RETE NEURALE E ASTROPARTICELLE - di Marco Illiano

SCOPO DEL DOCUMENTO

Questo documento descrive un percorso matematico – statistico finalizzato alla ricerca di un modello basato sulle reti neurali artificiali. Tale modello può essere utilizzato per effettuare simulazioni sull'andamento della media dei rilevatori AMD, dati i valori di alcune variabili ad un tempo t-k. Il lavoro è stato diviso in quattro parti:

1. In prima battuta sono stati prodotti alcuni **grafici di interpolazione** per verificare la presenza di variabili palesemente ridondanti.
2. E' stata poi applicata una tipologia di analisi fattoriale che va sotto il nome di **Analisi delle Componenti Principali** (ACP) per individuare quelle variabili esplicative che non danno un valido apporto informativo. L'ACP ha messo in evidenza anche i principali collegamenti e le correlazioni tra le variabili osservate (analisi conoscitiva del fenomeno)
3. Sono stati messi in evidenza i **ritardi temporali** nella correlazione tra le variabili con alcuni strumenti propri dell'analisi delle serie storiche
4. Infine è stato adottato un modello predittivo basato sulle **reti neurali artificiali**.

Il codice matlab del modello neurale, richiamabile anche da altri linguaggi di programmazione, genera un valore di riferimento che rappresenta la media teorica di un dato rilevatore AMD. Le variabili esplicative prese in considerazione riguardano principalmente l'attività solare, le condizioni meteo e l'attività sismica del luogo in cui la macchina è stata installata.
Il codice MATLAB, che è accompagnato step by step da codice di commento, è scaricabile, insieme ai dati di studio, dal link:
http://www.artescienza.altervista.org/zdetector_pozzuoli/NEURALE

Importazione Dati in Matlab da foglio excel

Il Data-set utilizzato è contenuto all'interno del file A_DATA_ANALYSIS.xls. Per la **"legenda"** delle variabili rimando all'appendice di questo documento.

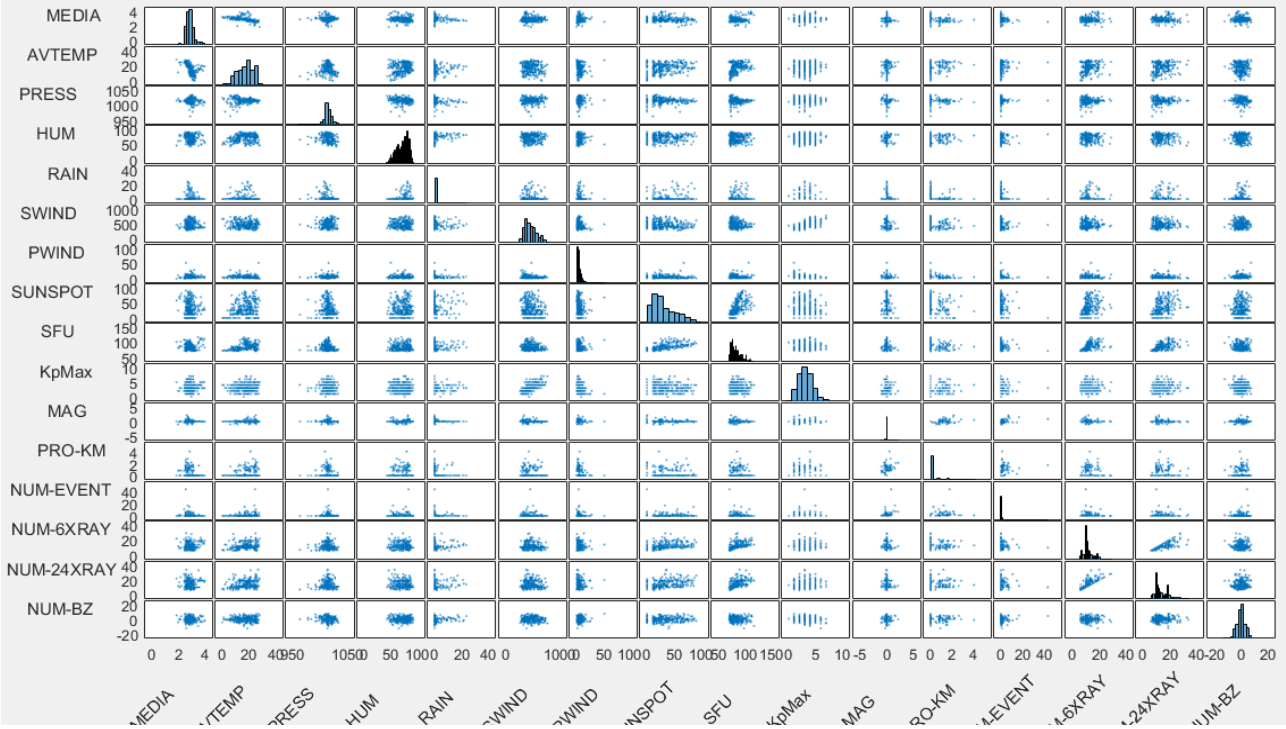
Le variabili non numeriche, come ad esempio NUM24XRAY, sono state trasformate in categorie con valori numerici. I dati sono stati importati dal foglio Excel escludendo le eventuali righe con celle in formato non numerico.
In Matlab:

```
%IMPORTAZIONE DA FILE EXCEL
[~, ~, raw0_0] = xlsread('D:\Templare\software\MATLAB\bin\MYPROG\A_DATA_ANALYSIS.xls','DATI_ASTROPARTICELLE','C7:R427');
raw = [raw0_0];
raw(cellfun(@(x) ~isempty(x) && isnumeric(x) && isnan(x),raw)) = {''};

%Creazione Matrice Dati finale di lavoro “Dati_Cosmici” 16x426
DATI_COSMICI = reshape([raw{:}],size(raw));

%Vettori della matrice di lavoro per la generazione di grafici bi-variabili
%Per Maggiori dettagli sul significato di ciascuna variabile si rimanda all’appendice.

MEDIA = DATI_COSMICI(:,1);
AVTEMP = DATI_COSMICI(:,2);
PRESS = DATI_COSMICI(:,3);
HUM = DATI_COSMICI(:,4);
RAIN = DATI_COSMICI(:,5);
SWIND = DATI_COSMICI(:,6);
PWIND = DATI_COSMICI(:,7);
SUNSPOT = DATI_COSMICI(:,8);
SFU = DATI_COSMICI(:,9);
KpMax = DATI_COSMICI(:,10);
MAG = DATI_COSMICI(:,11);
PROKM = DATI_COSMICI(:,12);
NUMEVENT = DATI_COSMICI(:,13);
NUM6XRAY = DATI_COSMICI(:,14);
NUM24XRAY = DATI_COSMICI(:,15);
NUMBZ = DATI_COSMICI(:,16);
```

Grafici di interpolazione tra le variabili

Con il **grafico a matrice** di Matlab è possibile individuare a colpo d’occhio **le relazioni più evidenti** tra le variabili prese a due a due.
Per generare un grafico a matrice in Matlab:

```
[H,AX] = plotmatrix(DATI_COSMICI);
for i = 1:length(AX)
   ylabel(AX(i,1),etichette{i},'Rotation',0,'HorizontalAlignment','right');
   xlabel(AX(end,i),etichette{i},'Rotation',45,'HorizontalAlignment','right');
end
```



Dal grafico a matrice, sono state selezionate **le variabili NUM6XRAY vs NUM24XRAY e SUNSPOT vs SFU per attuare l'interpolazione**: tutte variabili legate all'attività solare.
Di seguito gli indicatori di bontà di interpolazione offerti da Matlab: ***AdjRsquare***

NUM6XRAY vs NUM24XRAY =

adjrsquare: 7.8588e-01

SUNSPOT vs SFU

adjrsquare: 7.3552e-01

Le variabili hanno una relazione lineare abbastanza forte: NUM6XRAY è l'emissione massima di raggi X nelle ultime 6 ore mentre NUM24XRAY è l'emissione massima nelle 24 ore.
La correlazione tra SUNSPOT e SFU (solar flux units) è ampiamente documentata sul web.
Nel definire il modello neurale, verrà considerata la sola variabile NUM24XRAY e la sola variabile SUNSPOT.
Di seguito i grafici di interpolazione in Matlab:

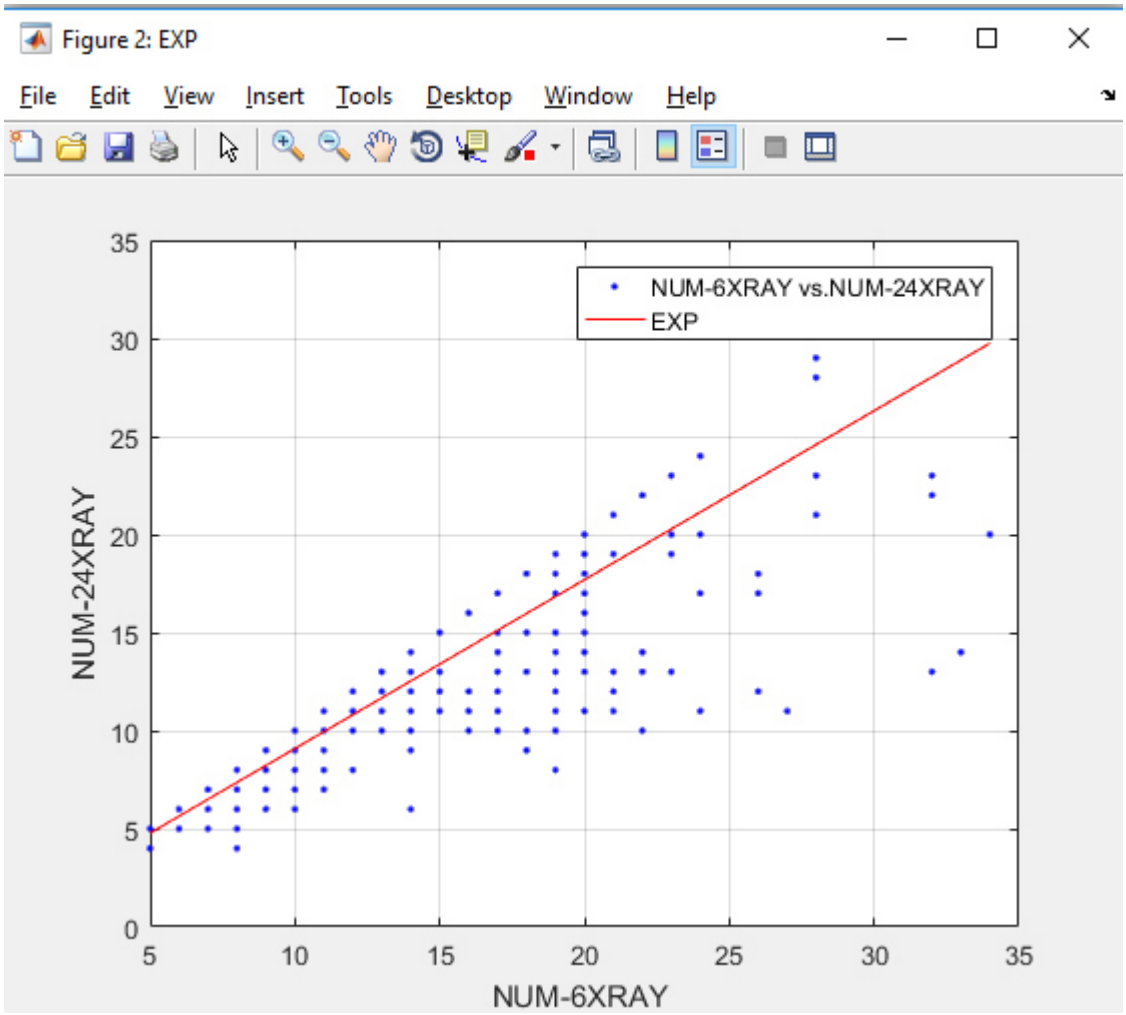
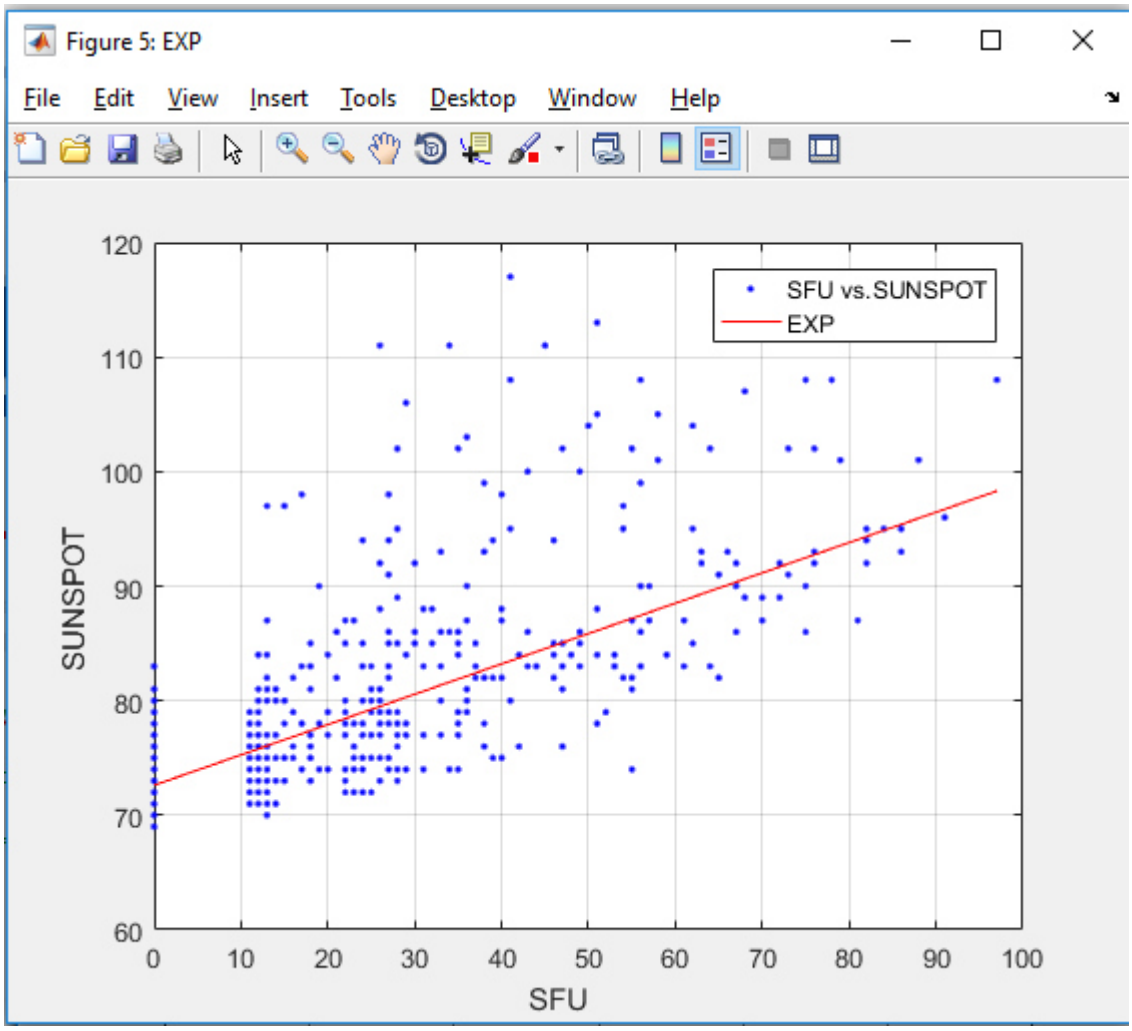
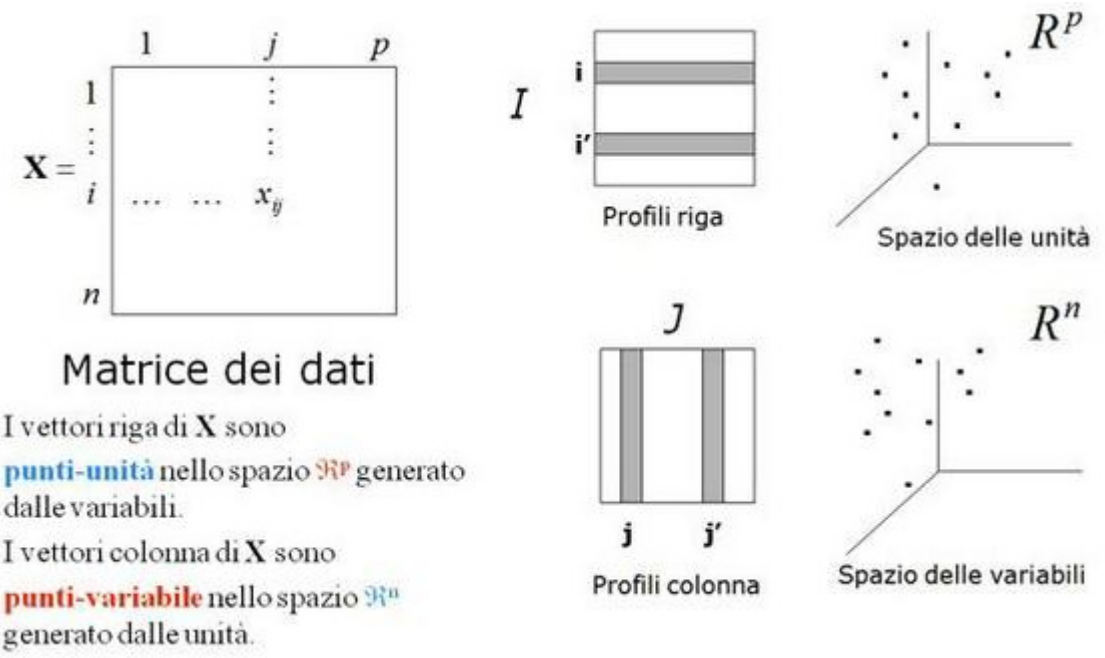


