X	X
Yr	X	X	X	X	X	X
Mnt	X	X	X	X	X	X
Holiday	X		X	X		
Weekday	X	X			X	
Workingday		X	X			X
Weathersit	X	X	X	X	X	X
Temp	X	X	X	X	X	X
Atemp						
Hum	X	X	X	X	X	X
Windspeed	X	X	X	X	X	X
MSE (medio)	4.21E+05	4.42E+05	1.16E+06	4.71E+05	4.64E+05	5.21E+05

La combinazione migliore è input04, quella che tiene in considerazione holiday e weekday. Notiamo però che input08 ha prestazioni solo di poco peggiori per cui scegliamo questa configurazione di caratteristiche.

Andiamo ora a analizzare quale tra season e mnt sia la feature che più influenza le performance del sistema. A partire da input08, creiamo pertanto input10 eliminando la feature mnt e input11 eliminando season.

```

input10 = input08;
input10(:,3) = [];
input11 = input08;
input11(:,1) = [];

```

	input10	input11
Season	X	
Yr	X	X
Mnt		X
Holiday		
Weekday	X	X
Workingday		
Weathersit	X	X
Temp	X	X
Atemp		
Hum	X	X
Windspeed	X	X
MSE (medio)	4.07E+05	4.79E+05

Le performance con input10 sono le migliori, addirittura le migliori ottenute fino a questo momento, quindi scegliamo questa configurazione di features.

Analizziamo ora il gruppo di features weathersit, hum e windspeed. Partendo da input10, creiamo input12 eliminando windspeed, input13 eliminando weathersit, input14 eliminando hum; vogliamo poi vedere le performance considerando una feature del gruppo singolarmente; creiamo quindi input15 contenente solo weathersit, input16 contenente solo hum e input17 contenente solo windspeed.

```

input12 = input10;
input12(:,7) = [];
input13 = input10;
input13(:,4) = [];
input14 = input10;
input14(:,6) = [];
input15 = input10;
input15(:,6:7) = [];
input16 = input10;
input16(:,7) = [];
input16(:,4) = [];
input17 = input10;
input17(:,6) = [];
input17(:,4) = [];

```

	input12	input13	input14	input15	input16	input17
Season	X	X	X	X	X	X
Yr	X	X	X	X	X	X
Mnt						
Holiday						
Weekday	X	X	X	X	X	X
Workingday						
Weathersit	X		X	X		
Temp	X	X	X	X	X	X

Atemp						
Hum	X	X			X	
Windspeed		X	X			X
MSE (medio)	4.42E+05	4.70E+05	4.92E+05	4.64E+05	5.28E+05	7.78E+05

La performance migliore si ottiene con input13, però si può notare che con input15 sono di poco peggiori e riusciamo a ridurre di più il set di feature ottimali, quindi decidiamo di prendere questa ultima configurazione.

Abbiamo ottenuto un sottoinsieme di 5 caratteristiche; proviamo a vedere se è quello ottimale o se si possono togliere altre feature; proviamo pertanto a vedere quanto cambiano le performance togliendo da questo set una feature alla volta.

```
input18 = input15;
input18(:,2) = [];
input19 = input15;
input19(:,1) = [];
input20 = input15;
input20(:,3) = [];
input21 = input15;
input21(:,4) = [];
input22 = input15;
input22(:,5) = [];
```

	input18	input19	input20	input21	input22
Season	X		X	X	X
Yr		X	X	X	X
Mnt					
Holiday					
Weekday	X	X		X	X
Workingday					
Weathersit	X	X	X		X
Temp	X	X	X	X	
Atemp					
Hum					
Windspeed					
MSE (medio)	1.51E+06	6.85E+05	5.78E+05	8.14E+05	1.03E+06

Togliendo una sola feature da input15 notiamo che le performance del sistema peggiorano notevolmente; pertanto preferiamo mantenere tutte le feature. Il subset ottimale scelto è quindi

- season
- yr
- weekday
- weathersit
- temp

Abbiamo scelto di tenere anche yr perché abbiamo notato che togliendo questa feature l'errore triplica rispetto all'errore medio; possiamo presupporre, data l'influenza che questa caratteristica esercita nella rete, che il servizio sia ancora in una fase iniziale o di sviluppo in cui le potenzialità del sistema non sono note al bacino delle utenze.

Calcolo del numero di nodi nell'hidden layer in Day

Vediamo ora di valutare quale sia il numero di nodi ottimale nell'hidden layer per ottenere le performance migliori nella rete; la testiamo per un numero di nodi da 1 a 15, facendo 15 test per ogni scelta del numero dei nodi. Per questo test facciamo riferimento allo script **DAY_Computation_Hidden_Nodes**.

#nodi	MSE (medio)
1	6.33E+05
2	7.17E+05
3	5.58E+05
4	4.32E+05
5	5.42E+05
6	5.58E+05
7	4.75E+05
8	5.33E+05
9	5.17E+05
10	5.33E+05
11	4.92E+05
12	4.67E+05
13	5.25E+05
14	5.92E+05
15	4.25E+05

La media più bassa in assoluto è quella relativa a 15 nodi nascosti che però sono troppi in proporzione al numero di campioni che riceve normalmente la rete. Possiamo notare che anche con 4 e 7 nodi nascosti le performance sono molto buone: decidiamo di fare altre 25 prove con queste due scelte. Per questo test facciamo riferimento allo script **DAY_Computation_Hidden_Nodes_4or7**.

#nodi	MSE (medio)
4	5.21E+05
7	4.83E+05

Su un numero di prove maggiori possiamo notare che l'errore è aumentato di molto per 4 nodi mentre è rimasto pressoché lo stesso per 7 nodi: scegliamo pertanto questo numero.

Rete neurale finale per Day

Possiamo quindi generare lo script della rete neurale ottenuta. Facciamo riferimento allo script **DAY_MLP_final** che esegue lo script DAY_Optimal_Features_script per caricare nel workspace le variabili necessarie, esegue il comando


```
hiddenLayerSize = 7;
```

per fissare il numero di nodi nascosti ottimale ottenuto dalle prove precedenti e il comando

```
num_training = 30;
```

per fare il training e testing della rete per 30 volte. Esegue infine lo script **DAY_MLP_simple_script**.

```
% Solve an Input-Output Fitting problem with a Neural Network
% Script generated by Neural Fitting app
% Created Wed May 20 17:55:09 CEST 2015
%
% This script assumes these variables are defined:
%
%   input - input data.
%   target - target data.

x = input15';
t = target';

% Choose a Training Function
% For a list of all training functions type: help nntrain
% 'trainlm' is usually fastest.
% 'trainbr' takes longer but may be better for challenging problems.
% 'trainscg' uses less memory. NFTOOL falls back to this in low memory
situations.
trainFcn = 'trainlm'; % Levenberg-Marquardt

% Create a Fitting Network

% hiddenLayerSize = 7;
%%to have the possibility of changing the value in the workspace
net = fitnet(hiddenLayerSize,trainFcn);

% Setup Division of Data for Training, Validation, Testing
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

for i=1:num_training
% Train the Network
[net,tr] = train(net,x,t);
end

% Test the Network
y = net(x);
e = gsubtract(t,y);
performance = perform(net,t,y)

% View the Network
view(net)

% Plots
% Uncomment these lines to enable various plots.
%figure, plotperform(tr)
%figure, plottrainstate(tr)
%figure, plotfit(net,x,t)
%figure, plotregression(t,y)
%figure, ploterrhist(e)
```

Possiamo vedere di seguito i risultati delle prove effettuate.

6.04255E+05	5.04565E+05
6.02496E+05	3.87287E+05
6.03703E+05	7.19009E+05
3.41355E+05	4.87246E+05
7.25363E+05	5.69702E+05
6.70125E+05	6.06845E+05
4.36277E+05	4.54980E+05
4.05335E+05	6.46310E+05
4.46874E+05	6.26678E+05
5.88029E+05	4.02970E+05
6.06845E+05	5.02720E+05
4.54980E+05	5.43591E+05
4.81981E+05	4.93581E+05
6.46310E+05	4.95365E+05
6.26678E+05	5.02720E+05
5.39473E+05	

Scelta del sottoinsieme di caratteristiche in *Hour*

Innanzitutto importiamo il dataset nel Working Space.

```
>>data = xlsread('D:\Documents\Bike-Sharing-Dataset\hour.csv')
```

Come possiamo vedere, abbiamo ottenuto una matrice di 17379 righe che costituiscono i diversi campioni e di 17 colonne rappresentanti:

- instant: indice di riga;
- dteday: data;
- season: 1=primavera, 2=estate, 3=autunno, 4=inverno;
- yr: 0=2011, 1=2012;
- mnth: mesi da 1 a 12;
- hr: ore da 0 a 23;
- holiday: 1=vacanza nazionale, 0=no;
- weekday: giorno della settimana da 0=domenica a 6=sabato;
- workingday: 0=weekend/festa, 1=feriale;
- weathersit: 1=sereno, poche nuvole, 2=nebbia+nuvole, 3=neve lieve, pioggia lieve, temporali+nubi sparse, 4=pioggia pesante, grandine, temporali+nebbia, neve+nebbia;
- temp: temperatura in Celsius normalizzata;
- atemp: temperatura percepita in Celsius normalizzata;
- hum: umidità normalizzata;
- windspeed: velocità del vento normalizzata;
- casual: numero di utenti occasionali;
- registered: numero di utenti registrati;
- cnt: numero totale di noleggi di biciclette, considerando utenti registrati e non;

Per questa parte facciamo riferimento allo script **HOUR_Optimal_Features_script**.

Creiamo una prima matrice degli input che include tutte le colonne eccetto instant e dteday che costituiscono l'”intestazione” del record e casual, registered e cnt che rappresentano l'output del sistema.

```
input01 = data(:,3:14);  
target = data(:,17);
```

Dando il comando nftool si accede all'app interna a nnstart che consente di risolvere il problema di fitting con una rete neurale a due livelli feed-forward (con una MLP).

```
>>nftool
```

Innanzitutto abbiamo testato le performance della rete (Mean Square Error sul test set) che riceve input01 come input e target come obiettivo, per sapere quali fossero le performance del sistema considerando tutte le features e quindi con informazioni ridondanti e avere un'idea del valore dell'errore di partenza.

	input01
Season	X
Yr	X
Mnt	X
Hr	X
Holiday	X
Weekday	X
Workingday	X
Weathersit	X
Temp	X
Atemp	X
Hum	X
Windspeed	X
MSE (medio)	2.9E+03

Le prime features che sembrano legate tra loro sono temp e atemp; partendo da input01, creiamo input02 eliminando la feature atemp e input03 eliminando la feature temp.

```
input02 = input01;  
input02(:,10) = [];  
input03 = input01;  
input03(:,9) = [];
```

	input02	input03
Season	X	X
Yr	X	X
Mnt	X	X
Hr	X	X
Holiday	X	X
Weekday	X	X
Workingday	X	X
Weathersit	X	X
Temp	X	
Atemp		X
Hum	X	X
Windspeed	X	X
MSE (medio)	3.7E+03	3.8E+03

Tra input02 e input03 scegliamo input02 poiché ha l'errore più basso.

Il secondo gruppo di features che andiamo a analizzare è holiday, weekday e workingday; partendo da input02, creiamo input04 eliminando la feature workingday, input05 eliminando la feature holiday e input06 eliminando la feature weekday. Vogliamo poi vedere cosa accade considerando solo una delle variabili del gruppo, pertanto creiamo input07 in cui manteniamo solo holiday, input08 in cui manteniamo solo weekday e input09 in cui manteniamo solo workingday.

```
input04 = input02;
input04(:,7) = [];
input05 = input02;
input05(:,5) = [];
input06 = input02;
input06(:,6) = [];
input07 = input02;
input07(:,6:7) = [];
input08 = input02;
input08(:,7) = [];
input08(:,5) = [];
input09 = input02;
input09(:,5:6) = [];
```

	input04	input05	input06	input07	input08	input09
Season	X	X	X	X	X	X
Yr	X	X	X	X	X	X
Mnt	X	X	X	X	X	X
Hr	X	X	X	X	X	X
Holiday	X		X	X		
Weekday	X	X			X	
Workingday		X	X			X
Weathersit	X	X	X	X	X	X
Temp	X	X	X	X	X	X
Atemp						

Hum	X	X	X	X	X	X
Windspeed	X	X	X	X	X	X
MSE (medio)	7.1E+03	3.9E+03	4.4E+03	9.4E+03	8.5E+03	4.9E+03

La combinazione migliore è input05, quella che tiene in considerazione weekday e workingday. Dal momento che le variabili sono ancora troppe scegliamo però input09 con una feature in meno e errore comunque accettabile.

Andiamo ora a analizzare quale tra season e mnt sia la feature che più influenza le performance del sistema. A partire da input09, creiamo pertanto input10 eliminando la feature mnt e input11 eliminando season.

```
input10 = input09;
input10(:,3) = [];
input11 = input09;
input11(:,1) = [];
```

	input10	input11
season	X	
Yr	X	X
Mnt		X
Hr	X	X
holiday		
weekday		
workingday	X	X
weathersit	X	X
temp	X	X
atemp		
Hum	X	X
windspeed	X	X
MSE (medio)	4.0E+03	4.7E+03

Le performance con input10 sono le migliori quindi scegliamo questa configurazione di features.

Analizziamo ora il gruppo di features weathersit, hum e windspeed. Partendo da input10, creiamo input12 eliminando windspeed, input13 eliminando weathersit, input14 eliminando hum; vogliamo poi vedere le performance considerando una feature del gruppo singolarmente; creiamo quindi input15 contenente solo weathersit, input16 contenente solo hum e input17 contenente solo windspeed.

```
input12 = input10;
input12(:,8) = [];
input13 = input10;
input13(:,5) = [];
input14 = input10;
input14(:,7) = [];
input15 = input10;
input15(:,7:8) = [];
input16 = input10;
input16(:,8) = [];
input16(:,5) = [];
input17 = input10;
```

```
input17(:,7) = [];
input17(:,5) = [];
```

	input12	input13	input14	input15	input16	input17
Season	X	X	X	X	X	X
Yr	X	X	X	X	X	X
Mnt						
Hr	X	X	X	X	X	X
Holiday						
Weekday						
Workingday	X	X	X	X	X	X
Weathersit	X		X	X		
Temp	X	X	X	X	X	X
Atemp						
Hum	X	X			X	
Windspeed		X	X			X
MSE (medio)	4.3E+03	4.3E+03	4.8E+03	6.3E+03	4.8E+03	5.1E+03

La performance migliore si ottiene con input12 e input13, però si può notare che con input16 l'errore è di poco peggiore e riusciamo a ridurre di più il set di feature ottimali, quindi decidiamo di prendere questa ultima configurazione.

Abbiamo ottenuto un sottoinsieme di 6 caratteristiche; proviamo a vedere se è quello ottimale o se si possono togliere altre feature; proviamo pertanto a vedere quanto cambiano le performance togliendo da questo set una feature alla volta.

```
input18 = input16;
input18(:,2) = [];
input19 = input16;
input19(:,3) = [];
input20 = input16;
input20(:,1) = [];
input21 = input16;
input21(:,4) = [];
input22 = input16;
input22(:,5) = [];
input23 = input16;
input23(:,6) = [];
```

	input18	input19	input20	input21	input22	input23
Season	X	X		X	X	X
Yr		X	X	X	X	X
Mnt						
Hr	X		X	X	X	X
Holiday						
Weekday						
Workingday	X	X	X		X	X
Weathersit						

Temp	X	X	X	X		X
Atemp						
Hum	X	X	X	X	X	
Windspeed						
MSE (medio)	9.0E+03	2.1E+04	6.3E+03	9.5E+03	7.8E+03	5.9E+03

Notiamo che le performance del sistema con input23 non peggiorano particolarmente; pertanto prendiamo questo set di 5 caratteristiche come il set ottimale

- season
- yr
- hour
- workingday
- temp

Abbiamo scelto di tenere anche yr perché abbiamo notato che togliendo questa feature l'errore triplica rispetto all'errore medio; possiamo presupporre, data l'influenza che questa caratteristica esercita nella rete, che il servizio sia ancora in una fase iniziale o di sviluppo in cui le potenzialità del sistema non sono note al bacino delle utenze.

Calcolo del numero di nodi nell'hidden layer in *Hour*

Vediamo ora di valutare quale sia il numero di nodi ottimale nell'hidden layer per ottenere le performance migliori nella rete; la testiamo per un numero di nodi da 1 a 15, facendo 15 test per ogni scelta del numero dei nodi. Per questo test facciamo riferimento allo script **HOURL_Computation_Hidden_Nodes**.

#nodi	MSE (medio)
1	2.15E+04
2	1.76E+04
3	1.27E+04
4	1.13E+04
5	9.00E+03
6	7.92E+03
7	7.42E+03
8	5.58E+03
9	6.33E+03
10	5.17E+03
11	4.67E+03
12	3.92E+03
13	4.58E+03
14	3.75E+03
15	3.58E+03

La media più bassa in assoluto è quella relativa a 15 nodi nascosti che però sono troppi in proporzione al numero di campioni che riceve normalmente la rete; scegliamo quindi 12 nodi nascosti.

Rete neurale finale per Hour

Possiamo quindi generare lo script della rete neurale ottenuta. Facciamo riferimento allo script **DAY_MLP_final** che esegue lo script **DAY_Optimal_Features_script** per caricare nel workspace le variabili necessarie, esegue il comando

```
hiddenLayerSize = 12;
```

per fissare il numero di nodi nascosti ottimale ottenuto dalle prove precedenti e il comando

```
num_training = 30;
```

per fare il training e testing della rete per 30 volte. Esegue infine lo script **DAY_MLP_simple_script**.

```
% Solve an Input-Output Fitting problem with a Neural Network
% Script generated by Neural Fitting app
% Created Sat May 23 18:28:26 CEST 2015
%
% This script assumes these variables are defined:
%
%   input - input data.
%   target - target data.

x = input';
t = target';

% Choose a Training Function
% For a list of all training functions type: help nntrain
% 'trainlm' is usually fastest.
% 'trainbr' takes longer but may be better for challenging problems.
% 'trainscg' uses less memory. NFTOOL falls back to this in low memory
situations.
trainFcn = 'trainlm'; % Levenberg-Marquardt

% Create a Fitting Network
%%hiddenLayerSize = 12; to change this parameter in the Workspace
net = fitnet(hiddenLayerSize,trainFcn);

% Setup Division of Data for Training, Validation, Testing
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

for i=1:num_training
% Train the Network
[net,tr] = train(net,x,t);
end

% Test the Network
y = net(x);
e = gsubtract(t,y);
performance = perform(net,t,y)
% View the Network
view(net)

% Plots
% Uncomment these lines to enable various plots.
%figure, plotperform(tr)
%figure, plottrainstate(tr)
%figure, plotfit(net,x,t)
```



```
%figure, plotregression(t,y)
%figure, ploterrhist(e)
```

Possiamo vedere di seguito i risultati delle prove effettuate.

