Intelligent Systems

Damiano Barone
Lorenzo Diana

A.Y. 2013/2014

Contents

1 Choice of best predictors indexes

The aim of this first step is create, for each index, a neural network that is able to forecast the value of ISE 100 index of one day based on previusli seven days values of the index, valuate the performance (MSE) of the network and chose which index is the best predictors for the ISE 100 index.
We have load data from file *data_akbilgic.xlsx* and we make a matrix conteining all index and e vector conteining the ISE 100 index in USD, that reppresent the targets value for our networks.

```
output=xlsread('data_akbilgic.xlsx','C:C', '', 'basic');
Input(:, 1:7)=xlsread('data_akbilgic.xlsx','D:J', '', 'basic');
```

For each index we created e MLP neural network at which we provide as output the value of ISE 100 from eighth day onward and as input a matrix where each row conteining values of seven days of the index: first row conteining days from first to seventh, second row days from second to eighth, third row days from third to ninth, and so on.

```
function Matrix=crea_matrice(input_)
    for i=1:(size(input_, 1)-7)
        Matrix(i, :)=input_(i:i+6)';
    end
end
```

We run many trial to find the best value for neurons number and training number, for each network, that would minimize MSE. Using 536 samples and reserving 70% for training set we can use from 5 to 9 neurons for the hidden layer of the network.

```
mse_avg=100;
num_tr=[8 10 12];
for n=5:9
    for t=1:size(num_tr, 2)
        for z=1:5
            for  i=1:7
                % create input matrix
                Campioni_input_allenamento_nn=
                        crea_matrice(Input(:, i));
                script_nn
                mse_nn(i)=performance;
            end
            if ((sum(mse_nn)/7)<mse_avg)
                mse_avg=sum(mse_nn)/7;
```

```
                best_mse_nn=mse_nn;
                best_t=t;
                best_n=n;
            end
        end
    end
end
```

We had modify the script obteined from matlab's tool so that it can automatically set the right value for neurons number and training number each time the script is called.

```
hiddenLayerSize = n;
```

```
for q=1:num_tr(t)
    [net,tr] = train(net,inputs,targets);
end
```

We compared the MSE average for each trial and we found the minimun value for it with 8 neurons in the hidden layer and 12 training, the value of MSE for each index that provide best MSE average is:

SP	DAX	FTSE	NIKKEI	BOVESPA	EU	EM
0,3389	0,3635	0,3922	0,3660	0,3481	0,3801	0,4007

Figure 1: MSE (e-03)

As best predictors of ISE 100 we chose SP, BOVESPA, NIKKEI and DAX indexes.

2 Forecasting model development

2.1 Computation best delays

We wanto to forecast only the movement of ISE 100, not the exactly value, so we change the daily value of ISE 100 whith 1 in the days whitch the ISE increase and whith 2 in the days it decrease, this will be the targets for our neural network. To this aim we use this function.

```
function ou=prepare_output(o)
    for e=1:size(o, 1)
```

```
        if (o(e)>0)
            ou(e)=1;
        else
            ou(e)=2;
        end
    end
end
```

The aviable sample was dividen in two period, first half of sample for the estimation period and the second half for forecasting period. As input, to train the network, we made a matrix that have as column the indexes chose at point 1 and as rows the rows of the estimation period, and as targets the first half of the ISE 100 value modified as said before.

```
indici_scelti_nn=[1 2 4 5];
output_ga=prepare_output(output);

% estimation period
Input_alenamento_ga=Input(1:floor(size(Input, 1)/2),
    indici_scelti_nn);
output_allenamento_ga=output_ga(1:floor(size(output_ga, 2)/2));
% forcasting period
Input_previsto_ga=Input(floor(size(Input,1)/2)+1:end,
    indici_scelti_nn);
output_previsto_ga=output_ga( floor(size(output_ga, 2)/2)+1:end );
```

