UPPSALA
UNIVERSITET

Signal processing

ECG reconstruction by RLS and LMS

Authors:

Oscar Uudelepp - 950128-1951

Elias Ericsson - 940127-3975

Uppsala

July 27, 2020

Chapter 1

Introduction

Reconstruction of signals is something that is needed in many fields of todays technology. For example, With the growing use of wireless transmission of data, it's essential for have a precise signal to achieve desirable results. Wireless transmissions often lose connection for small periods of time and then re-establish it fast again and during that downtime we usually don't notice it, like in a phone-call on a mobile device.
In this and many other cases a reconstruction of the signal can be done well enough and fast enough that we humans don't even notice the break in transmission, as devices nowadays have such powerful process-power.

Filtering in the form of RLS or otherwise can be used in countless other fields, for example when trying to predict what a market will look like based on information from other trades or what the weather will be like based on how it is in other areas. With more knowledge, data points and better applied filters to their tasks, more things can be predicted and therefor prepared for more accurately.
In 2010, PhysioNet held a challenge and one part of this challenge was to restore parts of biomedical signals and while this can be done in many ways, our focus was on using the recursive least squares adaptive filter algorithm to reconstruct a missing part of a ECG signal.

Chapter 2

Theory

Recursive Least Squares, RLS, and Least Mean Squars, LMS, were the two focused filters in this project.

Recursive Least Squares:
RLS minimizes the weighted linear least squares cost function and calculates the next value in the sequence as in formula 2.1.

$$y[n] = h[n] * \theta[n-1] + w[n] \tag{2.1}$$

Where the h[n] is our input and θ is the matrix of coefficients that generate new values. The w is a disturbance of the signal. To calculate θ the following four formulas are used.

$$\hat{\theta}[n] = \hat{\theta}[n-1] + K[n] * \bar{e}[n] \tag{2.2}$$

$$\bar{e}[n] = y[n] - h[n]^T * \bar{\theta}[n-1] \tag{2.3}$$

$$K[n] = \frac{P[n-1] * h[n]}{\lambda^n + h[n]^T * P[n-1] * h[n]} \tag{2.4}$$

$$P[n] = (I - K[n] * h^T[n]) * P[n-1] \tag{2.5}$$

In these equations the λ is a forgetting factor with a value between zero and one. The index n is the time where n is the current value and n-1 is the previous value and so on.

Least Mean Squares:
LMS minimizes the mean square error between the reference values and the approximated values and then calculates filter coefficients that can be used to approximate future values with only inputs.

$$x[n+1] = h[n+1] * y[n] + w[n] \tag{2.6}$$

Where y[n] are now the inputs and h[n] are the filter coefficients.

$$h[n+1] = h[n] + \mu * e[n] * y[n] \tag{2.7}$$

$$e[n] = y[n] - x[n] * h[n-1] \tag{2.8}$$

In these equations the μ is the step size with a value between zero and one. The index n is the time where n is the current value and n+1 is the next value and so on.

Chapter 3

Results

Table 3.1 displays the parameters for each filter that were used in the final reconstruction of the signal. These are not optimal and picked after a few tests as reasonable values.

Table 3.1: Chosen parameters for RLS and MLS filter

Type	N	M	p	λ	θ	μ	h	P
RLS	25	25	52	0.9992	0.2			10000 * index matrix
LMS	25	25	52			0.05	0.2	

Figure 3.1 and 3.2 display a part of the reconstructed signal with RLS and MLS respectively for patient 2. Results for the other patients are not shown in the report but can quickly be generated in MatLab using the .m files. Both show only a small portion of the reconstructed signal however they are very consistent and the zoomed in graphs portray the difference between the approximations and the reference signals. The reconstructed signals follow the reference signal at all times with minor deviations. Table 3.2 display the calculated values Q1 and Q2 for RLS and LMS with these specific values over all patients.