4.097E+03	4.782E+03
6.364E+03	4.839E+03
3.561E+03	4.394E+03
7.059E+03	4.271E+03
4.707E+03	3.582E+03
4.728E+03	4.068E+03
4.264E+03	3.978E+03
5.947E+03	4.318E+03
3.619E+03	5.335E+03
4.172E+03	6.464E+03
4.177E+03	4.286E+03
4.137E+03	3.720E+03
3.874E+03	4.500E+03
4.010E+03	6.510E+03
7.456E+03	4.415E+03
4.721E+03	

Rete RBF per Day

Lo script contenuto nel file **day_RBF_train_test.m** crea, a partire dalla matrice dei dati selezionati per Day (input15) e i relativi output (target), le seguenti strutture dati:

- train_set: una matrice 5x549 contenente il 70% dei dati iniziali (selezionati in modo random);
- output_train: un vettore 1x549 contenenti i relativi output;
- test_set: una matrice 5x182 contenente il 30% dei dati iniziali (selezionati in modo random);
- output_test: un vettore 1x182 contenenti i relativi output;

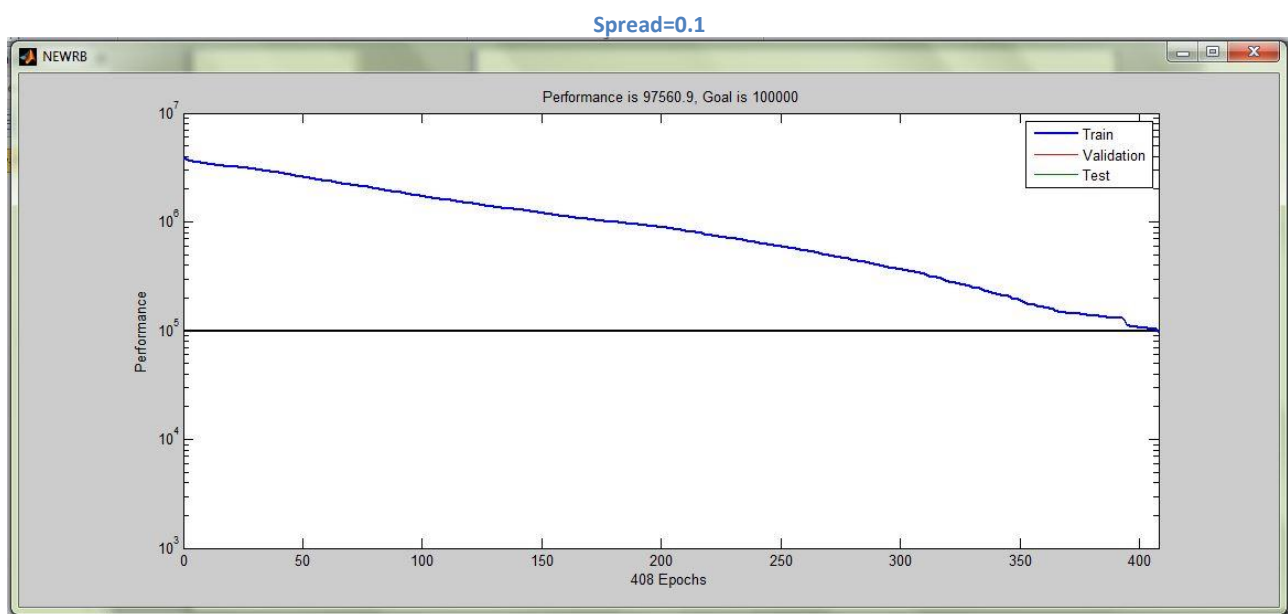
Per ogni configurazione sono state effettuate 10 prove; In tabella è stata riportata la media.

GOAL=100.000		
Spread	N° neuroni necessari	Performance (MSE sul test set)
0.00001	419	1.08E+07
0.0001	419	1.09E+07
0.001	420	1.05E+07
0.01	412	8.86E+06
0.1	310	1.97E+08

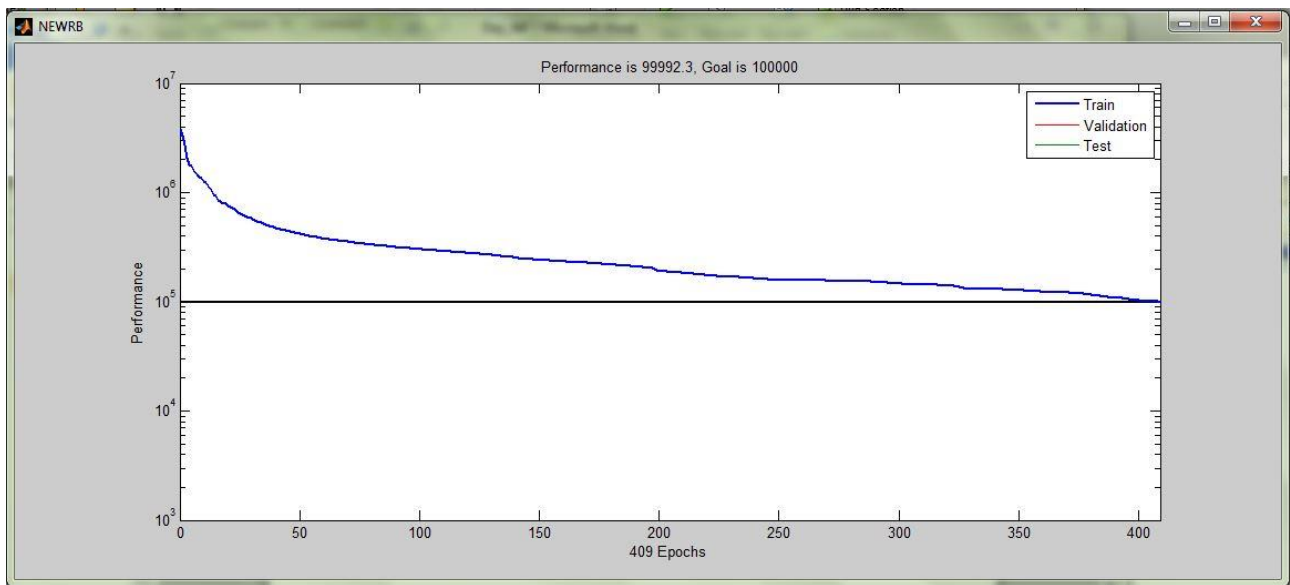
1	401	9.22E+08
10	549	La computazione viene arrestata prima di raggiungere il goal
100	549	
1000	549	
10 000	549	

Con goal=100.000 la migliore scelta possibile in termini di performance e numero di neuroni necessari sarebbe con spread=0.01.

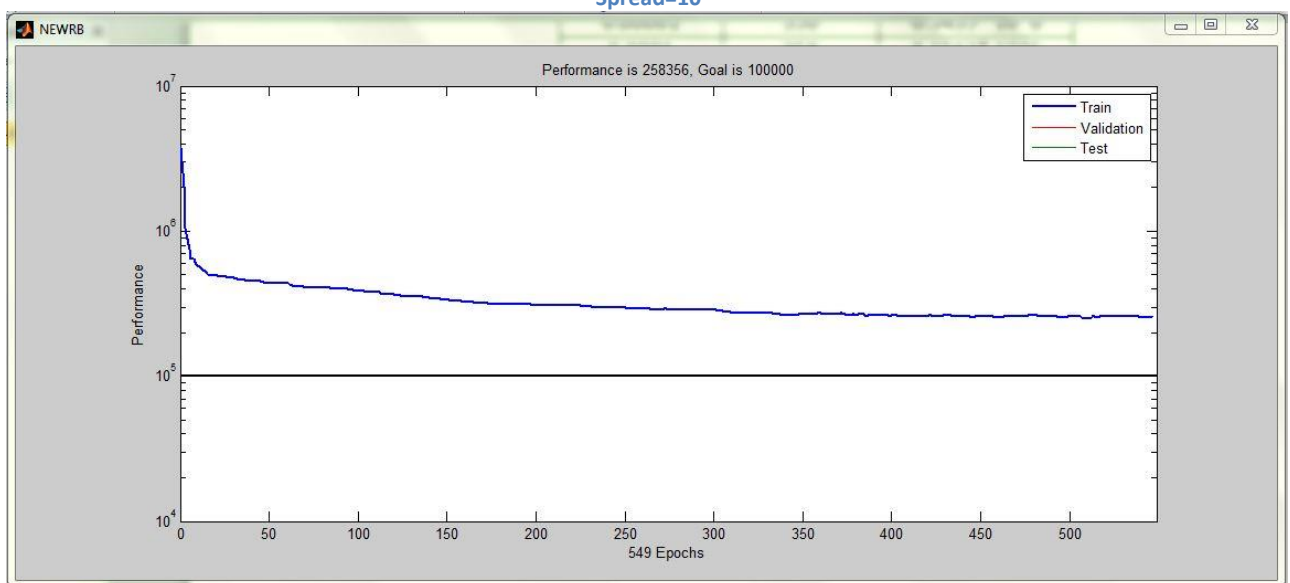
Tuttavia analizzando i grafici possiamo notare che, all'aumentare dello spread (nello specifico da 1 in poi), l'errore cala rapidamente all'inizio delle computazioni. Questa tendenza diminuisce progressivamente con l'avanzare delle epoche, fino ad arrivare ad effettuare variazioni minime.



Spread=1



Spread=10



Pertanto abbiamo provato ad analizzare l'andamento utilizzando goal superiori.

GOAL=150.000		
Spread	N° neuroni necessari	Performance (MSE sul test set)
0.00001	400	9.93E+06

0.0001	401	1.02E+07
0.001	403	9.72E+06
0.01	374	7.54E+06
0.1	257	4.18E+06
1	317	3.42E+07
10	549	La computazione viene arrestata prima di raggiungere il goal
100	549	
1000	549	
10 000	549	

Alzando il goal a 150.000 si può constatare un miglioramento rispetto all'andamento precedente. Lo spread migliore, sia per quanto riguarda il numero di nodi impiegati che le performance finali, passa da 0.01 a 0.1.

GOAL=200.000		
Spread	N° neuroni necessari	Performance (MSE sul test set)
0.00001	387	9.57E+06
0.0001	387	9.14E+06
0.001	386	9.35E+06
0.01	357	6.95E+06
0.1	232	3.69E+06
1	200	2.45E+06
10	549	La computazione viene arrestata prima di raggiungere il goal
100	549	
1000	549	
10 000	549	

Aumentando ulteriormente il goal il numero dei neuroni e l'errore sul test set continuano a calare. In questo caso la scelta migliore sarebbe uno spread uguale ad 1.

GOAL=250.000		
Spread	N° neuroni necessari	Performance (MSE sul test set)
0.00001	370	8.72E+06
0.0001	371	8.39E+06

0.001	373	8.55E+06
0.01	306	6.69E+06
0.1	219	2.84E+06
1	121	8.23E+05
10	396	6.39E+08
100	549	La computazione viene arrestata prima di raggiungere il goal
1000	549	
10 000	549	

Per quanto riguarda le prove effettuate con spread pari a 10 abbiamo constatato che:

- su 5 prove su 10 il training e il testing terminano fornendo i dati riportati in tabella (in media);
- sulle rimanenti 5 prove il training è stato arrestato automaticamente all'epoca n°549 senza aver raggiunto il goal (i valori non sono stati utilizzati ne calcolo delle medie riportate in tabella);

Per un goal pari a 250.000 il risultato migliore, per quanto riguarda sia il numeri di neuroni necessari che le performance sul test set, è quello con spread pari a 1.

Verifichiamo infine il comportamento con goal pari a 300.000

GOAL=300.000		
Spread	N° neuroni necessari	Performance (MSE sul test set)
0.00001	356	8.34E+06
0.0001	358	8.16E+06
0.001	357	8.35E+06
0.01	324	6.33E+06
0.1	199	2.16E+06
1	107	6.98E+05
10	182	9.14E+05
100	549	La computazione viene arrestata prima di raggiungere il goal
1000	549	
10000	549	

Infine abbiamo constatato che, per quanto riguarda le prove effettuate utilizzando un goal pari a 300.000, i risultati migliori si ottengono con spread pari a 1.

Ricapitolando, per meglio analizzare i dati finora ricavati, riportiamo nella tabella i valori relativi ai migliori risultati ottenuti per ogni goal esaminato. È importante notare che la valutazione sulla bontà di un risultato è effettuata basandosi sul valore delle performance, ovvero l'MSE sul test set.

Goal	Spread	N° neuroni necessari	Performance (MSE sul test set)
100.000	0.01	412	8.86E+06
150.000	0.1	257	4.18E+06
200.000	1	200	2.45E+06
250.000	1	121	8.23E+05
300.000	1	107	6.98E+05

In conclusione possiamo notare che all'aumentare del goal diminuisce non solo il numero dei neuroni necessari ma anche il MSE sul test set.

La soluzione migliore in assoluto è **goal=300.000** e **spread=1** e quindi scegliamo questa configurazione. Tuttavia, se non volessimo alzare troppo l'obiettivo, la configurazione **goal=250.000** e **spread=1** potrebbe essere un buon compromesso, non differendo di molto dalla soluzione precedente.

In seguito riportiamo lo script **day_RBF_train_test.m** pronto per essere eseguito con i parametri **goal=300.000** e **spread=1**. Preventivamente deve essere caricato il workspace **day_RBF_workspace**.

```
>>load('day_RF_workspace');
>>day_RBF_train_test;
```

Di seguito vediamo il contenuto di tale script.

```
%di_comodo_input e di_comodo_output contengono inizialmente input ed output. Poi
ci togliamo colonne fino a
%lasciarlo vuoto. train_set, test_set conterranno i dati di input,
%output_test, output_train quelli di train.
```

```
di_comodo_input=input15';
di_comodo_output=target';
```

```
% k è il numero di colonne -1... serve per l'intervallo della rand
k=730;
```

```
%assegno i valori a train_set ed output_train (549 sono circa il 70% dei
%campioni)
```

```
for i=1:549
    n=round(rand(1)*k+1);
    k=k-1;
    train_set(:,i)=di_comodo_input(:,n);
    di_comodo_input(:,n)=[];
    output_train(1,i)=di_comodo_output(1,n);
    di_comodo_output(:,n)=[];
end;
```

```

k=181;

%assegno i valori a test_set ed output_test (182 sono circa il 30% dei
%campioni)

for i=1:182
    n=round(rand(1)*k+1);
    k=k-1;
    test_set(:,i)=di_comodo_input(:,n);
    di_comodo_input(:,n)=[];
    output_test(1,i)=di_comodo_output(1,n);
    di_comodo_output(:,n)=[];
end;

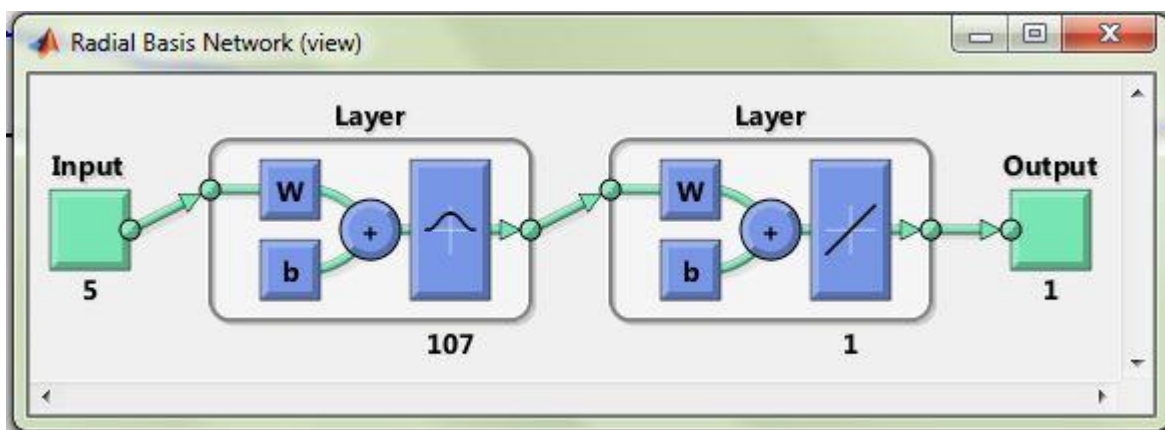
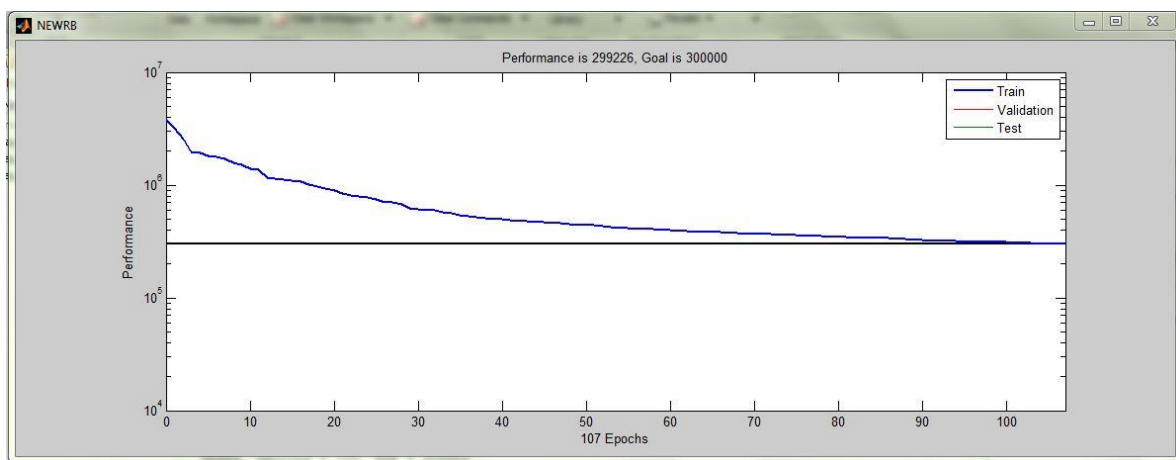
% Train the Network
net = newrb(train_set,output_train,300000,1);

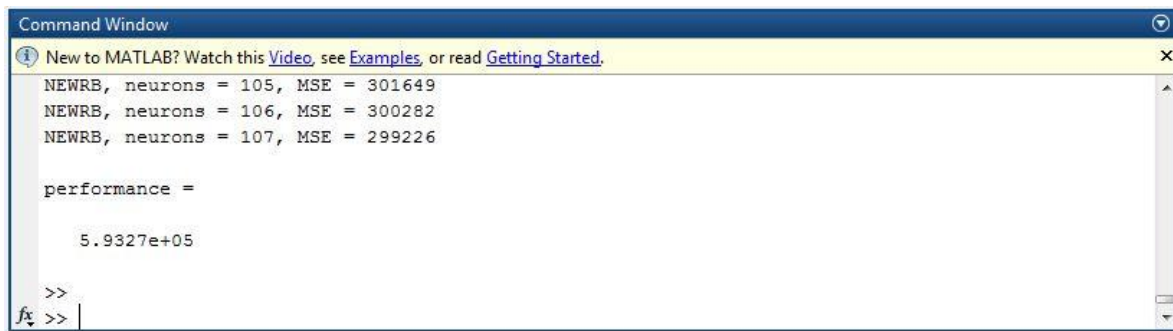
% Test the Network

y = net(test_set);
e = gsubtract(output_test,y);
performance = perform(net,output_test,y)

% View the Network
view(net)