We chose to train the network 12 times and we run *ga()* function with boundaries for input and output delays from 1 to 20, this function give us the best value for delays.

```
num_training_ga=12;
max_input_delay=20;
max_feedback_delay=20;

[best_delay, pcfd, exit, uscita, popolazione, punteggio] =
    ga(@fitfunc, 2, [], [], [], [], [1, 1], [max_input_delay,
        max_feedback_delay], [],[1 2]);
pcfd=100-pcfd;
```

To understand the goodness of each combination of input and output delay generated from *ga()* function we made a fitting function that take in input the delays, make and train a new time series network using the script generated from matlab's graphic tool and return the error of the network created. As error index we chose percentage of correctly forecasted days, but we 'invert' this value (see last row of function) because the *ga()* function only can minimize the value returned

from fitting function.

```
function PCFD_ts = fitfunc(delay)
    % set delays
    in_d=delay(1);
    feed_d=delay(2);

    script_ts

    % PCFD
    giorni_correttamente_predetti=round(cell2mat(outputs))==
        round(cell2mat(targets));
    PCFD_ts=(sum(giorni_correttamente_predetti)/
        size(cell2mat(targets), 2))*100;
    PCFD_ts=100-PCFD_ts;
end
```

We run this script more times in order to try different number of hidden layer size, and for each of this we found different input delay and output delays.

Neuron number	input delay	output delay	PCFD
4	9	8	66,4093
5	13	10	75,2941
6	16	9	69,8413
7	20	14	87,5000

Figure 2: Delays found by *ga()* and their PCFD

Before go to next step we try to found by *ga()* the best number of neurons, and olso the input and output delay, changing the code as follows.

```
% Changes in \textit{ga()}
[best_delay, pcfd, exit, uscita, popolazione, punteggio] =
    ga(@fitfunc, 3, [], [], [], [], [1, 1, 4],
        [max_input_delay, max_feedback_delay, 7],
            [],[1 2 3]);
```

```
% Amendments in \textit{fitfunc()}
num_neu=delay(3);
```

```
% Changes in script for time series
hiddenLayerSize = num_neu;
```

Neuron number	input delay	output delay	PCFD
7	18	20	83,0645

Figure 3: Values found by *ga()* for neurons number, input e output delays

2.2 Evaluation of performance

For eache pair of delays found we create a time series network and evaluate their performance on forecasting period, forecasting with 1, 2, ..., 40 step ahead for each network. For each step we compute MSE, MAPE and percentage of correctly forecasted days (PCFD), and for each network the average of this parameters. (the parameter of the last row in the table below give low performance so it will be not considered from now on, for convenience).

				Average on forecasting period		
Neuron number	input delay	output delay	PCFD	MSE	MAPE	PCFD
4	9	8	66,4093	0,2464	27,8545	61,2873
5	13	10	75,2941	0,2743	34,5009	57,6679
6	16	9	69,8413	0,2457	31,4179	59,2631
7	20	14	87,5000	0,2634	25,4804	57,472
7	18	20	83,0645	0,3091	33,3629	53,1343