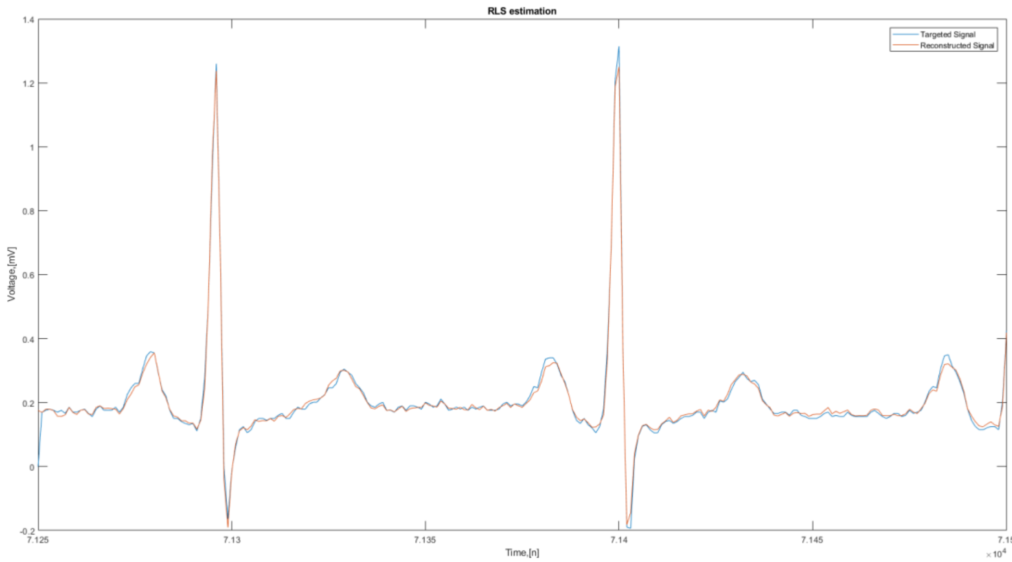

Figure 3.1: Reconstructed signal with RLS of patient 2.

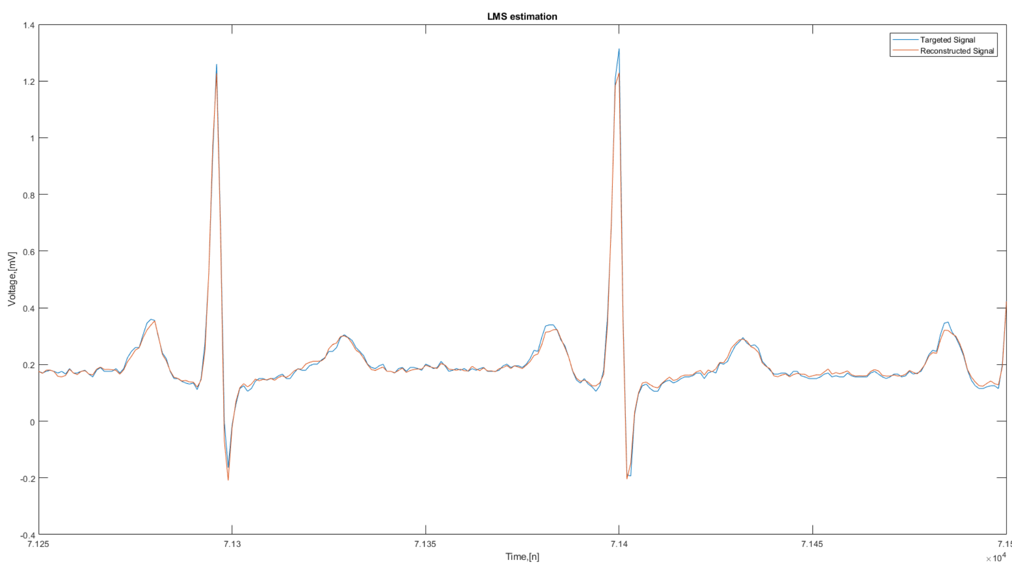

Figure 3.2: Reconstructed signal with LMS of patient 2.

Table 3.2: Q1 and Q2 for RLS and LMS

Patient	RLS Q1	RLS Q2	LMS Q1	LMS Q2
1	0.9633	0.9815	0.9623	0.9812
2	0.9947	0.9974	0.9942	0.9972
3	0.9796	0.9897	0.9754	0.9876
4	0.7712	0.8782	0.7290	0.8550
5	0.9433	0.9773	0.9156	0.9696
6	0.9239	0.9612	0.9226	0.9605
7	0.9714	0.9857	0.9713	0.9856
8	0.9936	0.9968	0.9890	0.9918

Table 3.3 shows the time it took to compute the filter coefficients and then reconstruct them afterwards, for each patient and each filter followed by an average.

Table 3.3: Computation time for RLS and LMS

Patient	RLS (seconds)	LMS (seconds)
1	1.357197	0.194069
2	1.242991	0.193467
3	1.347326	0.199887
4	1.350041	0.197917
5	1.38847	0.196987
6	1.454614	0.201918
7	1.460799	0.193632
8	1.349053	0.20202
average	1.368811375	0.197487125

Chapter 4

Discussion & Implementation

Both of the methods successfully reconstructed the signals with a good overall value of Q1 and Q2 with some exception to patient 4. For the patient 4 case, it is probably due to its signals being more randomly based than the others.