```





```
Command Window
New to MATLAB? Watch this Video, see Examples, or read Getting Started.
NEURB, neurons = 105, MSE = 301649
NEURB, neurons = 106, MSE = 300282
NEURB, neurons = 107, MSE = 299226

performance =

    5.9327e+05

>>
fx >>
```

Rete RBF per Hour

Per questa parte facciamo riferimento a **HOURL_RBF_Final**. Poiché abbiamo visto che per la MLP Neural Network l'input migliore era input23, continuiamo a prenderlo in considerazione anche per sviluppare la RBF Neural Network.

```
load('HOURL_data');
```

Per l'allenamento e il test della rete è stato necessario creare manualmente i due set necessari, il train set e il test set (per le RBF Neural Networks non serve anche un validation set perché l'overlearning si verifica per una scelta inappropriata dello spread e non è legato alla presentazione dei campioni di input). Pertanto abbiamo creato lo script **HOURL_RBF_Divide_Sets** per suddividere i 17379 campioni in due set: il train set conterrà 13034 campioni scelti in maniera random (circa il 70%) e il test set 4345 campioni scelti sempre in maniera random (i rimanenti, circa il 30%).

```
%%we have to divide the 17379 samples of the input matrix (input23) into the
%%train test and the test set
app_x = input23';
app_t = target';
```

```
k = 17378;
```

```
for i=1:13034 %%for the train set we keep about 70% of samples
    n = round(rand(1)*k+1);
    k = k-1;
    train_set(:,i) = app_x(:,n);
    train_out(:,i) = app_t(:,n);
    app_x(:,n) = [];
    app_t(:,n) = [];
end
k = 4344;
for i=1:4345 %%for the test set we keep about 30% of samples
    n = round(rand(1)*k+1);
    k = k-1;
    test_set(:,i) = app_x(:,n);
    test_out(:,i) = app_t(:,n);
    app_x(:,n) = [];
    app_t(:,n) = [];
end
```

Dal momento che per le MLP Neural Networks avevamo visto che l'MSE medio era di 4.721E+03, decidiamo di prendere 1000 come goal, poiché è dello stesso ordine di grandezza (e questo ci permette di fare meglio i

confronti con le performance ottenute con la MLP Neural Network) ma un po' più basso per consentire di vedere se con RBF Neural Network riusciamo a raggiungere performance leggermente migliori.

Per capire quale sia il valore dello spread ottimale dobbiamo effettuare alcune prove; poniamo un numero massimo di neuroni nell'hidden layer, 200, poiché siamo interessati a vedere come si modificano le performance del sistema alle prime iterazioni (dopo un po' MSE si stabilizza e pertanto è inutile continuare l'allenamento della rete), come abbiamo visto per *Day*.

Avevamo pensato di normalizzare i dati da dare in ingresso alla rete, in quanto la scelta dello spread è legata al tipo di valori che arrivano in ingresso; tuttavia, come possiamo notare dalla tabella sottostante, l'MSE non riusciva a diminuire. Abbiamo quindi deciso di lasciare i dati intatti, anche perché comunque la feature i cui valori si modificano di più è hr (va da 0 a 23) ma le altre features non hanno valori di molto più piccoli. Inoltre, in questo modo possiamo meglio confrontare le performance della MLP Neural Network rispetto a quelle della rete RBF, avendole testate sugli stessi dati.

spread	mse train 0 nodi	mse train 50 nodi	mse train 100 nodi	mse train 150 nodi	mse train 200 nodi	mse test (medio)
0.01	3.275E+04	2.896E+04	2.665E+04	2.492E+04	2.349E+04	2.686E+04
	3.242E+04	2.806E+04	2.595E+04	2.420E+04	2.318E+04	
	3.261E+04	2.850E+04	2.644E+04	2.491E+04	2.337E+04	
	3.275E+04	2.884E+04	2.673E+04	2.524E+04	2.360E+04	
	3.322E+04	2.890E+04	2.655E+04	2.464E+04	2.339E+04	
0.1	3.283E+04	1.686E+04	1.045E+04	7.957E+03	6.665E+03	7.678E+03
	3.228E+04	1.433E+04	8.902E+03	7.128E+03	6.008E+03	
	3.294E+04	1.640E+04	1.105E+04	7.896E+03	6.417E+03	
	3.304E+04	1.396E+04	9.208E+03	7.093E+03	6.000E+03	
	3.300E+04	2.473E+04	1.869E+04	1.348E+04	1.205E+04	
1	3.280E+04	1.177E+04	8.947E+03	7.193E+03	6.958E+03	7.638E+03
	3.260E+04	8.541E+03	8.037E+03	7.890E+03	7.846E+03	
	3.240E+04	1.128E+04	7.770E+03	7.102E+03	7.101E+03	
	3.223E+04	9.785E+03	8.167E+03	7.998E+03	7.992E+03	
	3.262E+04	1.218E+04	9.021E+03	6.942E+03	6.913E+03	
10	3.289E+04	1.391E+04	1.390E+04	1.390E+04	1.390E+04	1.345E+04
	3.294E+04	1.419E+04	1.389E+04	1.385E+04	1.388E+04	
	3.316E+04	1.400E+04	1.400E+04	1.398E+04	1.398E+04	
	3.316E+04	1.428E+04	1.395E+04	1.388E+04	1.399E+04	
	3.307E+04	1.423E+04	1.424E+04	1.424E+04	1.424E+04	

Eseguiamo lo script **HOURL_RBF_Final** per ogni valore dello spread che vogliamo esaminare: abbiamo scelto di testare 0.1, 0.5, 1, 5 e 10 per coerenza con la dimensionalità dei dati in ingresso.

```
goal = 1000;
spread = 1;           %%changed to find the best value
max_neurons = 100;
for i=1:5              %%we check the performances of the spread choosen 5 times in
a row
    HOURL_RBF_Divide_Sets;
    HOURL_RBF_simple_script;
```

end

Questo è il contenuto dello script **HOUR_RBF_simple_script**.

```
%%train and test sets have been previously created

%%Create and train the network
net = newrb(train_set,train_out,goal,spread,max_neurons);

% Test the Network
y = net(test_set);
error = gsubtract(test_out,y);
performance = perform(net,test_out,y)

% View the Network
view(net)

% Plots
% Uncomment these lines to enable various plots.
%figure, plotperform(tr)
%figure, plottrainstate(tr)
%figure, plotfit(net,x,t)
%figure, plotregression(t,y)
%figure, ploterrhist(e)
```

Questi sono i risultati dei test.

spread	mse train 0 nodi	mse train 50 nodi	mse train 100 nodi	mse train 150 nodi	mse train 200 nodi	mse test (medio)
0.1	3.261E+04	1.959E+04	1.469E+04	1.205E+04	1.020E+04	1.120E+04
	3.275E+05	1.875E+04	1.424E+04	1.153E+04	9.380E+03	
	3.322E+04	1.898E+04	1.438E+04	1.197E+04	1.011E+04	
	3.289E+04	1.897E+04	1.423E+04	1.170E+04	9.909E+03	
	3.294E+04	1.940E+04	1.500E+04	1.242E+04	1.047E+04	
0.5	3.280E+04	1.472E+04	1.032E+04	7.581E+03	5.883E+03	6.370E+03
	3.254E+04	1.494E+04	1.051E+04	7.493E+03	5.989E+03	
	3.320E+04	1.458E+04	1.014E+04	7.338E+03	5.895E+03	
	3.303E+04	1.446E+04	1.020E+04	7.133E+03	5.663E+03	
	3.311E+04	1.539E+04	1.030E+04	7.940E+03	6.171E+03	
1	3.329E+04	4.660E+03	3.637E+03	3.370E+03	3.236E+03	3.386E+03
	3.320E+04	4.356E+03	3.718E+03	3.485E+03	3.323E+03	
	3.298E+04	4.722E+03	3.712E+03	3.421E+03	3.314E+03	
	3.285E+04	4.539E+03	3.643E+03	3.415E+03	3.278E+03	
	3.301E+04	4.651E+03	3.555E+03	3.309E+03	3.199E+03	
5	3.294E+04	6.878E+03	5.270E+03	4.158E+03	3.932E+03	3.832E+03
	3.304E+04	7.102E+03	5.001E+03	4.112E+03	3.973E+03	
	3.300E+04	6.542E+03	4.472E+03	4.235E+03	4.084E+03	
	3.275E+04	6.688E+03	4.900E+03	4.186E+03	3.974E+03	
	3.242E+04	6.626E+03	4.887E+03	4.074E+03	3.968E+03	
10	3.262E+04	1.039E+04	6.955E+03	6.263E+03	5.938E+03	6.216E+03

3.223E+04	9.751E+03	6.980E+03	6.614E+03	6.162E+03
3.283E+04	1.022E+04	7.228E+03	6.999E+03	6.311E+03
3.228E+04	1.077E+04	7.028E+03	6.438E+03	5.806E+03
3.294E+04	8.101E+03	6.323E+03	5.885E+03	5.874E+03

Come è possibile vedere, i risultati migliori si ottengono per gli spread 1 e 5: proviamo quindi a vedere cosa accade per valori compresi tra 1 e 5 (2, 3 e 4).

spread	mse train 0 nodi	mse train 50 nodi	mse train 100 nodi	mse train 150 nodi	mse train 200 nodi	mse test
2	3.262E+04	4.753E+03	3.739E+03	3.478E+03	3.322E+03	3.522E+03
	3.223E+04	4.610E+03	3.736E+03	3.682E+03	3.438E+03	
	3.283E+04	4.669E+03	3.693E+03	3.509E+03	3.294E+03	
	3.228E+04	4.202E+03	3.624E+03	3.423E+03	3.384E+03	
	3.294E+04	4.477E+03	3.596E+03	3.325E+03	3.181E+03	
3	3.304E+04	4.897E+03	4.116E+03	3.768E+03	3.612E+03	3.483E+03
	3.300E+04	4.543E+03	3.842E+03	3.583E+03	3.473E+03	
	3.275E+04	5.294E+03	4.139E+03	3.744E+03	3.530E+03	
	3.242E+04	5.040E+03	4.166E+03	3.835E+03	3.603E+03	
	3.261E+04	4.802E+03	4.031E+03	3.705E+03	3.613E+03	
4	3.275E+04	5.828E+03	4.406E+03	3.789E+03	3.726E+03	3.608E+03
	3.322E+04	6.265E+03	4.426E+03	4.028E+03	3.997E+03	
	3.289E+04	5.686E+03	4.136E+03	4.008E+03	3.622E+03	
	3.294E+04	6.101E+03	4.273E+03	4.020E+03	3.846E+03	
	3.316E+04	5.980E+03	4.221E+03	3.875E+03	3.655E+03	

All'aumentare dello spread le performance degradano leggermente; proviamo quindi a vedere cosa accade con valori molto vicini a uno (0.9 e 0.8).

spread	mse train 0 nodi	mse train 50 nodi	mse train 100 nodi	mse train 150 nodi	mse train 200 nodi	mse test
0.9	3.314E+04	5.108E+03	3.894E+03	3.477E+03	3.312E+03	3.383E+03
	3.307E+04	4.931E+03	3.755E+03	3.395E+03	3.231E+03	
	3.321E+04	5.359E+03	3.901E+03	3.534E+03	3.312E+03	
	3.237E+04	5.059E+03	3.835E+03	3.392E+03	3.190E+03	
	3.314E+04	5.157E+03	3.963E+03	3.590E+03	3.356E+03	
0.8	3.268E+04	6.575E+03	4.330E+03	3.638E+03	3.258E+03	3.804E+03
	3.278E+04	6.413E+03	4.154E+03	3.660E+03	3.374E+03	
	3.339E+04	6.121E+03	4.318E+03	3.671E+03	3.452E+03	
	3.279E+04	6.005E+03	4.466E+03	3.752E+03	3.393E+03	
	3.297E+04	7.021E+03	4.565E+03	3.803E+03	3.450E+03	

Tra il valore 0.9 e 1 non c'è differenza in fatto di performance quindi possiamo scegliere entrambi i valori: per semplicità scegliamo lo spread 1. Inoltre possiamo notare che da 100 nodi in poi l'errore non varia più in maniera significativa: pertanto possiamo porre 100 come limite al numero massimo di nodi.

Modello di fitting Fuzzy

Modello Fuzzy-Mamdani per Day

Nella parte relativa alle MLP abbiamo selezionato le seguenti features:

- season
- yr
- weekday
- weathersit
- temp

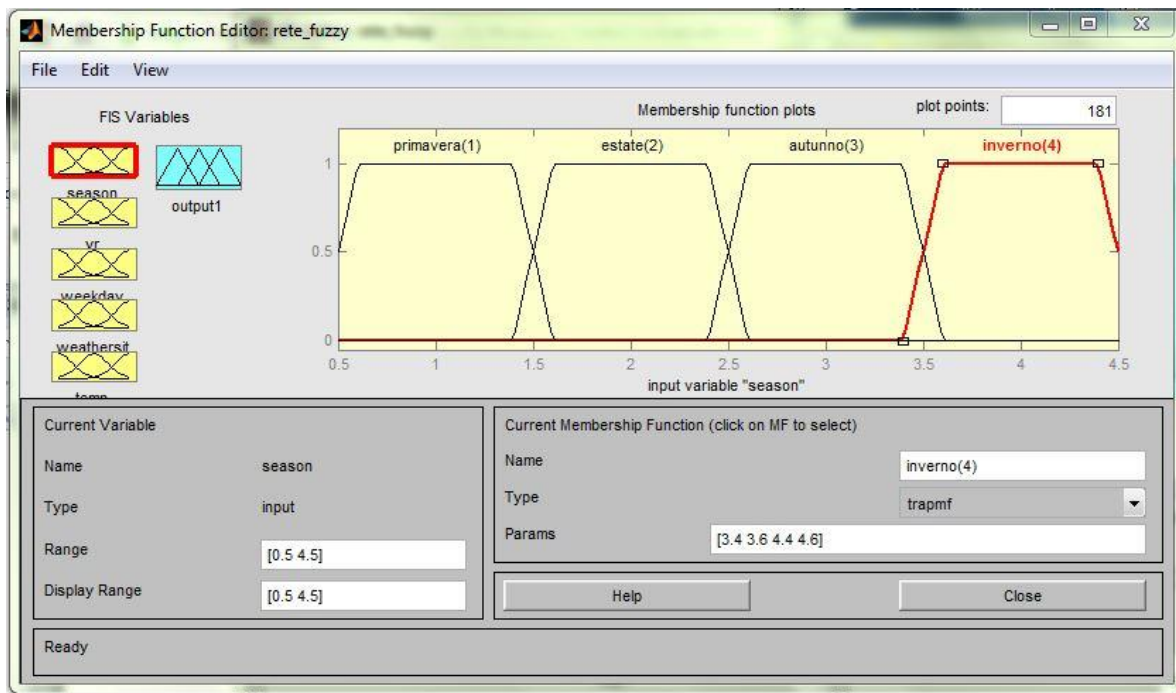
Per ognuna di esse, e per il count, abbiamo stabilito una membership function.

Scelta delle membership function

Quasi la totalità delle features selezionate appartengono a insiemi crisp. Per rimanere coerenti con il significato intrinseco delle caratteristiche, che non erano state pensate per essere considerate come insiemi fuzzy, abbiamo ritenuto opportuno implementare delle membership functions in modo che i valori di esse ricadessero nel core del fuzzy set stesso. Ciò non nuoce alla possibilità di poterle rendere, in futuro, un insieme fuzzy inserendo valori intermedi, ovviamente tenendo conto delle funzioni di appartenenza definite in questo momento.