Figure 4: Delays found by *ga()* their PCFD, and average of PCFD, MSE and MAPE on forecasting period

```
oriz_lim=40;
performance_forecast=zeros(1, oriz_lim)-1;
MAPE_forecast_ts=zeros(1, oriz_lim)-1;
PCFD_forecast_ts=zeros(1, oriz_lim)-1;

for orizzonte=1:oriz_lim;
    if (retrain_net==1)
        script_ts
    end
    input_nuovo_allenamento=Input_alenamento_ga;
    output_nuovo_allenamento=output_allenamento_ga;
    Input_previsto_ga_2=[Input_alenamento_ga(
        (end-max_delay-orizzonte+2):end, :); Input_previsto_ga];
    output_previsto_ga_2=[output_allenamento_ga(:,
        (end-max_delay-orizzonte+2):end) output_previsto_ga];
    outputs_forecast=cell(0);
    for pos=max_delay+1:(max_delay+size(output_previsto_ga, 2));
        rete=closeloop(net);
        inputSeries = tonndata(Input_previsto_ga_2((pos-max_delay):
            (pos+orizzonte-1), :),false,false);
        targetSeries = tonndata([output_previsto_ga_2(
            (pos-max_delay):(pos-1)) nan(1, orizzonte)]',
                false,false);
        [inputs_forecast,inputStates,layerStates,targets_forecast]=
```

```
25              preparets(rete,inputSeries,{},targetSeries);
26          out_rete = rete(inputs_forecast,inputStates,layerStates);
27          outputs_forecast = [outputs_forecast out_rete(end)];
28
29          % retrain
30          if (retrain_net==1)
31              input_nuovo_allenamento=[input_nuovo_allenamento;
32                  Input_previsto_ga(pos-max_delay, :)];
33              output_nuovo_allenamento=[output_nuovo_allenamento
34                  output_previsto_ga(pos-max_delay)];
35              inputSeries = tonndata(input_nuovo_allenamento,
36                  false,false);
37              targetSeries = tonndata(output_nuovo_allenamento',
38                  false,false);
39              [inputs,inputStates,layerStates,targets] =
40                  preparets(net,inputSeries,{},targetSeries);
41              for q=1:num_retraining_ga
42                  [net,tr] = train(net,inputs,targets,inputStates,
43                      layerStates);
44              end
45          end
46      end
47      % performances
48      performance_forecast(orizzonte) = sum(
49          (output_previsto_ga-my_round(
50          cell2mat(outputs_forecast))).*
51          (output_previsto_ga-my_round(
52          cell2mat(outputs_forecast))) )/size(output_previsto_ga, 2);
53      MAPE_forecast_ts(orizzonte)=sum( abs( ((output_previsto_ga-
54          my_round(cell2mat(outputs_forecast))./output_previsto_ga)
55          .*100 ) )/size(output_previsto_ga, 2);
56      % PCFD
57      giorni_correttamente_predetti=my_round(
58          cell2mat(outputs_forecast))==output_previsto_ga;
59      PCFD_forecast_ts(orizzonte)=(sum(giorni_correttamente_predetti)
60          /size(output_previsto_ga, 2))*100;
61  end
```

We use this function instead of standard *round()* to handle possible values lower than 0,5 or higher than 2,5.

```
1   function my_r = my_round(vet)
2       for i=1:max(size(vet))
3           if (vet(i)>=1.5)
4               my_r(i)=2;
5           else
6               my_r=1;
7           end
```

```
8         end
9     end
```

Below we show chart with performances of all network in terms of PCFD, MSE and MAPE.

