# Applications of Least Mean Square (LMS) Algorithm Regression in Time-Series Analysis

Eleftherios Giovanis

**Abstract**

In this paper we present a very brief description of least mean square algorithm with applications in time-series analysis of economic and financial time series. We present some numerical applications; forecasts for the Gross Domestic Product growth rate of UK and Italy, forecasts for S&P 500 stock index returns and finally we examine the day of the week effect of FTSE 100 for a short period. A full programming routine written in MATLAB software environment is provided for replications and further research applications.

**Keywords**: LMS, Least Mean Square Algorithm, MATLAB, time-series, stock returns, gross domestic product, forecast

## 1. Introduction

Gabor (1954) was the first to conceive the idea of a nonlinear adaptive filter in 1954 using a Volterra series. The Least Mean Square (LMS) algorithm, introduced by Widrow and Hoff in 1959 is an adaptive algorithm. LMS incorporates an iterative procedure that makes successive corrections to the weight vector in the direction of the negative of the gradient vector which eventually leads to the minimum mean square error. Compared to other algorithms LMS algorithm is relatively simple; it does not require correlation function calculation nor does it require matrix inversions. An *adaptive filter* is defined as a self-designing system that relies for its operation on a recursive algorithm, which makes it possible for the filter to perform satisfactorily in an environment where knowledge of the relevant statistics is not available. In this paper we do not compare the results with those of other methods, because our purpose is to present only the procedure and some empirical examples. Additionally, there are many models to compare with, as also even if we test for example ten different time-series in 50 countries it will not be enough.

Electronic copy available at: http://ssrn.com/abstract=1667440

## 2. Methodology

Assuming that all the signals involved are real-valued signals the LMS the elements for LMS algorithm are (Haykin, 1996; Hayes, 1996) :

Tap-weight vector $\bar{w} = [w_0, w_1, w_2, ....., w_{N-1}]^T$ (1)

Signal input $\bar{x}(n) = [x(n), x(n-1), x(n-2)....., x(n-N+1]^T$ (2)

Filter output $y(n) = \bar{w}^T \bar{x}(n)$ (3)

Error signal $e(n) = d(n) - y(n)$ (4)

Function (4) is the error or cost function and LMS is used in order to minimize it. A simple algorithm can be written as

for i=1:nk,

e(i)=d(i)-w(i)*x(i);

w(i)=w(i) + (m*e(i)*x(i))

end

, where *e, x* and *w* are defined as previously and *m* is the adaptation rate. The learning rule is just the relation

$w_{kj}(n+1) = w_{kj}(n) + \eta \cdot \Delta w_{kj}(n)$ (5)

,where the Greek letter $\eta$ denotes the learning rate. In figure 1 we present a simple process of LMS algorithm.

Electronic copy available at: http://ssrn.com/abstract=1667440

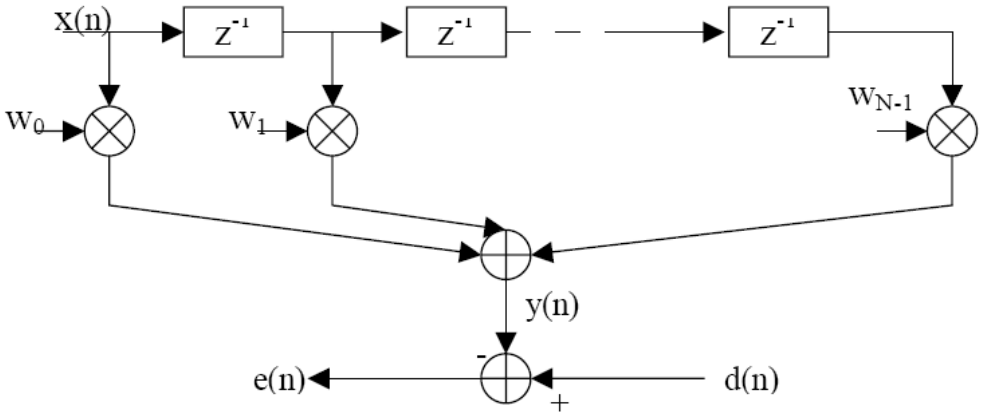


**Figure 1**. LMS algorithm procedure and transversal filter.

Additionally, in this study we expand the LMS algorithm and we use ordinary least squares regression based on the tap-weight vector after the training process. More specifically a simple linear regression is:

$$y = \beta_0 + \beta_1 \overline{w}x \tag{6}$$

, where $y$ is the dependent variable, $x$ is the independent variable, $\beta_0$, $\beta_1$ are the estimated coefficients and $w$ is the tap-weight vector after the training process. Relation (6) can be expanded including more independent variables. Actually equation (6) is a kind of weighted regression.

The first algorithm in the appendix minimized the cost function (4).In the second algorithm we change the cost function and in that case we minimize the sum squared of residuals error of linear regression (6).

The final approach is to take the forecast as follows:

$$forecast = y_t \cdot \overline{w} \tag{7}$$

, where $y_t$ is the last observation of the output or the actual dependent variable as we take lags.

## 3. Numerical Applications

As an example we take the UK gross domestic product (GDP) growth rate. We estimate an autoregressive model AR(3), which means that we take as inputs the lags of first, second and third order of GDP and without constant. The forecasting performance is not changed whether taking or not the constant, which is a vector of ones. We use the programming routine 1 provided in appendix, where we minimize the cost function (4). The estimation, in-sample or training period is 1991-2007 and the out-of-sample or testing period is 2008-2009 and the data are on quarterly frequency. The adaptation and learning rates *m* and *eta* are set up at 0.05. The number of maximum epochs is set up at 10 and the goal error at 7. We use lms_type=2. After the training process the error reached 3.6955 after 6 epochs. In figures 2 and 3 we present the actual versus forecasts in the in-sample and out-of-sample periods respectively.

