Machine Learning for Signal Processing and Pattern Classification.

SIMRAN

1. The given data is of the form (t_i, y_i) for $1 \le i \le N$. The goal is to find the polynomial that approximates the data by minimizing the energy of the residual:

$$E = \sum_i (y_i - p(t_i))^2$$

where $p(t)$ is a polynomial, eg: $p(t) = a_0 + a_1 t + a_2 t^2$

The problem can be viewed as solving the over determined system of equations,

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} \approx \begin{bmatrix} 1 & t_1 & t_1^2 \\ 1 & t_2 & t_2^2 \\ \vdots & \vdots & \vdots \\ 1 & t_N & t_N^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix}$$, which we denote as $y \approx \mathbf{A}a$.

Least square solution $a = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T y$ **.**

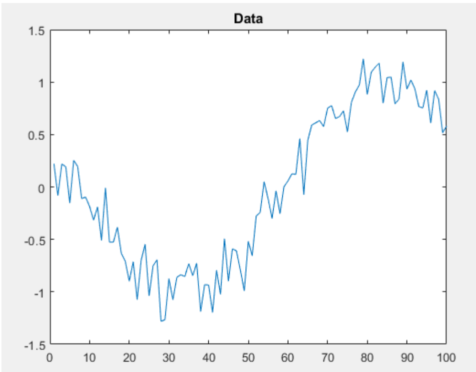
(i) Implement the same for the given data (data_1.txt).

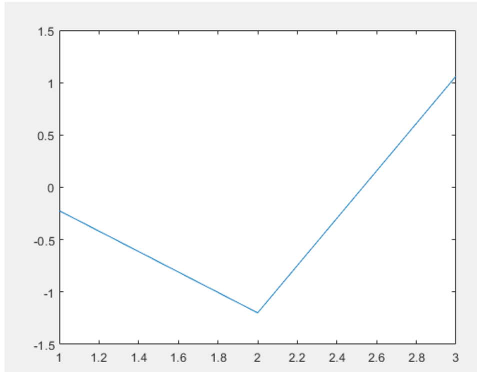
CODE:

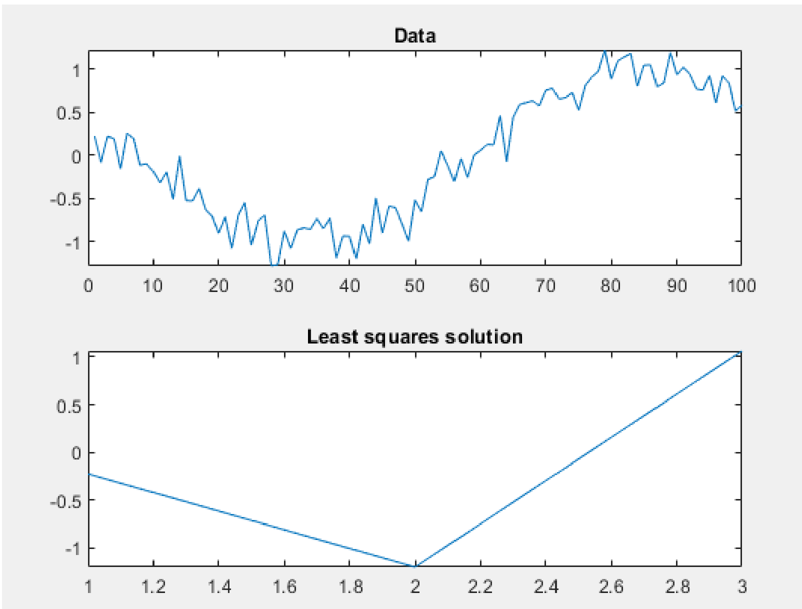
```
clc;
clear all;
close all;

load data_1.txt;  %loading the data
y = data_1(:,2);         % y values
t = data_1(:,1);         % t values

N = length(y); %taking the length of y

figure(1)
plot(y)      %ploting y data
title('Data')

e = ones(N, 1);
A = [e t t.^2];
%A matrix
F = A' * A;
x = F \ A' * y; %least square solution
```

```
figure(2)
plot(x) %ploting the obtained solution

figure;
subplot(2,1,1);plot(y);title('Data')
subplot(2,1,2);plot(x);title('Least squares solution')
```