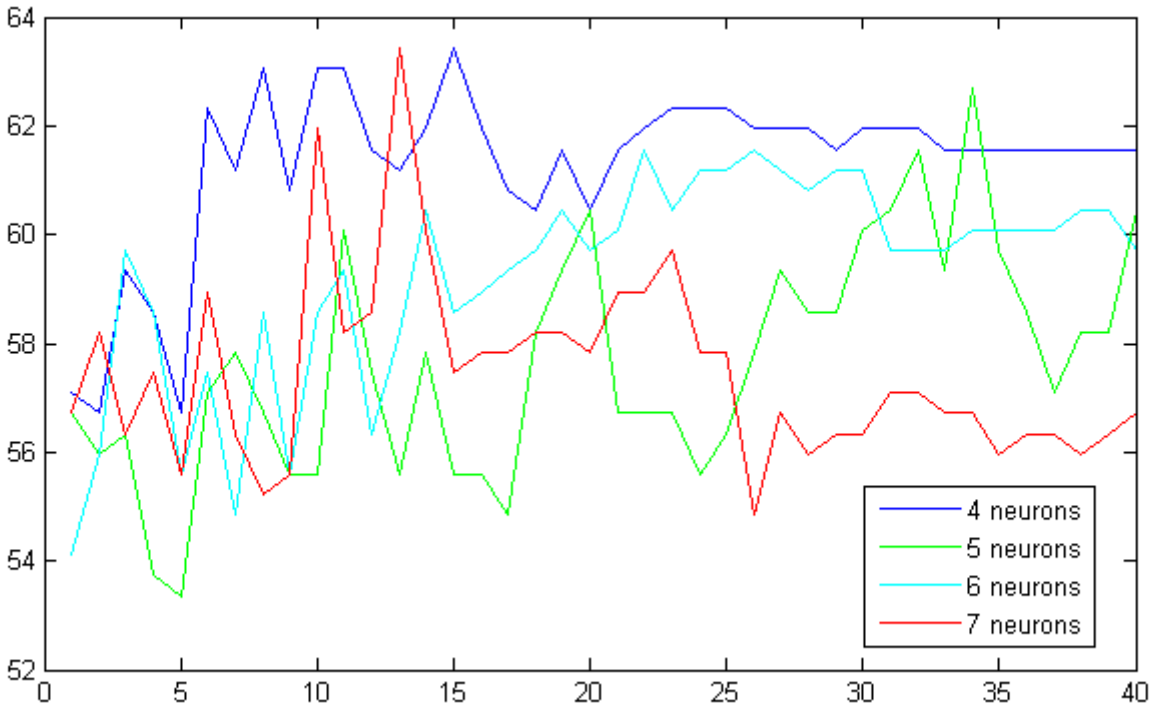

Figure 5: PCFD for eache step ahead forecasted

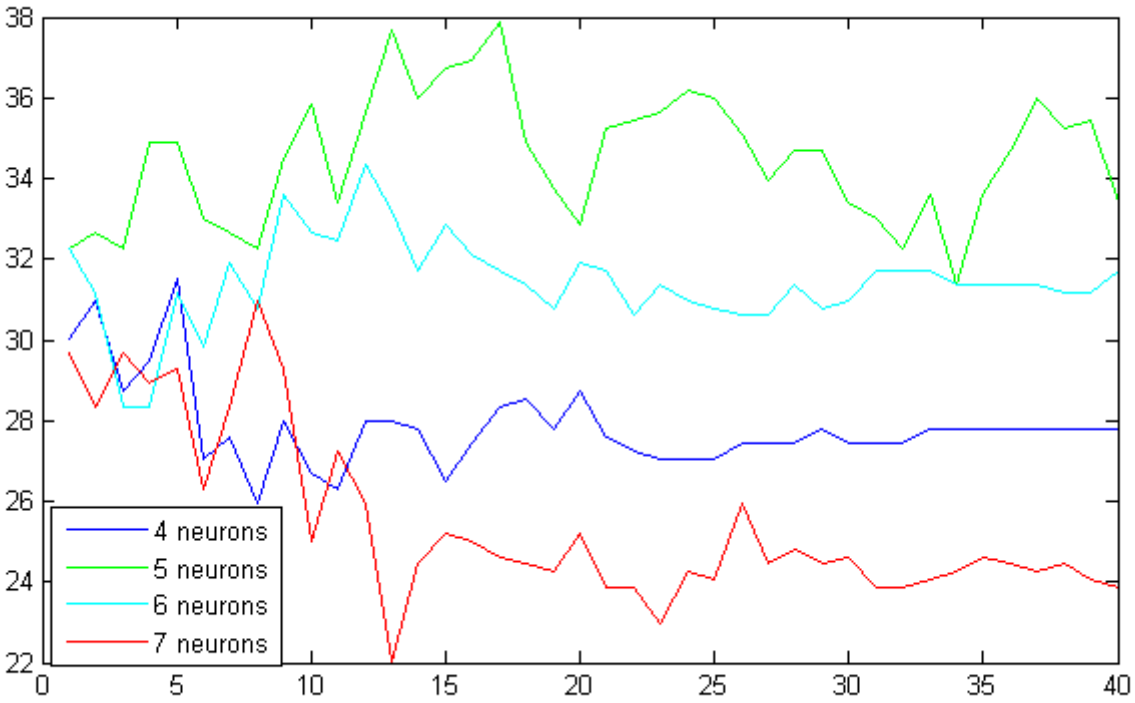

Figure 6: MAPE for eache step ahead forecasted

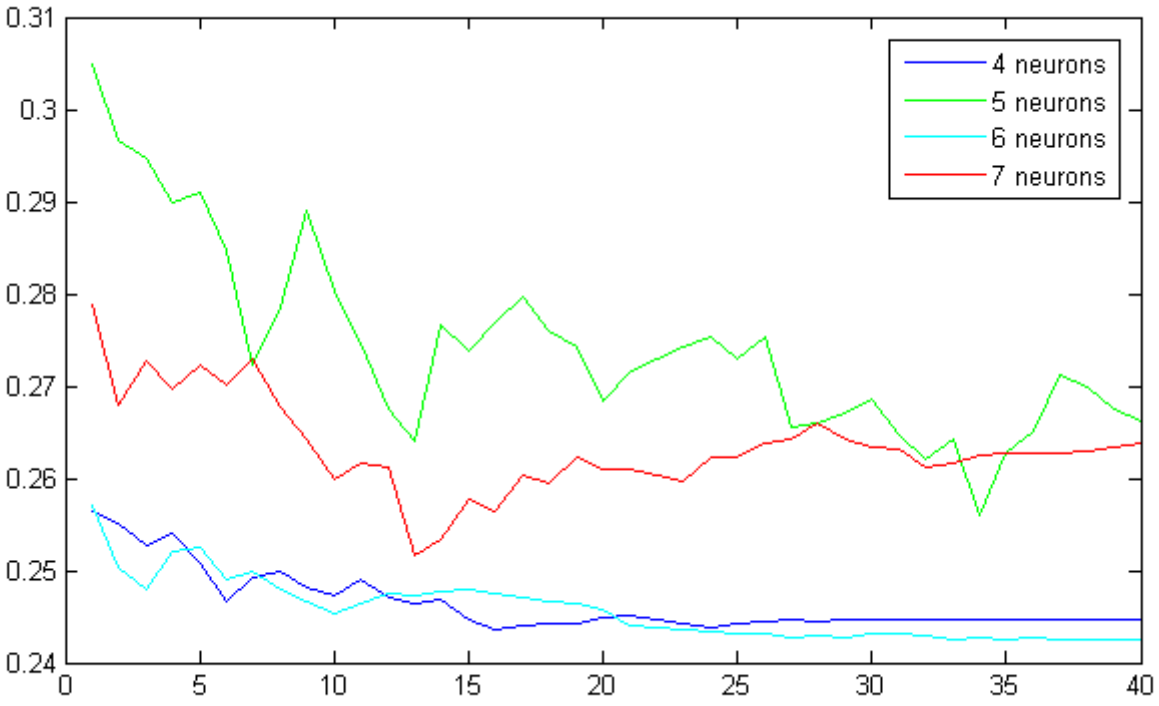

Figure 7: MSE for eache step ahead forecasted

We chose the network that have the best PCFD and we evaluate again his performance whith retrain option actived, that is after each forecast day we retrain the network whith the old estimation period plus the correct value for the day just forecast. We obtain that PCFD is decreased as can be seen in the follow charts where we compare performances with and without retrain the network.

			Average on forecasting period			
Neuron number	input delay	output delay	MSE	MAPE	PCFD	
4	9	8	0,2464	27,8545	61,2873	*whithout retrain*
			0,4358	32,8731	56,5578	*whith retrain*

