**Computer task 3**

**Data mining and neural networks**

All data is from Yahoo Finance website, and four time series are from Baidu, Yahoo, Ebay and HSBC, respectively. All data collected for 04/09/2012-30/08/2013 and only closing price has been used.

Choose Baidu closing value as a target g(t), the other three as supporting data, , , . Each of them has 249 data.

1)

These four time series are closing price of a certain stock so that all data is available except days when stock did not trade. From the data I downloaded, it is clear to see there is no missing value.

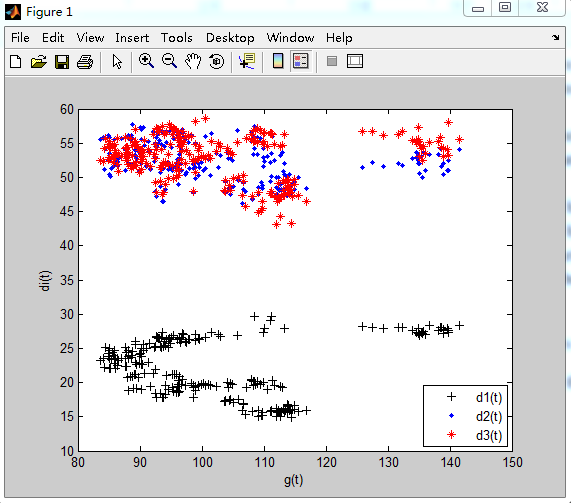


Figure 1

2) The principle of linear regression

According to the definition of linear regression, we have input vector and want to predict a real-valued output g(t). The linear regression model has the form

Where are unknown coefficients.

I obtained the value of : ,, , . Thus, the regression expression:

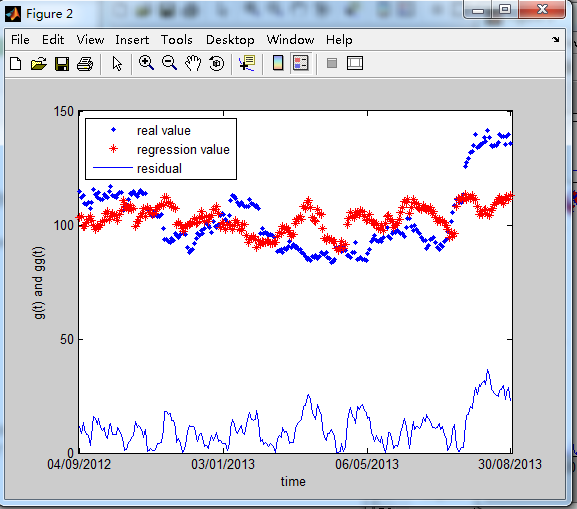


Figure 2

From the figure above, at first, the goodness of fit is relatively well and the residual between real-value and the regression value is small. Till 130th data, residual becomes larger. Another is from 225thto 250th, residual becomes maximal. It is also clear to see from red and blue scatter diagram. The residual variance is 44751, which means this linear regression is not good enough to describe the original data. Even worse, from the figure, the linear data could not describe the trend (up or down) of original data correctly. It is apparent that the estimated trend line obtained via simple linear regression does not quite capture the trend of the data.

3) regression with lagged variables

Stock values are not mess. Next-day value might be associated with data which belongs to a period of time ago. We can use today’s data to predict tomorrow. The linear regression model has the form

Where are unknown coefficients.

Using simple linear regression, I obtained the estimated coefficients: ,, , , . the regression expression: +0.0168.

Since this evaluation has taken the lagged variable into account, the approximation value is much more exact and accurate. The residual variance is 1490, which is much smaller compared to residual variance calculated in 2).

The residual between and is drawn in Figure 3. Apparently, the real value () and predicted value ( ) almost coincide such that the residual is very small. For the lack of yesterday’s data to predict the trend of data, it is less accurate.