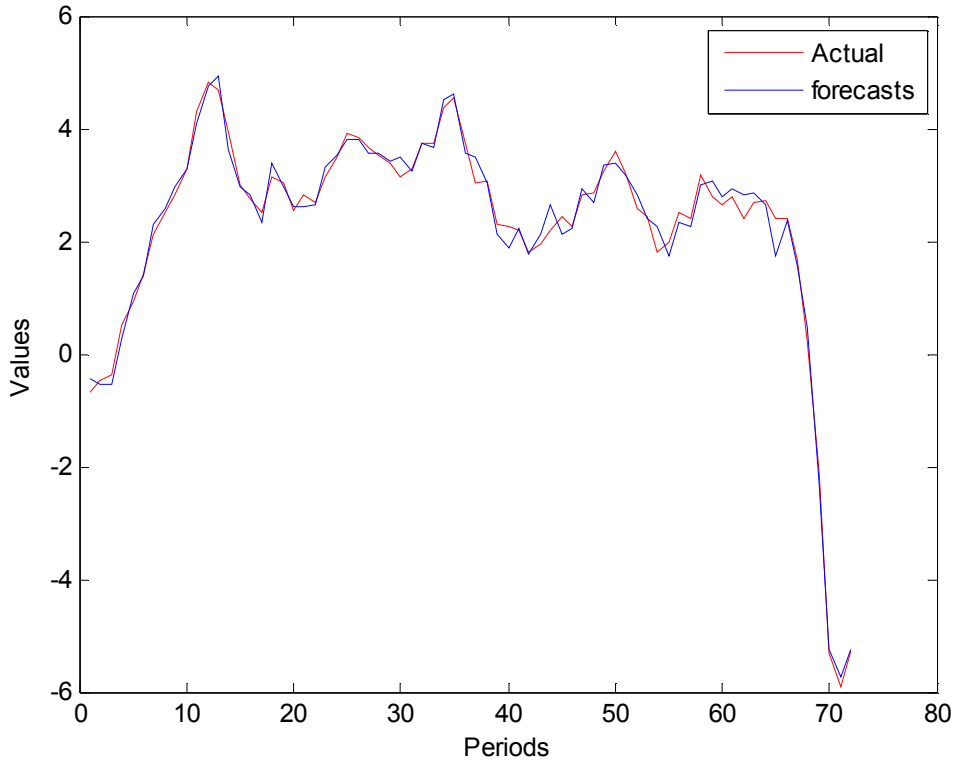


**Fig. 2** In-sample forecasts for UK GDP

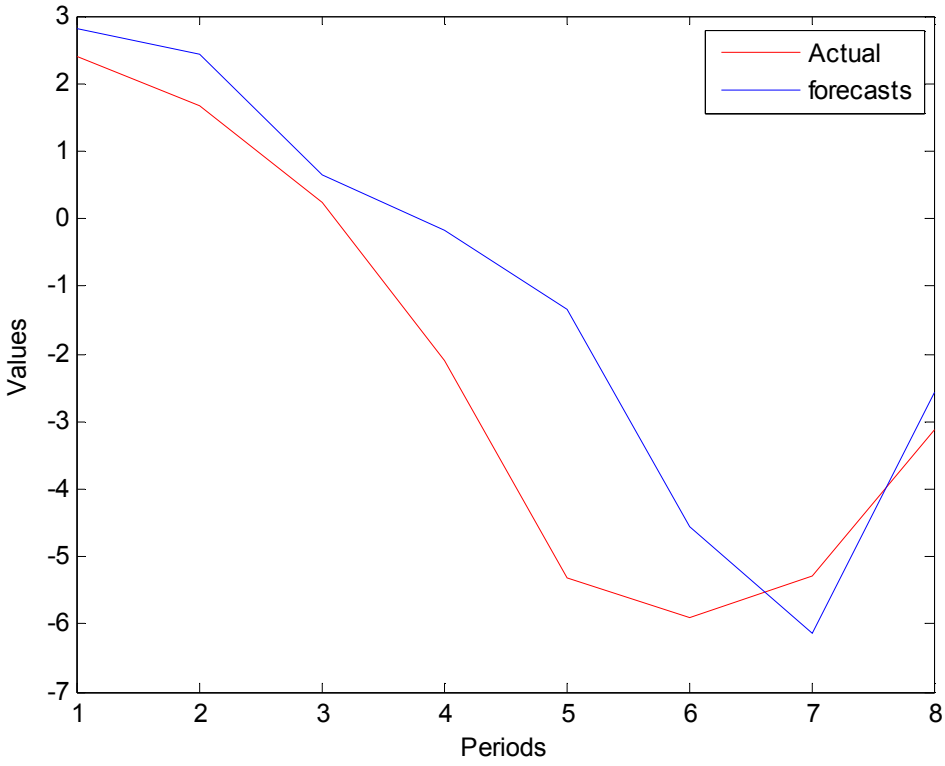


**Fig. 3** Out-of-sample forecasts UK GDP

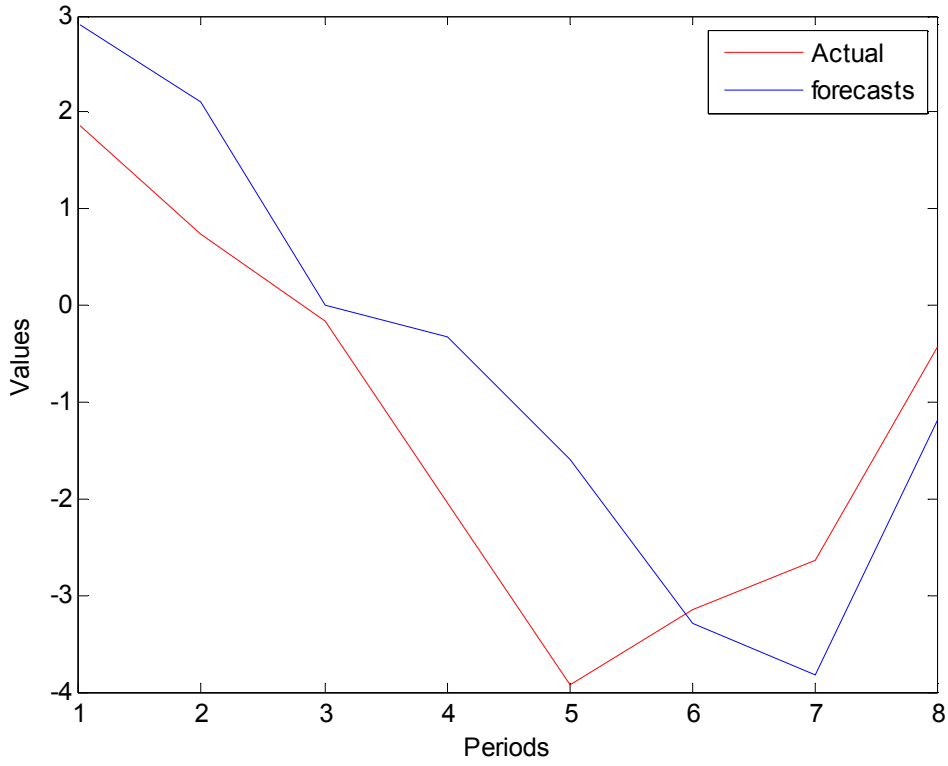


**Fig. 4** Out-of-sample forecasts for Italian GDP

We do not present the estimated coefficients as you can examined by yourself, but only AR(1) and AR(2) are statistically significant, as also we accept the null hypothesis of no autocorrelation based on p-values which is equal with 0.2291. In figure 4 we present the out-of-sample forecasts for Italian GDP.

In the next step we examine again UK GDP growth rate with the same settings but this time we minimize the squared error of regression residuals in equation (6) using MATLAB routine 2 in appendix. We define the same settings with the previous case, except from learning and adaptation rates, which are set up at 0.05. In figures 5 and 6 the in-sample and out-of-sample forecasts respectively versus the actual values are presented.

In appendix we provide two additional routines. MATLAB routines 3 and 4 are the same with routines 1 and 2 respectively, with the only difference that employ multi-period ahead forecasts.