Figure 8: Performance after and before retraining

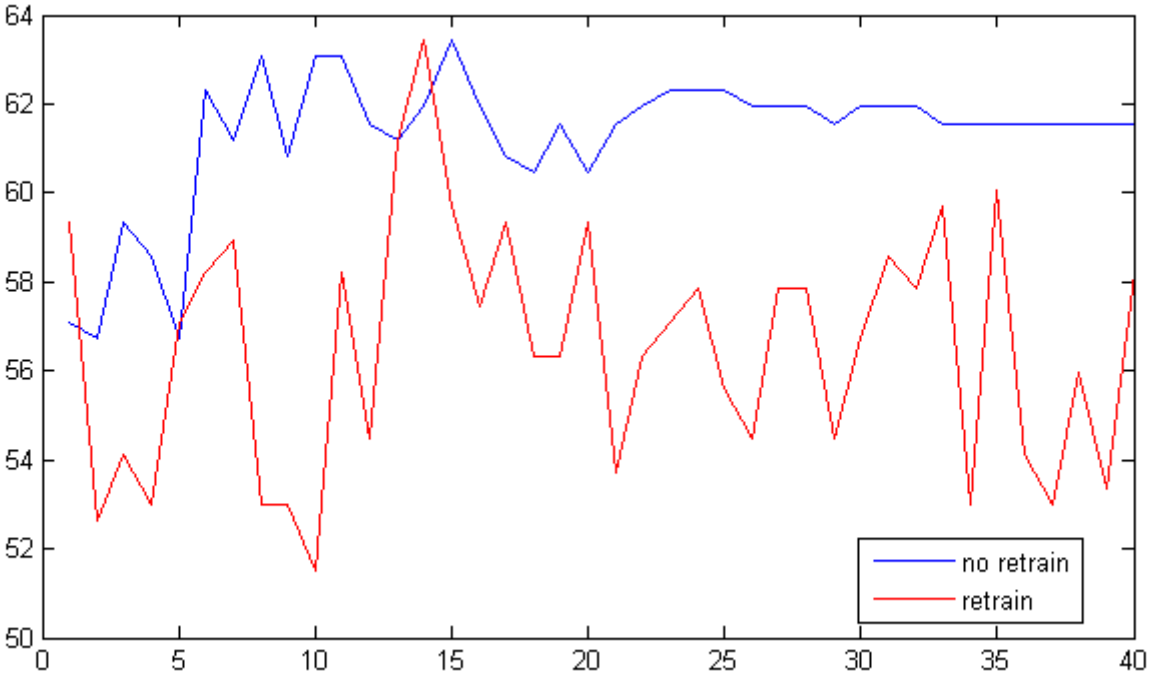

Figure 9: PCFD

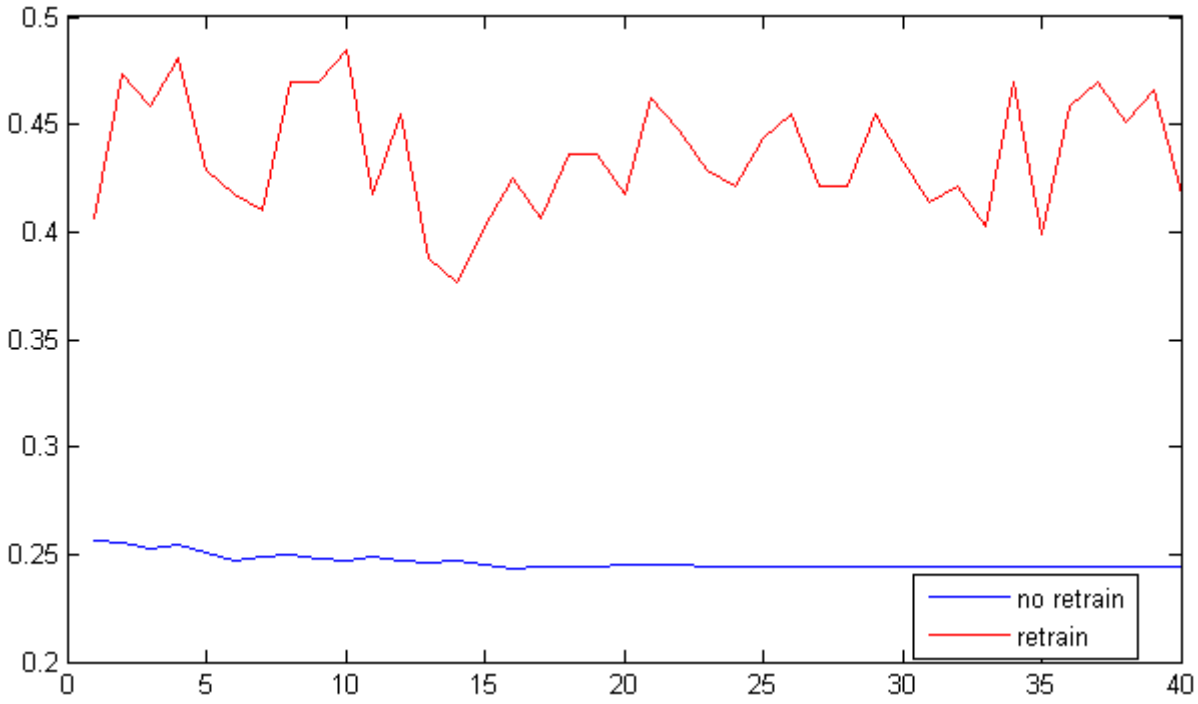

Figure 10: MSE

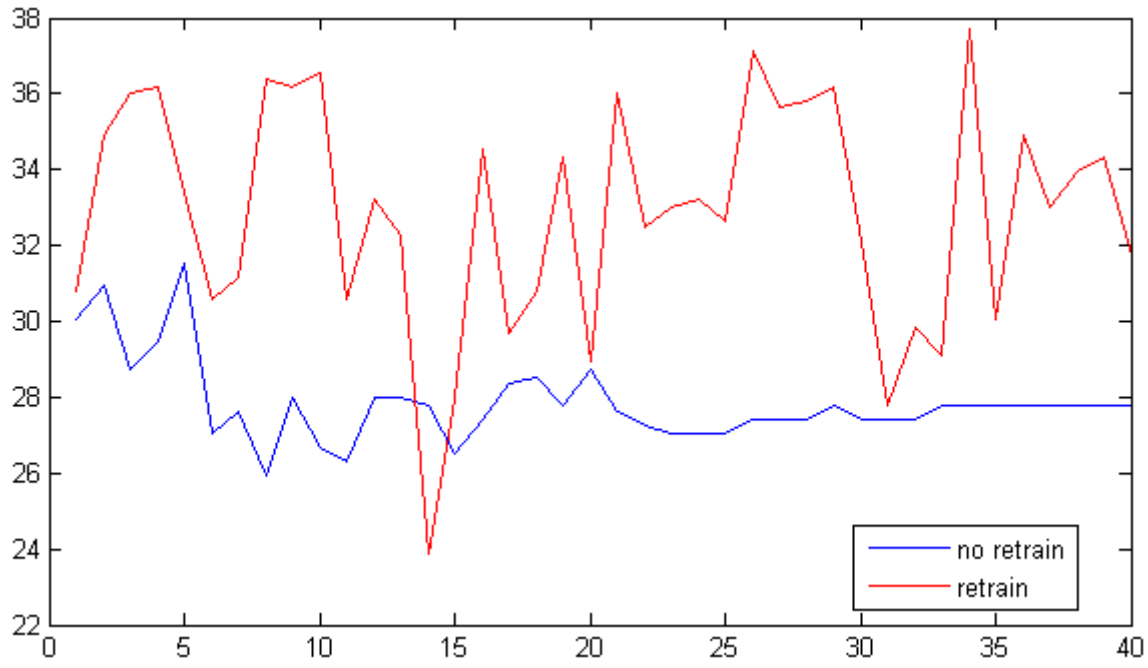

Figure 11: MAPE

3 fuzzy logic

To make test set, training set and checking set we take regular interval sample over all the aviable data. We have 536 samples so we take 70% of this for training set, 15% for checking set and the last 15% for test set.

```
x=1:6.7:536;
x=round(x);
% test set
targets_ISE=prepare_output(output(x));
Input_anfis_testing=Input(x, indici_scelti_nn);
% checking set
Input_anfis_checking=[Input((x+3), indici_scelti_nn),
    prepare_output(output(x+3))'];
% training set
Input_anfis_training=[Input(:, indici_scelti_nn),
    prepare_output(output)'];
Input_anfis_training([x x+3], :)=[];
```