Figure 2: EXP
File Edit View Insert Tools Desktop Window Help
NUM-6XRAY vs.NUM-24XRAY
EXP
NUM-24XRAY
NUM-6XRAY
0
5
10
15
20
25
30
35

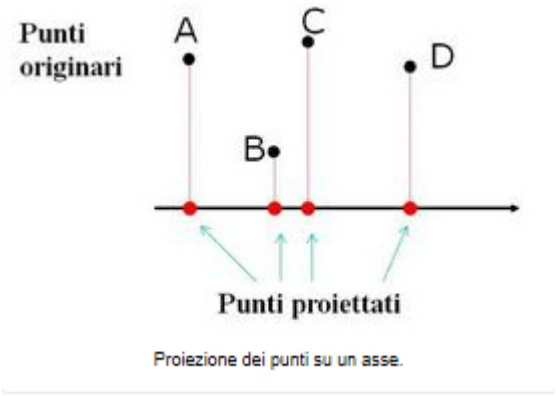


ANALISI COMPONENTI PRINCIPALI

Proseguendo l'analisi conoscitiva sui dati raccolti, si propone di sostituire il numero di variabili preso in considerazione, con un numero di variabili di sintesi più contenuto eliminando la ridondanza dei dati e garantendo la minima perdita di informazione. In tal modo è possibile interpretare **le relazioni più evidenti tra le variabili** prese in considerazione in uno spazio ridotto. Allo stesso tempo si possono individuare, se vi sono (tra le variabili originarie), variabili che non forniscono un contributo informativo valido (tali variabili verranno estromesse dal modello neurale). Le variabili di sintesi individuate con l'Analisi in Componenti Principali, sono non correlate (ortogonali) tra loro e sono combinazione lineare delle variabili originarie. In sintesi l'ACP individua uno spazio di dimensione ridotto su cui proiettare i punti dei casi osservati (la serie storica considerata) e i punti delle variabili originarie.



PROIEZIONE DEI PUNTI ORIGINALI SU UNA COMPONENTE PRINCIPALE:



La matrice con le variabili di sintesi (componenti principali) è la **Y = Xu** dove X è la matrice originaria dei dati e u è il vettore dei coefficienti che, applicati alle variabili, permettono di ottenere la combinazione lineare Y.
Il problema dell'ACP è trovare i vettori u che massimizzano la varianza della combinazione lineare Y con il vincolo che i vettori *u* abbiano modulo unitario e che siano ortogonali tra loro $u_i'u_j = 0$.
Massimizzando la varianza della combinazione lineare Y, si massimizza anche l'informazione contenuta in Y rispetto alla nuvola di punti originaria: il vettore dei coefficienti *u* con modulo unitario fornisce la direzione della componente principale (nuovo asse) che ottimizza il posizionamento delle proiezioni dei punti della nube dei valori originari nel nuovo spazio vettoriale delle componenti principali con dimensione ridotta.
Si tratta di un problema di massimo vincolato che viene risolto con il **moltiplicatore di Lagrange** con soluzione:
X'X = Λu

Il vettore *u* è il primo autovettore della matrice X'X mentre Λ è il corrispondente autovalore. La prima componente principale (vettore u) sarà quella con Λ maggiore e a seguire la seconda sarà quella con Λ maggiore dopo la prima e così via.
Spesso i dati hanno unità di misura differenti per cui occorre centrare (sulla media) e standardizzare le variabili. In tal modo la matrice **X'X diventa la matrice di correlazione R**.
Nel codice seguente di Matlab:

1) Si standardizzano le variabili dividendole per la varianza inversa al fine di produrre un'ACP sulla matrice di correlazione;

2) Si producono i vettori u (nel codice sono chiamati “wcoeff”) con **l’istruzione *pca*** di matlab.
3) Si rendono ortogonali i vettori u trovati.
4) Si calcolano le coordinate dei dati originali proiettati sulle componenti principali
5) Infine il programma produce alcuni grafici.

```
%Calcolo dellla matrice delle varianze inverse da passare alla funzione pca per standardizzare i dati
varianza_inv = 1./var(DATI_COSMICI);

%ACP
[wcoeff,score,latent,tsquared,explained] = pca(DATI_COSMICI,...
'VariableWeights',varianza_inv);  %ACP
coeff_orto = diag(sqrt(varianza_inv))*wcoeff; %rende ortonormale i coefficienti della ACP
orto_check = coeff_orto'*coeff_orto; % controlla se sono ortonormali
cscores = zscore(DATI_COSMICI)*coeff_orto; %proietto le righe sulle nuove componenti ortonormail...
%e ho lo score ossia le coordinate


%**************GRAFICO DELLE RIGHE (casi osservati) SULLE COMPONENTI PRINCIPALI************
plot(cscores(:,1),cscores(:,2),'+')
xlabel('1st Principal Component')
ylabel('2nd Principal Component')

%GRAFICO DELLA VARIANZA SPIEGATA DALLE COMPONENTI PRINCIPALI
latent  % la varianza spiegata dalle componenti principali
explained % varianza spiegata in percentuale
figure() %*********** grafico con la varianza spiegata dalle componenti principali
pareto(explained)
xlabel('Principal Component')
ylabel('Variance Explained (%)')

%GRAFICO DELLE PRIME 2 COPPIE DI COMPONENTI PRINCIPALI
% doppio grafico variabili e dati
figure
biplot(coeff_orto(:,1:2),'scores',cscores(:,1:2),'varlabels',variable);
xlabel('Component 1');

ylabel('Component 2');
%GRAFICO con la terza e quarta COMPONENTE PRINCIPALE
```

```
figure
biplot(coeff_orto(:,3:4),'scores',cscores(:,3:4),'varlabels',variable);
xlabel('Component 3');
ylabel('Component 4');
```

Grafico della percentuale di variabilità spiegata dalle componenti principali

Le prime 4 componenti spiegano circa il 60% dell'informazione totale contenuta nei dati.
Infatti ogni colonna dell'istogramma seguente è la percentuale della variabilità totale contenuta nella componente principale corrispondente.
Non è una percentuale molto alta probabilmente per il fatto che i dati raccolti coprono un arco temporale ancora molto limitato (il rilevatore è attivo da poco più di un anno).

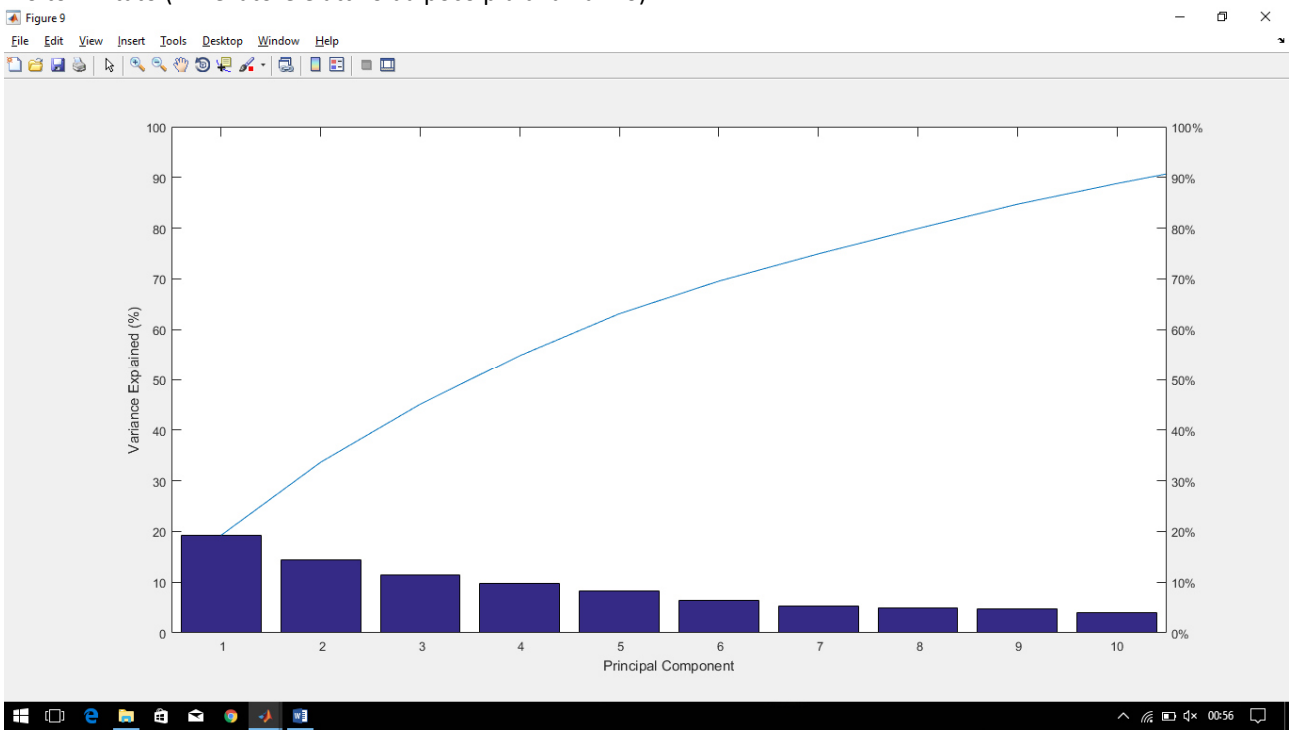
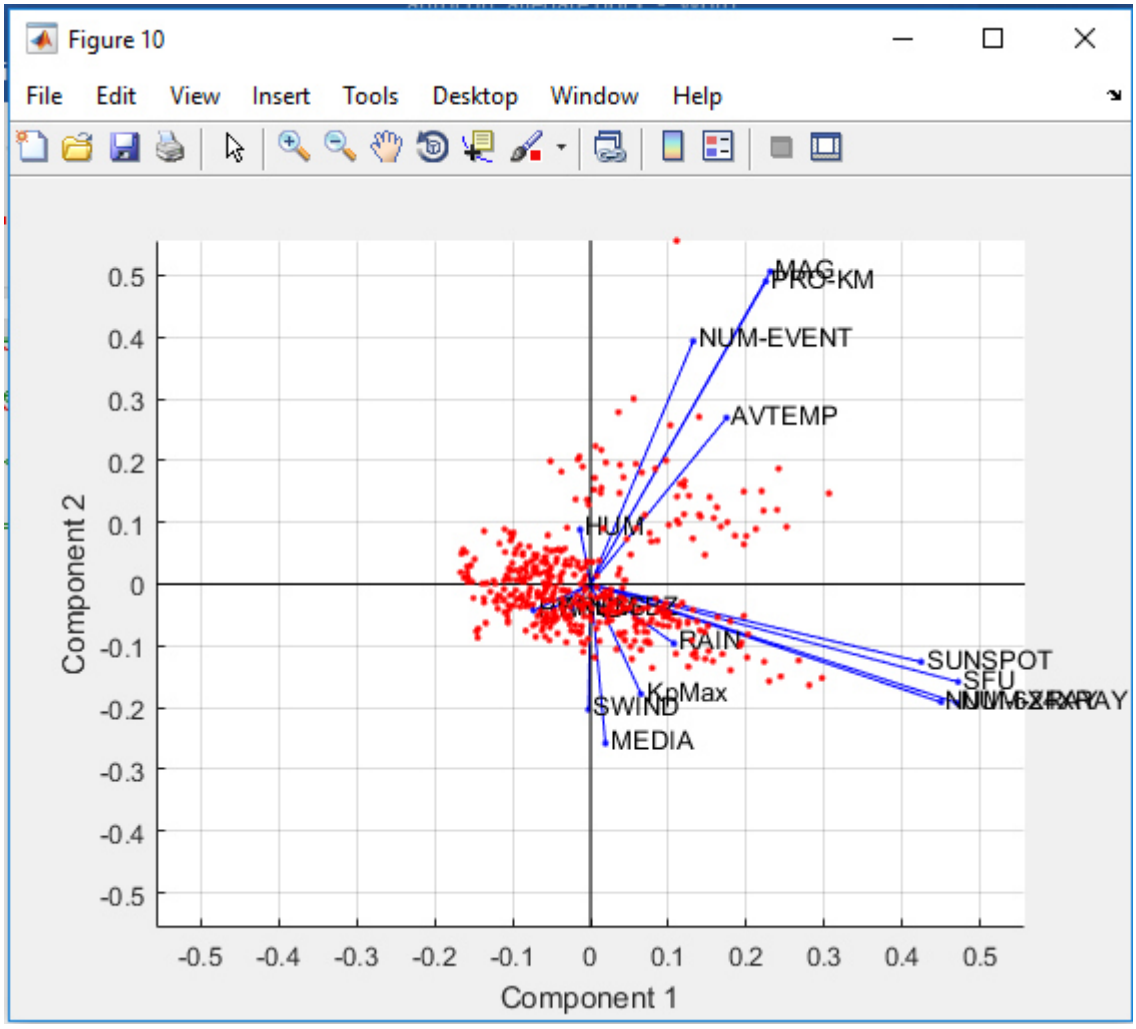
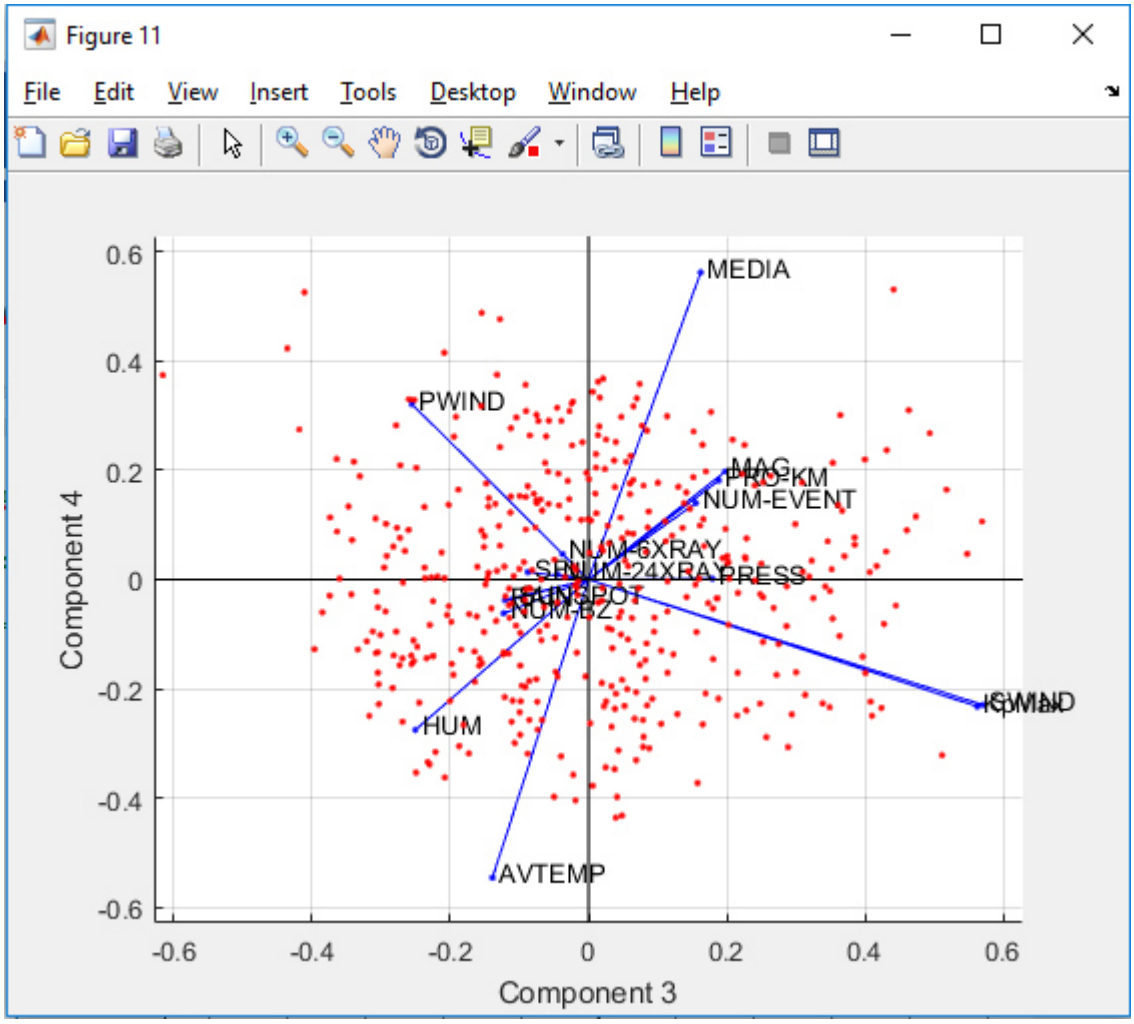