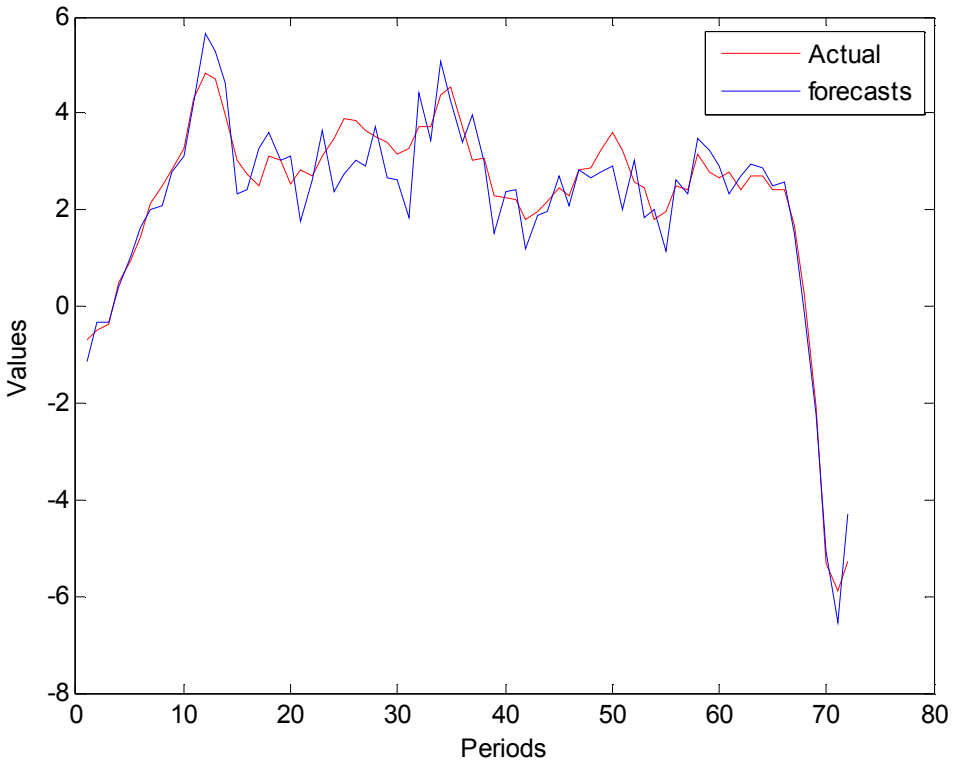


**Fig. 5** In-sample forecasts for UK GDP

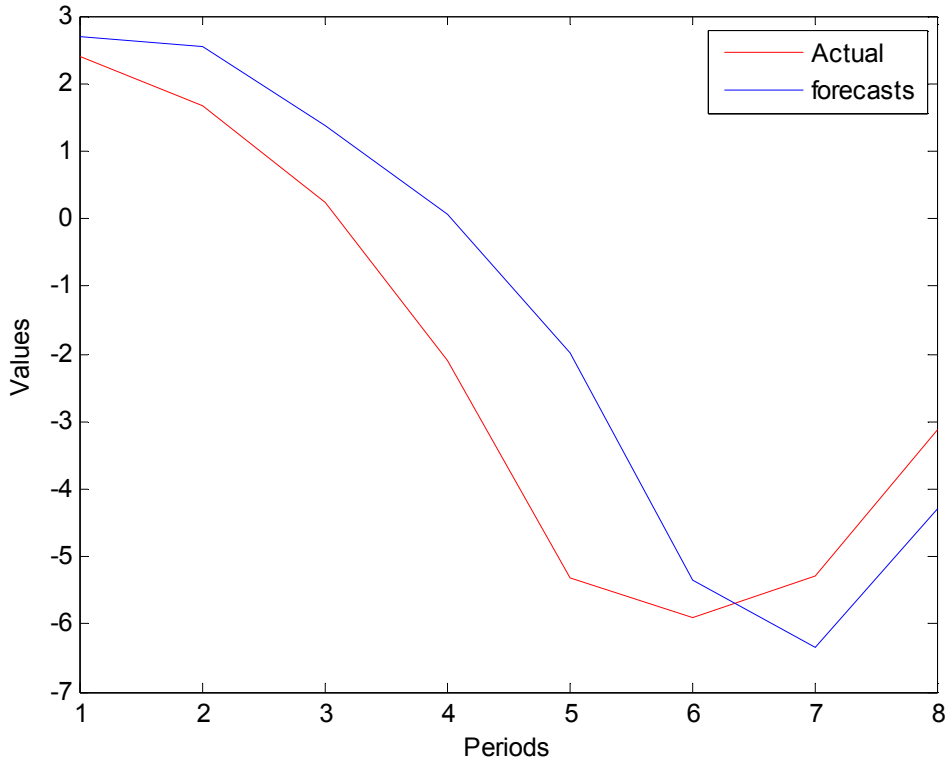


**Fig. 6** Out-of-sample forecasts UK GDP

Next we present one example more for the S&P 500 stock index returns. We take as sample the 100 last trading days of 2009, where the first 90 are obtained as the training period and the last 10 for testing. The settings are the same, but the learning and adaptation rates are set up at 0.8 and we obtain an AR(2) process without constant. We follow the process of relation (7) and MATLAB routine 5 in appendix. In figures 7 and 8 the in-sample and out-of-sample forecasts respectively are reported.

The final example we examine is the day of the week effect for FTSE 100 stock index returns. The regression we estimate is:

$$r_t = \beta_1 D_1 + \beta_2 D_2 + \beta_3 D_3 + \beta_4 D_4 + \beta_5 D_5 + \varepsilon_t \qquad (8)$$

,where $r_t$ denotes the stock index logarithmic returns, $\beta_i$ denotes the estimated coefficients for $i=1,2..5$, $D_1$ is the dummy variable for Monday taking value 1 for returns on Monday and 0 otherwise and so on until dummy $D_5$ is the dummy variable for Friday.