We generate different Sugeno-type FIS using different function for the input (like triangular, gaussian, ecc...), both linear and constant function for the output and for training method both hibrid and backpropagation.

```
num_epoch=30;

fun=['trimf' 'psigmf' 'gbellmf' 'gaussmf' 'gauss2mf' 'pimf'
    'dsigmf' 'trapmf'];
fun_supp=[1 5 6 11 12 18 19 25 26 33 34 37 38 43 44 49];
o_fun=['linear' 'constant'];
o_fun_supp=[1 6 7 14];

for o=1:2:4;
    for f=1:2:16;
        for a=0:1
            fis = genfis1(Input_anfis_training, 3, fun(fun_supp(f):
                fun_supp(f+1)), o_fun(o_fun_supp(o):
                o_fun_supp(o+1)));
            [fis,error,stepsize,chkFis,chkErr] = anfis(
            Input_anfis_training, fis,
            [num_epoch NaN NaN NaN NaN],
            [NaN NaN NaN NaN], Input_anfis_checking, a);

            % performance evalutation
            out=evalfis(Input_anfis_testing, chkFis);
            out=out';
            giorni_correttamente_predetti=my_round(out)==
```

```
                targets_ISE;
            if (a==0)
                PCFD_b(o, f)=(sum(giorni_correttamente_predetti)/
                    size(targets_ISE, 2))*100;
                MSE_b(o, f)=sum( (targets_ISE-my_round(out)).*
                    (targets_ISE-my_round(out)) )/
                    size(targets_ISE,2);
                MAPE_fis_b(o, f)=sum( abs( ((targets_ISE-
                    my_round(out))./targets_ISE).*100 ) )/
                    size(targets_ISE, 2);
            else
                PCFD_h(o, f)=(sum(giorni_correttamente_predetti)/
                    size(targets_ISE, 2))*100;
                MSE_h(o, f)=sum( (targets_ISE-my_round(out)).*
                    (targets_ISE-my_round(out)) )/
                    size(targets_ISE, 2);
                MAPE_fis_h(o, f)=sum( abs( ((targets_ISE-
                    my_round(out))./targets_ISE).*100 ) )/
                    size(targets_ISE, 2);
            end
        end
    end
end
```

Training FIS for 30 ephocs, we decide the constant function is the best for the output, and the gaussian membership function (gaussmf) is the best for input as we can see from MAPE, MSE and PCFD. As training algorithm the best is hybrid, because the backpropagation give unusual result.

PCFD	trimf	psigmf	gbellmf	gaussmf	gauss2mf	pimf	dsigmf	trapmf
linear	61.2500	60.0000	63.7500	65.0000	55.0000	60.0000	60.0000	
constant	63.7500	62.5000	60.0000	67.5000	61.2500	65.0000	62.5000	62.5000

MAPE	trimf	psigmf	gbellmf	gaussmf	gauss2mf	pimf	dsigmf	trapmf
linear	37.5000	32.5000	33.7500	29.3750	53.7500	56.8750	32.5000	
constant	25.0000	25.6250	29.3750	21.8750	27.5000	25.6250	25.6250	28.1250

MSE	trimf	psigmf	gbellmf	gaussmf	gauss2mf	pimf	dsigmf	trapmf
linear	1.3500	0.6125	1.1375	0.6625	3.7125	3.6750	0.6125	
constant	0.4625	0.3750	0.4375	0.3250	0.4250	0.5375	0.3750	0.4750

Figure 12: Errors with hybrid training algorithm

PCFD	trimf	psigmf	gbellmf	gaussmf	gauss2mf	pimf	dsigmf	trapmf
linear	52.5000	52.5000	52.5000	52.5000	52.5000	52.5000	52.5000	52.5000
constant	52.5000	52.5000	52.5000	52.5000	52.5000	52.5000	52.5000	52.5000

MAPE	trimf	psigmf	gbellmf	gaussmf	gauss2mf	pimf	dsigmf	trapmf
linear	23.7500	23.7500	23.7500	23.7500	23.7500	23.7500	23.7500	23.7500
constant	23.7500	23.7500	23.7500	23.7500	23.7500	23.7500	23.7500	23.7500

MSE	trimf	psigmf	gbellmf	gaussmf	gauss2mf	pimf	dsigmf	trapmf
linear	0.4750	0.4750	0.4750	0.4750	0.4750	0.4750	0.4750	0.4750
constant	0.4750	0.4750	0.4750	0.4750	0.4750	0.4750	0.4750	0.4750

Figure 13: Errors with backpropagation training algorithm