This projects main task, after actually writing a functional RLS and LMS code, is to choose the parameters for the filters. In our case this was done for the signals from patient one and for better results we should have made them an average of all patients. The parameters N and M are telling the filter how many previous values to take into the calculations and this is also why our codes loops are starting at the iteration number of the highest of N or M so that it won't try to read values before time zero. Higher N and M would mean that the filters can use more data which would in the end result in better made approximations but that would also mean higher computational time required. We found that with N and M set to about 25 the results didn't become more accurate if we increased the value and therefore we kept them there.

The forgetting factor λ will weigh older values less and make the tracking for the filter more or less susceptible to big changes, a lower λ will let the filter change values faster. Choosing a λ was done by incrementally increasing λ, using the found N=M=25, and storing the values of Q1 and Q2 in an array. This could also have been done by iteration to find an optimal value. The best mean value of Q1 and Q2 for the given λ was yielded at $\lambda = 0.9992$. In the beginning we thought we had picked a reasonable λ at 0.9998 but there was an anomaly when reconstructing the signal of patient 5. The 'a' coefficient made a sudden jump at around data point 60000 causing a disarray in the following estimations. Thus creating a bad Q1 and Q2 for the filter. We then reduced λ to 0.9992 which allowed the filter to successfully return to it's correct '2' coefficient fast enough not to ruin the values of Q1 and Q2 completely. But for the LMS filter it was done successfully without the error occurring.

Except for patient 4, the performance of the RLS filter was overall satisfactory for all patients however the computation time for the filter was higher than the LMS, as expected. It took around 1.3 seconds to compute the estimate of patient 2 while LMS needed 0.19 seconds. This can be reduced by reducing the N and M to lower values, but at the cost of quality of the reconstruction. Depending on what the filters purpose is this can be changed.

In the LMS filter the step size determines the speed of the convergence, the tracking capability and the fluctuations of the estimation. A higher μ yields a faster convergence and better tracking capability for non-stationary signals but will allow larger fluctuations which can create larger errors if the targeted signal is "choppy". A lower μ will have the opposite effect and depending on what the filter is estimating the μ should be optimized for that situation in particular. The method of choosing μ done by same technique as finding the lambda. The best mean value of Q1 and Q2 for the given μ was yielded at $\mu = 0.05$. The performance of the LMS was overall satisfactory with also an exception for patient 4.

Chapter 5

Conclusion & Recommendations

This project was quite broad and difficult to start of with and get our heads around but once the code started to take place it was very interesting and we learned a lot. Is this a good filter to use in a real world scenario for ECG? We have no idea but hopefully it's not even close to what the real competitors of the challenge were able to come up with. It's very interesting but we just have no knowledge about ECG and medicinal procedures. These filters could be developed in the same way but for other fields as well, potentially marketing, economic growth or weather tracking. It would be interesting to see how our filters would hold up in these fields and then change parameters to fit them.

To optimize these filters even further we could do more iterations to test different values on the parameters, for both the RLS and the LMS filters. The computations would become heavy quite quickly as each extra test of one parameter would add the full range of tests on all the others and it would end up in a computational case of N * M * λ * θ. Where the parameters in this equation are not their values but their range and resolution. The project wasn't about optimizing these parameters but it was a interesting side-task to find the best values for Q1 and Q2. Furthermore you could find even better results by making a filter that is both adaptive in the sense as RLS and LMS already are but also adaptive to which patient it is but that would seem like cheating as you can't know what medical problem a potential 9:th patient would have.

We are satisfied with our result and how we managed the challenge of this project and we definitely came out the with a lot more knowledge and understanding of the statistical subject of filters and their implementations. There are a huge amount of things to do before an actual step can be made towards the end goal, it is very hard to know that any part of your solution is correct until most other parts also are correct. This creates a steep start for the project but once your code is running there are a lot of things to play with to learn which is great for this type of learning.