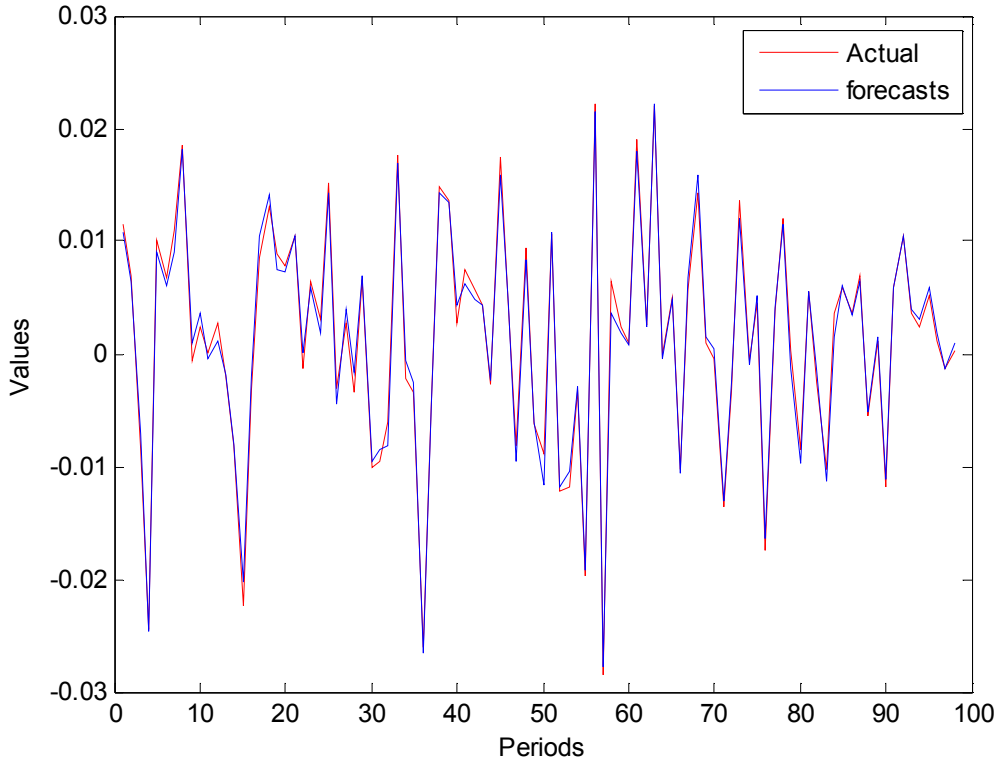


**Fig. 7** In-sample for S&P 500 index returns

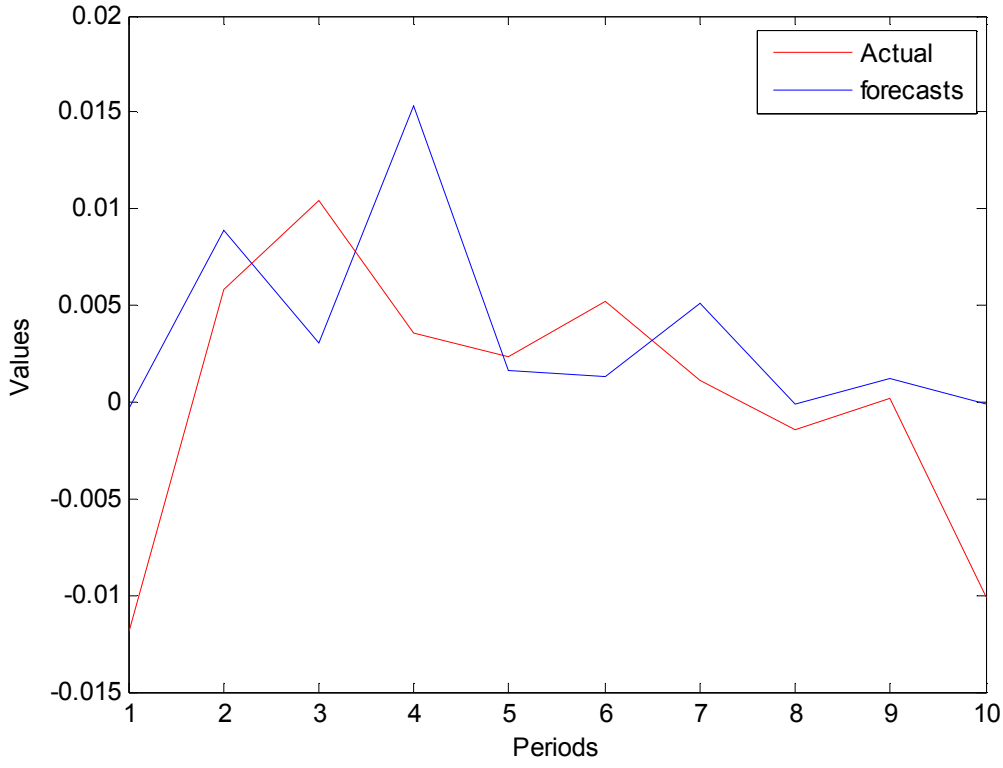


**Fig. 8** Out-of-sample for S&P 500 index returns

In that case the inputs which are the dummy variables are multiplied with the tap-weight vector. This is useful because the classification of one and zero is not correct, because it is not enough to set up 1 for the returns on the specific day. More specifically, each day presents different returns on each week and the traditional classification is not clear. For this reason different weights are assigned in each day in each week. We examine 2009. In table 1 we present the estimated results for the day of the week effect and we conclude that there is only a reverse Monday effect where the only significant and positive returns are reported on Monday. This is not absolutely because of the procedure we follow. It might be a reverse Monday effect because of the short and specific period, where 2009 might resent positive Monday returns. On the other hand it should be noticed that with ordinary least squares we found significant positive returns only on Wednesday, Thursday and Friday, with the highest returns on Wednesday.

**Table 1.** Estimation for the day of the week effect with LMS algorithm

| **Estimated coefficients** | | | | | **Diagnostic tests** | |
|---|---|---|---|---|---|---|
| $\beta_1$ | $\beta_2$ | $\beta_3$ | $\beta_4$ | $\beta_5$ | Q-stat (5) | ARCH-LM (4) |
| 0.0162 | 0.0040 | -0.0039 | -0.0028 | 0.0043 | 5.2721 | 11.6802 |
| (1.707)*** | (0.470) | (-0.450) | (-0.356) | 0.490) | [0.2701] | (0.0394 |

t-statistics in parentheses, p-values in brackets *** denotes significance in $\alpha$=0.10, Q-stat is for Ljung-Box Q-statistic, ARCH-LM test for ARCH effects

## References


Gabor, D. (1954). Communication theory and cybernetics. *IRE Transactions on Circuit Theory*, Vol. CT-1, pp. 19-31.

Hayes, M. H. (1996). Statistical Signal Processing and Modeling, Wiley, 1996.

Haykin, S. (1996). *Adaptive Filter Theory*, 3rd Edition, Prentice Hall, 1996.

Widrow, B., and M.E. Hoff, Jr. (1960). Adaptive switching circuits. *IRE WESCON Convention Record*, pp. 96-104.

## Appendix

## MATLAB routine 1

## One period ahead forecast with cost function (4) minimization

```
clear all;
load file.mat
% data includes vector y which is a univariate time-series

lms_type=2; % LMS type
constant=0  % 0 for no constant and 1 for constant
eta=0.5     % Learning rate
m=0.5       % Adaptation or step size rate