OUTPUT:







2. One approach to predict future values of a time series is based on linear prediction, eg

$$y(n) \approx a_1 y(n-1) + a_2 y(n-2) + a_3 y(n-3)$$

If past data $y(n)$ is available, then the problem of finding a_i can be solved using least squares. Finding $a = (a_0, a_1, a_3)^T$ can be viewed as one of solving an over determined system of equations. For example, if $y(n)$ is available for $0 \le n \le N-1$, and we seek a third order linear predictor then the overdetermined system of equations are given by,

$$\begin{bmatrix} y(3) \\ y(4) \\ \vdots \\ y(N-1) \end{bmatrix} \approx \begin{bmatrix} y(2) & y(1) & y(0) \\ y(3) & y(2) & y(1) \\ \vdots & \vdots & \vdots \\ y(N-2) & y(N-3) & y(N-4) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

, which we can write as $\bar{y} = \mathbf{A}a$ where $\mathbf{A}$ is a matrix of size $(N-3) \times 3$. The least squares solution is given by $a = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T y$. Note that $\mathbf{A}^T \mathbf{A}$ is small of size 3×3 only. Hence, a is obtained by solving a small linear system of equations.

Implement the fourth order linear predictor for the given data.

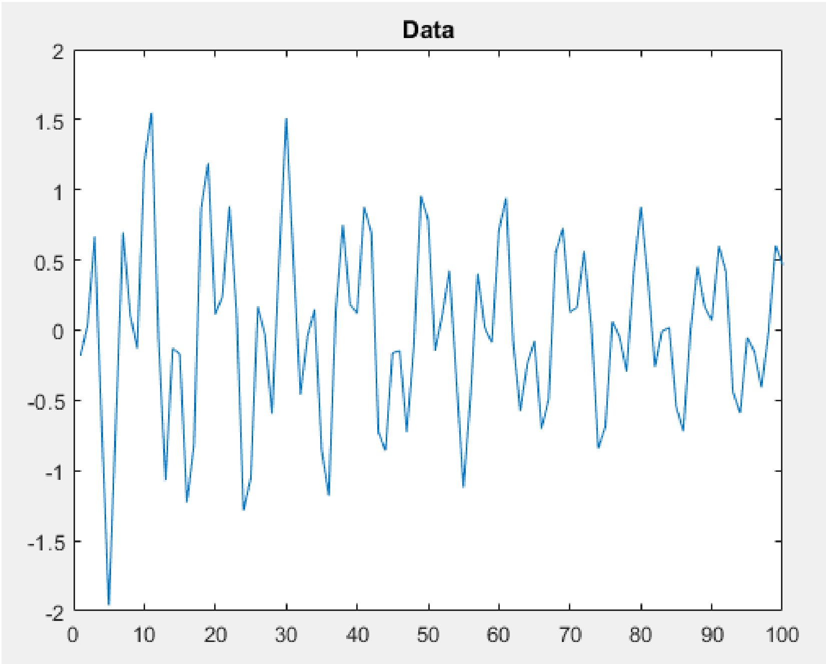
CODE:

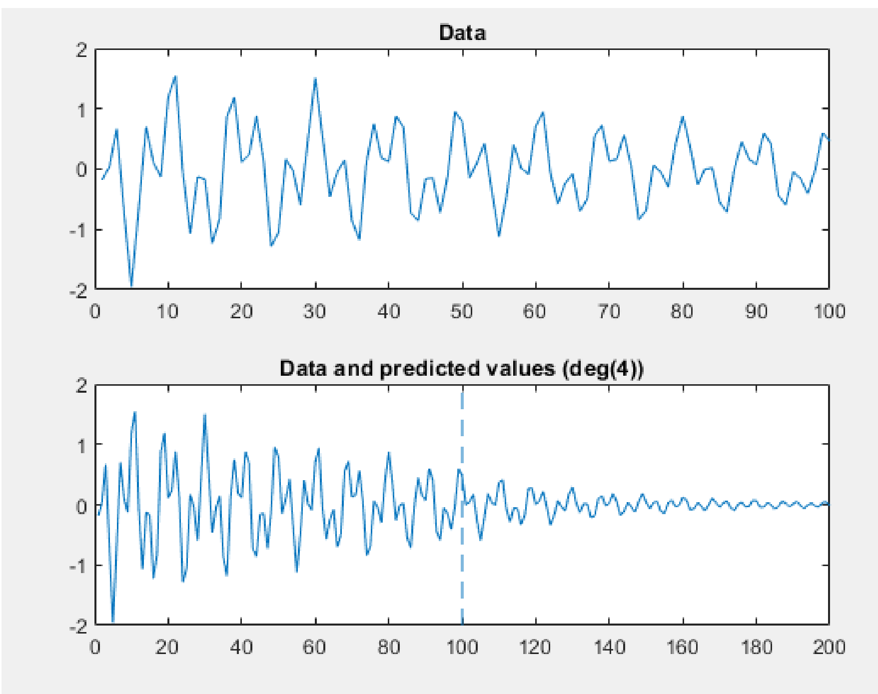
```
clc;
clear all;
close all
%% Load data

load data_2.txt;

y = data_2;           % data value

%% Display data

figure(1)
plot(y)
title('Data')

L = 100;

%% 4th order linear predictor

N = length(y);
H = [y(4:N-1) y(3:N-2) y(2:N-3) y(1:N-4)];
```

```
b = y(5:N);
a = (H' * H) \ (H' * b)          % a : coefficients of linear
predictor

g1 = [y; zeros(L, 1)];
for i = N+1:N+L
    g1(i) = a(1) * g1(i-1) + a(2) * g1(i-2) + a(3) * g1(i-3)
+a(4) *g1(i-4);
end



figure
subplot(2,1,1);plot(y);title('Data')
subplot(2,1,2);plot(g1);line([N N], [-2 2], 'linestyle', '--
');title('Data and predicted values (deg(4))')
```

OUTPUT:

```
a =
          1.42
         -1.67
          1.36
         -0.90
```


Data
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
0
10
20
30
40
50
60
70
80
90
100


Data
2
1
0
-1
-2
0
10
20
30
40
50
60
70
80
90
100
Data and predicted values (deg(4))
2
1
0
-1
-2
0
20
40
60
80
100
120
140
160
180
200

3. Derive the solution for signal smoothing using least square approach using the third order derivative

$$D = \begin{bmatrix} 1 & -3 & 3 & -1 & & \\ & 1 & -3 & 3 & -1 & \\ & & \cdot & \cdot & \cdot & \cdot \\ & & & \cdot & \cdot & \cdot \\ & & & & \cdot & \cdot \end{bmatrix}$$

CODE:

```
%% Least squares smoothing

clc;
clear all;
close all;

%% Load data

load data_3.txt;

y = data_3;          % data value
N = length(y);

%% Display data

figure(1)
clf
plot(y)
title('Data')

%% Smoothing (degree = 3)
% D is the third-order difference matrix.
% It approximates the third-order derivative.
% In order to exploit fast banded solvers in Matlab,
% we define D as a sparse matrix using 'spdiags'.

e = ones(N, 1);
D = spdiags([e -3*e 3*e -e], 0:3, N-3, N); %third order
derivative coefficient matrix


%%
% Observe the first and last corners of D.
% (D is too big to display, so we show
```

```
% the first and last corners only.)

%%
% First corner of D:

full(D(1:5, 1:5))

%%
% Last corner of D:

full(D(end-4:end, end-4:end))

%%
% Solve the least square problem

lam = 0.5;
F = lam*(speye(N)) +(D' * D);% F is a banded matrix
%
x = F \ (lam*y);                                    % Matlab uses a
fast solver for banded systems)

% F = (speye(N)) +lam*(D' * D);% F is a banded matrix
%
% x = F \ y;


figure;
subplot(2,1,1);plot(y);title('Data')
subplot(2,1,2);plot(x);title('Least squares smoothing')
```

OUTPUT:

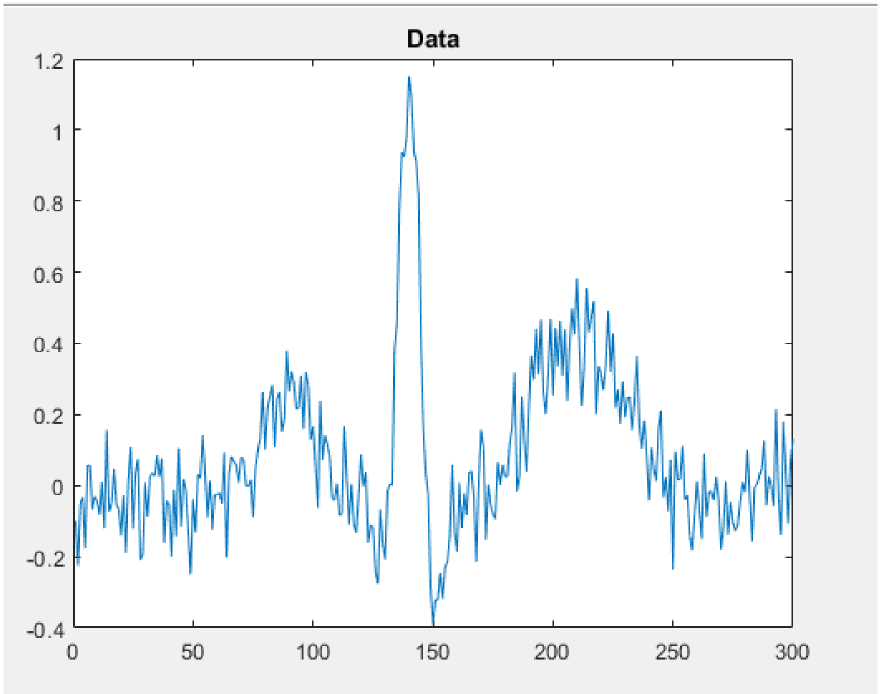
ans =

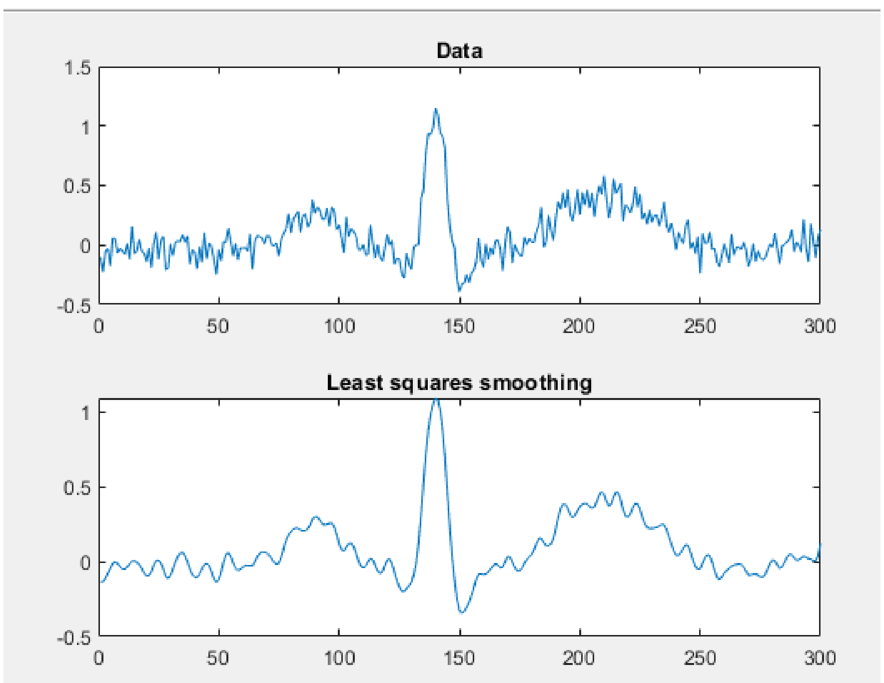
1	**-3**	**3**	**-1**	**0**
0	**1**	**-3**	**3**	**-1**
0	**0**	**1**	**-3**	**3**
0	**0**	**0**	**1**	**-3**
0	**0**	**0**	**0**	**1**

ans =

-1	**0**	**0**	**0**	**0**
3	**-1**	**0**	**0**	**0**
-3	**3**	**-1**	**0**	**0**

1	**-3**	**3**	**-1**	**0**
0	**1**	**-3**	**3**	**-1**


Data
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
0
50
100
150
200
250
300


Data
1.5
1
0.5
0
-0.5
0
50
100
150
200
250
300
Least squares smoothing
1
0.5
0
-0.5
0
50
100
150
200
250
300

4. Consider the data given below:

Point id	X_1	X_2	d
1	1	1	-1
2	0.5	0.5	-1
3	3	3	1
4	4	5	1

Implement SVM hard using cvx.