Chapter 6

Codes

```
close all
clear all
%Patient Number
P = 2;
%Filter order
N = 25;
M = 25;
p =N+M+2;

% Gets the  signals from patient
% [xto, xlo,x2o,xm] = eli(P);
xto= importdata('ECG_2_II.mat');
xlo= importdata('ECG_2_AVR.mat');
x2o= importdata('ECG_2_V.mat');
xm= importdata('ECG_2_II_missing.mat');
%Makes the signals zero mean.
meanx1 = mean(xlo);
meanx2 = mean(x2o);
meanxt = mean(xto);

xt = xto - meanxt;
x1 = xlo- meanx1;
x2 = x2o- meanx2;

% LMS filter parameters
mymu = 0.05;
% mybeta=0.2;
% myeps=0.0001;
%
% LMS filter initilization
h=0.2*ones(p,1);

% number of iterations
Nsim=length(xt);
% keep filter coefficients and error in an array to plot later
hA=zeros(p,Nsim+1);
eA=zeros(1,Nsim);
hA(:,1)=h;
%
tic
for i= max(N,M)+1:Nsim -p
    %observations for this step
    y=[x1(i:-1:i-N);x2(i:-1:i-M)];
    %error
    e=xt(i,1)-h.'*y;
   % h update.
     h= h + y*mymu*e;
```

```
tic
for i= max(N,M)+1:Nsim -p
    %observations for this step
    y=[x1(i:-1:i-N);x2(i:-1:i-M)];
    %error
    e=xt(i,1)-h.'*y;
   % h update.
     h= h + y*mymu*e;
    %%%%%%%
    % store the values to plot later
    hA(:,i+1)=h;
    eA(1,i)=e;
end

for n=Nsim+1:(length(x1))

%  c= [x1(n+p-1:-1:n);x2(n+p-1:-1:n)];
  c=[x1(n:-1:n-N);x2(n:-1:n-M)];
xt(n) = h'*c;
end

 % adds the mean back to the predicted signal
 xt = xt+ meanxt;

 toc
 mse = mean((xm-(xt(Nsim+1:length(x1)))).^2);
x_var = var(xt(Nsim+1:length(x1),1));
covM = cov(xm, xt(Nsim+1:length(x1)));
Q1 = 1 - (mse/var(xm));
Q2 = covM(1,2)/(sqrt(x_var*var(xm)));
meanQ = (Q1+Q2)/2
figure
%puts the signal together with its missing part
xz = [xto;xm];
plot(xz);

hold on
p2 =plot(xt);
xlabel('Time,[n]')
 ylabel('Voltage,[mV]')
title('LMS estimation')
legend('Targeted Signal','Reconstructed Signal')
 xlim([71250 71500])
 hold off
%
% Below code is written for p=2
hf1=figure;
% Plot LMS filter coefficients
```

Figure 6.1: Code page 1 of RLS.

```
% Below code is written for p=2
hf1=figure;
% Plot LMS filter coefficients
h11=plot(hA(1,:)',':','Linewidth',2);
hold on
h12=plot(hA(2,:)',':','Linewidth',2);
hold on
% Plot the LMMSE filter coefficients (values LMS is trying to find)
% These are the same with the parameters of the process
% h21=plot(ones(Nsim+1,1)* c1, '-k','Linewidth',2);
% h22=plot(ones(Nsim+1,1)* c2, '--k','Linewidth',2);
xlim([1,Nsim+1]);
hxlabel=xlabel('Iteration');
hylabel=ylabel('Filter Coefficients');
legend([h11 h12], 'a-Estimate', 'b-Estimate');
title('LMS estimation')
grid on;
set(gca,'FontSize',16);
set(hxlabel,'FontSize', 18);
set(hylabel,'FontSize', 18);
set(gca,'FontSize',16);
epsName=sprintf('figAR2_NLMS.eps');
set(hf1, 'PaperPositionMode', 'auto');



% saveas(hf1,epsName,'epsc')
% % eA=eA./max(abs(eA)); sound(eA,Fs);
% %Look at the error

% figure
% plot(eA.^2)
%
```