☀ **season:** 4 stagioni con un intervallo di rappresentazione pari a [0.5 4.5]

- ⇒ spring (1)
 - type=trapmf
 - params= [0.4 0.6 1.4 1.6]
- ⇒ summer (2)
 - type=trapmf
 - params= [1.4 1.6 2.4 2.6]
- ⇒ fall (3)
 - type=trapmf
 - params= [2.4 2.6 3.4 3.6]
- ⇒ winter (4)
 - type=trapmf
 - params= [3.4 3.6 4.4 4.6]



☀ **yr**: 2 anni con un intervallo di rappresentazione pari a [-0.5 1.5]



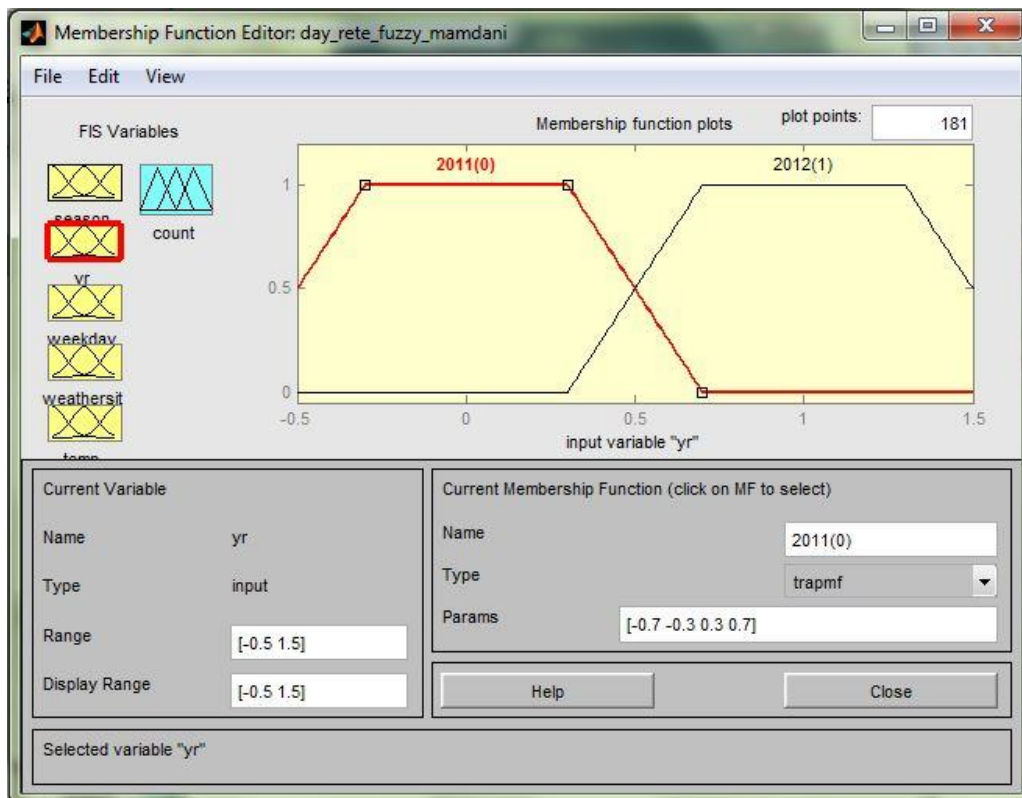
anno0 (0)

- type=trapmf
- params= [-0.7 -0.3 0.3 0.7]



anno1 (1)

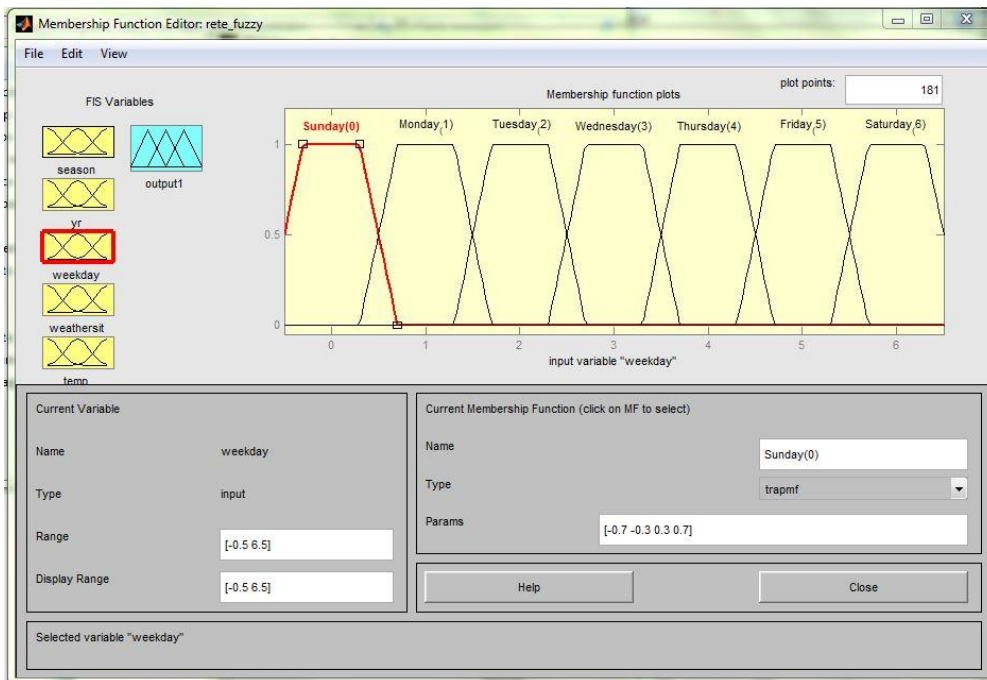
- type=trapmf
- params= [0.3 0.7 1.3 1.7]



☀ **weekday**: 7 giorni della settimana con un intervallo di rappr. pari a [-0.5 6.5]

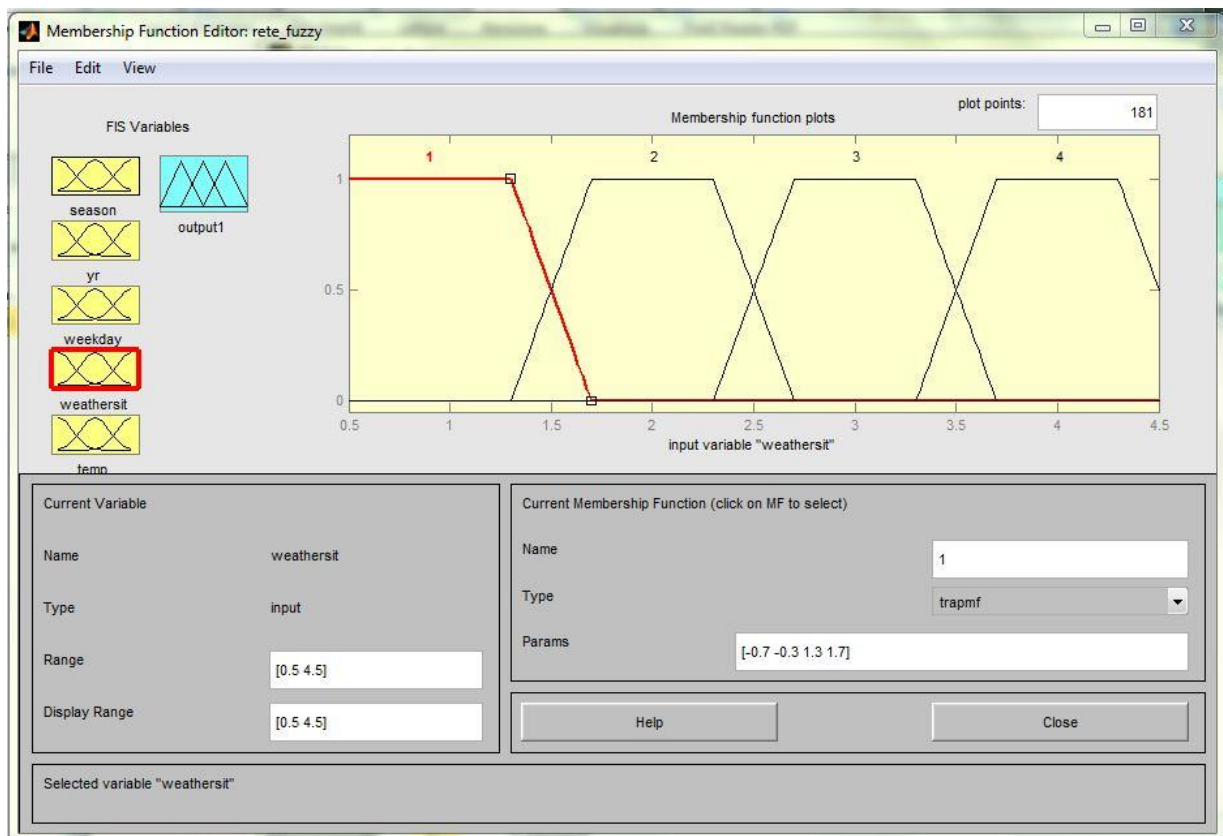
- ⇒ Sunday (0)
 - type=trapmf
 - params= [-0.7 -0.3 0.3 0.7]
- ⇒ Monday (1)
 - type=trapmf
 - params= [0.3 0.7 1.3 1.7]
- ⇒ Tuesday (2)
 - type=trapmf
 - params= [1.3 1.7 2.3 2.7]
- ⇒ Wednesday (3)
 - type=trapmf
 - params= [2.3 2.7 3.3 3.7]
- ⇒ Thursday (4)
 - type=trapmf
 - params= [3.3 3.7 4.3 4.7]
- ⇒ Friday (5)
 - type=trapmf
 - params= [4.3 4.7 5.3 5.7]
- ⇒ Saturday (6)
 - type=trapmf

- params= [5.3 5.7 6.3 6.7]



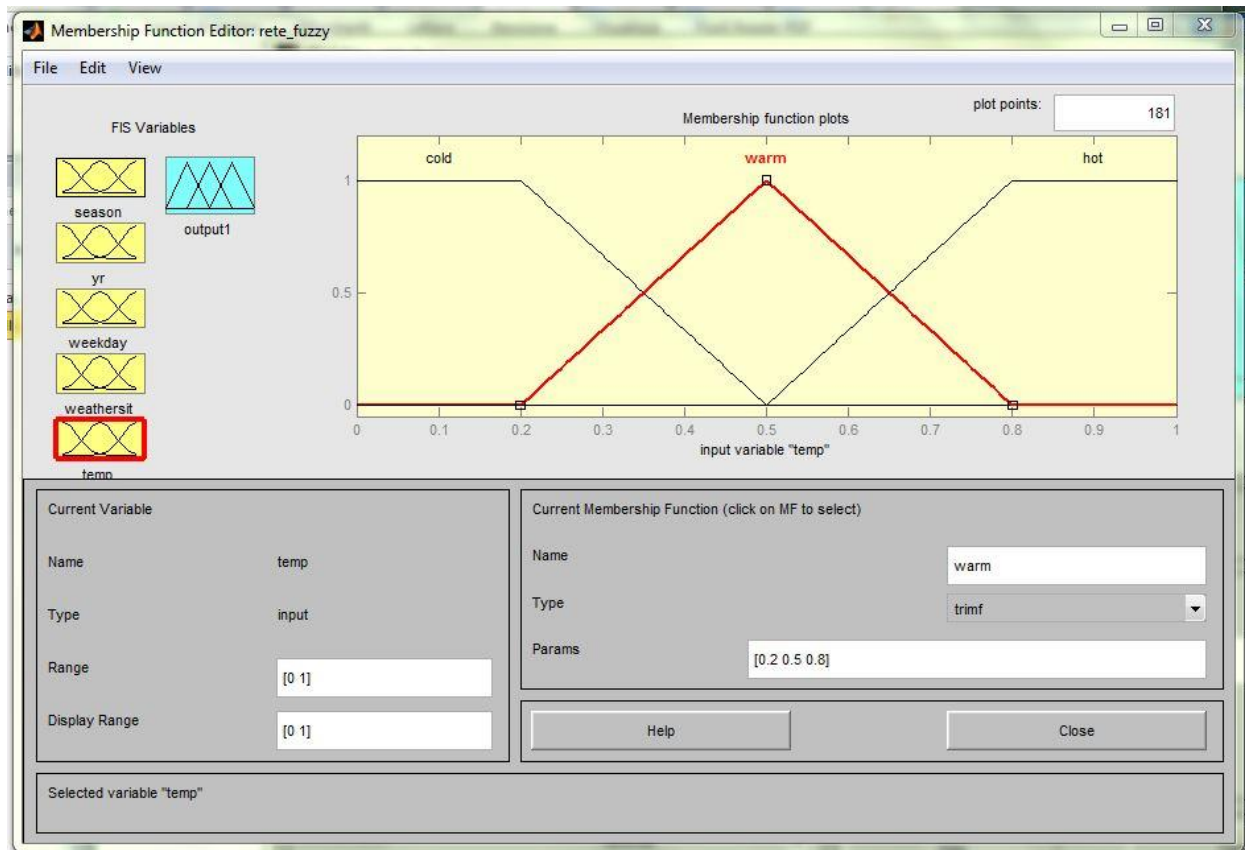
☀ **whethersit:** 4 fasce di condizioni climatiche dalla migliore (1) alla peggiore (4) con un intervallo di rappresentazione pari a [0.5 4.5]

- ⇒ 1
 - type=trapmf
 - params= [-0.7 -0.3 1.3 1.7]
- ⇒ 2
 - type=trapmf
 - params= [1.3 1.7 2.3 2.7]
- ⇒ 3
 - type=trapmf
 - params= [2.3 2.7 3.3 3.7]
- ⇒ 4
 - type=trapmf
 - params= [3.3 3.7 4.3 4.7]



☀ **temp**: 3 fasce di temperatura con un intervallo di rappresentazione pari a [0 1]

- ⇒ cold
 - type=trapmf
 - params= [0 0 0.2 0.5]
- ⇒ warm
 - type=trimf
 - params= [0.2 0.5 0.8]
- ⇒ hot
 - type=trapmf
 - params= [0.5 0.8 1 1]



Rimane da creare la membership function per il count (l'output).

Per quanto riguarda i dati a disposizione il massimo numero di bici noleggiate è 8714 mentre il minimo numero di bici noleggiate è pari a 22.

Nella scelta dei parametri per le nostre membership function prendiamo come valore minimo 0 e come valore massimo 10.000. La scelta di sovrastimare il numero massimo di bici noleggiabili ci permette di utilizzare il sistema sviluppato anche in caso di piccole espansioni.

☀ **count:** intervallo di rappresentazione pari a [0 10.000]



low

- type=trapmf
- params= [0 0 500 3500]



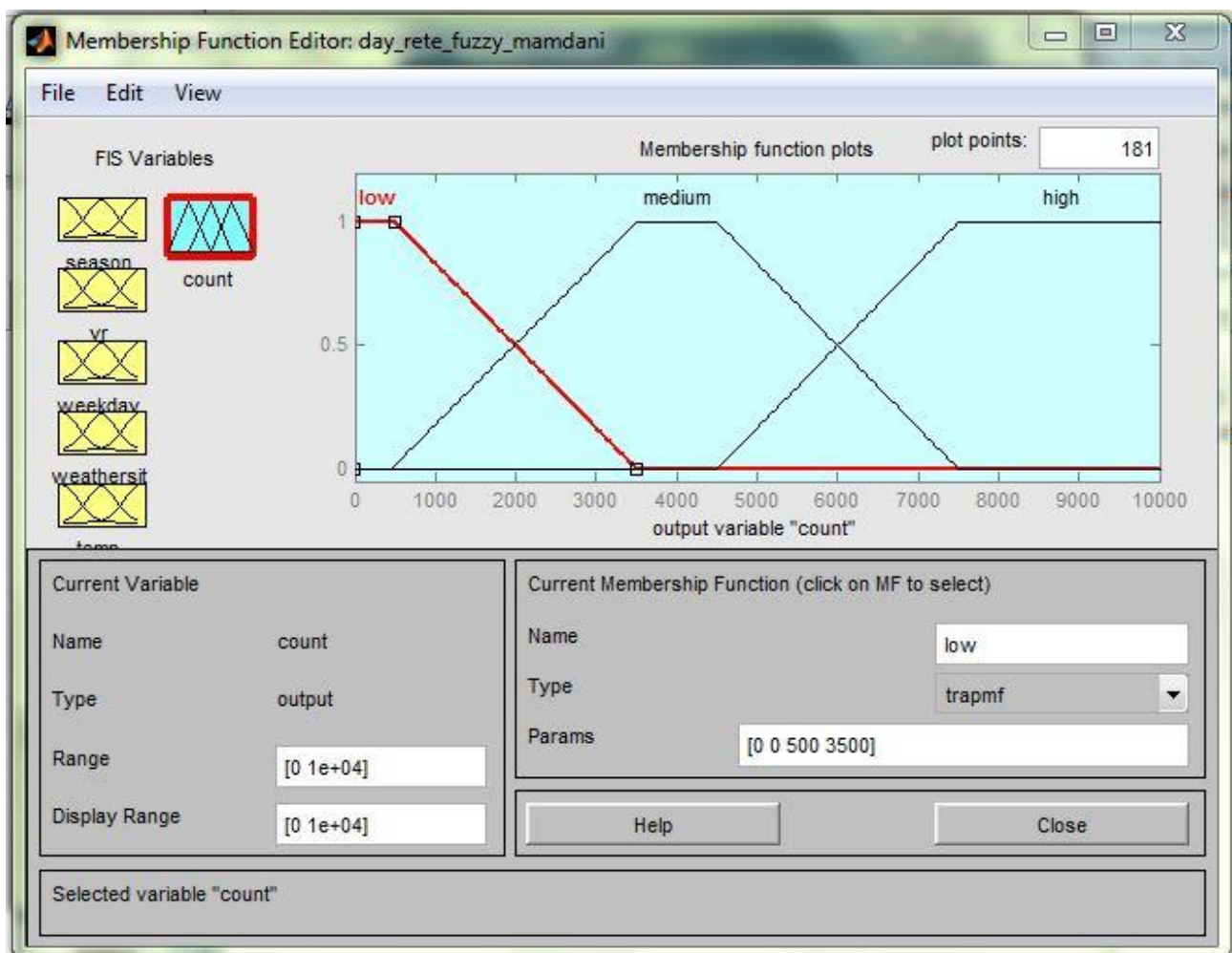
medium

- type=trapmf
- params= [450 3500 4500 7500]



high

- type=trapmf
- params= [4500 7500 10000 10000]



Scelta delle regole

Analizzando il file day abbiamo individuato le seguenti relazioni tra le feature ed l'output.