Grafico con la proiezione sulle componenti principali

Le coordinate delle variabili sulle componenti sono nient'altro che i coefficienti di correlazione delle variabili con le componenti principali. Esse rappresentano anche il contributo delle variabili all'informazione complessiva contenuta nelle componenti. Nell'interpretare il grafico le variabili più vicine all'origine non danno un contributo significativo alle componenti**. Le variabili con coordinata più alta in valore assoluto (correlazione più alta) sulla componente possono caratterizzare la componente in sede di interpretazione**.



Sulla Componente 1 le variabili con coordinata maggiore sono le variabili sismiche (MAG,NUM-EVENT,PROFKM). Quindi la prima componente la possiamo assegnare all'attività sismica: all'aumentare del numero di eventi sismici e della magnitudo la MEDIA (posizionata all'opposto rispetto all'origine) delle coincidenze diminuisce.
Su questo argomento ci sono studi che indicano che in aree vulcaniche le acque termali del sottosuolo subiscono un abbassamento della quantita' di gas radon disciolto durante l'attività vulcanica seguito da un aumento in stato di riposo. A tal proposito, l'abitazione in tufo presso cui è in funzione il rilevatore si trova su un vulcano (la solfatara) e pesca con le sue fondamenta da sorgenti termali che sfociano in mare, di fronte, a pochi metri di distanza. Si potrebbe concludere che l'abbassamento della concentrazione di radon in queste acque termali produce l'abbassamento delle coincidenze all'aumentare dell'attività vulcanica/sismica nell'area flegrea. Bisogna comunque dire che la concentrazione di radon dipende anche dalle piogge che vanno a riempire gli spazi all'interno delle strutture di tufo facendo ridurre le emissioni: occorrerebbe depurare il dato dai giorni di pioggia avuti nel periodo preso in considerazione.
La seconda componente principale è caratterizzata dall'attività solare (macchie solari, emissioni X, emissioni radio). Il grafico non mostra una relazione significativa (sono quasi ortogonali) con la media del rilevatore AMD5 ma ripeto: il rilevatore è in funzione da meno di un anno con un'attività solare abbastanza contenuta nel periodo.



Sulla terza componente risulta evidente la contrapposizione tra le variabili meteorologiche (AVTEMP, HUM) e la MEDIA per cui possiamo assegnare questa componente alle condizioni meteorologiche: all'aumentare di temperature e umidità la MEDIA diminuisce.

La quarta componente viene caratterizzata fortemente dall'attività solare (SOLARWIND e kpMAX contrapposte alla pressione dinamica PWIND). Viene confermato che la MEDIA non risulta particolarmente influenzata dall'attività solare nel breve periodo considerato.

Da notare che la variabile NUMBZ ossia il campo magnetico interplanetario ha coordinate piccole in tutte e 4 le componenti prese in considerazione. La variabile quindi non fornisce un soddisfacente apporto informativo alla variabilità globale dei dati.

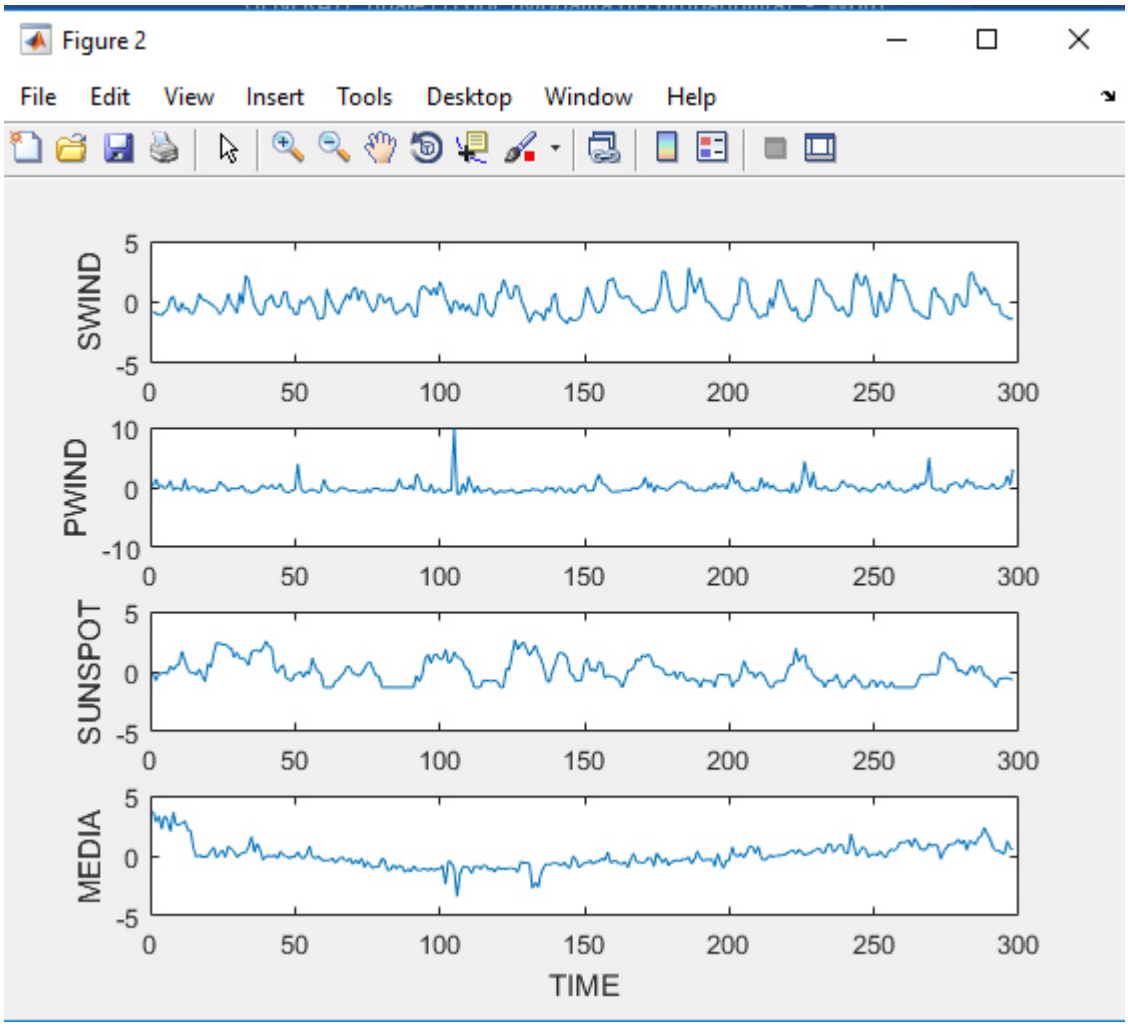
La variabile NUM_BZ non verrà considerata nel modello neurale preditivo.

SERIE STORICHE: AUTOCORRELAZIONE – CROSS CORRELAZIONE

Il Dataset di studio è costituito da **una serie storica** con dati raccolti quotidianamente. Nel cercare un modello generale basato su reti neuronali, occorre considerare l'eventuale ritardo temporale k con cui le variabili esplicative vanno ad influire sul comportamento della variabile MEDIA generata dal rilevatore al tempo *t*.

Nelle serie storiche si cerca di **individuare la struttura dei dati del passato (trend, periodicita') per proiettarla nel futuro** in sede di previsione. Generalmente le variabili hanno, come si dice, memoria di se stessa: ad esempio la MEDIA rilevata dalla macchina AMD5 di Pozzuoli, come si vedrà più avanti, ha una forte correlazione (**autocorrelazione**) con i suoi stessi valori dei periodi precedenti. Inoltre la MEDIA al tempo t ha una discreta correlazione, seppur inferiore all'autocorrelazione, con i valori al tempo *t-k* delle altre variabili prese in

considerazione **(cross-correlazione)** . Tale correlazione è ancora più evidente in situazioni in cui vi è una spiccata periodicità. Tutte le variabili prese in considerazione hanno una certa periodicità come si può vedere da questo grafico disegnato sui valori standardizzati :



Dobbiamo quindi considerare un'**autocorrelazione** della MEDIA (variabile indipendente) con se stessa su un ritardo kx e una **cross - correlazione** della MEDIA con le altre variabili su un ritardo kz.
I valori di **autocorrelazione** sono calcolati in Matlab con la funzione ***nncorr*** che genera questi valori andando a splittare la variabile con un ritardo k crescente e calcolando il coefficiente di correlazione sulla parte in comune per ogni ritardo (in figura k = 2 ossia 2 periodi di ritardo):

split autocorrelazione ritardo k=2

x_1	x_2	x_3	...	...	...	...	x_{n-2}	x_{n-1}	x_n		
		x_1	x_2	x_3	...	...	...	...	x_{n-2}	x_{n-1}	x_n

Parte comune su cui calcolare l'autocorrelazione - ritardo k=2

x_3	...	...	...	...	x_{n-2}	x_{n-1}	x_n
x_1	x_2	x_3	...	...	...	...	x_{n-2}

Per la **cross-correlazione**, si procede andando a splittare la variabile indipendente rispetto ad ognuna delle variabili esplicative (per k = 2):

split cross-correlazione ritardo k=2

x_1	x_2	x_3	...	...	...	...	x_{n-2}	x_{n-1}	x_n		
		z_1	z_2	z_3	...	...	...	...	z_{n-2}	z_{n-1}	z_n

Parte comune su cui calcolare la cross-correlazione - ritardo k=2

x_3	...	...	...	...	x_{n-2}	x_{n-1}	x_n
z_1	z_2	z_3	...	...	...	...	z_{n-2}

Con il sistema di analisi statistica denominata **regressione multivariata**, la funzione che approssima i dati è la seguente:

$$\begin{aligned} x_i &= a_1 x_{i-1} + a_2 x_{i-2} \ldots \\ &\quad + c_1 z_{i-1} + c_2 z_{i-2} \ldots \\ &\quad + b + \varepsilon_i \end{aligned}$$

dove la variabile al tempo i, X_i, dipende dai suoi valori nei periodi precedenti (con ritardi i-1 …i-2…) e dai valori delle altre variabili esplicative con i ritardi presi in considerazione (i-1 …i-2…). Il **modello neurale** arriva a risultati molto simili a questa funzione ma con maggiore velocità di elaborazione, con una soluzione di tipo non lineare e con maggiore capacità di avvicinarsi ai valori anomali.
Dalla funzione Matlab ***nncorr*** viene fuori un **vettore di coefficienti di autocorrelazione/cross-correlazione** per ciascuna variabile, con valori generalmente decrescenti mano a mano che il ritardo k aumenta (le variabili hanno memoria che tende a perdersi nel tempo specie quando esiste un trend**). Quando questi coefficienti hanno valori elevati, siamo in presenza di strutture di dati che sono stabili e ripetibili nel tempo.**
Il punto chiave è trovare il ritardo più significativo, in termini di valore di autocorrelazione, da associare alla MEDIA e i **ritardi più significativi di cross-correlazione** delle variabili esplicative vs. la MEDIA, che siano validi anche in termini **inferenziali** (generalizzazione dei risultati a partire da un campione).
Nel seguente codice Matlab viene **ricercato un valore soglia** che successivamente verrà applicato a ciascuna variabile per selezionare i ritardi più significativi.