nlag=2      % Number of lags
nforecast=8 % Number of one period ahead forecasts

for jj=nforecast:-1:1
y=data(1:end-jj,:)
    clear x
for pp=1:nlag

    x(:,pp)=lagmatrix(y,pp)
end



i1  =  find(isnan(x));
i2  =  find(isnan(diff([x ; zeros(1,size(x,2))]) .* x));
if (length(i1) ~= length(i2)) || any(i1 - i2)
   error('Series cannot contain NaN).')
end

if any(sum(isnan(x)) == size(x,1))
   error('A realization of ''x'' is completely missing (all
NaN''s).')
end

first_Row  =  max(sum(isnan(x))) + 1;
x             =  x(first_Row:end , :);
y=y(first_Row:end,:)

[t nj]=size(y)


if constant == 0
        x=x
elseif constant==1
 x=[ones(t,1) x];
end
[nk ni]=size(x)
%num_hidden=floor(ni/2+sqrt(t));
num_hidden=ni

n = randn(nk,1);
% Add noise
n = n * std(y)/(10*std(n));

d = y + n;
% The dependent variable with noise
```

```
a=-0.5
b=0.5

%rand('state',sum(100*clock))            % Resets it to a different
state each time.
rand('state',0)                         % Resets the generator to its
initial state.
w=a + (b-a) *rand(nk,1);
%w=[randn(u,1) zeros(u,nk1-1)];
dw=zeros(size(w))
epochs=1;
maxepochs=10;
Error=10;
goal_error=7;

while (epochs<maxepochs & Error>goal_error)

    Error=0;
if lms_type==1;
for i=1:nk,
   e(i)=d(i)-w(i)*x(i);
   w(i)=w(i) + (m*e(i)*x(i))
   y1(i)=w(i)*x(i)
end

elseif lms_type==2;
for i=1:nk,
   e(i)=d(i)-w(i)*x(i);
   w(i)=w(i) + (m*e(i)*x(i))/(x(i)'*x(i))
   y1(i)=w(i)*x(i)
end
end
Error=e*e'

w=w+eta*dw


epochs=epochs+1
arraygbest ( epochs )= Error;
indexiter ( epochs ) = epochs;
end

for i=1:nk
    for ii =1:ni
   newx(i,ii)=w(i).*x(i,ii);
    end
end

bols=inv(newx'*newx)*newx'*y
%bols=pinv(newx)*y
res=y-newx*bols
yhat(jj,:)=newx(end,:)*bols
end
yf=newx*bols

s2 = (y-newx*bols)'*(y-newx*bols)/(nk-ni);
```

```
Vb=s2*inv(newx'*newx);              % Get the variance-covariance
matrix
se=sqrt(diag(Vb));              % Get coefficient standard errors
tstudent=bols./se;

[H,p_Jung,Qstat,CriticalValue] =lbqtest(res,4,0.05)



for iii=1:nforecast
yfore(iii,:)=yhat(end-iii+1,:)
iii=iii+1
end
test_y=data(end-nforecast+1:end,:)

figure, plot(y,'-r'); hold on; plot(d,'-b');
xlabel('Periods')
ylabel('Values')
%title('In_sample forecasts')
h1 = legend('Actual','forecasts',1);
figure, plot(test_y,'-r'); hold on; plot(yfore,'-b');
xlabel('Periods')
ylabel('Values')
%title('Out_of_sample forecasts')
h = legend('Actual','forecasts',1);
figure, plot (indexiter , arraygbest );
xlabel('epochs')
ylabel('Error')
title('Number of Epochs')
```