Figure 6.2: Code page 2 of RLS.

```matlab
xTreal8 = ECG_8_II_missing;
x8 = ECG_8_AVR - mean(ECG_8_AVR);
y8 = ECG_8_V - mean(ECG_8_V);
xT8 = ECG_8_II - mean(ECG_8_II);
xTmean8 = mean(ECG_8_II);


% START OF ACTUAL CODE

% Change the number on the right side of the equation to swap between
% patience.
x = x2;
y = y2;
xT = xT2;
xTmean = xTmean2;
xTreal = xTreal2;


% Iteration lengths
Ndata = length(xT);
Nfull = length(x);
Nrec = Nfull-Ndata;

% number of unknowns, length of theta vector
N=25;
M=25;
p=M+N+1+1;    % Current value for each a and b plus the current value


% Initilizations
mylambda=0.9992;            % Tested and found to be a good value
mytheta=0.2*ones(p,1);
P=10000*eye(p);             % set as high.

% Vectors for storing values.
eA=zeros(Ndata,1);
thetaA(:,1)=mytheta;
thetaA2 = zeros(p,1);
xPredict= zeros(Nrec,1);


tic
for i=max(N,M)+1:Ndata
    % Creating a row vector for H
    h=[x(i:-1:i-N) ;y(i:-1:i-M)];

    % error (this is prediction error, not the ls error )
    e = xT(i) - h' * mytheta;
```

```matlab
tic
for i=max(N,M)+1:Ndata
    % Creating a row vector for H
    h=[x(i:-1:i-N) ;y(i:-1:i-M)];

    % error (this is prediction error, not the ls error )
    e = xT(i) - h' * mytheta;

    % update kalman gain
    K= P * h/(mylambda^i + h' * P * h);

    % update theta
    mytheta = mytheta + K * e;

    % update P
    P = (eye(p)- K * h') * P;

    % store the values to plot later
    thetaA(:,i+1)=mytheta;
    eA(i,1)=e;
end

for i=Ndata+1:Nfull
   h=[x(i:-1:i-N) ;y(i:-1:i-M)];
   xPredict(i-Ndata) = h' * mytheta;
end

xPredict = xPredict + xTmean;
toc

hf1=figure;
h12=plot(xTreal,'r');
legend('real value')
hold on

h11=plot(xPredict,'b');
legend('prediction value')
hold on

xlim([800,1150]);
hxlabel=xlabel('Time');
hylabel=ylabel('Value');

% %Look at the error
etot = sum(eA)./(Nrec);
% figure

% plot(eA.^2)
```

Figure 6.3: Code page 1 of LMS.

```matlab
hf1=figure;
h12=plot(xTreal,'r');
legend('real value')
hold on

h11=plot(xPredict,'b');
legend('prediction value')
hold on

xlim([800,1150]);
hxlabel=xlabel('Time');
hylabel=ylabel('Value');

% %Look at the error
etot = sum(eA)./(Nrec);
% figure

% plot(eA.^2)
eAT=zeros(Nrec,1);
for i=1:Nrec
    eAT(i,1) = eA(i) * eA(i);
end

% hf2=figure;
% h11=plot(eAT);
% hold on

eA2=zeros(Nrec,1);
for i=1:Nrec
    eA2(i,1) = (xTreal(i)-xPredict(i))^2;
end




Q1 = 1 - (mean(eA2) / var(xTreal));
Q2matrix = cov(xTreal,xPredict) / sqrt(  var(xTreal) * var(xPredict) );
Q2 = Q2matrix(2);
if Q2 < 0
    Q2 = 0;
end
if Q1 < 0
    Q1 = 0;
end

Q1
Q2% % Below code is written for p=2
```

```matlab
% hf2=figure;
% h11=plot(eAT);
% hold on

eA2=zeros(Nrec,1);
for i=1:Nrec
    eA2(i,1) = (xTreal(i)-xPredict(i))^2;
end




Q1 = 1 - (mean(eA2) / var(xTreal));
Q2matrix = cov(xTreal,xPredict) / sqrt(  var(xTreal) * var(xPredict) );
Q2 = Q2matrix(2);
if Q2 < 0
    Q2 = 0;
end
if Q1 < 0
    Q1 = 0;
end

Q1
Q2% % Below code is written for p=2
hf1=figure;
% Plot RLS filter coefficients, i.e. theta_1 and theta_2
h11=plot(thetaA(1,:)',':','Linewidth',2);
hold on
h12=plot(thetaA(2,:)',':','Linewidth',2);
hold on
% % Plot the parameters of the process
% h21=plot(ones(Nsim+1,1)* a1, '-k','Linewidth',2);
% h22=plot(ones(Nsim+1,1)* a2, '--k','Linewidth',2);
xlim([1,Ndata+1]);
hxlabel=xlabel('Iteration');
hylabel=ylabel('Filter Coefficients');
legend([h11 h12 ], 'a-Estimate', 'b-Estimate');
title('RLS estimation')
grid on;
set(gca,'FontSize',16);
set(hxlabel,'FontSize', 18);
set(hylabel,'FontSize', 18);
set(gca,'FontSize',16);
epsName=sprintf('figAR2_RLS.eps');
set(hf1, 'PaperPositionMode', 'auto');
```

Figure 6.4: Code page 2 of LMS.