1. Si effettuano **200 step di cross-correlazione** (***nncorr)*** su due variabile random con distribuzione normale e si ottengono **200 valori di cross-correlazione** che superano il valore più alto al 90% di ciascuna distribuzione ottenuta. Si considera **la mediana dei valori ottenuti come valore soglia** per la scelta dei ritardi delle variabili vere e proprie (*sigthresh95* nel codice).

```
rng('default') %imposta seme per variabile random
Ltrn = floor(0.90*(2*Ntrn-1)) % 535: indice per prendere successivamente i
valori di autocorrelazione piu' significativi (vedi oltre) sopra la soglia
del 90%; Ntrn è il numero di osservazioni del campione di addestramento

Nrep = 200 % 200==>numero di ripetizioni per generare la distribuzione dei
valori di autocorrelazione delle variabili random
rng(0)
for i = 1:Nrep
    n1 = zscore(randn(1,Ntrn),1); %vettore random a distribuzione normale
    n2 = zscore(randn(1,Ntrn),1); %vettore random a distribuzione normale
    xcorrn = nncorr( n1,n2, Ntrn-1, 'biased'); %CROSSCORRELAZIONE
```

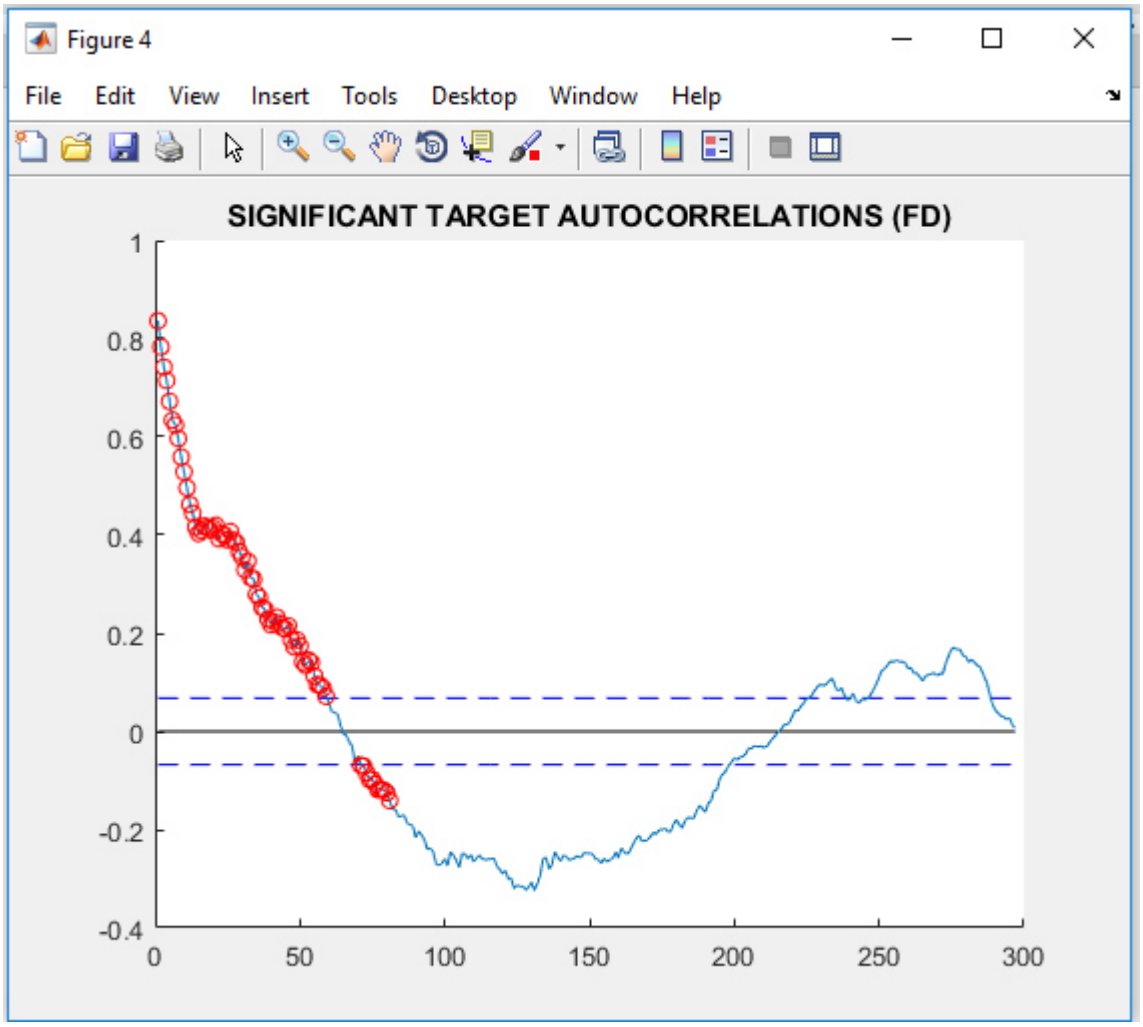
```
    sortabsxcorrn = sort(abs(xcorrn)); % ordina il vettore trovato
    thresh95(i) = sortabsxcorrn(Ltrn); % prende il valore più alto al 90%
allo step
end
minthresh95 = min(thresh95) % minimo
medthresh95 = median(thresh95) % mediana
meanthresh95 = mean(thresh95) % media
stdthresh95 = std(thresh95) % errore standard
maxthresh95 = max(thresh95) % massimo
sigthresh95 = medthresh95 % mediana scelta come soglia per le variabili
```

2. Nel codice seguente si opera **l'autocorrelazione (nncorr) sulla variabile MEDIA** e si selezionano i ritardi più significativi confrontando i valori di autocorrelazione trovati con il valore soglia. Infine si cercano i **salti sulla progressività numerica** dei ritardi significativi trovati: i salti avvengono nel punto in cui la serie dei ritardi scende sotto la soglia (vedremo più avanti cosa significa graficamente).

```
%autocorrelazione sulla MEDIA standardizzata
autocorrt = nncorr(zMEDIA_ttrn,zMEDIA_ttrn,Ntrn-1,'biased');
%confronto con il valore soglia sigthresh95
sigflag950 = find(abs(autocorrt(Ntrn+1:end))>=sigthresh95);
numsigflags = length(sigflag950)
find(diff(sigflag950) > 1) %trovo i salti nel vettore dei ritardi sigflag950
sigflag95 = sigflag950(1:70) %stampo i primi valori di autocorr > della
soglia
```

3. si genera **un grafico dove si evidenziano in rosso** i primi valori dei ritardi (in ascissa) significativi che sono oltre la soglia (in figura la soglia è segnata con linee tratteggiate). I salti trovati con l'istruzione *find* sono i valori degli ultimi ritardi significativi prima di andare sotto la soglia (tratto del grafico all'interno delle linee tratteggiate). Per la MEDIA possiamo scegliere un ritardo fino a 59 giorni prima del salto sotto soglia. **Per i primi 59 ritardi la struttura della variabile resta stabile.**

```
%grafico ritardi/autocorrelazione per la MEDIA
plt = plt+1, figure(plt);
hold on
plot(1:Ntrn-1, -sigthresh95*ones(1,Ntrn-1),'b--')
plot(1:Ntrn-1, zeros(1,Ntrn-1),'k')
plot(1:Ntrn-1, sigthresh95*ones(1,Ntrn-1),'b--')
plot(1:Ntrn-1, autocorrt(Ntrn+1:2*Ntrn-1))
plot( sigflag95, autocorrt(Ntrn+sigflag95), 'ro' )
title('SIGNIFICANT TARGET AUTOCORRELATIONS (FD)')
```



3. Si effettua **la cross-correlazione tra MEDIA e ciascuna variabile esplicativa** e si scelgono i ritardi significativi prima del salto sotto soglia come in precedenza. Si stampa il grafico con i primi valori significativi in rosso. Come si vede dal primo grafico, anche nel caso della variabile temperatura (AVTEMP) e nel caso delle macchie solari (SUNSPOT) **vi è un buon margine di ritardi per i quali la struttura delle variabili, unite alla MEDIA, resta stabile**.

```
%%%cross-correlazione INPUT/TARGET con nncorr
%zAVTEMP_trn
crosscorrx1t = nncorr( zAVTEMP_trn,zMEDIA_ttrn, Ntrn-1,'biased');
%trovo i valori significativi superiori alla soglia
sigi1lag950 = -1 + find(abs(crosscorrx1t( Ntrn:end ) ) >= sigthresh95);
numsigi1lags = length( sigi1lag950 ) % 468
find( diff( sigi1lag950 ) > 1) % trovo I salti nel vettore dei ritardi
sigflag950
sigi1lag95 = sigi1lag950(1:51) %stampo i primi valori di autocorr buoni
% grafico ritardi tra MEDIA e AVTEMP
plt = plt + 1, figure(plt);
hold on
plot( 0:Ntrn-1, -sigthresh95*ones(1,Ntrn), 'b--')
plot( 0:Ntrn-1, zeros(1,Ntrn), 'k')
plot( 0:Ntrn-1, sigthresh95*ones(1,Ntrn), 'b--')
plot( 0:Ntrn-1, crosscorrx1t(Ntrn:2*Ntrn-1))
plot( sigi1lag95, crosscorrx1t(Ntrn+sigi1lag95) ,'ro')
title( ' SIGNIFICANT zAVTEMP-TARGET CROSSCORRELATIONS (ID) ' )
```

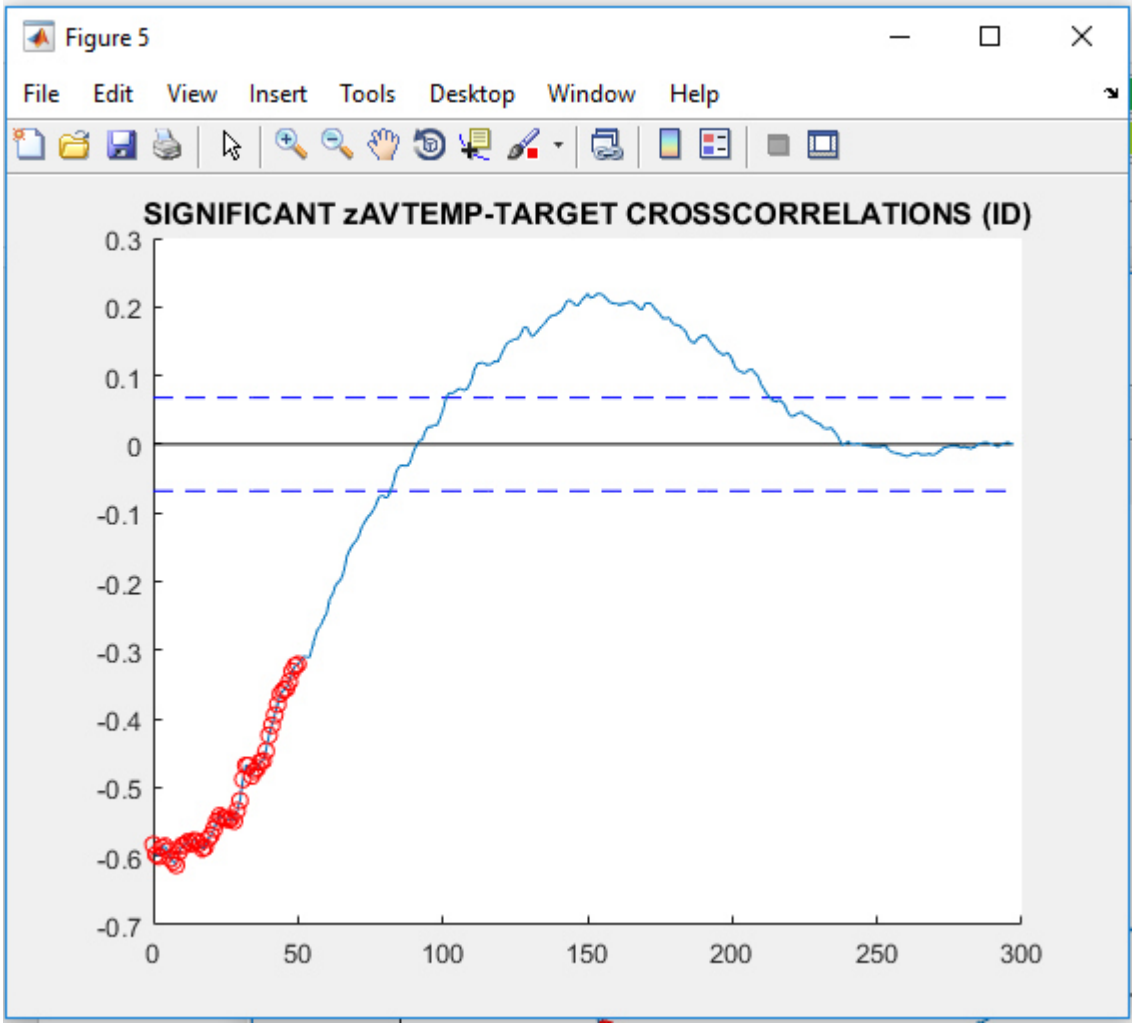


Figure 5
File Edit View Insert Tools Desktop Window Help
SIGNIFICANT zAVTEMP-TARGET CROSSCORRELATIONS (ID)
0.3
0.2
0.1
0
-0.1
-0.2
-0.3
-0.4
-0.5
-0.6
-0.7
0
50
100
150
200
250
300