## MATLAB routine 2

## One period ahead forecast with sum squared error residuals minimization

```
clear all;
load file.mat
% data includes vector y which is a univariate time-series

lms_type=2; % LMS type
constant=0  % 0 for no constant and 1 for constant
eta=0.5     % Learning rate
m=0.5       % Adaptation or step size rate
nlag=2      % Number of lags
nforecast=8 % Number of one period ahead forecasts

for jj=nforecast:-1:1
y=data(1:end-jj,:)
    clear x
for pp=1:nlag

    x(:,pp)=lagmatrix(y,pp)
end



i1  =  find(isnan(x));
i2  =  find(isnan(diff([x ; zeros(1,size(x,2))]) .* x));
if (length(i1) ~= length(i2)) || any(i1 - i2)
```

```
   error('Series cannot contain NaN).')
end

if any(sum(isnan(x)) == size(x,1))
   error('A realization of ''x'' is completely missing (all
NaN''s).')
end

first_Row  =  max(sum(isnan(x))) + 1;
x          =  x(first_Row:end , :);
y=y(first_Row:end,:)

[t nj]=size(y)


if constant == 0
        x=x
elseif constant==1
 x=[ones(t,1) x];
end
[nk ni]=size(x)
%num_hidden=floor(ni/2+sqrt(t));
num_hidden=ni

n = randn(nk,1);
% Add noise
n = n * std(y)/(10*std(n));

d = y + n;
% The dependent variable with noise


a=-0.5
b=0.5

%rand('state',sum(100*clock))           % Resets it to a different
state each time.
rand('state',0)                        % Resets the generator to its
initial state.
w=a + (b-a) *rand(nk,1);
%w=[randn(u,1) zeros(u,nk1-1)];
dw=zeros(size(w))
epochs=1;
maxepochs=10;
Error=10;
goal_error=7;

while (epochs<maxepochs & Error>goal_error)

    Error=0;
if lms_type==1;
for i=1:nk,
   e(i)=d(i)-w(i)*x(i);
   w(i)=w(i) + (m*e(i)*x(i))
   y1(i)=w(i)*x(i)
end

elseif lms_type==2;
for i=1:nk,
```

```
   e(i)=d(i)-w(i)*x(i);
   w(i)=w(i) + (m*e(i)*x(i))/(x(i)'*x(i))
   y1(i)=w(i)*x(i)
end
end
w=w+eta*dw

 w=w+eta*dw
for i=1:nk
    for ii =1:ni
   newx(i,ii)=w(i).*x(i,ii);
    end
end
%bols=inv(newx'*newx)*newx'*y
bols=pinv(newx)*y
res=y-newx*bols
Error=res'*res
yhat(jj,:)=newx(end,:)*bols

epochs=epochs+1
arraygbest ( epochs )= Error;
indexiter ( epochs ) = epochs;
end
 end
yf=newx*bols

s2 = (y-newx*bols)'*(y-newx*bols)/(nk-ni);
Vb=s2*inv(newx'*newx);              % Get the variance-covariance
matrix
se=sqrt(diag(Vb));              % Get coefficient standard errors
tstudent=bols./se;

[H,p_Jung,Qstat,CriticalValue] =lbqtest(res,4,0.05)

for iii=1:nforecast
yfore(iii,:)=yhat(end-iii+1,:)
iii=iii+1
end
test_y=data(end-nforecast+1:end,:)

figure, plot(y,'-r'); hold on; plot(d,'-b');
xlabel('Periods')
ylabel('Values')
%title('In_sample forecasts')
h1 = legend('Actual','forecasts',1);
figure, plot(test_y,'-r'); hold on; plot(yfore,'-b');
xlabel('Periods')
ylabel('Values')
%title('Out_of_sample forecasts')
h = legend('Actual','forecasts',1);
figure, plot (indexiter , arraygbest );
xlabel('epochs')
ylabel('Error')
title('Number of Epochs')
```