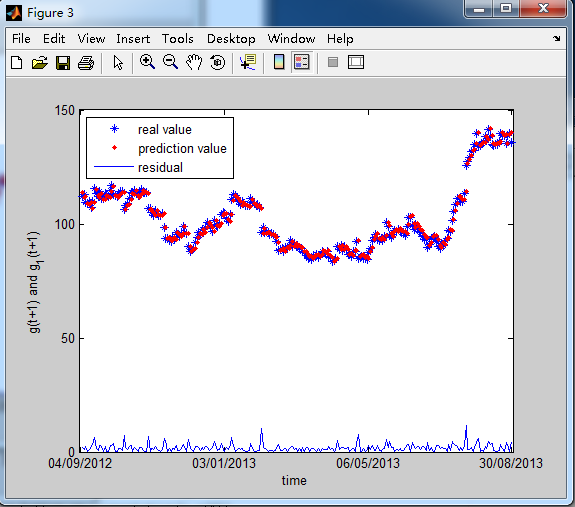


Figure 3

4)

In order to compare result of forecasting to the next-day forecast with today’s origin data, we use subtraction calculating the difference. Theoretically, the difference is not substantial as the next-day forecast value would have to be based on present value. It is clear to see the difference approaches to zero, which is less than residual of regression with lagged variables.

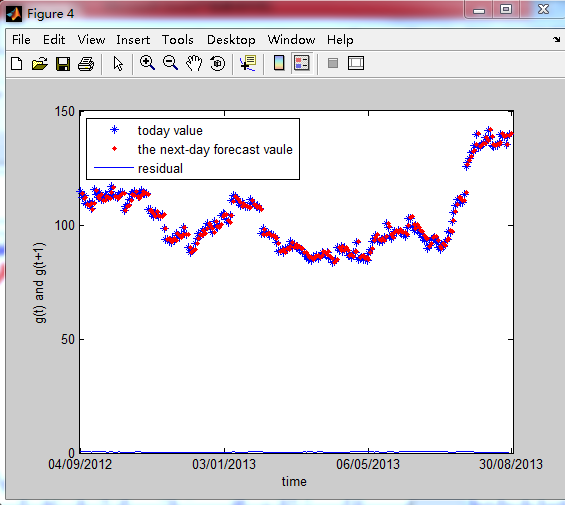


Figure 4

5) bottom-up segmentation algorithm

The Bottom-Up algorithm is the natural complement to the Top-Down algorithm.

① the algorithm begins by creating the finest possible approximation of the time series, so that n/2 segments joining adjacent points in a n-length time series.

② compute the cost of merging adjacent segments

③ iteratively merge the lowest cost pair until a stopping criterion is met

(the stopping criterion is based on error tolerance value: stop when mean error exceeds the selected level of tolerance)

The pseudocode for the algorithm is shown as follows:

|  |
| --- |
| **Algorithm** Seg\_TS = Bottom\_Up(T , max\_error)  **for** i=1:2:length(T) // Create initial fine approximation.  Seg\_TS = concat(Seg\_TS, create\_segment(T[i:i+1]));  **end**;  **for** i=1:length(Seg\_TS) – 1 // Find cost of merging each pair of segments.  merge\_cost(i) = calculate\_error([merge(Seg\_TS(i), Seg\_TS(i+1))]);  **end**;  **while** min(merge\_cost) < max\_error // While not finished.  index = min(merge\_cost); // Find “cheapest” pair to merge.  Seg\_TS(index) = merge(Seg\_TS(index), Seg\_TS(index+1))); // Merge them.  delete(Seg\_TS(index+1)); // Update records.  merge\_cost(index) = calculate\_error(merge(Seg\_TS(index), Seg\_TS(index+1)));  merge\_cost(index-1) = calculate\_error(merge(Seg\_TS(index-1), Seg\_TS(index)));  **end**; |

Table 1

Levels of tolerance (20% of average price, 10% of average price, 5% of average price and 1% of average price).

Since the algorithm must divide the data into n/2 segments, so that it can only deal with even data. Choosing 248 data to execute the program, there is 124 segments, then, calculate the cost of merging adjacent segments. We can obtain 123 costs of 124 segments. If the cost is more than levels of tolerance, this segment cannot merge with its latter segment.

20% of average price

By calculation, segment 12,13,14,52,53,80,81,114,115 cannot be merged.

10%of average price

By calculation, segment 4,5,9,10,12,13,14,19,20,21,24,25,38,39,43,44,45,52,53,80,81,95,96,111,112,114,115,116,118,119,123,124 cannot be merged.

5%of average price

By calculation, segment 4,5,9,10,12,13,14,19,20,21,24,25,26,30,31,38,39,40,43,44,45,52,53,74,75,80,81,88,89,94,95,96,107,108,110,111,112,114,115,116,117,118,119,123,124 cannot be merged.