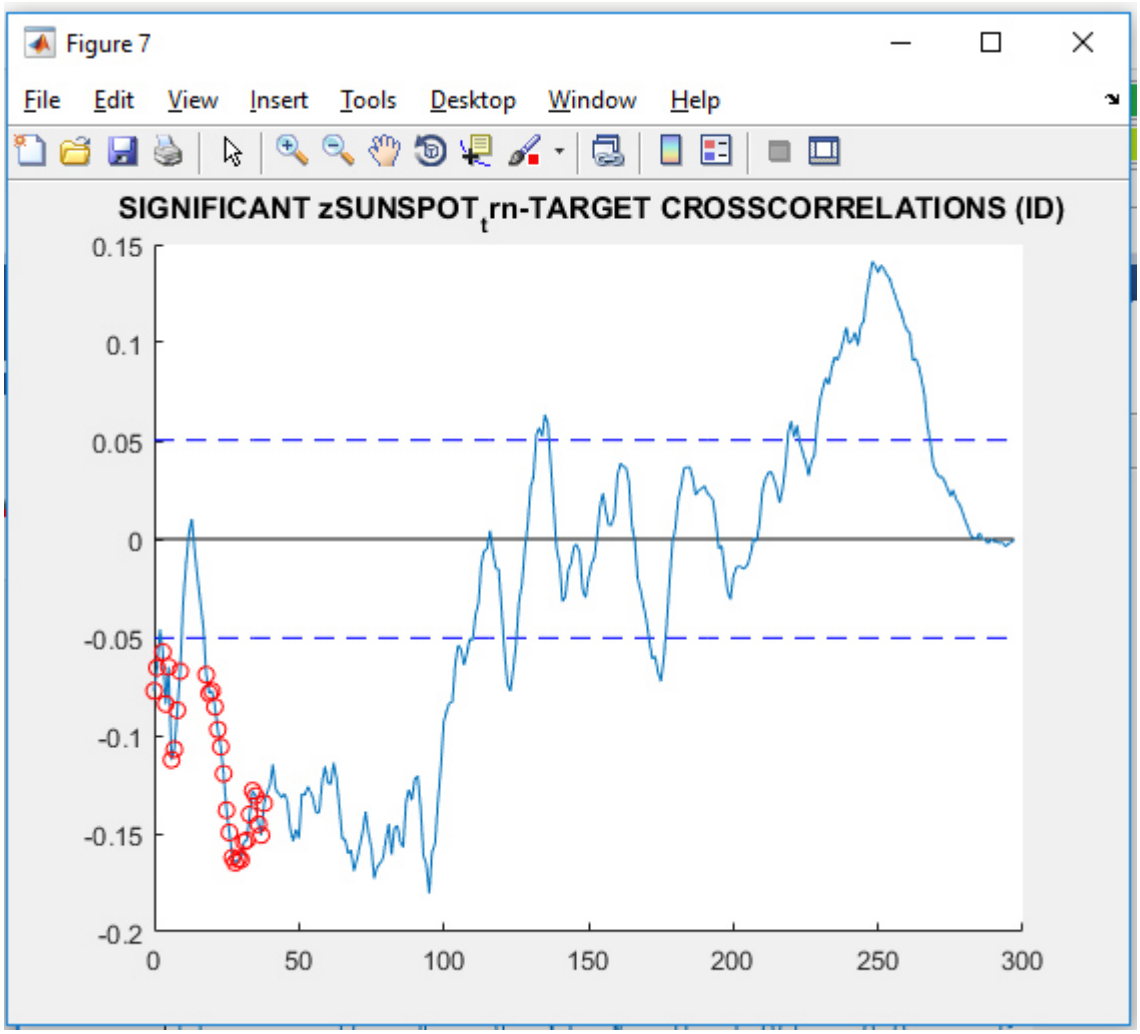
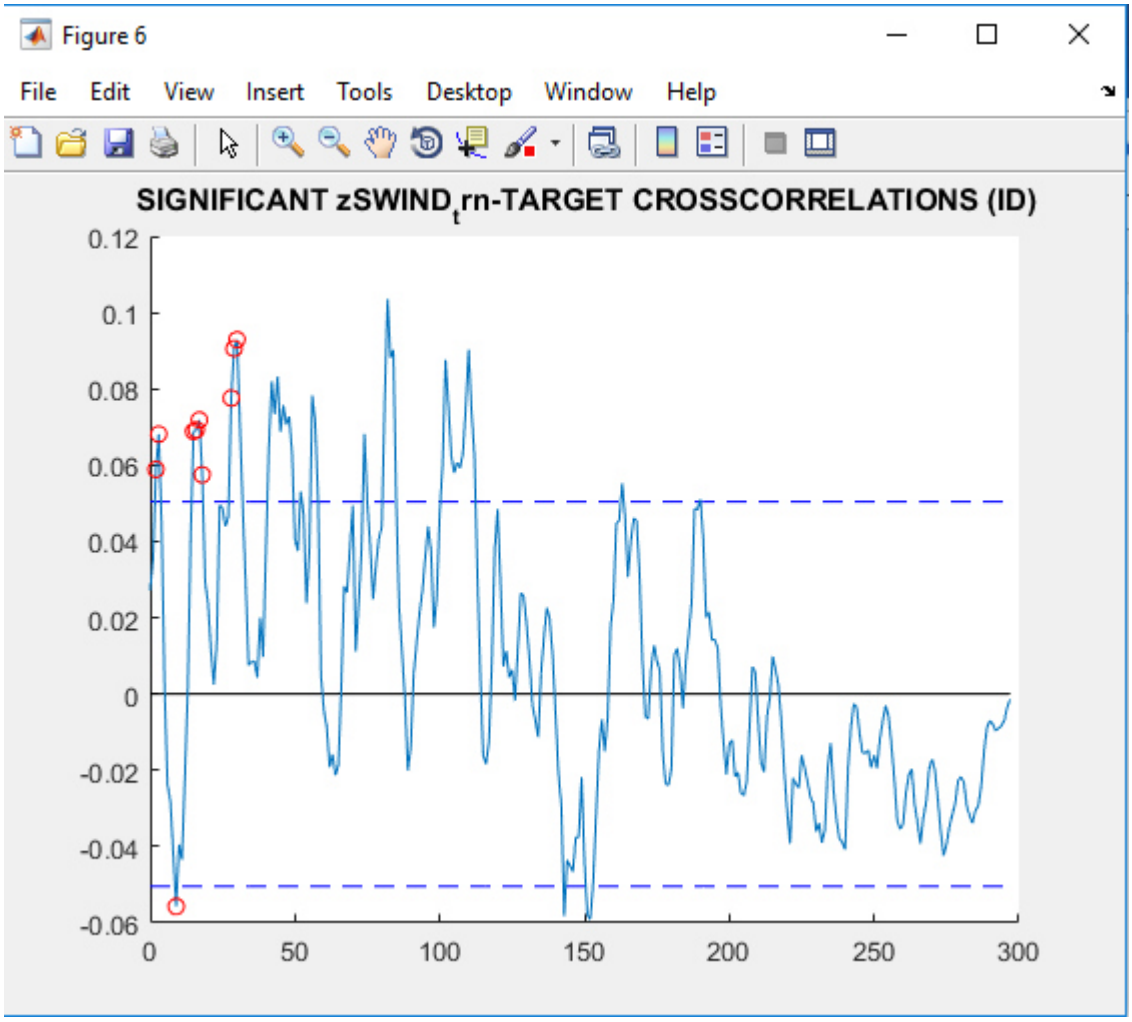


Figure 7
File Edit View Insert Tools Desktop Window Help
SIGNIFICANT zSUNSPOT$_t$rn-TARGET CROSSCORRELATIONS (ID)
0.15
0.1
0.05
0
-0.05
-0.1
-0.15
-0.2
0
50
100
150
200
250
300



Nell'ultimo grafico (solar wind) si nota che i ritardi significativi, prima di entrare sotto soglia, sono pochi. Quindi per questa variabile occorre scegliere un ritardo abbastanza contenuto.
A tal proposito il modello neurale di Matlab per le serie storiche ha il limite di assumere come parametro **un solo range di ritardi (ID)** per la cross-correlazione, comune a tutte le variabili. Quindi per la cross-correlazione si può scegliere solo **il ritardo minimo compatibile con tutte le variabili** considerate. Dato che il ritardo minimo per alcune variabili è di 2 giorni, avremo:

ID 1:2 **per le variabili esplicative**

FD 1:59 **per la MEDIA**

La scelta dei ritardi da 1 : 59, con una quantità di dati cosi' limitata (427 osservazioni x 12 variabili), risulta eccessiva perché porta ad una **riduzione notevole dei gradi di libertà**: nella ricerca di una funzione multidimensionale che dovrebbe prevedere i dati futuri, considerare +59+2 parametri vuol dire **ingessare il modello** rispetto alla sua capacità di generalizzazione. Come si vedrà più avanti, per una rete neurale che opera su serie storiche, i parametri (i pesi *w* della rete) crescono molto rapidamente: nel calcolo del numero di pesi/parametri dobbiamo, non solo considerare il numero di connessioni tra i neuroni, ma anche il numero di ritardi considerati che rappresentano essi stessi dei nodi di input alla rete (*num_pesi_w = **(ID***I + **FD***O + 1)*h + (h+1)*O*) . **Per questi motivi, andremo a considerare solo i primi 5 ritardi per la variabile MEDIA:**

ID 1:2

FD 1:5

RETI NEURONALI

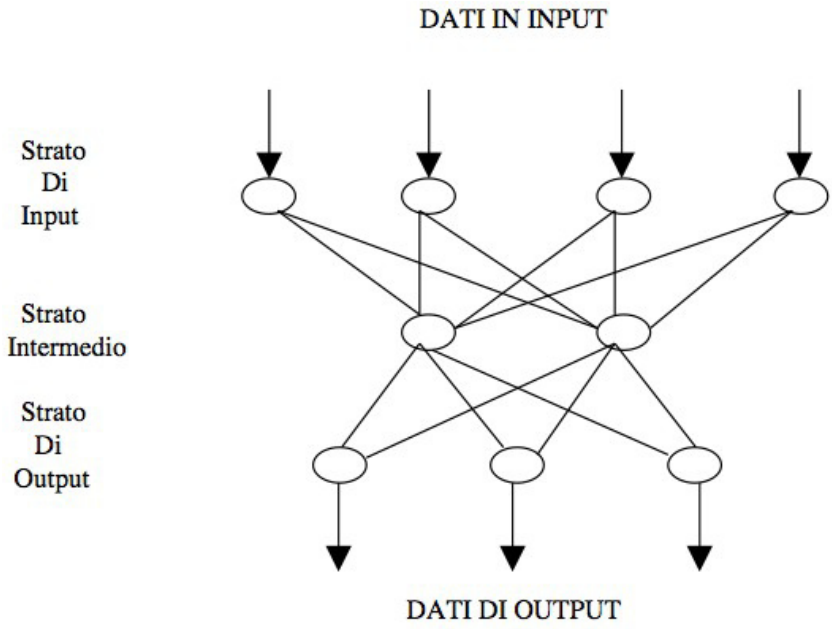
Le reti neurali artificiali somigliano alle reti neurali biologiche sotto diversi aspetti. Le reti neurali biologiche sono costituite da miliardi di cellule nervose (neuroni) interconnesse tra loro. L'input di un neurone è costituito dai segnali provenienti da altri neuroni connessi. Quando i segnali di input superano una certa soglia il neurone si attiva generando un segnale bioelettrico che si propaga verso altri neuroni.
Le reti neurali artificiali offrono una risposta in termini di:

- parallelismo dell'elaborazione come i neuroni biologici
- distribuzione della conoscenza che permette di bypassare neuroni danneggiati
- robustezza, per fornire una risposta coerente in output anche quando i dati di input sono parziali
- apprendimento dall'esperienza che permette alla rete di adattarsi continuamente alle nuove informazioni
- tipologia di modello non lineare

STRUTTURA DI UNA RETE NEURALE

Le reti neurali artificiali sono costituite da unità computazionali elementari (neuroni). Generalmente i neuroni artificiali sono organizzati in strati:

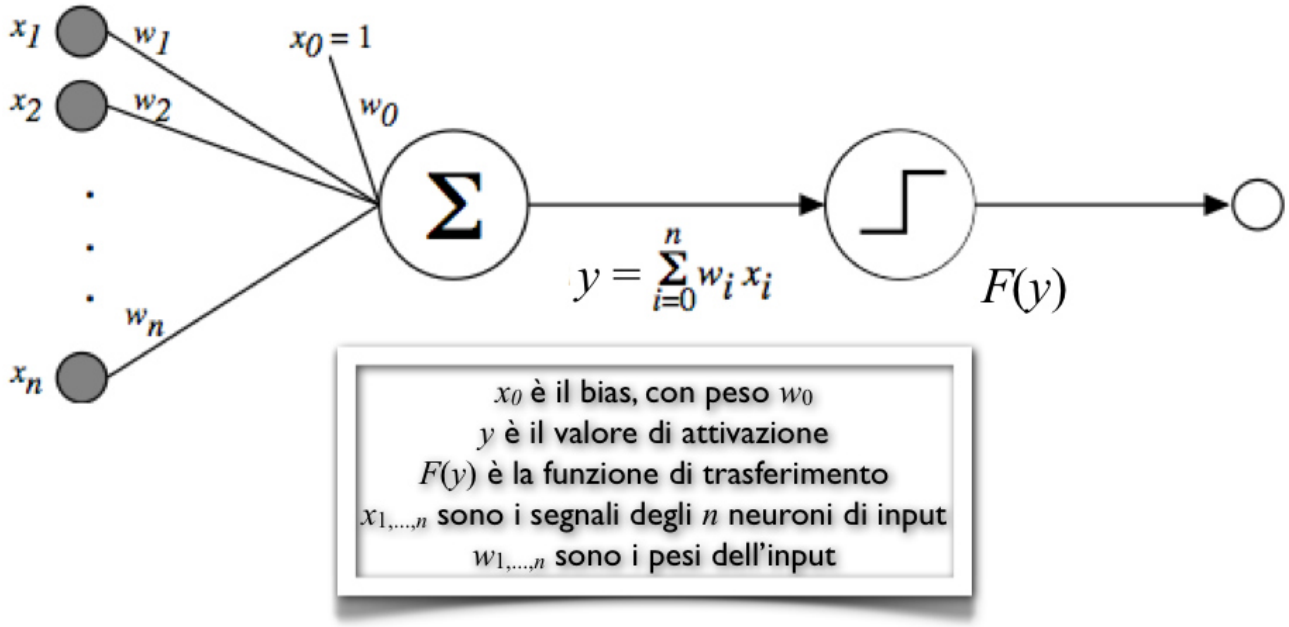
- Strato di INPUT di *n* neuroni pari al numero di input dei dati.
- Uno o più strati nascosti/intermedi di *m* neuroni
- Strato di output di *p* neuroni a seconda del numero di output voluti



In figura è possibile vedere una rete di tipo *feedforward* dove i segnali procedono soltanto dai neuroni di uno strato ai neuroni dello strato successivo.

MODELLO ELEMENTARE DI NEURONE ARTIFICIALE

Il modello di neurone artificiale proposto nel 1943 da McCulloch e Pitts è ancora valido ai giorni nostri:



Ogni neurone riceve *n* segnali di input dagli altri neuroni (vettore x di input) e applica una funzione di attivazione *y* dove intervengono i valori delle **connessioni detti pesi sinaptici *w*:** i pesi sono i parametri della funzione y. La funzione di trasferimento *F(y)* genera un valore di output che viene passato ad un neurone di un altro strato oppure costituisce il risultato della rete se il neurone considerato è un neurone di output.
I neuroni dello strato di input non ricevono in input i dati da elaborare da altri neuroni e non eseguono nessun calcolo passando i valori del data-set di studio direttamente ai neuroni degli strati intemedi.
La conoscenza appresa dalla rete è contenuta nei valori dei pesi sinaptici *w* che vengono determinati durante i processi di addestramento/apprendimento.