CODE:

```
clear all
clc
A=[1 0.5 3 4;1 0.5 3 5]';
d=[-1 -1 1 1]';
e=ones(size(A,1),1);

cvx_begin quiet
    variable w(2);
    variable gama;
    dual variable u;
    minimize (w'*w)
    subject to
        u:d.*(A*w-gama*e)>=e;
cvx_end
format bank
disp('w vector')
w
disp('gama')
gama
disp('u vector (lagange multipliers)')
u

plot(A(1:2,1), A(1:2,2),'*');% plot-1 points
hold on
plot(A(3:4,1), A(3:4,2),'o');% plot+1 points
hold on
x1=-1:4;
x2=-(u(1)/u(3))*x1+(gama/u(3));
plot(x1,x2); % draw the classifier line
```

```
hold on
x2=-(u(1)/u(3))*x1+((-1+gama)/u(3));
plot(x1,x2) % draw the lower bounding line

hold on
x2=-(u(1)/u(3))*x1+((1+gama)/u(3));
plot(x1,x2) % draw the upper bounding line
```

OUTPUT:

w vector

w =

0.50

0.50

gama

gama =

2.00

u vector (lagange multipliers)

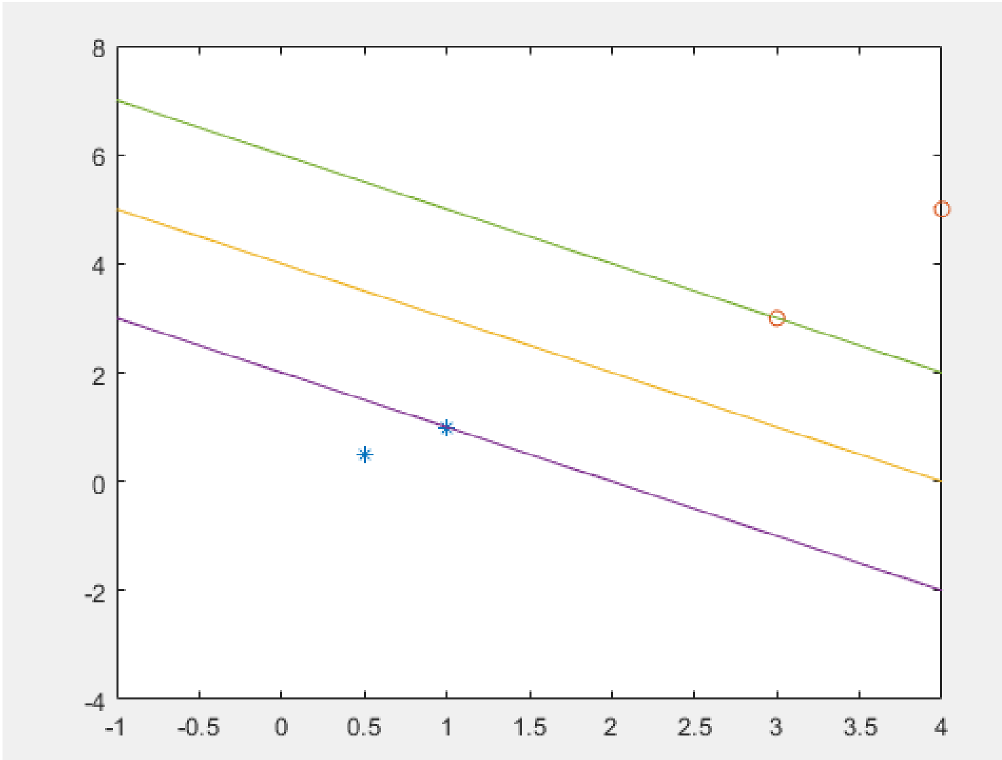
u =

0.50

0.00

0.50

0.00



Support vectors are: (X1,X2) = (1,1) and (3,3).

