Machine Learning for Signal Processing and Pattern Classification.

MISSING SMAPLE ESTIMATION – LEAST SQUARE APPROACH

SIMRAN

1.

AIM:

To obtain the least square solution for the below given problem formulation for missing samples estimation.

$$\min_{v} \left\| D(S^t y + S_c^t v \right\|_2^2$$

SOLUTION:

The least square solution for estimating missing samples is:

$$v = -\left(S_c D^t D S_c^t\right)^{-1} S_c D^t D S^t y$$

Where

1. S is a matrix having 1's where data is present
2. Sc is the complement of S
3. D is the second order derivative
4. y is the original signal

CODE:

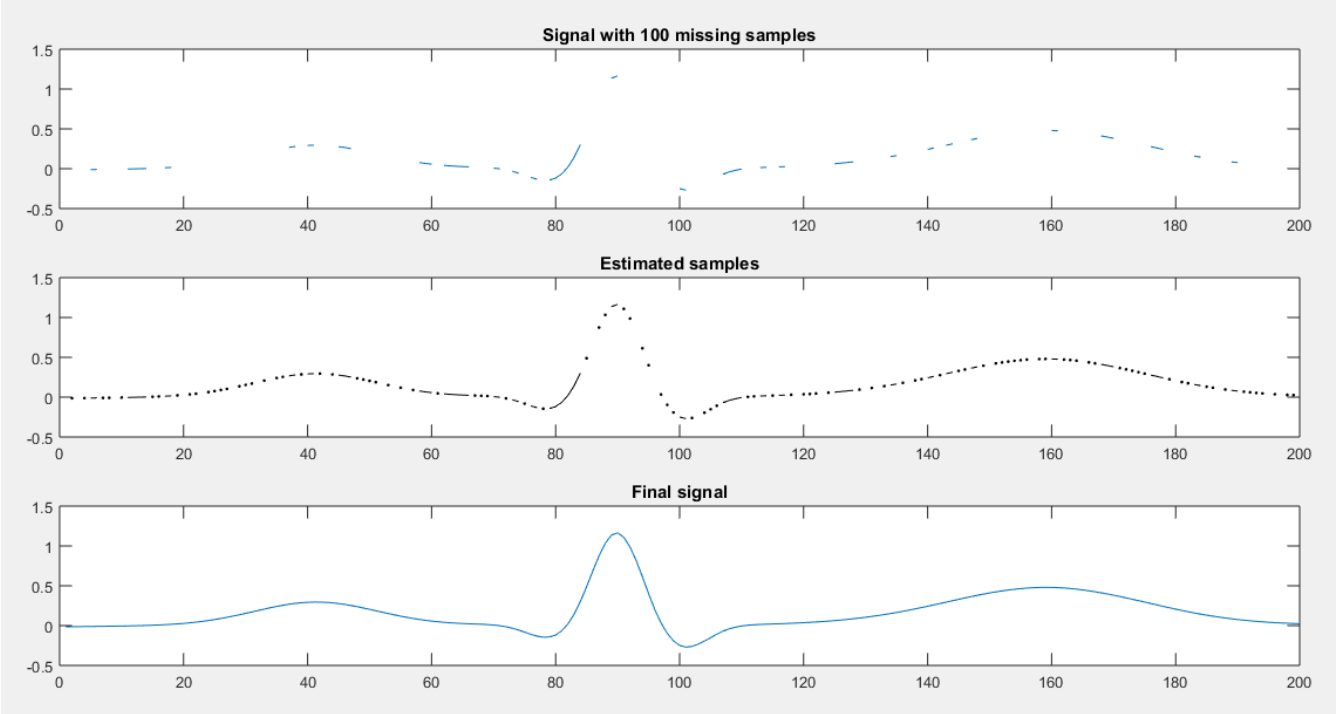
```
%% Estimation of missing data

clc;
clear all;
close all

%% Load data

load data.txt;           % load the data from data.txt
y = data;                % y : data value
N = length(y);           % N : the length of the data in y
n = 1:N;                 % n : the indexes from 1 to N

%% Define matrix D
% D represents the second-order derivitive
% (2nd-order difference).
% D is defined as a sparse matrix so that Matlab
```

```
% subsequently uses fast solvers for banded systems.

e = ones(N, 1);
D = spdiags([e -2*e e], 0:2, N-2, N);

%% Define matrices S and Sc

k = isfinite(y);  % k  : logical vector.1 where elements are
finite valve.0 where elements are infinite(NaN)
S = speye(N);     % S  : sparse matrix with size N.
S(~k, :) = [];    % S  : sampling matrix in which samples are
missing removing that row
Sc = speye(N);    % Sc : sparse matrix with size N.
Sc(k, :) = [];    % the rows at which values are present are
omitted
number_of_missing_values = sum(~k) %count the number of missing
data

%% Estimate missing data
% Least square estimation of missing data.
% Note that the system matrix is banded so the system
% equations can be solved very efficiently with a fast banded
system solver.
% By defining S and D as sparse matrices, Matlab calls a fast
% banded system solver by default.

v = -(Sc * (D' * D) * Sc') \ ( Sc * (D') * D * S' * y(k));
% v : estimated samples
% y : input matrix
% D : second order derivative matrix
% S : a matrix with values 1 at present data
% Sc: complement of S which contains 1 at places of missing data

%% Fill in unknown values
% Place the estimated samples into the signal.

x = zeros(N,1); % x variable with zeros
                % length of x=length of input matrix(N)
x(k) = y(k);    % original signal is preserved
x(~k) = v;      % Estimated values are insert at places of
missing data
figure(1);
subplot(3,1,1);plot(n, y);title('Signal with 100 missing
samples');
subplot(3,1,2);plot(n, y, 'k', n(~k), x(~k)
,'k.');title('Estimated samples');
subplot(3,1,3);plot(n, x );title('Final signal');
```