If weathersit=4 then count=low
If season!=4 & weathersit=3 then low
If season=4 & weathersit=1 & temp=hot then count=high
If season=4 & weathersit=1 & temp=warm then count=medium
If season=4 & weathersit=1 & temp=cold then count=low
If season=4 & weathersit=2 & temp=hot then count=high
If season=4 & weathersit=2 & temp=warm then count=medium
If season=4 & weathersit=2 & temp=cold then count=low
If season=4 & weathersit=3 & temp=hot then count=medium
If season=4 & weathersit=3 & temp=cold then count=low
If season=3 & weathersit=1 & temp=warm then count=high

If season=3 & weathersit=1 & temp=hot then count=medium
If season=3 & weathersit=2 & temp=hot then count=high
If season=3 & weathersit=2 & temp=warm then count=medium
If season=2 & weathersit=1 & temp=warm then count=high
If season=2 & weathersit=1 & temp=cold then count=medium
If season=2 & weathersit=2 & temp=hot then count=high
If season=2 & weathersit=2 & temp=warm then count=medium
If season=2 & weathersit=2 & temp=cold then count=low
If season=1 & weathersit=1 & temp=warm then count=medium
If season=1 & weathersit=1 & temp=cold then count=low
If season=1 & weathersit=2 & temp=hot then count=high
If season=1 & weathersit=2 & temp=warm then count=medium
If season=1 & weathersit=2 & temp=cold then count=low

Le regole evidenziate possono essere combinate per formare un numero minore di regole equivalenti.

If season!=3 & weathersit=2 & temp=hot then count=high
If season!=3 & weathersit=2 & temp=warm then count=medium
If season!=3 & weathersit=2 & temp=cold then count=low

Valutazione delle prestazioni

Per valutare le prestazioni della nostra rete fuzzy abbiamo preparato lo script **day_fuzzy_mamdani_testset_con_media.m**.

Preventivamente è necessario caricare il workspace **day_fuzzy_mamdani_workspace.mat** che contiene le strutture dati necessarie all'esecuzione dello script. La rete fuzzy implementata è presente sia nello workspace che nel file allegato **day_rete_fuzzy_mamdani.fis**.

```
>> load('day_fuzzy_mamdani_workspace');
>> day_fuzzy_mamdani_testset_con_media;
```

```
media=0;
```

```
for l=1:100
```

```
    %di_comodo_input e di_comodo_output contengono inizialmente input ed output. Poi ci
    %tolgiamo colonne fino a
    %lasciarlo vuoto.
```

```
di_comodo_input=input15;
di_comodo_output=target;
```

```
% k è il numero di colonne -1... serve per l'intervallo della rand
k=730;
```

```
for i=1:400
    n=round(rand(1)*k+1);
    k=k-1;
    example_feature(i,:)=di_comodo_input(n,:);
    di_comodo_input(n,:)=[];
    example_output(i,1)=di_comodo_output(n,1);
    di_comodo_output(n,:)=[];
end;
```

```
% Test the Network
output= evalfis(example_feature, rete_fuzzy);
```

```
%errore su ogni singolo esempio
e = gsubtract(example_output, output);
```

```
%mi manca solo il MSE
MSE=0;
```

```
for i=1:400;
    MSE=MSE+e(i,1)*e(i,1);
end
```

```
MSE=MSE/400;
```

```
media=media+MSE;
```

```
MSE    %stampa a video il MSE
```

```
end;
```

```
media=media/100    %aggiorna la media e la visualizza sulla command window
```

Il codice contiene un ciclo che per 100 volte estrae in modo totalmente random un sottoinsieme delle features disponibili. Questi valori vengono messi all'interno della matrice `example_features`, mentre i relativi output vengono messi nell'array `example_output`.

```
>>evalfis;
```

Tramite il comando `evalfis` possiamo testare la rete ed usare la differenza tra i valori ottenuti e quelli desiderati per calcolarne il MSE.

Alla fine delle iterazioni `media` contiene la media di tutti i 100 MSE ottenuti. Dopo aver eseguito lo script abbiamo ottenuto i seguenti MSE

2.5575E+06	2.5629E+06	2.6161E+06	2.7635E+06	2.7681E+06
2.7226E+06	2.6383E+06	2.7702E+06	2.7435E+06	2.6494E+06
2.7505E+06	2.6512E+06	2.6120E+06	2.8567E+06	2.8360E+06
2.7882E+06	2.8733E+06	2.4990E+06	2.5393E+06	2.6656E+06
2.5139E+06	2.9662E+06	2.7045E+06	2.7017E+06	2.7255E+06
2.5813E+06	2.9481E+06	2.7969E+06	2.6980E+06	2.7800E+06
3.0341E+06	2.5562E+06	2.9766E+06	2.9266E+06	2.8521E+06
2.7814E+06	2.6718E+06	2.7818E+06	2.8012E+06	2.7124E+06
2.7234E+06	2.8010E+06	2.8836E+06	2.6268E+06	2.6404E+06
2.7634E+06	2.5020E+06	2.6185E+06	2.9082E+06	2.9329E+06
2.8200E+06	2.7468E+06	2.5829E+06	2.8062E+06	2.7581E+06
2.4581E+06	2.6506E+06	2.7081E+06	2.7302E+06	2.9175E+06
2.7115E+06	2.8299E+06	2.6376E+06	2.8425E+06	2.5757E+06
2.9387E+06	2.7082E+06	2.9278E+06	2.7007E+06	2.7295E+06
2.7617E+06	2.6957E+06	2.7201E+06	2.8277E+06	2.7227E+06
2.8459E+06	2.8188E+06	2.6919E+06	2.8394E+06	2.8567E+06
2.6688E+06	2.6582E+06	2.8192E+06	2.5998E+06	2.7083E+06
3.0591E+06	2.8969E+06	2.7857E+06	2.7704E+06	2.5585E+06
2.9801E+06	2.9576E+06	2.9079E+06	2.7215E+06	2.7227E+06
2.5896E+06	2.8485E+06	2.6223E+06	2.7153E+06	2.7603E+06

La media finale è **2.7496E+06**.

Modello Fuzzy-Mamdani per Hour

Nella parte relativa alle MLP abbiamo selezionato le seguenti features:

- season
- yr
- hour
- workingday
- temp

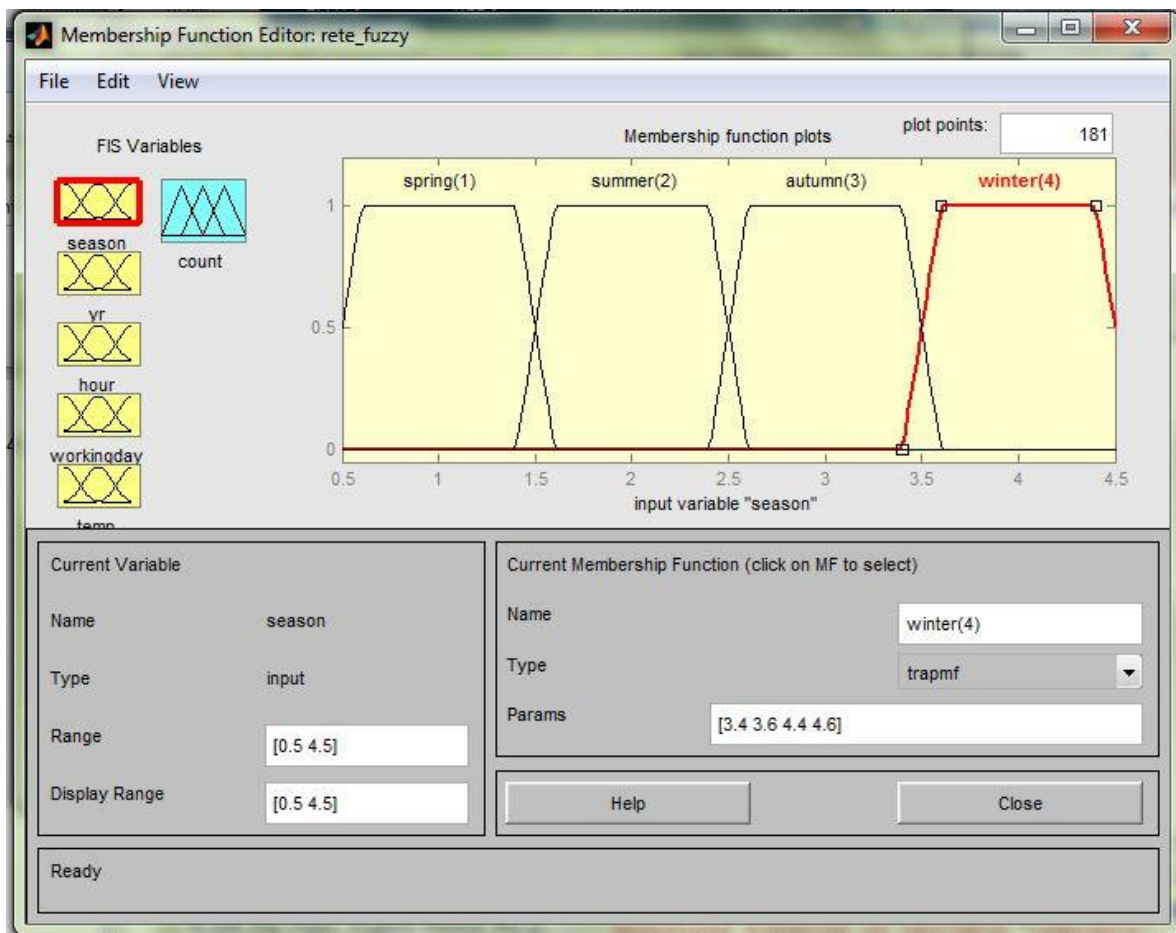
Per ognuna di esse, e per il count, abbiamo stabilito le seguenti membership function.

Scelta delle membership function

Quasi la totalità delle features selezionate appartengono a insiemi crisp. Per rimanere coerenti con il significato intrinseco delle caratteristiche, che non erano state pensate per essere considerate come insiemi fuzzy, abbiamo ritenuto opportuno implementare delle membership functions in modo che i valori di esse ricadessero nel core del fuzzy set stesso. Ciò non nuoce alla possibilità di poterle rendere, in futuro, un insieme fuzzy inserendo valori intermedi, ovviamente tenendo conto delle funzioni di appartenenza definite in questo momento.

☀ **season:** 4 stagioni con un intervallo di rappresentazione pari a [0.5 4.5]

- ⇒ spring (1)
 - type=trapmf
 - params= [0.4 0.6 1.4 1.6]
- ⇒ summer (2)
 - type=trapmf
 - params= [1.4 1.6 2.4 2.6]
- ⇒ fall (3)
 - type=trapmf
 - params= [2.4 2.6 3.4 3.6]
- ⇒ winter (4)
 - type=trapmf
 - params= [3.4 3.6 4.4 4.6]



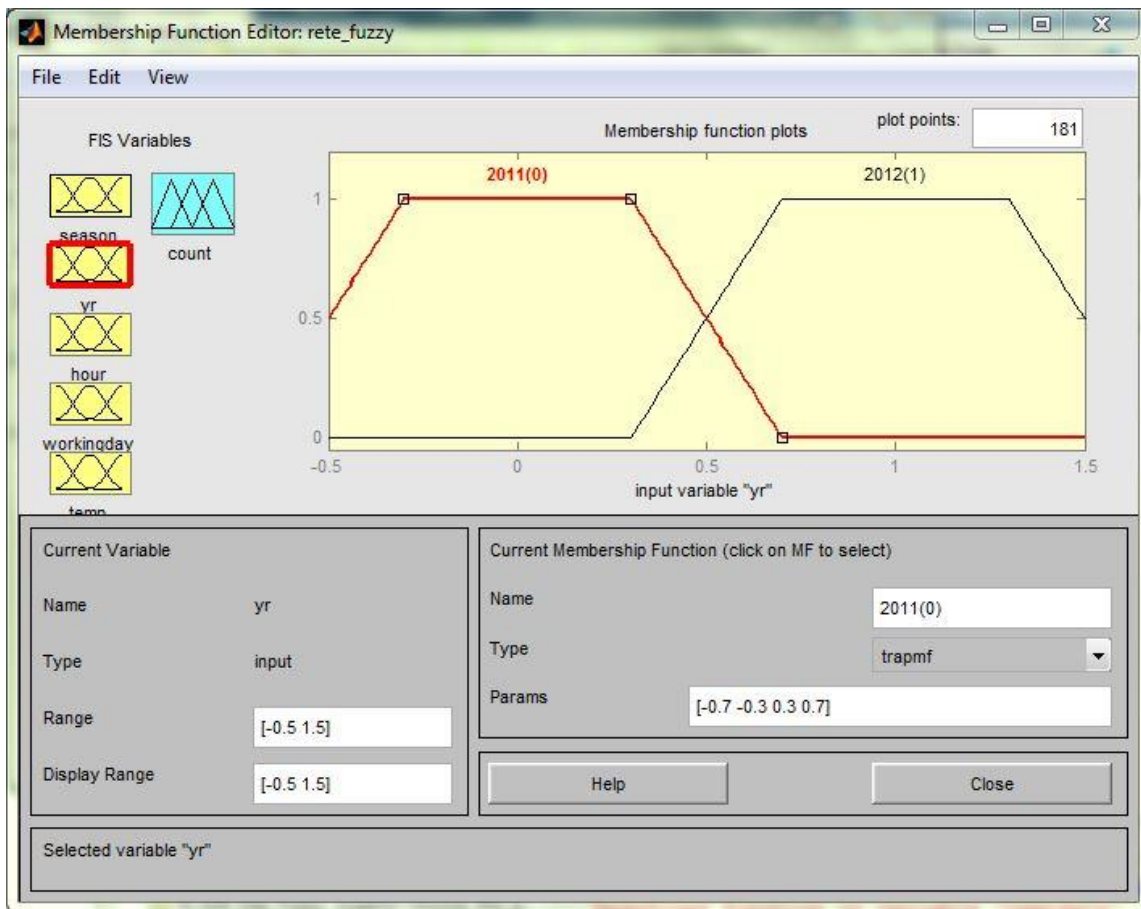
☀ **yr:** 2 anni con un intervallo di rappresentazione pari a [-0.5 1.5]

- ⇒ 2011 (0)
 - type=trapmf
 - params= [-0.7 -0.3 0.3 0.7]



2012 (1)

- type=trapmf
- params= [0.3 0.7 1.3 1.7]



★ **hour:** 0-23 ore con un intervallo di rappresentazione pari a [0 24]



Night

- type=trapmf
- params= [0 0 5 7]



Morning

- type=trapmf
- params= [5 7 11 13]



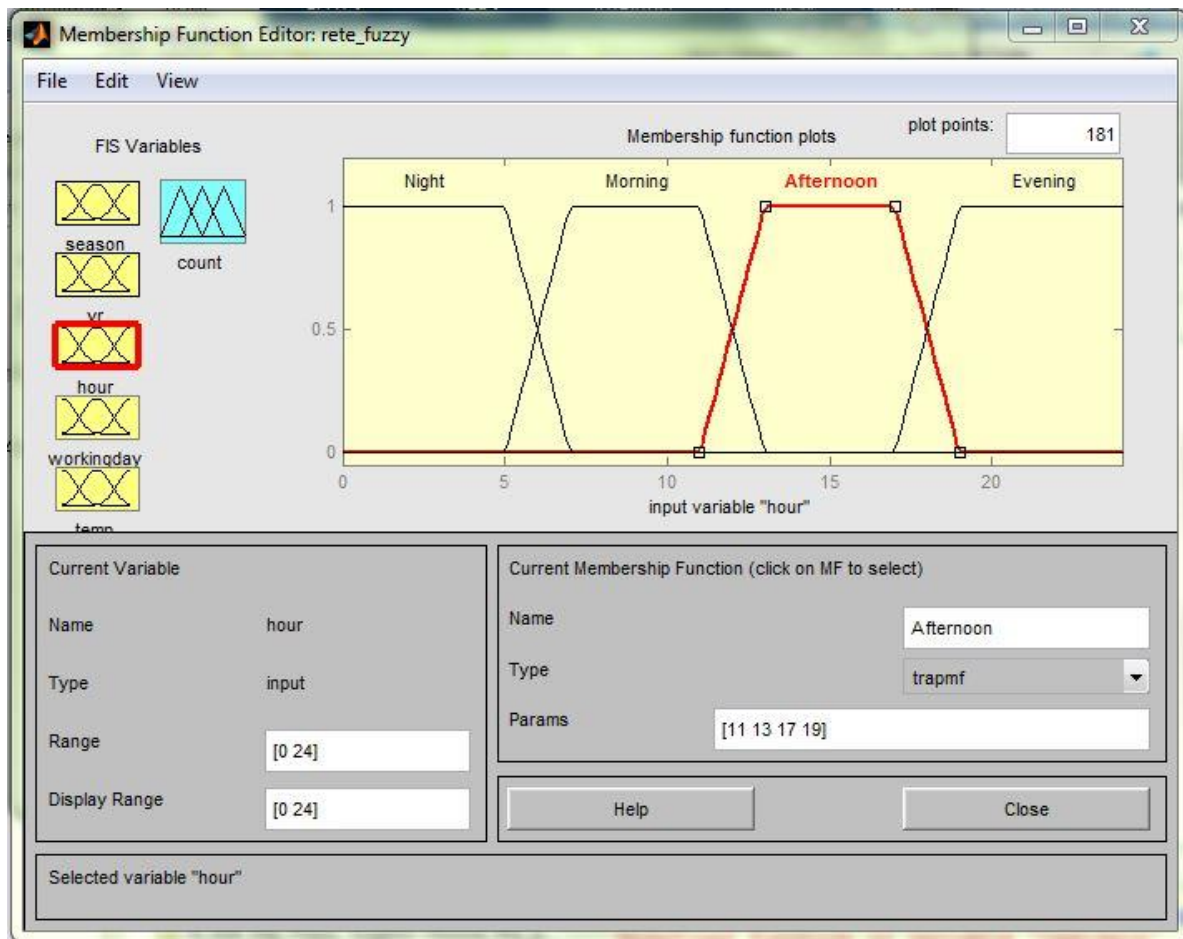
Afternoon

- type=trapmf
- params= [11 13 17 19]