ALGORITMI DI ADDESTRAMENTO DELLA RETE

L'addestramento della rete si effettua a partire da un data-set composto da *n* variabili di input e una o più variabili di output/target per *p* casi di studio (per semplificare consideriamo un'unica variabile di output):

VARIABILI INPUT					**VAR DI OUTPUT TARGET**
x_{11}	x_{12}	.	.	x_{1n}	y_1
x_{21}	x_{22}	.	.	x_{2n}	y_2
.	.	.	.	.	.
.	.	.	.	.	.
X_{p1}	x_{p2}	.	.	x_{pn}	y_p

Per ogni vettore di *n* variabili *x* passate allo strato di input (una riga della matrice di INPUT), la rete produce un valore in output o_i (uscente dalla funzione di trasferimento del neurone di output). Tale output va confrontato con il valore target y_i del data-set di addestramento determinando il **valore di errore $e_i = y_i - o_i$**

Gli algoritmi di addestramento hanno come obiettivo la ricerca del minimo della funzione somma del quadrato degli errori (SSE: sum of square errors) modificando ad ogni step i pesi sinaptici *w*:

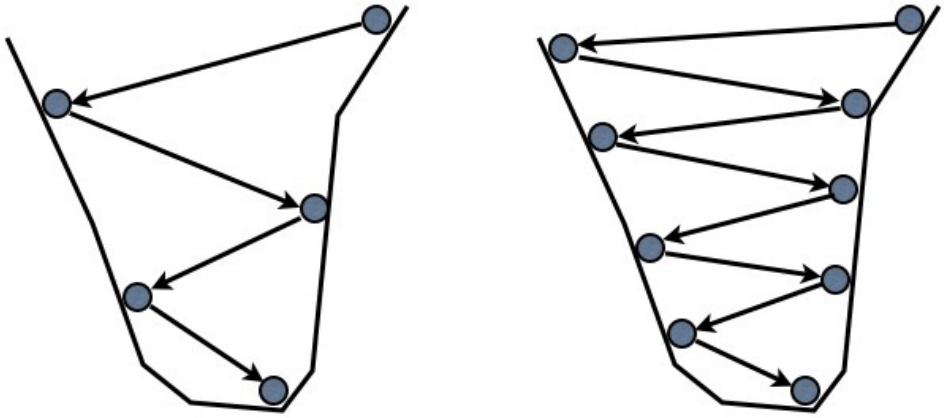
FUNZIONE SSE

$$E(\boldsymbol{x},|\boldsymbol{w}) = \frac{1}{2}\sum_{p=1}^{P}\sum_{m=1}^{M} e_{p,m}^2$$

Dove ***x*** è il vettore di input;
w è il vettore dei pesi delle connessioni;
$\boldsymbol{e_{p,m}}$ è l'errore calcolato sull'output *m* (nel nostro caso *m = 1* perché abbiamo un solo neurone di output) relativo al caso di studio *p* ed è uguale alla differenza:

$$\boldsymbol{e_{p,m} = y_{p,m} - o_{p,m}}$$

dove:
y è il vettore degli output target;
o è il vettore degli output generato dalla rete ossia dalla funzione di trasferimento dello strato di output.



In Figura è possibile vedere il risultato sulla funzione dell'errore SSE di due diversi algoritmi di addestramento: **ad ogni step di addestramento (pallini grigi) si modificano i pesi *w* secondo determinati criteri producendo la convergenza verso il minimo della funzione SSE** (Il primo algoritmo converge più velocemente del secondo).
Per l'addestramento della rete neurale artificiale vi sono vari algoritmi che si differenziano per come si modificano i valori dei pesi sinaptici ***w*** ad ogni step.

Il metodo del gradiente discendente modifica i pesi *w* allo step *k+1* con l'equazione:

$$w_{k+1} = w_k - \alpha g_k \qquad \textbf{(1)}$$

Dove α si chiama tasso di apprendimento e *g* è il gradiente/pendenza con segno negativo (derivata di primo ordine della funzione SSE):

$$g = \frac{\partial E(x,w)}{\partial w}$$

Tale metodo ha diversi inconvenienti tra cui la difficoltà nella scelta dell'alfa che è determinante nella velocità con cui la funzione converge al minimo. In certe situazioni l'algoritmo del gradiente discendente può addirittura divergere rispetto al minimo.

Il **metodo di Newman** modifica i pesi ***w*** allo step ***k+1*** con l'equazione:

$$w_{k+1} = w_k - H_k^{-1} g_k \qquad \textbf{(2)}$$

dove ***H*** è la matrice Hessiana delle derivate seconde della funzione somma dell'errore quadratico SSE: con la hessiana si ottiene un adeguamento del tasso di apprendimento (nel metodo precedente il tasso α era fisso) a seconda del punto della funzione in cui ci troviamo allo step k. Il problema con l'algoritmo di Newman è la difficoltà di calcolo della matrice hessiana delle derivate seconde della funzione ***E(x, w).***

Il **metodo Gauss-Newton** modifica i pesi ***w*** allo step ***k+1*** con l'equazione:

$$w_{k+1} = w_k - \left(J_k^T J_k\right)^{-1} J_k e_k \qquad \textbf{(3)}$$

dove la matrice ***J*** è una jacobiana che permette una buona approssimazione della hessiana utilizzando derivate prime anziché derivate seconde.

L'algoritmo Gauss-Newton risolve il problema della lentezza nel convergere al minimo della funzione SSE, ma per funzioni complesse nello spazio degli errori, l'algoritmo può portare a divergere dal minimo.

Il **metodo Levemberg – Marquardt (USATO IN MATLAB)** modifica i pesi ***w*** allo step ***k+1*** con l'equazione:

$$w_{k+1} = w_k - \left(J_k^T J_k + \mu I\right)^{-1} J_k e_k \qquad \textbf{(4)}$$

Rispetto all'algoritmo precedente viene introdotto il parametro μ mentre ***I*** è la matrice identità.

Il coefficiente μ è variabile ed è stato introdotto allo scopo di evitare che l'algoritmo diverga rispetto al minimo della SSE:

1. Quando uno step dell'algoritmo porta ad un valore della funzione dell'errore migliore, allora il parametro μ viene ridotto per far aumentare la velocità di convergenza (la variazione dei pesi diventa simile al metodo Gauss – Newton);
2. Quando invece il valore della funzione dell'errore peggiora (diverge dal minimo), il parametro μ viene aumentato per ridurre la variazione dei pesi allo scopo di cercare un valore della funzione SSE migliore in una posizione più vicina al punto del precedente step: ad esempio, se lo step peggiorativo ha prodotto un salto fuori dalla zona di minimo della SSE, si riprova con una variazione più piccola nelle vicinanze del punto sulla funzione raggiunto con lo step precedente.

Gli step dell'addestramento sono:

1. Si parte da pesi sinaptici w_k scelti casualmente e si fa lavorare la rete sui dati di addestramento;
2. Si calcolano gli errori ***e;*** la funzione SSE ***E;*** e la Jacobiana ***J***;
3. Si modificano i pesi w_{k+i} utilizzando un valore di μ iniziale casuale (oppure un valore determinato negli step successivi)
4. Si calcola la funzione SSE ***E*** con i pesi modificati.
5. Se l'errore è diminuito ($E_{k+1} < E_k$) allora μ viene diviso per 10 per accelerare la convergenza (nella (4) il termine in parentesi è elevato a meno uno quindi esso aumenta quando diminuiamo μ).
6. Se L'errore è aumentato, μ viene moltiplicato per 10 per rallentare la convergenza.
7. Si ritorna al passo numero 2
8. L'algoritmo si ferma quando la funzione SSE ***E*** raggiunge un valore minimo di soglia oppure quando SSE inizia ad aumentare sul campione "validation test" di cui si parlerà più oltre.

Preparazione dei dati per la rete

Con il codice matlab seguente:

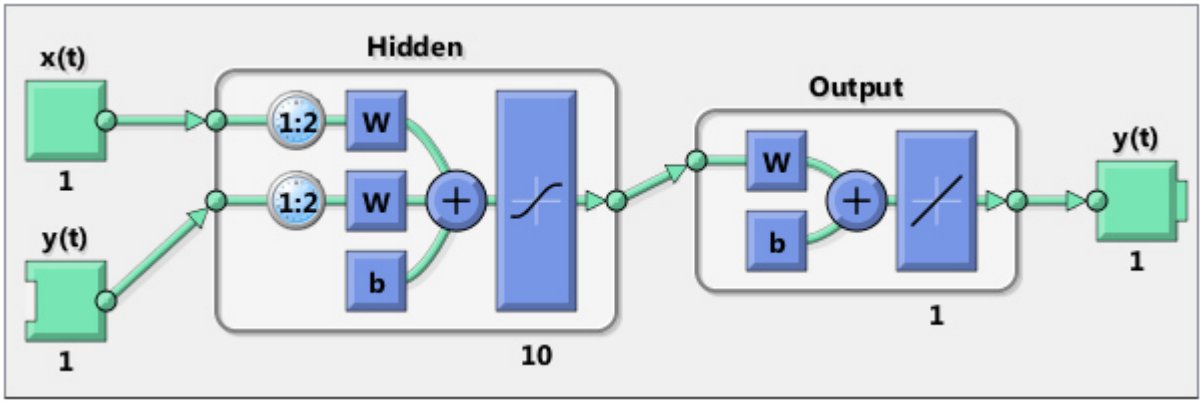
1. si leggono I dati da file excel e si crea la matrice `DATI_COSMICI`
2. si genera un vettore per ogni variabile considerata
3. si crea una tabella x (minuscolo) con tutte le variabili selezionate in precedenza: abbiamo escluso `NUM24XRAY, NUMBZ, SFU, KpMax`

4. si trasformano le tabelle in formato Matlab Cell-Array**: tabelle T e X che contengono i dati per l'addestramento della rete.**

```
%% IMPORTA DATI DA EXCEL
[~, ~, raw0_0] =
xlsread('D:\Templare\software\MATLAB\bin\MYPROG\A_DATA_ANALYSIS.xls','DATI_ASTRO
PARTICELLE','C2:R427');
raw = [raw0_0];
raw(cellfun(@(x) ~isempty(x) && isnumeric(x) && isnan(x),raw)) = {''};

%% ESCLUDI RIGHE CON DATI NON NUMERICI
I = ~all(cellfun(@(x) (isnumeric(x) || islogical(x)) && ~isnan(x),raw),2); %
raw(I,:) = [];

%% DATASET DI LAVORO
DATI_COSMICI = reshape([raw{:}],size(raw));

%% VETTORI VARIABILI SINGOLE
MEDIA = DATI_COSMICI(:,1);
AVTEMP = DATI_COSMICI(:,2);
PRESS = DATI_COSMICI(:,3);
HUM = DATI_COSMICI(:,4);
RAIN = DATI_COSMICI(:,5);
SWIND = DATI_COSMICI(:,6);
PWIND = DATI_COSMICI(:,7);
SUNSPOT = DATI_COSMICI(:,8);
SFU = DATI_COSMICI(:,9);
KpMax = DATI_COSMICI(:,10);
MAG = DATI_COSMICI(:,11);
PROKM = DATI_COSMICI(:,12);
NUMEVENT = DATI_COSMICI(:,13);
NUM6XRAY = DATI_COSMICI(:,14);
NUM24XRAY = DATI_COSMICI(:,15);
NUMBZ = DATI_COSMICI(:,16);

plt = 0;
x= DATI_COSMICI(:,2:8)'; %TABELLA CON VARIABILI INPUT IN ORIZZONTALE
x= [x;DATI_COSMICI(:,11:13)']; % aggiungo variabili 11 - 13
x= [x;DATI_COSMICI(:,15)'];   % aggiungo variabile 15
%TABELLE IN FORMATO CELL-ARRAY DA SOTTOPORRE ALLA RETE
X  = num2cell(x,1);     %trasformo la marice x in cell array
T = num2cell(MEDIA',1); %trasformo il vettore MEDIA in cell array
```

MODELLO PREDITTIVO CON RETI NEURALI

Matlab mette a disposizione un tool chiamato Neural Network Toolbox che include molte funzioni per il disegno e l'addestramento di reti neurali.
La tipologia di rete che più si adatta **alle serie storiche** si chiama in MATLAB **NARX network** ed ha una struttura schematizzata del tipo:



Questa **struttura** ha uno strato nascosto con 10 neuroni che svolgono una funzione di trasferimento sigmoide e uno strato di output con un neurone che svolge una funzione lineare.
Nella figura di esempio i valori delle variabili esplicative x (con ritardo ID=1:2 ossia x_{t-1} e x_{t-2}) e i valori della variabile target y (con ritardi FD=1:2 ossia y_{t-1} e y_{t-2})) sono passati a 10 neuroni hidden che a loro volta passano i risultati della loro funzione di trasferimento al nodo di output che genera il valore y_t.
Matematicamente la rete equivale ad una funzione del tipo:
$y(t)=F(y(t-1),y(t-2),\ldots,y(t-ky),x(t-1),x(t-2),\ldots,x(t-kx))$
In una rete neurale, un po' come avviene per l'analisi di regressione, se aumentiamo il numero di nodi, e quindi il numero di pesi w che li uniscono, diminuiscono i gradi di libertà. Il sistema si adatta in modo troppo perfetto ai dati (**OVERFIT**) per cui non è in grado di prevedere o comunque avvicinare i valori anomali**. La rete perde di capacità di generalizzazione**.
Quindi, nel disegno di una rete, il primo obiettivo è decidere il numero dei nodi nascosti. (Hidden)
Per fare cio' si procede **all'addestramento di un certo numero di reti con una quntità di nodi via via crescenti** (quindi cresce anche il numero di pesi che uniscono i nodi) fino al limite in cui il numero dei pesi w è uguale al numero delle osservazioni del dataset. Si raccolgono progressivamente i parametri di performance delle reti e infine **si sceglie quella rete che ha una discreta performance con il numero minimo di nodi.**
Con ID ed FD rispettivamente i ritardi considerati per la MEDIA e i ritardi per le variabili esplicative, calcoliamo il numero massimo di nodi (Hmax nel codice) per cui **il numero complessivo dei pesi è uguale al numero di osservazioni (Nw = Ntrneq);** il numero di pesi sarà Nw = (NID*I + NFD*O + 1)*h + (h+1)*O:

```
NID = length(ID); %     numero ritardi MEDIA
NFD = length(FD); %     numero ritardi applicati alle variabili esplicative
Hmax = (Ntrneq-O)/(NID*I+NFD*O+O+1) % massimo numero di nodi oltre il quale  si
va in overfit  -> Nw = Ntrneq

%Ntrneq è il numero di osservazioni;
%O è il numero di nodi di output (nel nostro caso 1)
%I è il numero di INPUT (LE VARIABILI ESPLICATIVE)
%NID e NFD sono il numero di ritardi considerati per la MEDIA e per le variabili
esplicative.
% il numero di pesi è Nw = ( NID*I + NFD*O + 1)*h + (h+1)*O;
```

A questo punto per ogni rete con un numero di nodi hidden che va da 0 ad `Hmax`**:**

- si disegna la rete con il comando ***narxnet***;
- si preparano i dati di input in modo che considerino anche i ritardi stabiliti sulle variabili, con la funzione ***preparets***
- si sceglie un valore minimo target da raggiungere per la funzione dell'errore quadratico medio MSEgoal: lo 0.001 della variabilità (informazione) totale della MEDIA.
- Si addestra 10 volte la rete con il comando ***train*** e si salvano i dati di **performance NMSEo e R2a (mean square error; R square)** per confrontarli successivamente con i dati delle altre reti con numero di nodi maggiore.