1%of average price

By calculation, there are fewer segments which can be merged.

Reference

[1]Lecture Notes

[2] Segmenting Time Series: A Survey and Novel Approach

[3]How to Separate Regular Prices from Promotional Prices

Appendix

close all;

clear;

clc;

format compact;

load t\_series.mat;

g=time(:,1);

d1=time(:,2);d2=time(:,3);d3=time(:,4);

figure(1)

plot(g,d1,'K+');hold on

plot(g,d2,'b.');hold on

plot(g,d3,'r\*');hold on

xlabel('g(t)');

ylabel('di(t)');

legend('d1(t)','d2(t)','d3(t)',4);

gg=L\_regression(d1,d2,d3,g);

[rv1,r1]=SSerr(g,gg);

t=1:length(g);

figure(2)

plot(t,g,'b.');hold on

plot(t,gg,'r\*');hold on

plot(t,r1);hold on

xlabel('time');ylabel('g(t) and gg(t)');

legend('real value','regression value','residual',2);

set(gca,'xtick',[1 84 167 249]);

set(gca,'XTickLabel',{'04/09/2012','03/01/2013','06/05/2013','30/08/2013'});

ga=g(2:249);

g\_1=lr(g(1:248),d1(1:248),d2(1:248),d3(1:248),ga);

[rv2,r2]=SSerr(ga,g\_1);

t1=1:length(ga);

figure(3)

plot(2:249,ga,'b\*');hold on

plot(2:249,g\_1,'r.');hold on

plot(2:249,r2);hold on

xlabel('time');ylabel('g(t+1) and g\_1(t+1)');

legend('real value','prediction value','residual',2);

set(gca,'xtick',[1 84 167 249]);

set(gca,'XTickLabel',{'04/09/2012','03/01/2013','06/05/2013','30/08/2013'});

residual=abs(g\_1-g(1:248));

figure(4)

plot(t1,g(1:248),'b\*');hold on

plot(2:249,g\_1,'r.');hold on

plot(t1,residual);

xlabel('time');ylabel('g(t) and g(t+1)');

legend('today value','the next-day forecast vaule','residual',2);

set(gca,'xtick',[1 84 167 249]);

set(gca,'XTickLabel',{'04/09/2012','03/01/2013','06/05/2013','30/08/2013'});

AP=[0.2,0.1,0.05,0.01];

v=AP.\*mean(g);

[segment1,best1,re1,err1] = bottom\_up\_segmentation(g(1:248),v(1));

[segment2,best2,re2,err2] = bottom\_up\_segmentation(g(1:248),v(2));

[segment3,best3,re3,err3] = bottom\_up\_segmentation(g(1:248),v(3));

[segment4,best4,re4,err4] = bottom\_up\_segmentation(g(1:248),v(4));

function gg=L\_regression(a,b,c,d)

z=[ones(length(a),1),a,b,c];

temp=inv(z'\*z);

B=temp\*(z'\*d);

for i=1:length(a)

gg(i,1)=B(1)+B(2)\*a(i)+B(3)\*b(i)+B(4)\*c(i);

end

function [re\_v,re]=SSerr(x,y)

n=length(x);re\_v=0;

for i=1:n

re\_v=re\_v+(x(i)-y(i))^2;

re(i,1)=abs(x(i)-y(i));

end

function g\_1=lr(a,b,c,d,e)

z1=[ones(length(a),1),a,b,c,d];

temp1=inv(z1'\*z1);

B=temp1\*(z1'\*e);

for i=1:length(a)

g\_1(i,1)=B(1)+B(2)\*a(i)+B(3)\*b(i)+B(4)\*c(i)+B(5)\*d(i);

end

function [segment,best,r,err] = bottom\_up\_segmentation(data,value)

% d=data(1:(length(data)-1));

d=data;

t=(1:length(d))';

x=reshape(t,2,length(t)/2);

y=reshape(d,2,length(d)/2);

for i=1:(length(d))/2-1

c=polyfit([x(:,i);x(:,i+1)],[y(:,i);y(:,i+1)],1);

best(1:4,i)=(c(1)\*([x(:,i);x(:,i+1)]))+c(2);

err(i)=sum(([y(:,i);y(:,i+1)]-best(1:4,i)).^2);

end

segment=find(err>value);

r=find(err<value);