## MATLAB routine 3

## Multi period ahead forecast with cost function (4) minimization

```
nforecast=4
for ii=1:nforecast
clear all;
load trainfile.mat


lms_type=2;
constant=0
nlag=3
m=0.5
eta=0.5

y=train_data

for pp=1:nlag

    x(:,pp)=lagmatrix(y,pp)
end



i1  =  find(isnan(x));
i2  =  find(isnan(diff([x ; zeros(1,size(x,2))]) .* x));
if (length(i1) ~= length(i2)) || any(i1 - i2)
   error('Series cannot contain NaN).')
end

if any(sum(isnan(x)) == size(x,1))
   error('A realization of ''x'' is completely missing (all
NaN''s).')
end

first_Row  =  max(sum(isnan(x))) + 1;
x             =  x(first_Row:end , :);
y=y(first_Row:end,:)

[t nj]=size(y)



if constant == 0
        x=x
elseif constant==1
 x=[ones(t,1) x];
end
[nk ni]=size(x)
%num_hidden=floor(ni/2+sqrt(t));
num_hidden=ni

n = randn(nk,1);
% Add noise
n = n * std(y)/(10*std(n));

d = y + n;
% The dependent variable with noise
```

```
a=-0.5
b=0.5

%rand('state',sum(100*clock))            % Resets it to a different
state each time.
rand('state',0)                         % Resets the generator to its
initial state.
w=a + (b-a) *rand(nk,1);
%w=[randn(u,1) zeros(u,nk1-1)];
dw=zeros(size(w))
epochs=1;
maxepochs=10;
Error=10;
goal_error=7;

while (epochs<maxepochs & Error>goal_error)

    Error=0;


if lms_type==1;
for i=1:nk,
   e(i)=d(i)-w(i)*x(i);
   w(i)=w(i) + (m*e(i)*x(i))
   y1(i)=w(i)*x(i)
end

elseif lms_type==2;
for i=1:nk,
   e(i)=d(i)-w(i)*x(i);
   w(i)=w(i) + (m*e(i)*x(i))/(x(i)'*x(i))
   y1(i)=w(i)*x(i)
end
end
Error=e*e'

w=w+eta*dw


epochs=epochs+1
arraygbest ( epochs )= Error;
indexiter ( epochs ) = epochs;
end

for i=1:nk
    for ii =1:ni
   newx(i,ii)=w(i).*x(i,ii);
    end
end

%bols=inv(newx'*newx)*newx'*y
bols=pinv(newx)*y
res=y-newx*bols

yhat=newx(end,:)*bols

train_data=[ train_data;yhat]
```

```
save(trainfile, ' train_data ')
end
nforecast=4
load testfile.mat
test_y= test_data (end-nforecast+1:end,:)
yfore= train_data (end-nforecast+1:end,:)

h1 = legend('Actual','forecasts',1);
figure, plot(test_y,'-r'); hold on; plot(yfore,'-b');
xlabel('Periods')
ylabel('Values')
%title('Out_of_sample forecasts')
```

**MATLAB routine 4**

**Multi period ahead forecast with sum squared error residuals minimization**

```
nforecast=4
for ii=1:nforecast
clear all;
load trainfile.mat

lms_type=2;
constant=0
nlag=3
m=0.5
eta=0.5

y=train_data

for pp=1:nlag

    x(:,pp)=lagmatrix(y,pp)
end

 i1  =  find(isnan(x));
i2  =  find(isnan(diff([x ; zeros(1,size(x,2))]) .* x));
if (length(i1) ~= length(i2)) || any(i1 - i2)
   error('Series cannot contain NaN).')
end

if any(sum(isnan(x)) == size(x,1))
   error('A realization of ''x'' is completely missing (all
NaN''s).')
end

first_Row  =  max(sum(isnan(x))) + 1;
x             =  x(first_Row:end , :);
y=y(first_Row:end,:)

[t nj]=size(y)

 if constant == 0
        x=x
elseif constant==1
 x=[ones(t,1) x];
end
```

```
[nk ni]=size(x)
%num_hidden=floor(ni/2+sqrt(t));
num_hidden=ni
 n = randn(nk,1);
% Add noise
n = n * std(y)/(10*std(n));

d = y + n;
% The dependent variable with noise

a=-0.5
b=0.5

%rand('state',sum(100*clock))            % Resets it to a different
state each time.
rand('state',0)                         % Resets the generator to its
initial state.
w=a + (b-a) *rand(nk,1);
%w=[randn(u,1) zeros(u,nk1-1)];
dw=zeros(size(w))
epochs=1;
maxepochs=10;
Error=10;
goal_error=7;

while (epochs<maxepochs & Error>goal_error)

    Error=0;
if lms_type==1;
for i=1:nk,
   e(i)=d(i)-w(i)*x(i);
   w(i)=w(i) + (m*e(i)*x(i))
   y1(i)=w(i)*x(i)
end

elseif lms_type==2;
for i=1:nk,
   e(i)=d(i)-w(i)*x(i);
   w(i)=w(i) + (m*e(i)*x(i))/(x(i)'*x(i))
   y1(i)=w(i)*x(i)
end
end

w=w+eta*dw

 for i=1:nk
    for ii =1:ni
   newx(i,ii)=w(i).*x(i,ii);
    end
end

%bols=inv(newx'*newx)*newx'*y
bols=pinv(newx)*y
res=y-newx*bols
Error=res'*res
epochs=epochs+1
arraygbest ( epochs )= Error;
indexiter ( epochs ) = epochs;
end
```

```
yhat=newx(end,:)*bols

train_data=[ train_data;yhat]
save(trainfile, ' train_data ')
end
nforecast=4
load testfile.mat
test_y= test_data (end-nforecast+1:end,:)
yfore= train_data (end-nforecast+1:end,:)

h1 = legend('Actual','forecasts',1);
figure, plot(test_y,'-r'); hold on; plot(yfore,'-b');
xlabel('Periods')
ylabel('Values')
%title('Out_of_sample forecasts')
```