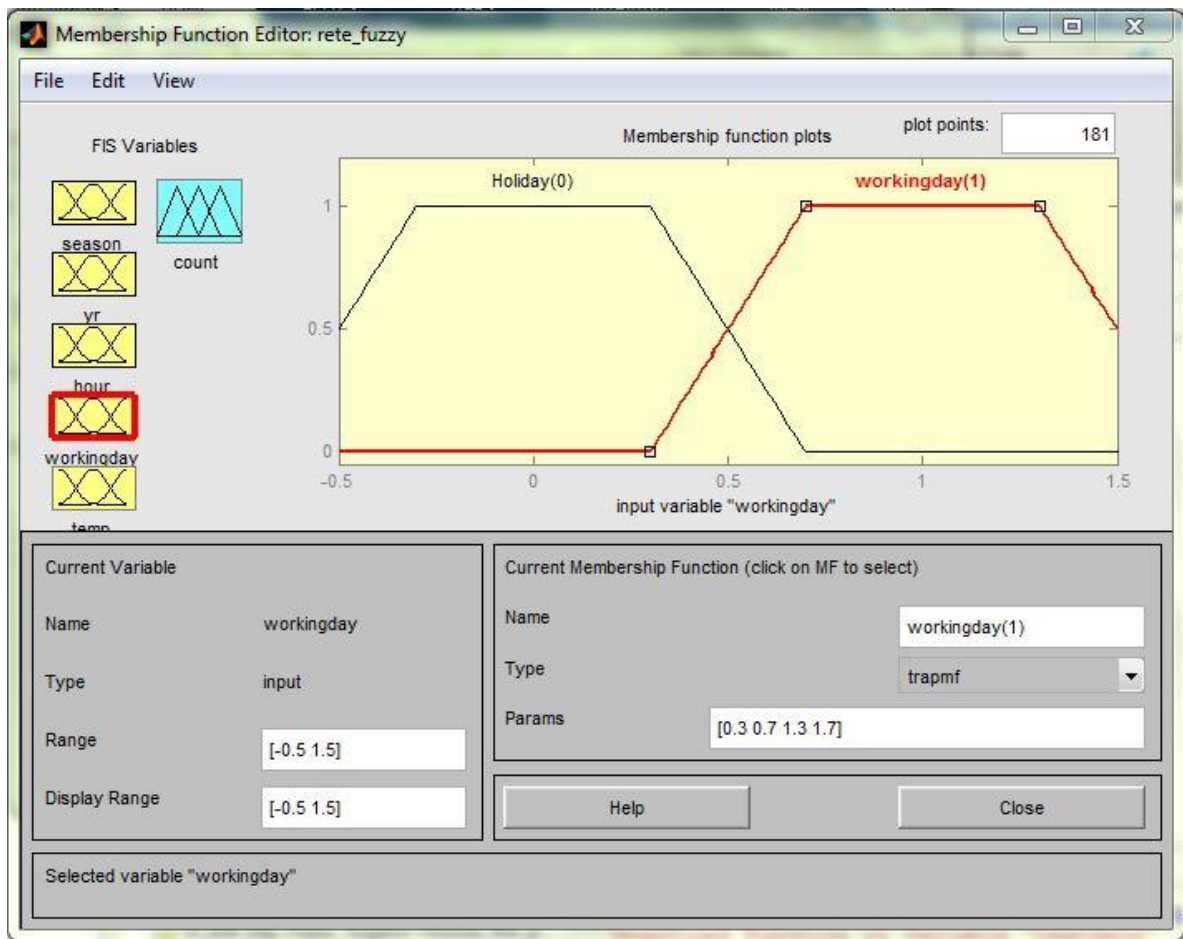
Evening

- type=trapmf
- params= [17 19 24 24]



☀ **workingday:** 0-1 con un intervallo di rappresentazione pari a [-0.5 1.5]

- ⇒ holiday (0)
 - type=trapmf
 - params= [-0.7 -0.3 0.3 0.7]
- ⇒ workingday(1)
 - type=trapmf
 - params= [0.3 0.7 1.3 1.7]



☀ **temp**: 3 fasce di temperatura con un intervallo di rappresentazione pari a [0 1]



cold

- type=trapmf
- params= [0 0 0.2 0.5]



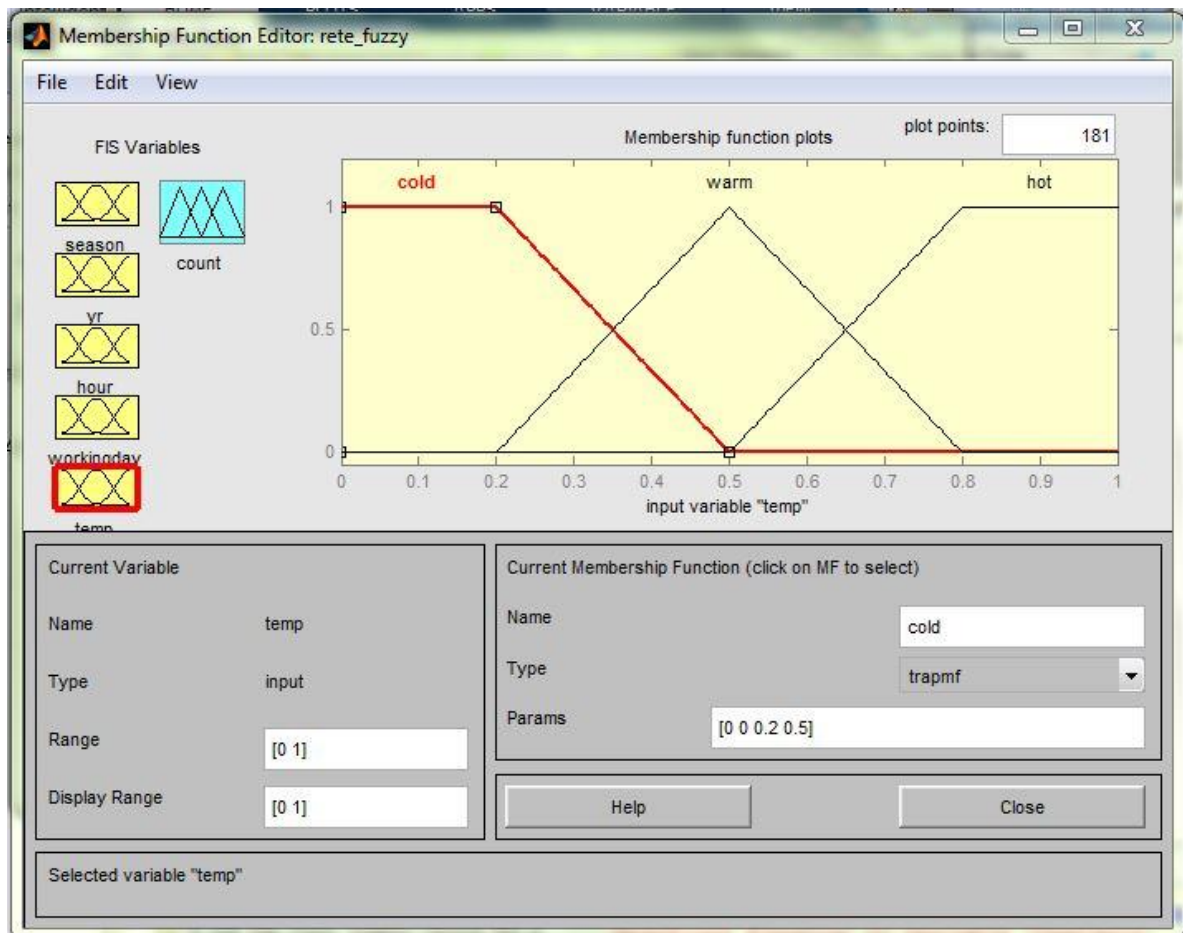
warm

- type=trimf
- params= [0.2 0.5 0.8]



hot

- type=trapmf
- params= [0.5 0.8 1 1]



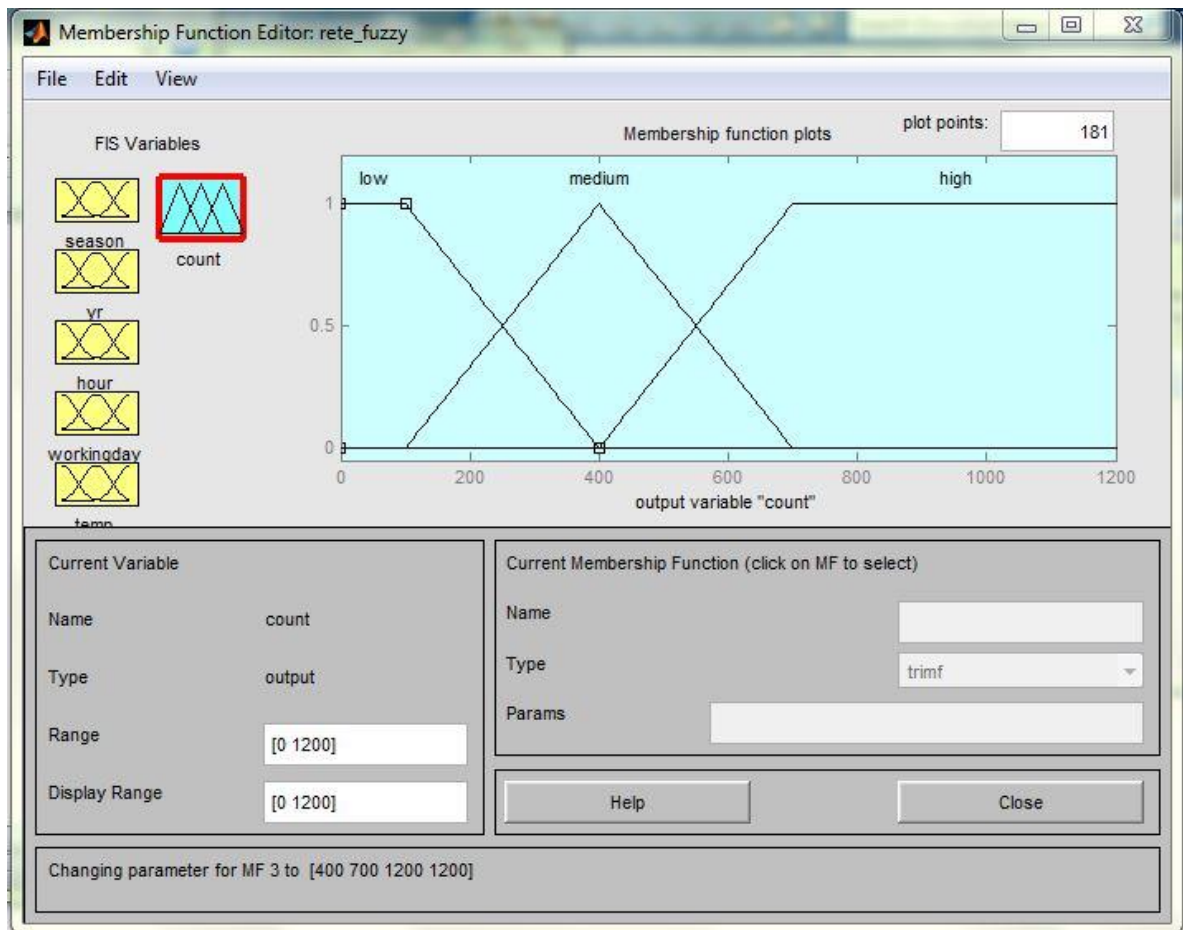
Rimane da creare la membership function per il count (l'output).

Per quanto riguarda i dati a disposizione il massimo numero di bici realmente noleggiate è 977 mentre il minimo numero di bici realmente noleggiate è pari a 1.

Nella scelta dei parametri per le nostre membership function prendiamo come valore minimo 0 e come valore massimo 1.200. La scelta di sovrastimare il numero massimo di bici noleggiabili ci permette di utilizzare il sistema sviluppato anche nel caso in cui, in un prossimo futuro, avvengano piccole espansioni.

★ **count:** intervallo di rappresentazione pari a [0 1200]

- ⇒ low
 - type=trapmf
 - params= [0 0 100 400]
- ⇒ medium
 - type=trimf
 - params= [100 400 700]
- ⇒ high
 - type=trapmf
 - params= [400 700 1200 1200]



Scelta delle regole

Analizzando il file hour abbiamo individuato le seguenti relazioni tra le features ed l'output desiderato (count).

If season=4 & hour=night then count=low
If season=4 & hour=morning & temp=warm then count=low
If season=4 & hour=morning & temp=cold then count=high
If season=4 & hour=afternoon & temp=hot then count=high
If season=4 & hour=afternoon & temp=warm then count=medium
If season=4 & hour=afternoon & temp=cold then count=low
If season=4 & hour=evening & temp=hot then count=high
If season=4 & hour=evening & temp=warm then count=medium
If season=4 & hour=evening & temp=cold then count=low
If season=3 & hour=night then count=low
If season=3 & hour=morning & temp=warm then count=high

If season=3 &hour=morning & temp=hot then count=low
If season=3 &hour=afternoon & temp=hot then count=high
If season=3 &hour=afternoon & temp=warm then count=low
If season=3 &hour=evening & temp=warm then count=high
If season=3 &hour=evening & temp=hot then count=low
If season=2 &hour=night then count=low
If season=2 &hour=morning & temp=cold then count=low
If season=2 &hour=morning & temp!=cold then count=medium
If season=2 &hour=afternoon & temp=cold then count=low
If season=2 &hour=afternoon & temp=hot then count=high
If season=2 &hour=evening & temp=warm then count=high
If season=2 &hour=evening & temp=cold then count=low
If season=1 &hour=night then count=low
If season=1 &hour=morning & temp=warm then count=high
If season=1 &hour=morning & temp=cold then count=low
If season=1 &hour=afternoon & temp=cold then count=low
If season=1 &hour=afternoon & temp=warm then count=high
If season=1 &hour=evening & temp=warm then count=high
If season=1 &hour=evening & temp=cold then count=low

Le regole evidenziate possono essere combinate per formare un numero minore di regole equivalenti.

If hour=night then count=low
If season=1 &hour!=night & temp=warm then count=high
If season=1 &hour!=night & temp=cold then count=low

Valutazione delle prestazioni

Per valutare le prestazioni della nostra rete fuzzy abbiamo preparato lo script **hour_fuzzy_mamdani_testset_con_media.m**.

Preventivamente è necessario caricare il workspace **hour_fuzzy_mamdani_workspace.mat** che contiene le strutture dati necessarie all'esecuzione dello script. La rete fuzzy implementata è presente sia nello workspace che nel file allegato **hour_rete_fuzzy_mamdani.fis**.

```

>>load('hour_fuzzy_mamdani_workspace');
>>hour_fuzzy_mamdantestset_con_media;

media=0;
for l=1:100
%di_comodo_input e di_comodo_output contengono inizialmente input ed      output.
Poi ci togliamo colonne fino %lasciarlo vuoto.

    di_comodo_input=input23;
    di_comodo_output=target;

    % k è il numero di colonne -1... serve per l'intervallo della rand
    k=17378;

    % Estraggo 7500 campioni random
    for i=1:7500
        n=round(rand(1)*k+1);
        k=k-1;
        example_feature(i,:)=di_comodo_input(n,:);
        di_comodo_input(n,:)=[];
        example_output(i,1)=di_comodo_output(n,1);
        di_comodo_output(n,:)=[];
    end;

    % Test the Network
    output= evalfis(example_feature,rete_fuzzy);

    %errore su ogni singolo esempio
    e = gsubtract(example_output,output);

    %mi manca solo il MSE
    MSE=0;

    for i=1:7500;
        MSE=MSE+e(i,1)*e(i,1);
    end

    MSE=MSE/7500;

    media=media+MSE;

    MSE
end;

media=media/100

```

Il codice contiene un ciclo che per 100 volte estrae in modo totalmente random un sottoinsieme delle features disponibili. Questi valori vengono messi all'interno della matrice `example_features`, mentre i relativi output vengono messi nell'array `example_output`.

```
>>evalfis;
```

Tramite il comando `evalfis` possiamo testare la rete ed usare la differenza tra i valori ottenuti e quelli desiderati per calcolarne il MSE. Alla fine delle iterazioni `media` contiene la media di tutti i 100 MSE ottenuti.

Alla fine delle iterazioni media contiene la media di tutti i 100 MSE ottenuti. Dopo aver eseguito lo script abbiamo ottenuto i seguenti MSE

1.1474E+05	1.1276E+05	1.1201E+05	1.1256E+05	1.1418E+05
1.1517E+05	1.1350E+05	1.1316E+05	1.1363E+05	1.1480E+05
1.1492E+05	1.1599E+05	1.1387E+05	1.1309E+05	1.1174E+05
1.1476E+05	1.1277E+05	1.1348E+05	1.1380E+05	1.1461E+05
1.1301E+05	1.1443E+05	1.1653E+05	1.1402E+05	1.1390E+05
1.1192E+05	1.1412E+05	1.1234E+05	1.1411E+05	1.1236E+05
1.1263E+05	1.1444E+05	1.1549E+05	1.1429E+05	1.1435E+05
1.1420E+05	1.1341E+05	1.1106E+05	1.1222E+05	1.1461E+05
1.1162E+05	1.1282E+05	1.1498E+05	1.1263E+05	1.1439E+05
1.1412E+05	1.1539E+05	1.1336E+05	1.1511E+05	1.1403E+05
1.1399E+05	1.1426E+05	1.1481E+05	1.1242E+05	1.1546E+05
1.1514E+05	1.1268E+05	1.1180E+05	1.1163E+05	1.1319E+05
1.1555E+05	1.1288E+05	1.1211E+05	1.1423E+05	1.1648E+05
1.1456E+05	1.1489E+05	1.1541E+05	1.1548E+05	1.1417E+05
1.1379E+05	1.1296E+05	1.1312E+05	1.1521E+05	1.1423E+05
1.1294E+05	1.1496E+05	1.1362E+05	1.1391E+05	1.1450E+05
1.1498E+05	1.1544E+05	1.1216E+05	1.1324E+05	1.1349E+05
1.1074E+05	1.1310E+05	1.1346E+05	1.1328E+05	1.1466E+05
1.1405E+05	1.1360E+05	1.1287E+05	1.1483E+05	1.1156E+05
1.1576E+05	1.1486E+05	1.1416E+05	1.1289E+05	1.1339E+05

La media di tutti questi valori è pari a **1.1380E+05**.