5. Use matlab/python programming to prove that "Convolution in time domain is equal to the multiplication in frequency domain".

CODE:

```
clear all
clc

x=[1 2 3 4] % 1st signal
y=[5 6 7] % 2nd signal
timedomain_conv=conv(x,y) %convolving the two signals in time
domain.

m=length(x); % m:length of the 1st signal
n=length(y); % n:length of the 2nd signal
l=m+n-1; % l:length of the convolved sequence

w=fft(x,l); % fft of the 1st signal
z=fft(y,l); % fft of the 2nd signal
```

```
a=w.*z; % multipying the coefficients of the signals(frequency
domain).
frequencydomain_mul=ifft(a) % converting it back to time domain
using ifft.
```

OUTPUT:

x =

1.00 2.00 3.00 4.00

y =

5.00 6.00 7.00

timedomain_conv =

5.00 16.00 34.00 52.00 45.00 28.00

frequencydomain_mul =

5.00 16.00 34.00 52.00 45.00 28.00

6. Use the matlab/python programming to compute the DFT matrix for length 'N', which must be the power of 2. (Hint: W_N^{nk}', where $W_N = e^{-j*2*pi/N}$, n = 0 to N-1 and k=0 to N-1). Avoid using loop

CODE:

```
clear all
clc
x=[2 4 8 16 32];
if(log2(x))
    N=length(x); % N is the length of signal
    n=0:N-1;   % n is the number of columns
    theta=2*pi*n/N;  % theta value are taken
    k=(0:N-1)';  % k is the number of rows
    W=exp(-i*k*theta)*x'  % coefficients of DFT
    w=dftmtx(N)*x' % built-in command for dft coefficients
    X=fft(x)'  % built-in command for fft
else
    disp('Not a valid signal')
end
```

OUTPUT:

W =

62.00

-6.29

-19.71

-19.71

-6.29

w =

62.00

-6.29

-19.71

-19.71

-6.29

X =

62.00

-6.29

-19.71

-19.71

-6.29

Taking x=[1 2 3 4 5 6 7 8]

OUTPUT:

x =

Columns 1 through 6

1.00 2.00 3.00 4.00 5.00 6.00

Columns 7 through 8

7.00 8.00

Not a valid signal

7. Use the below given formulation to classify the given data

$$\min_{\xi,w,\gamma} \sum_i \xi_i$$

$$d_i(w^T x_i - \gamma) \geq 1 - \xi_i; \forall i$$

$$\xi_i \geq 0; \forall i$$

The classifier is: $sign(w^T x - \gamma)$

CODE:

```
clear all
clc

load('cardio.mat')
d=y;
a=size(X)
e=ones(size(X,1),1);
cvx_begin quiet
    variable w(21);
    variable gama;
    variable psii(1831);
    minimize sum(psii)
    subject to
        d.*(X*w-gama*e)>=1-psii;
        psii>=0;
cvx_end

labels=sign(X*w-gama); % predict the class label using the given
calssifier
```

8. Apply SVM hard margin formulation to classify the given data ('cardio.mat')

CODE:

```
clear all
clc

load('cardio.mat')
d=y;
d(d==0) = -1; % converting the labels of 0 to -1

e=ones(size(X,1),1);
A = (d*d').*(X*X');

cvx_begin quiet
    variables u(1831)
    minimize (((1/2)*u'*A*u) - (e'*u));
    subject to
        u'*y == 0;
        u>=0;
cvx_end

w=sum(u*d'*X);% calculating w

gamma=(w*X(99,:)') + 1; % calculating gamma

predict_labels=sign(w*X'-gamma); % predicting the labels
```

OUTPUT:

w =

Columns 1 through 7

-9.91	-288.39	118.73	-107.81	183.90	176.96	640.33

Columns 8 through 14

420.16	53.40	352.52	-277.11	41.67	-80.54	-37.73

Columns 15 through 21

30.65	5.19	-452.36	-449.45	-414.66	353.38	-216.11

gamma =

-2911.55

predict_labels will predicts the labels of the classes using sign classifier.(1831)

9. Develop a simple 1D-CNN architecture to classify the given data (‘cardio.mat’)

Code and output in Q9.ipython file.