- **Si passa alla prossima rete con un nodo hidden in più** e si ripete: il disegno della struttura, i 10 addestramenti e il salvataggio delle performance nelle matrici NMSEo ed R2a (mean square error; R square)

```
for h = Hmin: dH: Hmax   %per numero nodi crescente con passo 1 fino ad HMAX
    j = j+1
    trainFcn = 'trainlm';     %algoritmo di addestramento Levemberg-Marquardt
    neto = narxnet( ID, FD, h ); %disegno della rete
    Nw = ( NID*I + NFD*O + 1)*h + (h+1)*O;      %numero di pesi; h = numero di
nodi
    Ndof = Ntrneq-Nw    %numero gradi di libertà: numero osservazioni meno
numero pesi
%Training, Validation, Testing
    neto.divideFcn = 'dividerand';        %tipo divisione dei dati
    neto.divideParam.trainRatio = 70/100; % percentuale dati di addestramento
    neto.divideParam.valRatio = 15/100;   %percentuale dati di validazione
    neto.divideParam.testRatio = 15/100;  %percentuale dati di test

    %preparazione dati con i ritardi per l'addestramento;
    [Xo Xoi Aoi To] = preparets(neto,X,{},T);
    if j == 1, to = cell2mat(To), varto1 = var(to,1),
      varto0 = var(to,0), end
      MSEgoal = 0.001*max( Ndof, 0 )*varto0/Ntrneq %MSE (mean square error)
OBIETTIVO PER LA RETE
      MinGrad = MSEgoal/100
      neto.trainParam.goal = MSEgoal;
      neto.trainParam.min_grad = MinGrad;
      for i = 1:Ntrials  %Ntrials = 10 : 10 addestramenti della rete per ogni
nodo in più
          s(i,j) = rng;
          neto = configure(neto,Xo,To);
          [neto tro Yo Eo Xof Aof ] = train(neto,Xo,To,Xoi,Aoi); %ADDESTRAMENTO
          stopcriterion{ i, j } = tro.stop;
          numepochs (i ,j ) = tro.num_epochs;
          NMSEo( i, j ) = mse(Eo)/varto1;       %SALVATAGGIO DATI PERFORMANCE MSE
          R2a(i,j) = 1-mse(Eo)/varto1;          %SALVATAGGIO DATI PERFORMANCE R2
      end
  end
```

Il codice seguente **stampa i dati di performance** con cui operare la scelta della dimensione ottimale della rete. Si seleziona una struttura che abbia buone performance con un numero minimo di nodi:

```
result1 = [ (Hmin:dH:Hmax); R2a ]
minNMSE0 = min(NMSEo)
medNMSE0 = median(NMSEo)
meanNMSE0 = mean(NMSEo)
stdNMSE0 = std(NMSEo)
maxNMSE0 = max(NMSEo)
minR2 = min(R2a)
medR2 = median(R2a)
meanR2 = mean(R2a)
stdR2 = std(R2a)
maxR2 = max(R2a)
```

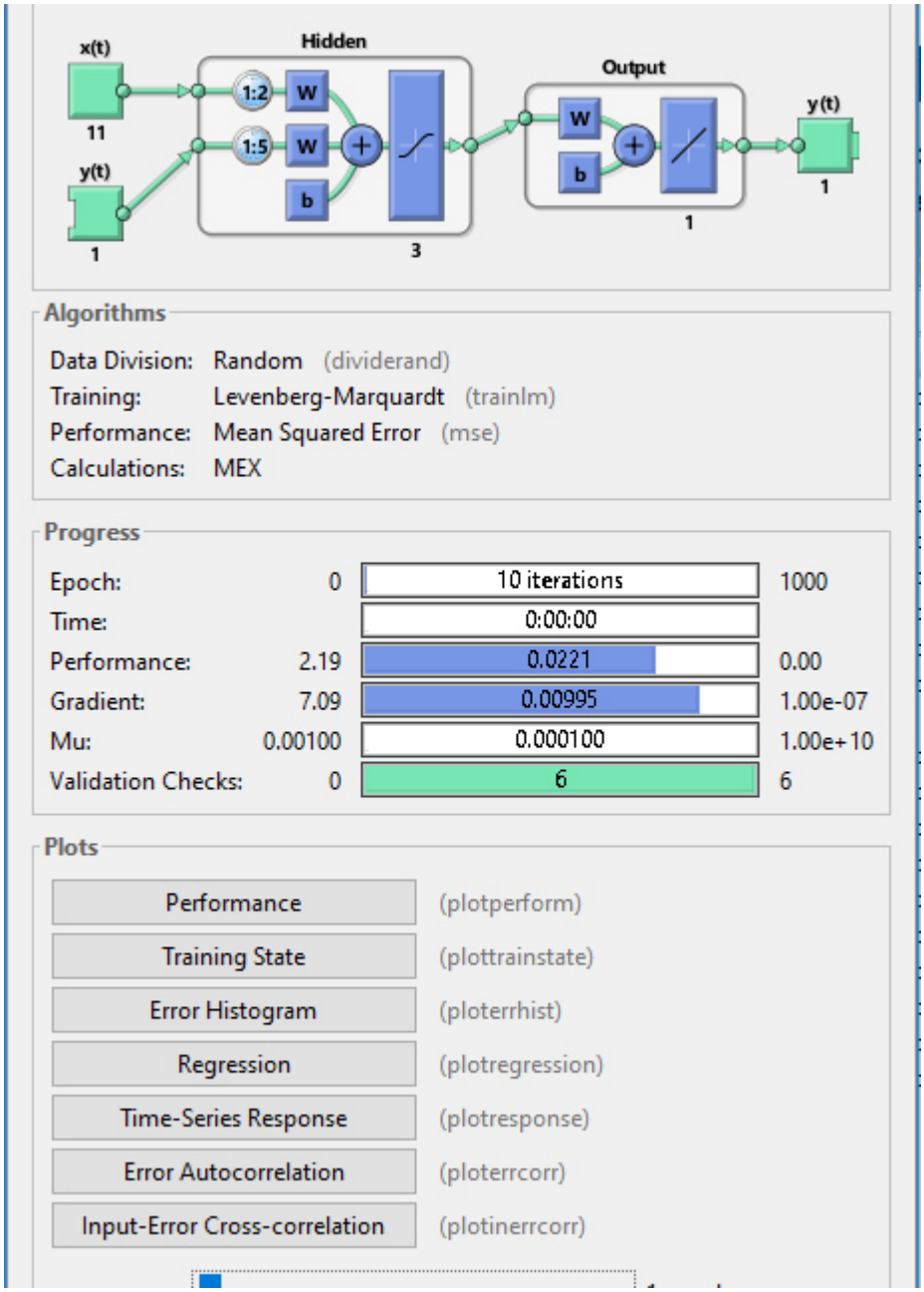
Ed ecco i **valori R square per ogni rete** con numero di nodi hidden che va da 0 a 10 (in giallo) e per n° 10 addestramenti (in verde). In coda alla tabella vi sono i valori di media e mediana:

	NUM NODI	0	1	2	3	4	5	6	7	8	9	10
NUM ADDEST.		R2	R2	R2	R2	R2	R2	R2	R2	R2	R2	R2
1	R2	6.8093e-01	6.8495e-01	6.9612e-01	6.8404e-01	6.3363e-01	7.1846e-01	7.1594e-01	7.5224e-01	7.2913e-01	7.4923e-01	6.8587e-01
2	R2	6.9072e-01	6.5655e-01	6.9787e-01	7.3709e-01	6.2639e-01	6.4488e-01	7.3187e-01	7.2275e-01	7.0436e-01	6.3528e-01	7.2049e-01
3	R2	6.8562e-01	6.9197e-01	7.0043e-01	7.2674e-01	6.2952e-01	6.9860e-01	7.2182e-01	7.3972e-01	7.0302e-01	7.4938e-01	6.9938e-01
4	R2	6.8513e-01	6.7917e-01	6.1827e-01	7.0798e-01	6.9728e-01	7.7049e-01	6.7567e-01	6.8136e-01	7.4093e-01	6.9580e-01	7.4531e-01
5	R2	6.9015e-01	6.8887e-01	6.8752e-01	7.0923e-01	7.3429e-01	7.4675e-01	7.0777e-01	7.2829e-01	7.8285e-01	4.7890e-01	6.2847e-01
6	R2	6.8716e-01	6.7716e-01	6.8554e-01	7.2682e-01	7.1500e-01	7.0781e-01	6.8386e-01	6.7060e-01	6.9446e-01	7.7862e-01	7.4625e-01
7	R2	6.7290e-01	6.8967e-01	6.8480e-01	7.0488e-01	7.0266e-01	6.8849e-01	6.9117e-01	7.2254e-01	6.9750e-01	7.7836e-01	6.0608e-01
8	R2	6.8222e-01	6.8761e-01	6.7134e-01	6.8105e-01	7.1969e-01	7.5106e-01	7.1264e-01	7.1793e-01	7.7601e-01	8.0677e-01	7.1697e-01
9	R2	6.8753e-01	6.7447e-01	7.0107e-01	7.0347e-01	7.3095e-01	7.1705e-01	6.5714e-01	7.4975e-01	6.7526e-01	7.5282e-01	6.5522e-01
10	R2	6.9193e-01	6.8348e-01	6.9682e-01	6.2430e-01	6.9001e-01	7.1148e-01	7.2236e-01	6.4949e-01	7.7078e-01	7.2403e-01	7.9734e-01
	NUM NODI	0	1	2	3	4	5	6	7	8	9	10
medR2		6.8639e-01	6.8421e-01	6.9182e-01	7.0643e-01	6.9997e-01	7.1426e-01	7.1020e-0	7.2264e-01	7.1674e-01	7.4930e-01	7.0818e-01
meanR2		6.8543e-01	6.8139e-01	6.8398e-01	7.0056e-01	6.8794e-01	7.1551e-01	7.0202e-01	7.1347e-01	7.2743e-01	7.1492e-01	7.0014e-01

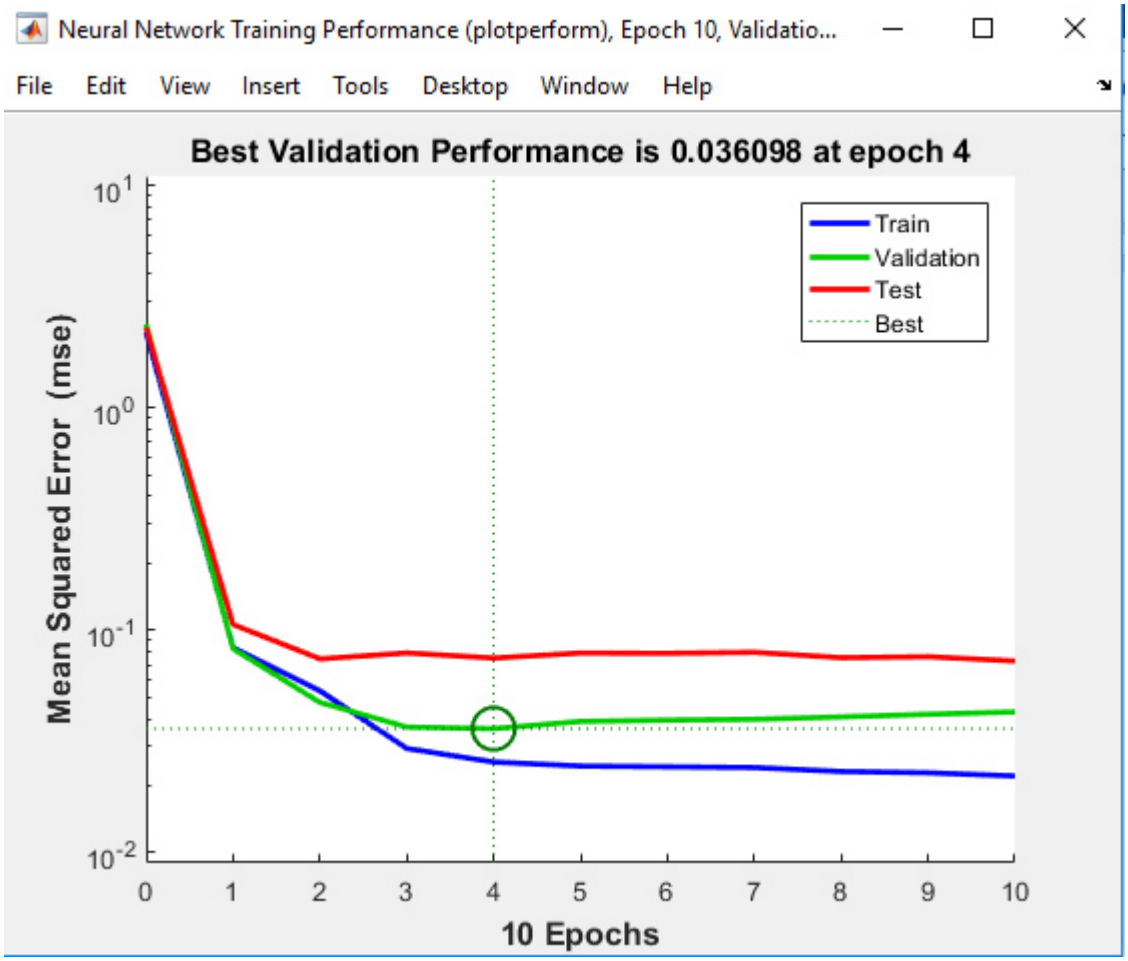
Esaminando la tabella si può notare che a partire dalla rete con 3 nodi nascosti, i valori di media e mediana di R2 sono superiori a 0.70 (la rete spiega il 70% della variabilità totale dei dati). Dato che per le reti con oltre 3 nodi, l'incremento del valore di R2 è abbastanza contenuto, **scegliamo una rete con 3 nodi Hidden.**

La schermata seguente mostra il processo di apprendimento con 3 neuroni mentre viene sviluppato da Matlab: in figura si notano il numero di iterazioni eseguite, il gradiente, il valore di μ etc.