ANFIS per Day

Come prima cosa è necessario importare il workspace

```
>> load('day_ANFIS_workspace.mat')
```

successivamente possiamo eseguire lo script **day_ANFIS_create_the_set.m** il quale crea tre set di dati così formati:

- training_set: 511 campioni (circa il 70% del totale);
- testing_set: 110 campioni (circa il 15% del totale);
- checking_set: 110 campioni (circa il 15% del totale).

```
>> day_ANFIS_create_the_set;
```

```

%completa conterra l'intero set di esempi
completa=input15;
completa(:,6)=target(:,1);

%ora bisogna creare 3 set: trainig, testing e checking
%training=70% (511 sample)-testing=15% (110 sample)-checking=15% (110 sample)

di_comodo=completa;

% k è il numeto di colonne -1... serve per l'intervallo della rand.
% iniziamo da trainig
k=510;

for i=1:511
    n=round(rand(1)*k+1);
    k=k-1;
    training_set(i,:)=di_comodo(n,:);
    di_comodo(n,:)=[];
end;

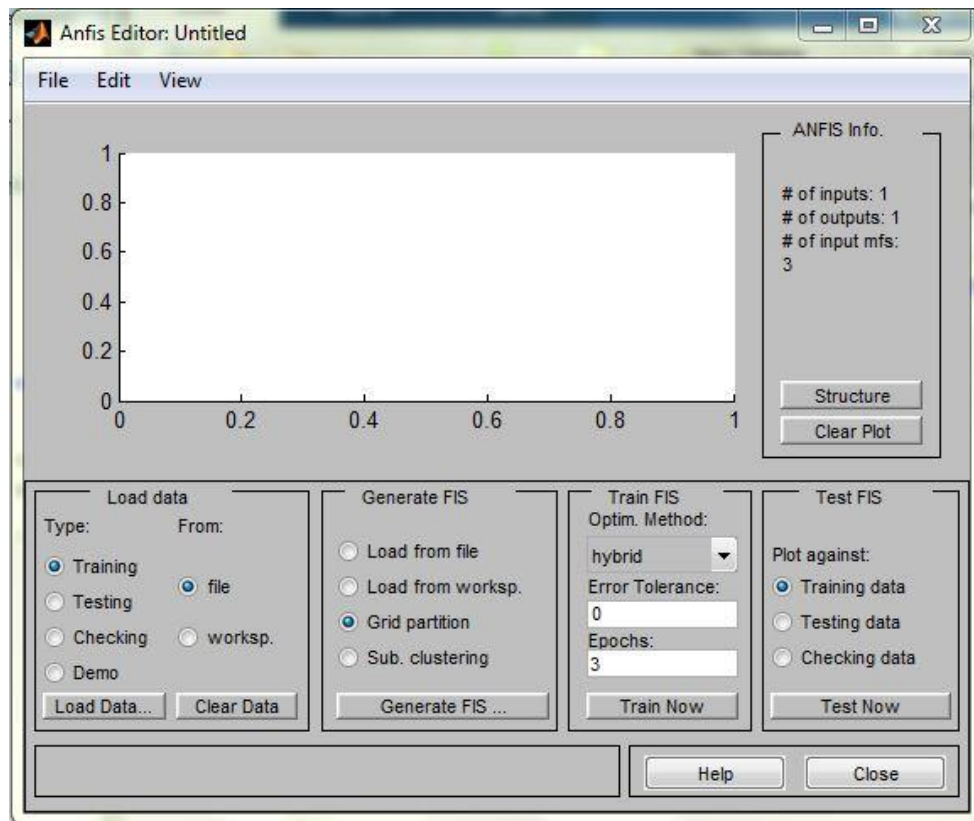
%ora facciammo testing e checking
k=219;

for i=1:110
    n=round(rand(1)*k+1);
    k=k-1;
    testing_set(i,:)=di_comodo(n,:);
    di_comodo(n,:)=[];
    n=round(rand(1)*k+1);
    k=k-1;
    checking_set(i,:)=di_comodo(n,:);
    di_comodo(n,:)=[];
end;

```

Utilizzando il comando anfisedit, dato attraverso la command window, si apre la seguente finestra in cui possiamo caricare i tre set creati mediante lo script.

```
>> anfisedit
```

I parametri che testiamo sono i seguenti:

Generate FIS

- Grid partition
 - INPUT- MF Type
 - trimf;
 - trapmf;
 - gbellmf;
 - gaussmf
 - gauss2mf;
 - pimf
 - dsigmf
 - psigmf
 - OUTPUT-MF Type
 - constant
 - linear
- Sub clustering

Train FIS

- Optimization method
 - Backpropagation
 - Hybrid
- Epoch: abbiamo scelto di provare questi quattro valori
 - 3
 - 10
 - 20
 - 40

Grid partition

In tabella abbiamo riportato i risultati dei test.

				Errore sul train	Errore sul test	Errore sul check
trimf	Constant	Back propagation	3	4111.86	6323.91	6418.78
			10	4111.84	6323.91	6418.77
			20	4111.78	6323.89	6418.75
			40	4111.63	6323.82	6418.69
		Hybrid	3	749.86	4061.90	4459.50
			10	676.81	4085.47	4479.67
			20	548.43	4192.97	4532.13
			40	518.97	4170.48	4575.21
	Linear	Back propagation	3	4111.85	6323.91	6418.77
			10	4111.72	6323.87	6418.74
			20	4111.44	6323.79	6418.65
			40	4110.77	6323.48	6418.33
		Hybrid	3	355.99	33342.84	9870.22
			10	348.18	22520.10	14409.56
			20	348.18	22520.10	14409.56
			40	348.18	2252.10	14409.56

				Errore sul train	Errore sul test	Errore sul check
trapmf	Constant	Back propagation	3	4111.86	6323.91	66418.78
			10	4111.83	6323.90	6418.77
			20	4111.76	6323.88	6418.76
			40	4111.63	6323.86	6418.74
		Hybrid	3	742.62	4032.61	4164.52
			10	679.62	4019.79	4123.24
			20	551.21	4005.34	4119.09
			40	531.66	3999.93	4115.43
	Linear	Back propagation	3	4111.84	6323.90	6418.77
			10	4111.69	6323.86	6418.73
			20	4111.37	6323.78	6418.67
			40	4110.73	6323.66	6418.58
		Hybrid	3	370.99	12388.06	8830.58
			10	351.28	15571.43	6587.97
			20	351.05	15571.43	6587.97
			40	351.05	15571.43	6587.97

				Errore sul train	Errore sul test	Errore sul check
gbellmf	Constant	Back propagation	3	4111.86	6323.91	6418.78
			10	4111.84	6323.90	6418.77
			20	4111.77	6323.86	6418.73
			40	4111.61	6323.77	6418.64
		Hybrid	3	673.49	3476.23	3838.41
			10	620.96	3526.83	4481.52
			20	508.33	7489.19	5594.87
			40	484.43	3990.69	3879.87
	Linear	Back propagation	3	4111.87	6323.90	6418.77
			10	4111.72	6323.86	6418.72
			20	4111.42	6323.68	6418.72
			40	4110.70	6323.29	6418.10
		Hybrid	3	336.93	16804.22	17828.84
			10	319.07	13067.81	7484.61
			20	319.07	13067.81	7484.61
			40	319.07	13067.29	8252.02

				Errore sul train	Errore sul test	Errore sul check
gaussmf	Constant	Back propagation	3	4111.86	6323.91	6418.78
			10	4111.84	6323.90	6418.77
			20	4111.78	6323.86	6418.73
			40	4111.60	6323.78	6418.65
		Hybrid	3	491.13	4662.78	3932.33
			10	491.17	4656.78	3936.42
			20	491.21	4648.26	3938.52
			40	491.96	4954.15	4018.16
	Linear	Back propagation	3	4111.85	6323.91	6418.77
			10	4111.72	6323.86	6418.73
			20	4111.39	6323.68	6418.54
			40	4110.65	6323.35	6418.17
		Hybrid	3	341.25	15615.14	16282.19
			10	332.22	18575.67	20336.99
			20	320.61	14318.80	12378.44
			40	320.61	14318.80	12378.44

				Errore sul train	Errore sul test	Errore sul check
gauss2mf	Constant	Back propagation	3	4111.86	6323.91	6418.78
			10	4111.83	6323.90	6418.77
			20	4111.76	6323.89	6418.76
			40	4111.66	6323.87	6418.75
		Hybrid	3	735.17	3764.92	4122.98
			10	735.17	3764.92	4122.98
			20	735.17	3764.92	4122.98
			40	735.17	3764.92	4122.98
	Linear	Back propagation	3	4111.84	6323.90	6418.77
			10	4111.69	6323.86	6418.74
			20	4111.36	6323.80	6418.69
			40	4110.90	6323.74	6418.61
		Hybrid	3	348.11	13533.82	10376.83
			10	329.09	18603.56	16143.23
			20	348.11	13533.82	10376.83
			40	320.61	14318.80	12378.44

				Errore sul train	Errore sul test	Errore sul check
pimf	Constant	Back propagation	3	4111.86	6323.91	6418.78
			10	4111.83	6323.90	6418.77
			20	4111.76	6323.89	6418.76
			40	4111.62	6323.86	6418.74
		Hybrid	3	499.47	5763.35	4384.83
			10	499.47	5763.35	4187.87
			20	499.47	5763.35	4168.20
			40	499.47	5763.35	4187.87
	Linear	Back propagation	3	4111.84	6323.72	6418.77
			10	4111.68	6323.74	6418.73
			20	4111.36	6323.78	6418.68
			40	4111.24	6323.78	6418.66
		Hybrid	3	399.89	62955.30	12273.83
			10	361.25	12679.95	7663.35
			20	357.83	11021.15	6542.32
			40	357.83	11021.15	6542.32

				Errore sul train	Errore sul test	Errore sul check
dsigmf	Constant	Back propagation	3	4111.86	6323.91	6418.78
			10	4111.83	6323.90	6418.77
			20	4111.77	6323.89	6418.76
			40	411.63	6323.85	6418.74
		Hybrid	3	563.97	8796.72	5389.81
			10	519.31	4293.14	4506.39
			20	486.18	4312.87	4978.66
			40	483.59	4060.55	4519.57
	Linear	Back propagation	3	4110.73	6323.65	6418.56
			10	4110.78	6323.66	6418.57
			20	4110.94	6323.69	6418.59
			40	4111.25	6323.76	6418.65
		Hybrid	3	350.07	12406.80	8030.38
			10	335.69	11056.37	8946.51
			20	329.50	11302.76	7589.09
			40	318.72	13170.30	10010.05

				Errore sul train	Errore sul test	Errore sul check
psigmf	Constant	Back propagation	3	4111.86	6323.91	6418.78
			10	4111.83	6323.90	6418.77
			20	4111.77	6323.89	6418.76
			40	4111.63	6323.85	6418.74
		Hybrid	3	497.10	4085.51	4425.62
			10	497.11	4089.35	4413.71
			20	497.12	4091.04	4404.07
			40	505.34	3779.62	4217.34
	Linear	Back propagation	3	4111.87	6323.91	6418.77
			10	4111.72	6323.87	6418.74
			20	4111.41	6323.79	6418.67
			40	4111.77	6323.66	6418.56
		Hybrid	3	352.71	12360.30	8580.09
			10	338.94	10766.38	8415.34
			20	323.64	11312.01	8047.28
			40	318.72	13171.24	10012.04

Analisi dei dati ottenuti tramite Grid Partition

Per aiutare la valutazione dei risultati raccolti riportiamo in tabella il migliore e il peggiore per ogni configurazione presa in analisi.

			min-epoche		max-epoche	
trimf	→ Constant	→ Back propagation	6323.82	-40	6323.91	-3/10
		→ Hybrid	4061.90	-3	4192.97	-20
	→ Linear	→ Back propagation	6323.48	-40	6323.91	-3
		→ Hybrid	22520.10	-10/20/40	33342.84	-3
trapmf	→ Constant	→ Back propagation	6323.86	-40	6323.91	-3
		→ Hybrid	3999.93	-40	4032.61	-3
	→ Linear	→ Back propagation	6323.66	-40	6323.90	-3
		→ Hybrid	12388.06	-3	15571.43	-10/20/40
gbellmf	→ Constant	→ Back propagation	6323.77	-40	6323.91	-3
		→ Hybrid	3476.23	-3	7489.19	-20
	→ Linear	→ Back propagation	6323.29	-40	6323.90	-3
		→ Hybrid	13068.81	-10/20/40	16804.22	-3
gaussmf	→ Constant	→ Back propagation	6323.78	-40	6323.91	-3
		→ Hybrid	4648.26	-20	4954.15	-40
	→ Linear	→ Back propagation	6323.35	-40	6323.91	-3
		→ Hybrid	14318.80	-20/40	18575.67	-10
gauss2mf	→ Constant	→ Back propagation	6323.87	-40	6323.91	-3
		→ Hybrid	3764.92 per ogni epoca valutata			
	→ Linear	→ Back propagation	6323.74	-40	6323.90	-3
		→ Hybrid	13533.82	-3/20	18603.56	-10
pimf	→ Constant	→ Back propagation	6323.86	-40	6323.91	-3
		→ Hybrid	5763.35 per ogni epoca valutata			
	→ Linear	→ Back propagation	6323.72	-3	6323.78	-20/40
		→ Hybrid	11021.15	-20/40	62955.30	-3
dsigmf	→ Constant	→ Back propagation	6323.85	-40	6323.91	-3
		→ Hybrid	4060.55	-40	8796.72	-3
	→ Linear	→ Back propagation	6323.65	-3	6323.76	-40
		→ Hybrid	11056.37	-10	13170.39	-40
psigmf	→ Constant	→ Back propagation	6323.85	-40	6323.91	-3
		→ Hybrid	3779.62	-40	4085.51	-3
	→ Linear	→ Back propagation	6323.66	-40	6323.91	-3
		→ Hybrid	10766.38	-10	13171.24	-40

La prima cosa che abbiamo notato è che i risultati sono legati alla scelta della combinazione constant/linear+back propagation/hybrid mentre la variazione di funzione di ingresso scelta non aiuta più di tanto a migliorare (o a peggiorare) il risultato. Allo stesso modo abbiamo notato che nemmeno variare il numero di epoche utilizzate per il train aiuta particolarmente ad abbassare l'errore che, sebbene nella maggior parte dei casi diminuisce, riporta variazioni infinitesime.

Detto questo possiamo vedere che la combinazione linear+hybrid ritorna i risultati peggiori in assoluto mentre la combinazione hybrid+constant è quella che pare comportarsi meglio.

Sub-clustering

In linea teorica, il sub-clustering è da considerarsi più accurato del grid partition, in pratica tuttavia possiamo considerarlo equivalente al risultato migliore ottenuto tramite questo secondo metodo.

Avendo analizzato tutte le possibili combinazioni di parametri nel grid partition evitiamo di effettuare questa parte anche se, in una applicazione reale per un committente, un'analisi più accurata non potrebbe essere omessa.

ANFIS per Hour

Come prima cosa è necessario importare il workspace

```
>> load('hour_ANFIS_workspace.mat')
```

Successivamente possiamo eseguire lo script **hour_ANFIS_create_the_set.m** il quale crea tre set di dati così formati:

- training_set: 12179 campioni (circa il 70% del totale);
- testing_set: 2600 campioni (circa il 15% del totale);
- checking_set: 2600 campioni (circa il 15% del totale).

```
>> hour_ANFIS_create_the_set;
```

```
%completa conterra l'intero set di esempi
completa=input23;
completa(:,6)=target(:,1);

%ora bisogna creare 3 set: trainig, testing e checking

di_comodo=completa;

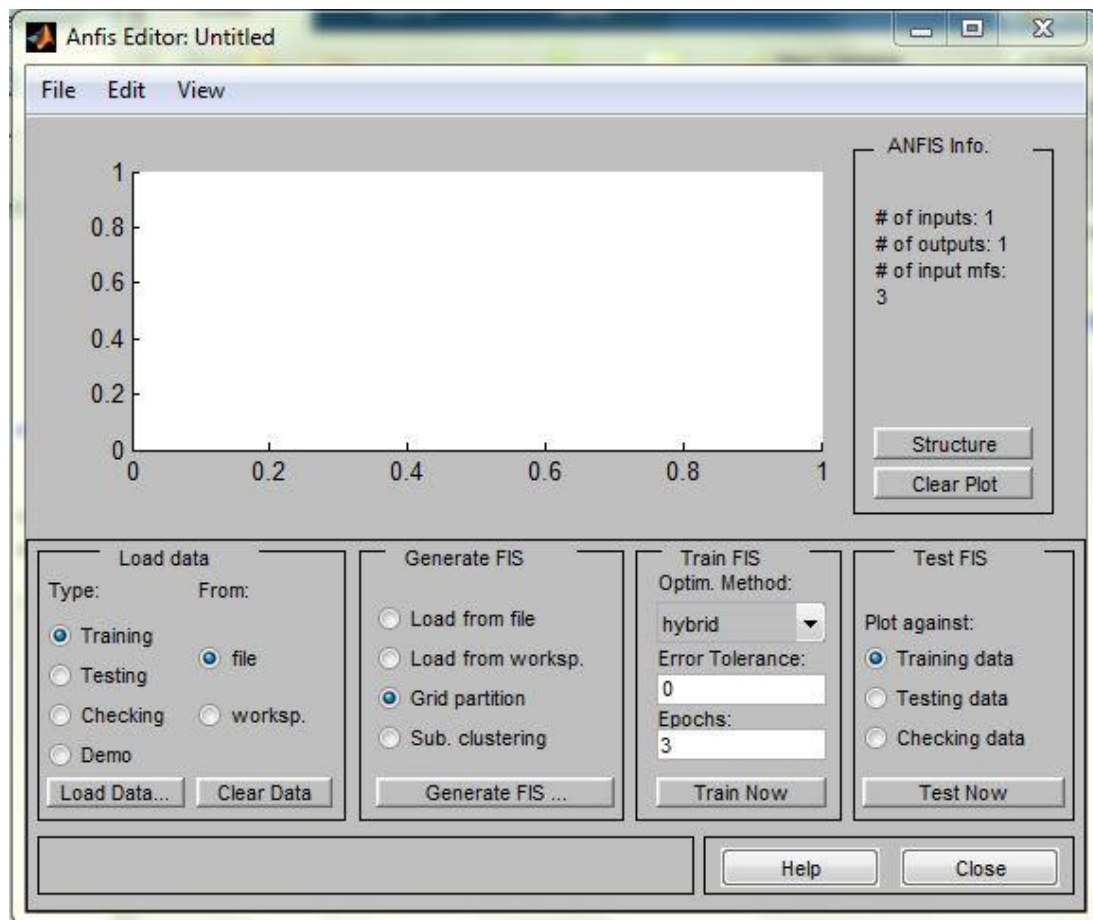
% k è il numeto di colonne -1... serve per l'intervallo della rand.
% iniziamo da trainig
k=12178;

for i=1:12179
    n=round(rand(1)*k+1);
    k=k-1;
    training_set(i,:)=di_comodo(n,:);
    di_comodo(n,:)=[];
end;

k=5199;
for i=1:2600
    n=round(rand(1)*k+1);
    k=k-1;
    testing_set(i,:)=di_comodo(n,:);
    di_comodo(n,:)=[];
    n=round(rand(1)*k+1);
    k=k-1;
    checking_set(i,:)=di_comodo(n,:);
    di_comodo(n,:)=[];
end;
```

Utilizzando il comando `anfisedit`, dato attraverso la command window, si apre la seguente finestra in cui possiamo caricare i tre set creati mediante lo script.

```
>> anfisedit
```



I parametri che testiamo sono i seguenti:

Generate FIS

- Grid partition
 - INPUT- MF Type
 - trimf;
 - trapmf;
 - gbellmf;
 - gaussmf
 - gauss2mf;
 - pimf
 - dsigmf
 - psigmf
 - OUTPUT-MF Type
 - constant
 - linear
- Sub. Clustering.