## MATLAB routine 5

## One period ahead forecast with relation (7) and cost function (4)

```
clear all;
load file.mat
% data includes vector y which is a univariate time-series

lms_type=2; % LMS type
constant=0  % 0 for no constant and 1 for constant
eta=0.8     % Learning rate
m=0.8       % Adaptation or step size rate
nlag=2      % Number of lags
nforecast   % Number of one period ahead forecasts

for jj=nforecast:-1:1
y=data(1:end-jj,:)
    clear x
for pp=1:nlag

    x(:,pp)=lagmatrix(y,pp)
end



i1  =  find(isnan(x));
i2  =  find(isnan(diff([x ; zeros(1,size(x,2))]) .* x));
if (length(i1) ~= length(i2)) || any(i1 - i2)
   error('Series cannot contain NaN).')
end

if any(sum(isnan(x)) == size(x,1))
   error('A realization of ''x'' is completely missing (all
NaN''s).')
end

first_Row  =  max(sum(isnan(x))) + 1;
x             =  x(first_Row:end , :);
y=y(first_Row:end,:)

[t nj]=size(y)
```

```
if constant == 0
        x=x
elseif constant==1
 x=[ones(t,1) x];
end
[nk ni]=size(x)
%num_hidden=floor(ni/2+sqrt(t));
num_hidden=ni

n = randn(nk,1);
% Add noise
n = n * std(y)/(10*std(n));

d = y + n;
% The dependent variable with noise


a=-0.5
b=0.5

%rand('state',sum(100*clock))            % Resets it to a different
state each time.
rand('state',0)                         % Resets the generator to its
initial state.
w=a + (b-a) *rand(nk,1);
%w=[randn(u,1) zeros(u,nk1-1)];
dw=zeros(size(w))
epochs=1;
maxepochs=10;
Error=10;
goal_error=0.005;

while (epochs<maxepochs & Error>goal_error)

    Error=0;

if lms_type==1;
for i=1:nk,
   e(i)=d(i)-w(i)*x(i);
   w(i)=w(i) + (m*e(i)*x(i))
   y1(i)=w(i)*x(i)
end

elseif lms_type==2;
for i=1:nk,
   e(i)=d(i)-w(i)*x(i);
  w(i)=w(i) + (m*e(i)*x(i))/(x(i)'*x(i))
  y1(i)=w(i)*x(i)
end
end
Error=e*e'

w=w+eta*dw


epochs=epochs+1
```

```
arraygbest ( epochs )= Error;
indexiter ( epochs ) = epochs;
end


yhat(jj,:)=w(i)*y(end,:)

end



for iii=1:nforecast
yfore(iii,:)=yhat(end-iii+1,:)
iii=iii+1
end
test_y=data(end-nforecast+1:end,:)

figure, plot(y,'-r'); hold on; plot(y1,'-b');
xlabel('Periods')
ylabel('Values')
%title('In_sample forecasts')
h1 = legend('Actual','forecasts',1);
figure, plot(test_y,'-r'); hold on; plot(yfore,'-b');
xlabel('Periods')
ylabel('Values')
%title('Out_of_sample forecasts')
h = legend('Actual','forecasts',1);
figure, plot (indexiter , arraygbest );
xlabel('epochs')
ylabel('Error')
title('Number of Epochs')
```

**MATLAB routine 6**

**LMS algorithm for the day of the week effects on FTSE 100 stock returns**

```
clear all;
load file.mat

% data is a matrix where the first five columns are the dummy
variables and the last is the stock returns.


lms_type=1;
constant=0
weights=1
nforecast=10
m=0.08 % Adaptation rate
eta=0.08
x=data(:,1:end-1)
y=data(:,end)
```

```
[nk ni]=size(x)
%num_hidden=floor(ni/2+sqrt(t));
num_hidden=ni

n = randn(nk,1);
% Add noise
n = n * std(y)/(10*std(n));

d = y + n;
% The dependent variable with noise


a=-0.5
b=0.5

%rand('state',sum(100*clock))           % Resets it to a different
state each time.
rand('state',0)                        % Resets the generator to its
initial state.
w=a + (b-a) *rand(nk,1);
%w=[randn(u,1) zeros(u,nk1-1)];
dw=zeros(size(w))
epochs=1;
maxepochs=2;
Error=10;
goal_error=0.005;

while (epochs<maxepochs & Error>goal_error)

    Error=0;
 if lms_type==1;
for i=1:nk,
   e(i)=d(i)-w(i)*x(i);
   w(i)=w(i) + (m*e(i)*x(i))
   y1(i)=w(i)*x(i)
end

elseif lms_type==2;
for i=1:nk,
   e(i)=d(i)-w(i)*x(i);
  w(i)=w(i) + (m*e(i)*x(i))/(x(i)'*x(i))
  y1(i)=w(i)*x(i)
end
end
Error=e*e'
for i=1:nk
    if isnan(w)
 w(i)=0
    end
end
w=w+eta*dw


epochs=epochs+1
arraygbest ( epochs )= Error;
indexiter ( epochs ) = epochs;
end
```

```
for i=1:nk
    for ii =1:ni
   newx(i,ii)=w(i).*x(i,ii);
    end
end

%bols=inv(newx'*newx)*newx'*y
bols=pinv(newx)*y
res=y-newx*bols
s2 = (y-newx*bols)'*(y-newx*bols)/(nk-ni);
Vb=s2*inv(newx'*newx);              % Get the variance-covariance
matrix
se=sqrt(diag(Vb));              % Get coefficient standard errors
tstudent=bols./se;

[H,p_Jung,Qstat,CriticalValue] =lbqtest(res,5,0.05)
[H1,pValue,ARCHstat,CriticalValue1] = archtest(res,5,0.05)
```