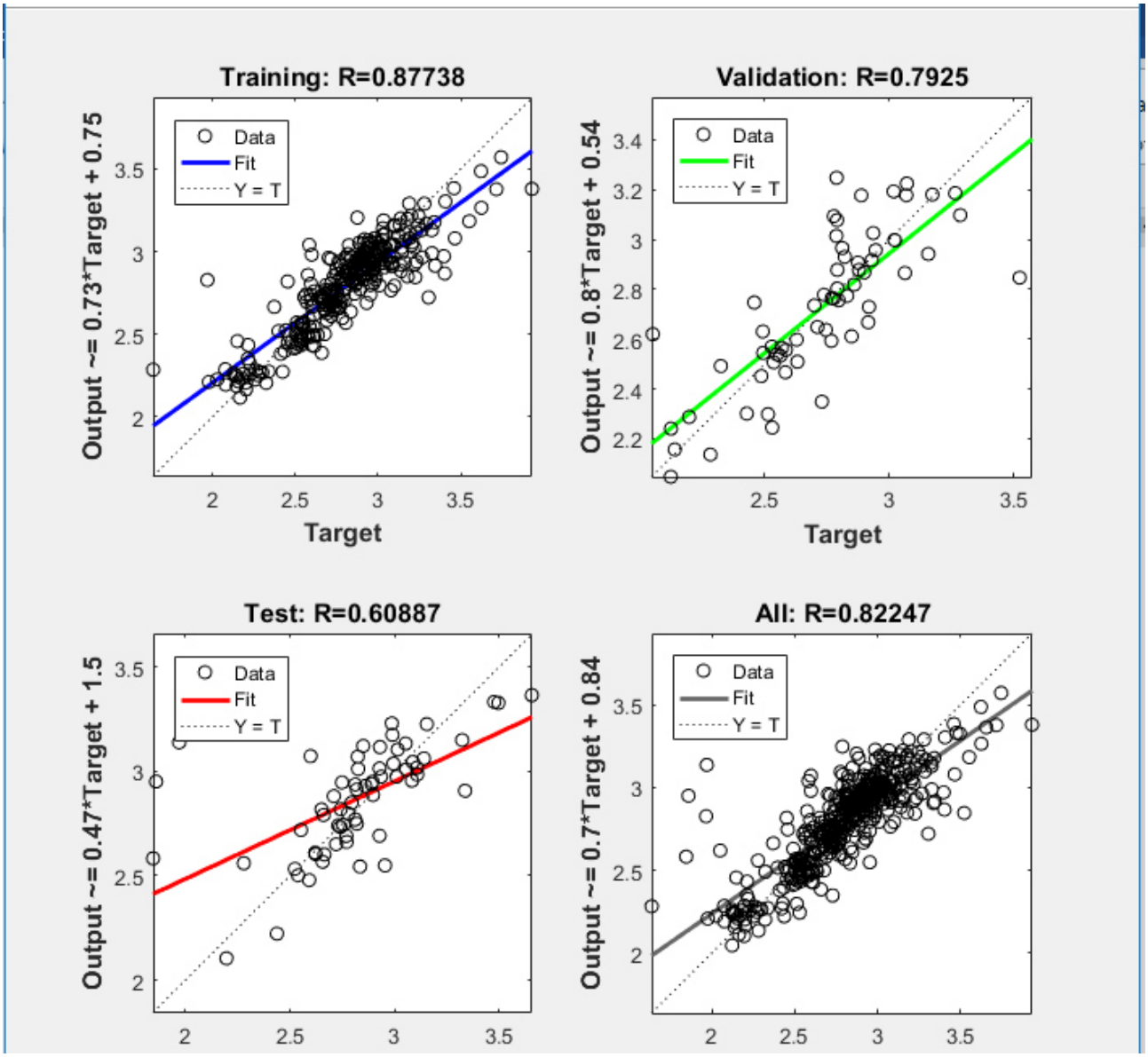
Ma soprattutto questa schermata mette a disposizione alcuni tasti che generano dei **grafici sulle performance** della rete e sull'adattamento ai dati.

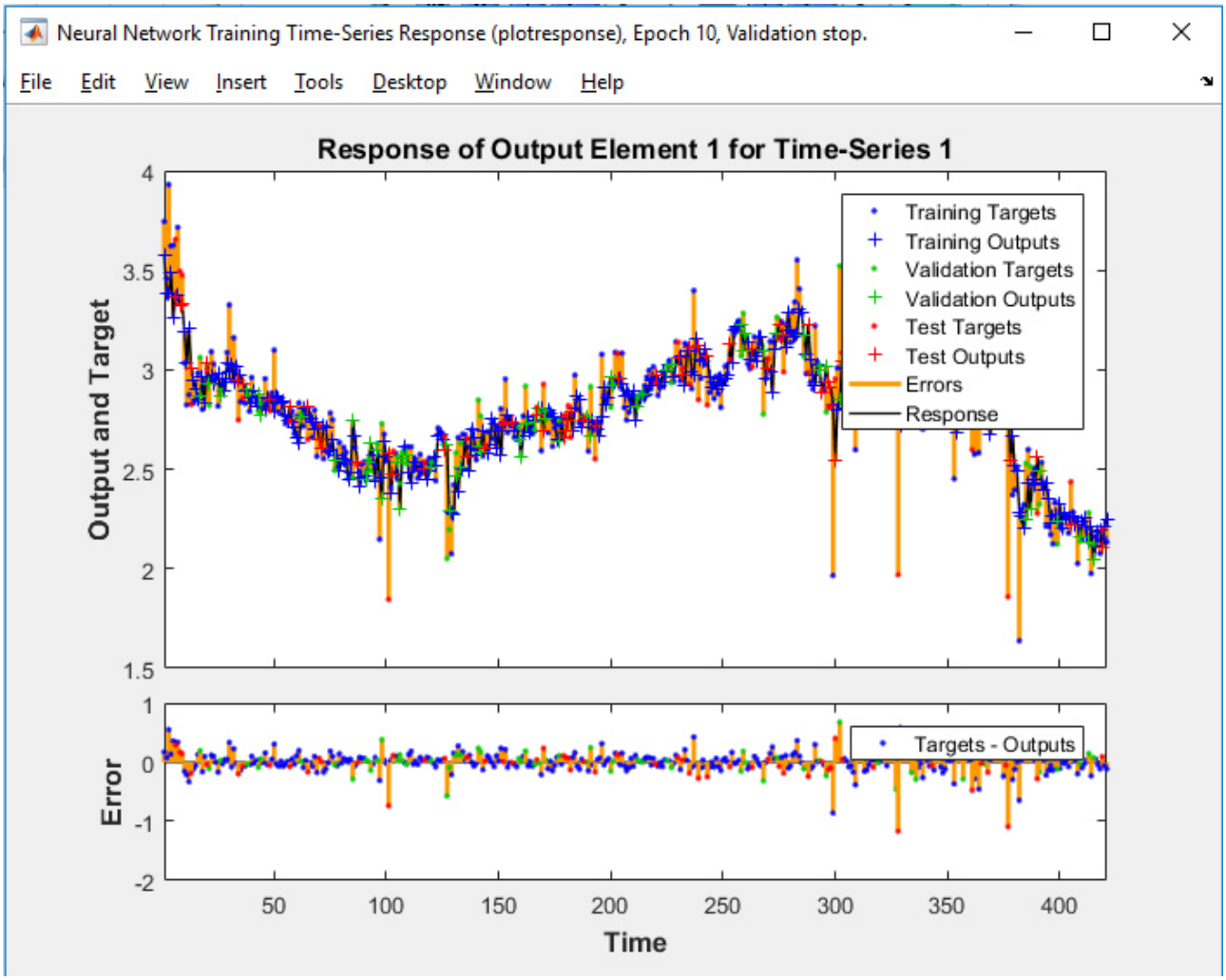


Schiacciando il pulsante Performance otteniamo Il grafico che illustra la variazione del MSE (Mean Square Error) durante il processo di addestramento. Mette anche in evidenza il punto in cui si ferma l'addestramento: al quarto step quando il valore del MSE sul campione di validazione raggiunge il minimo. Dopo questo step la rete va in overfitting: l'errore sul training set continua a diminuire mentre l'errore sul validation set aumenta facendo ridurre la capacità di generalizzazione della rete.



Schiacciando **il pulsante "Regression**" otteniamo un grafico che mostra la nube dei punti nel piano **- Output rete/ target osservati -** per i dati di training, di validation e di test: R è il coefficiente di correlazione tra questi dati.
L'ultimo grafico mostra, in altro modo, la distanza tra i valori di output e i valori target per i tre set di dati: training validatione e test set:


Training: R=0.87738
Data
Fit
Y = T
Output ~= 0.73*Target + 0.75
Target
Validation: R=0.7925
Output ~= 0.8*Target + 0.54
Test: R=0.60887
Output ~= 0.47*Target + 1.5
All: R=0.82247
Output ~= 0.7*Target + 0.84



.

GENERAZIONE FUNZIONE PER MATLAB E PER C++

Il codice Matlab consente la preparazione in automatico di librerie di funzioni, non solo per il loro utilizzo all'interno di Matlab stesso, ma anche per la ***include*** **in altri linguaggi di programmazione come il C++.**

Nel seguente codice si crea e si addestra la rete con la struttura selezionata in precedenza con 3 nodi nascosti (Hidden). Successivamente si genera una funzione per Matlab con l'istruzione ***genFunction*** passando come parametro il nome della rete e il nome che vogliamo dare alla funzione (`'rete_Neurale_FCN'`)

```
neto = narxnet( ID, FD, 3 );  %genero la struttura della rete con 3 nodi Hidden
neto.divideFcn = 'dividerand';
neto.divideParam.trainRatio= 70/100; %divisione dati campione di addestramento
neto.divideParam.valRatio = 15/100;   %divisione dati campione di validazione
neto.divideParam.testRatio = 15/100; %divisione dati campione di test
[Xo Xoi Aoi To] = preparets(neto,X,{},T); %preparazione dati di addestramento
con i ritardi
[neto tro Yo Eo Xof Aof ] = train(neto,Xo,To,Xoi,Aoi); %addestramento rete

genFunction(neto,'rete_Neurale_FCN');%generazione funzione rete_Neurale_FCN.m
[yo,xof,aof] = rete_Neurale_FCN(Xo,Xoi,Aoi);     %utilizzo della funzione e
generazione risultato della rete yo
accuracy2 = max(abs(cell2mat(Yo)-cell2mat(yo))) %confronto risultato funzione yo
con il risultato dell'addestramento Yo
```

```
mcc -W lib:libneto -T link:lib rete_Neurale_FCN %generazione libreria libneto
per C++
```

In quest'ultima istruzione si genera la funzione ***libneto*** che può essere inclusa all'interno del codice C++. Per portare a buon fine l'istruzione ***mcc***, occorre che sul computer siano installate le **librerie SDK.**
Il risultato dell'istruzione *genFunction* è illustrato nel seguente codice Matlab dove nelle prime righe di commento viene spiegato il corretto utilizzo della **funzione *rete_Neurale_FCN*** e dei suoi parametri:

```
function [Y,Xf,Af] = rete_Neurale_FCN(X,Xi,~)
%RETE_NEURALE_FCN neural network simulation function.
%
% Generated by Neural Network Toolbox function genFunction, 06-Aug-2017
01:14:29.
%
% [Y,Xf,Af] = rete_Neurale_FCN(X,Xi,~) takes these arguments:
%
%   X = 2xTS cell, 2 inputs over TS timesteps
%   Each X{1,ts} = 11xQ matrix, input #1 at timestep ts.
%   Each X{2,ts} = 1xQ matrix, input #2 at timestep ts.
%
%   Xi = 2x5 cell 2, initial 5 input delay states.
%   Each Xi{1,ts} = 11xQ matrix, initial states for input #1.
%   Each Xi{2,ts} = 1xQ matrix, initial states for input #2.
%
%   Ai = 2x0 cell 2, initial 5 layer delay states.
%   Each Ai{1,ts} = 3xQ matrix, initial states for layer #1.
%   Each Ai{2,ts} = 1xQ matrix, initial states for layer #2.
%
% and returns:
%   Y = 1xTS cell of 2 outputs over TS timesteps.
%   Each Y{1,ts} = 1xQ matrix, output #1 at timestep ts.
%
%   Xf = 2x5 cell 2, final 5 input delay states.
%   Each Xf{1,ts} = 11xQ matrix, final states for input #1.
%   Each Xf{2,ts} = 1xQ matrix, final states for input #2.
%
%   Af = 2x0 cell 2, final 0 layer delay states.
%   Each Af{1ts} = 3xQ matrix, final states for layer #1.
%   Each Af{2ts} = 1xQ matrix, final states for layer #2.
%
% where Q is number of samples (or series) and TS is the number of timesteps.
```

Dopo la costruzione della rete neurale e il suo addestramento, le previsioni avvengono con la funzione: [Y,Xf,Af] = rete_Neurale_FCN(X,Xi,~) dove se si inseriscono nella matrice "X" i dati degli ultimi "k" giorni delle variabili esplicative e della variabile da prevedere, si ottiene in "Y" il valore previsto della media giornaliera nel giorno "k+1".

CONCLUSIONE

In conclusione, il percorso seguito in queste pagine ha messo in evidenza le potenzialità del programma matlab applicato ai modelli matematico-statistici per l'analisi dei dati. In particolare il **modello neurale** è un valido strumento per la simulazione dei dati raccolti dal rilevatore AMD ma potrebbe anche essere utilizzato per mettere in evidenza quei valori anomali che non risultano prevedibili in base alle condizioni meteo, alla situazione dell'attività solare e al livello delle radiazioni ambientali. **Su questi valori si potrebbe concentrare lo studio per l'individuazione dei tanto ricercati UHECR.**
Il limite, per adesso, è costituito dal numero basso di osservazioni registrate dal rilevatore di Pozzuoli ma ogni anno il modello verrà perfezionato sperando di poter contare anche su un incremento dell'attività solare.

APPENDICE

MEDIA: media giornaliera amd5

AVTEMP: temperatura media

PRESS: pressione atmosferica

HUM: umidità

SWIND: vento solare in km/s (VERDE< 400; GIALLO 400-600; ROSSO 600-800) SATELLITE LAGRANGIANO ACE

PWIND: densità particelle per cm^2 che serve al calcolo della pressione dinamica P (P<=1 -> quiete)

NUM6XRAY: MAX emissione ULTIME 6 ORE raggi x lunghezza d'onda 1-8 angstrom satellite GOES 15 (CATEGORIE A,B,C,M,X; A,B QUIETE) in W/m^2

NUM24XRAY: MAX emissione nelle 24 ORE raggi x lunghezza d'onda 1-8 angstrom satellite GOES 15 (CATEGORIE A,B,C,M,X; A,B QUIETE) in W/m^2

SUNSPOT: numero macchie solari; indicatore per il calcolo del Wolf: funzione del numero di macchie e dei raggruppamenti delle macchie.

SFU: solar flux unit; emissione di onde radio sulla lunghezza d'onda dei 10.7cm. 1sfu = 10^-26 jansky - W/m^2/Hz

KpMAX: indice geomagnetico planetario: influenza dell'attività solare sul campo magnetico terrestre. QUIETE Kp< 3; All'aumentare di KP il fronte delle aurore si sposta verso sud.

NUMBz: campo magnetico interplanetario; componente vettoriale che rappresenta l'influenza del campo magnetico solare su quello terrestre. Quiete: valore positivo nord.

MAG: Magnitudo sisma; valori anche negativi per l'aumento della precisione strumentale rispetto alla scala Richter.

KPROM: profondità sisma in KM

NUMEVENT: numero eventi sisma