Train FIS

- Optimization method
 - Backpropagation
 - Hybrid
- Epoch: abbiamo scelto di provare questi quattro valori
 - 3
 - 10
 - 20
 - 40.

Dal momento che in *Day* i risultati migliori sono stati ottenuti usando un algoritmo ibrido abbiamo deciso di verificare l'andamento delle prestazioni solo per esso

In tabella sono riportati i risultati dei test.

				Errore sul train	Errore sul test	Errore sul check
trimf	Constant	Hybrid	3	104.55	608.79	556.64
			10	104.15	324.75	315.36
			20	104.23	420.29	392.08
			40	104.09	348.76	325.42
	Linear	Hybrid	3			
			10			
			20			
			40			

				Errore sul train	Errore sul test	Errore sul check
trapmf	Constant	Hybrid	3	102.23	245.60	256.36
			10	99.02	248.29	258.59
			20	99.02	248.29	258.59
			40	99.09	248.15	258.38
	Linear	Hybrid	3			
			10			
			20			
			40			

				Errore sul train	Errore sul test	Errore sul check
--	--	--	--	------------------	-----------------	------------------

Gbellmf	Constant	Hybrid	3	100.86	238.62	251.53
			10	100.58	245.13	259.18
			20	100.22	260.35	274.14
			40	99.59	543.56	477.53
	Linear	Hybrid	3			
			10			
			20			
			40			

				Errore sul train	Errore sul test	Errore sul check
gaussmf	Constant	Hybrid	3	101.18	482.30	457.75
			10	101.23	509.17	473.60
			20	101.41	568.42	509.04
			40	101.06	432.51	432.06
	Linear	Hybrid	3			
			10			
			20			
			40			

				Errore sul train	Errore sul test	Errore sul check
gauss2mf	Constant	Hybrid	3	100.68	305.38	293.66
			10	99.63	280.40	277.05
			20	99.01	531.17	456.74
			40	98.27	302.15	313.19
	Linear	Hybrid	3			
			10			
			20			
			40			

				Errore sul train	Errore sul test	Errore sul check
primf	Constant	Hybrid	3	103.06	260.57	267.56
			10	100.14	249.08	257.82
			20	98.93	246.08	254.79
			40	98.28	253.10	261.17
	Linear	Hybrid	3			
			10			
			20			
			40			

				Errore sul	Errore sul	Errore sul
--	--	--	--	------------	------------	------------

				train	test	check
dsigmf	Constant	Hybrid	3	101.35	333.58	314.47
			10	99.87	318.42	302.85
			20	99.18	318.81	299.02
			40	98.99	325.17	302.02
	Linear	Hybrid	3			
			10			
			20			
			40			

				Errore sul train	Errore sul test	Errore sul check
psigmf	Constant	Hybrid	3	99.79	326.04	301.68
			10	100.07	321.78	298.29
			20	99.46	338.28	316.52
			40	98.99	325.17	302.02
	Linear	Hybrid	3			
			10			
			20			
			40			

Seconda parte

Nella seconda parte si fa riferimento soltanto ai dati giornalieri. In questa parte si intende predire il numero di noleggi a Washington DC basandoci sul sottoinsieme di feature ottimali, selezionate nella prima parte con l'aiuto della MLP Neural Network (input15).

NN Time Series Tool su Day, anno 2011

Useremo i dati del 2011 per sviluppare, usando NN Time Series Tool e adottando una strategia ad anello aperto, un modello che preveda il noleggio delle biciclette. Come prima cosa vanno quindi selezionati i dati relativi all'anno 2011: per fare ciò utilizziamo lo script **Forecasting_pt1_Divide_Data**.

```
load('Forecasting_workspace');
```

```
input2011=input15;  
target2011=target;
```

```
k=731;  
i=1;  
while (i<=k)  
    if (input2011(i,2)==1)  
        input2011(i,:)=[];  
        target2011(i,:)=[];  
        k=k-1;  
    else  
        i=i+1;  
    end  
end
```

```
>>ntstool
```

Utilizziamo ora ntstool interno a nnstart per realizzare la rete con cui faremo la predizione. Scegliamo di usare il modello NARX, dal momento che abbiamo a disposizione i valori passati sia dell'output $y(t)$ sia dell'input $x(t)$ e decidiamo di sfruttare tutte le informazioni per ottenere un modello il più accurato possibile.

Lo script finale utilizzato è **Forecasting_pt1_final**. Dopo aver creato il data set di partenza, si impostano i parametri che utilizziamo per fare i test differenziati. Proviamo a vedere cosa accade con ritardi variabili da 1 a 10 e numero di nodi variabile da 1 a 10. I ritardi per la serie di input $x(t)$ potrebbero anche essere diversi da quelli della serie di output $y(t)$ ma per semplicità noi li assumiamo uguali.

```
Forecasting_pt1_Divide_Data;
```

```
num_training = 15;  
for i=1:10  
    hiddenLayerSize = i;    %%changed to find the best value  
    for j=1:10  
        inDelay = j;        %%changed to find the best value  
        outDelay = j;        %%changed to find the best value  
        fprintf('#nodes = %d \t\t#delays = %d', hiddenLayerSize, inDelay)  
        Forecasting_pt1_simple_script;  
    end  
end
```

Questo è il contenuto dello script **Forecasting_pt1_simple_script**, generato da ntstool. Lo abbiamo modificato solo per consentire di cambiare alcuni parametri prima della sua esecuzione tramite **Forecasting_pt1_final**.

```
% Solve an Autoregression Problem with External Input with a NARX Neural Network
% Script generated by Neural Time Series app
% Created Sun Jun 07 15:33:58 CEST 2015
%
% This script assumes these variables are defined:
%
%   input2011 - input time series.
%   target2011 - feedback time series.

X = tonndata(input2011,false,false);    %%inputSeries
T = tonndata(target2011,false,false);    %%targetSeries

% Choose a Training Function
% For a list of all training functions type: help nntrain
% 'trainlm' is usually fastest.
% 'trainbr' takes longer but may be better for challenging problems.
% 'trainscg' uses less memory. NTSTOOL falls back to this in low memory
situations.
trainFcn = 'trainlm'; % Levenberg-Marquardt

% Create a Nonlinear Autoregressive Network with External Input
inputDelays = 1:inDelay;
feedbackDelays = 1:outDelay;
%%%hiddenLayerSize = 10;      so we can change it before execution of the
%%%script
net = narxnet(inputDelays,feedbackDelays,hiddenLayerSize,'open',trainFcn);

% Prepare the Data for Training and Simulation
% The function PREPARETS prepares timeseries data for a particular network,
% shifting time by the minimum amount to fill input states and layer states.
% Using PREPARETS allows you to keep your original time series data unchanged,
while
% easily customizing it for networks with differing numbers of delays, with
% open loop or closed loop feedback modes.
[x,xi,ai,t] = preparets(net,X,{},T);    %% x = input
                                         %% xi = inputStates
                                         %% ai = layerStates
                                         %% t = targets

% Setup Division of Data for Training, Validation, Testing
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% Train the Network
for i=1:num_training
[net,tr] = train(net,x,t,xi,ai);
end

% Test the Network
y = net(x,xi,ai);
e = gsubtract(t,y);
performance = perform(net,t,y)

% View the Network
```

```

view(net)

% Plots
% Uncomment these lines to enable various plots.
%figure, plotperform(tr)
%figure, plottrainstate(tr)
%figure, plotregression(t,y)
%figure, plotresponse(t,y)
%figure, ploterrcorr(e)
%figure, plotinerrcorr(x,e)

% Closed Loop Network
% Use this network to do multi-step prediction.
% The function CLOSELOOP replaces the feedback input with a direct
% connection from the outout layer.
netc = closeloop(net);
netc.name = [net.name ' - Closed Loop'];
%view(netc)
[xc,xic,aic,tc] = preparets(netc,X,{},T);
yc = netc(xc,xic,aic);
closedLoopPerformance = perform(netc,tc,yc)

% Step-Ahead Prediction Network
% For some applications it helps to get the prediction a timestep early.
% The original network returns predicted y(t+1) at the same time it is given
y(t+1).
% For some applications such as decision making, it would help to have predicted
% y(t+1) once y(t) is available, but before the actual y(t+1) occurs.
% The network can be made to return its output a timestep early by removing one
delay
% so that its minimal tap delay is now 0 instead of 1. The new network returns
the
% same outputs as the original network, but outputs are shifted left one
timestep.
nets = removedelay(net);
nets.name = [net.name ' - Predict One Step Ahead'];
%view(nets)
[xs,xis,ais,ts] = preparets(nets,X,{},T);
ys = nets(xs,xis,ais);
stepAheadPerformance = perform(nets,ts,ys)

```

Di seguito abbiamo i risultati delle varie prove.

# hidden nodes	# delays	MSE (medio)
1	1	5.82E+05
1	2	6.65E+05
1	3	6.48E+05
1	4	6.44E+05
1	5	6.72E+05
1	6	9.22E+05
1	7	6.52E+05
1	8	6.06E+05
1	9	7.70E+05
1	10	9.73E+05

2	1	5.72E+05
2	2	6.91E+05
2	3	5.46E+05
2	4	7.24E+05
2	5	7.26E+05
2	6	7.45E+05
2	7	6.85E+05
2	8	1.13E+06
2	9	8.84E+05
2	10	8.15E+05
3	1	5.34E+05
3	2	5.97E+05
3	3	5.77E+05
3	4	6.90E+05
3	5	7.99E+05
3	6	8.16E+05
3	7	8.66E+05
3	8	8.29E+05
3	9	8.18E+05
3	10	9.23E+05
4	1	5.64E+05
4	2	6.43E+05
4	3	6.23E+05
4	4	7.36E+05
4	5	7.30E+05
4	6	6.91E+05
4	7	7.90E+05
4	8	9.48E+05
4	9	1.02E+06
4	10	7.72E+05
5	1	5.90E+05
5	2	6.56E+05
5	3	6.15E+05
5	4	7.28E+05
5	5	7.78E+05
5	6	7.50E+05
5	7	8.38E+05
5	8	9.32E+05
5	9	8.71E+05
5	10	8.75E+05
6	1	5.77E+05
6	2	6.45E+05
6	3	5.92E+05
6	4	7.84E+05
6	5	7.43E+05

6	6	7.72E+05
6	7	8.68E+05
6	8	8.73E+05
6	9	9.43E+05
6	10	8.10E+05
7	1	5.03E+05
7	2	6.69E+05
7	3	7.04E+05
7	4	7.23E+05
7	5	7.49E+05
7	6	6.95E+05
7	7	8.86E+05
7	8	7.89E+05
7	9	8.67E+05
7	10	7.93E+05
8	1	6.81E+05
8	2	6.42E+05
8	3	6.16E+05
8	4	7.42E+05
8	5	8.11E+05
8	6	7.00E+05
8	7	7.24E+05
8	8	9.13E+05
8	9	8.66E+05
8	10	9.16E+05
9	1	5.09E+05
9	2	7.95E+05
9	3	6.69E+05
9	4	7.47E+05
9	5	7.52E+05
9	6	8.82E+05
9	7	7.93E+05
9	8	7.95E+05
9	9	9.56E+05
9	10	9.47E+05
10	1	6.36E+05
10	2	7.19E+05
10	3	7.38E+05
10	4	8.10E+05
10	5	8.25E+05
10	6	8.76E+05
10	7	8.72E+05
10	8	7.93E+05
10	9	9.05E+05
10	10	8.31E+05

Come possiamo vedere la combinazione migliore in assoluto è 7 nodi nascosti e un ritardo; tuttavia la combinazione 3 nodi e un ritardo ha performance di poco peggiori e scegliamo questa perché così minimizziamo il numero di nodi.

Strategia ad anello chiuso su *Day*, anno 2012

Come prima cosa selezioniamo da *Day* i dati relativi all'anno 2012 con lo script **Forecasting_pt2_Divide_Data**.

```
load('Forecasting_workspace');

input2012=input15;
target2012=target;

k=731;
i=1;
while (i<=k)
    if (input2012(i,2)==0)
        input2012(i,:)=[];
        target2012(i,:)=[];
        k=k-1;
    else
        i=i+1;
    end
end
```

Possiamo ora avviare lo script **Forecasting_pt2_final** in cui impostiamo innanzitutto il valore del ritardo, per semplicità lo stesso per i campioni di ingresso e quelli di uscita, e la dimensione dell'hidden layer. Abbiamo utilizzato i valori ottimali ottenuti dall'analisi della parte precedente: in un'applicazione reale questo non sarebbe corretto perché i dati relativi ai due periodi 2011 e 2012 non sono gli stessi, però per semplicità possiamo considerarli una scelta valida.

```
delay = 1;
hiddenLayerSize = 3;
```

Stabiliamo ora il numero di giorni che vogliamo prevedere: 1, 2, 7, 10 o 15.

```
forecastedDays = 15;
```

Avviene infine la chiamata a **Forecasting_pt2_simple_script** il cui contenuto è il seguente.

```
%%creazione delle serie di input e di output
X = tonndata(input2012,false,false);
T = tonndata(target2012,false,false);

%creation of the network

inputDelays = 1:delay;
feedbackDelays = 1:delay;

net = narxnet(inputDelays, feedbackDelays, hiddenLayerSize);

%%MULTI-STEP PREDICTION
```

```

%%We divide the pool of data into 12 blocks corresponding approximately to
%%the 12 months; the first 15 days of each month are used for train; the
%%last 15 days or part of them is used to evaluate the performance in
%%forecasting

for i=0:11

%train: effettuato sulla rete in ciclo aperto
inputSeries = X((i*30+1):(i*30+15+delay));
targetSeries = T((i*30+1):(i*30+15+delay));
[xs,xi,ai,ts] = preparets(net, inputSeries, {}, targetSeries);
net = train(net, xs, ts, xi, ai);
%view(net);
y = net(xs, xi, ai);
end
for j=0:11
inputSeriesPred = X((j*30+15+delay):(30*(j+1)+1)-(15-forecastedDays));
targetSeriesPred = [T(j*30+15+delay), con2seq(nan(1,forecastedDays))];
%%NAN to force the prediction

netc = closeloop(net);
%view(netc);
[xs,xi,ai,ts] = preparets(netc, inputSeriesPred, {}, targetSeriesPred);
yPred = netc(xs, xi, ai);

fprintf('Performance nella previsione di %d giorni per il mese %d in anello
chiuso', forecastedDays, j+1);
perfAhead = perform (netc, yPred, T(end-forecastedDays+1:end)) %%accuracy
of forecasting is evaluated comparing results
%%with the data of the original series
end

```

Abbiamo deciso di dividere l'insieme dei dati in 12 blocchi corrispondenti approssimativamente ai 12 mesi per testare le performance della rete finale con diversi input. Per ogni blocco, abbiamo usato i primi 15 record per il training della rete; abbiamo poi usato gli altri 15 o parte di essi per valutare le performance delle previsioni. Dopo aver allenato la rete in ciclo aperto, chiudiamo l'anello e generiamo le due serie da usare nella previsione; la serie da usare come target viene riempita di NAN per forzare la rete a calcolare i valori. L'accuratezza dei risultati ottenuti è testata confrontando i valori previsti con quelli effettivamente ottenuti.

Soprattutto nel caso di previsione di 1 e 2 giorni in avanti sono andati sprecati molti record di esempio, ma possiamo vedere come, nonostante questo limite, siamo così riusciti a ottenere risultati facilmente confrontabili tra le varie previsioni.

Riportiamo in seguito l'esito delle 5 prove effettuate per 1, 2, 7, 10 e 15 in avanti a livello di performance medie.

1 giorno in avanti				
3.33E+08	3.85E+07	9.31E+06	5.67E+06	5.83E+06
7.85E+07				

2 giorni in avanti				
1.10E+07	3.06E+07	5.52E+07	2.06E+07	1.45E+07
2.64E+07				

7 giorni in avanti				
1.13E+08	2.37E+07	2.38E+07	1.39E+07	1.23E+07
3.73E+07				

10 giorni in avanti				
3.83E+08	2.01E+07	1.44E+07	1.32E+07	1.05E+07
8.82E+07				

15 giorni in avanti				
4.31E+06	1.18E+07	8.71E+06	1.57E+07	1.79E+08
4.39E+07				