OUPUT:

number_of_missing_values =

100



2.

AIM:

What are the role of two matrices S and S_c? How will you derive those matrices from the given data, in which few samples are missing?

CODE:

```
clc;
clear all;
close all
y =[0.7342,0.1234,NaN,0.2345,NaN,0.1324,NaN,0.2326,NaN,0.7345]';
% y : data value
N = length(y);          % N : the length of the data in y
```

```
n = 1:N;                  % n : the indexes from 1 to N
k = isfinite(y);  % k  : logical vector.1 where elements are
finite valve.0 where elements are infinite(NaN)
S = speye(N);      % S  : sparse matrix with size N.
S(~k, :) = []    % S  : sampling matrix in which samples are
missing removing that row
Sc = speye(N);     % Sc : sparse matrix with size N.
Sc(k, :) = []    % the rows at which values are present are
omitted
```

S is the sampling matrix in where the value is 1 when data is present.

Sc is matrix which is the complement of the sampling matrix. The value is 1 in the missing value position.

Function isfinite is used to calculate if the data is finite or not. If it is NaN then isfinite put 0 in this position and if t is finite then puts 1 in this position.

The position at which 0 is there that is ignored and remaining are put in S matrix.

S matrix dimension is number of known samples X length of the signal y

Sc matrix dimension is missing values X length of the signal y

OUPUT:

S =

(1,1) 1

(2,2) 1

(3,4) 1

(4,6) 1

(5,8) 1

(6,10) 1

Sc =

(1,3) 1

(2,5) 1

(3,7) 1

(4,9) 1

3.

AIM:

To justify the above given problem formulation suits for the signal with 10 samples in which the samples at odd locations following the even locations are missing.

CODE:

```
clc;
clear all;
close all

y =[0.7342,0.1234,NaN,0.2345,NaN,0.1324,NaN,-0.2326,NaN,0.7345]'
% y : data value

N = length(y);  % the length of the data in y(10)
n = 1:N;        % the indexes from 1 to N


%% Define matrix D
% D represents the second-order derivitive
% (2nd-order difference).
% D is defined as a sparse matrix so that Matlab
% subsequently uses fast solvers for banded systems.

e = ones(N, 1);
D = spdiags([e -2*e e], 0:2, N-2, N);

%% Define matrices S and Sc

k = isfinite(y);  % k  : logical vector.1 where elements are
finite valve.0 where elements are infinite(NaN)
S = speye(N);     % S  : sparse matrix with size N.
S(~k, :) = [];    % S  : sampling matrix in which samples are
missing removing that row
Sc = speye(N);    % Sc : sparse matrix with size N.
Sc(k, :) = [];    % the rows at which values are present are
omitted
number_of_missing_values = sum(~k)% count the number of missing
data

%% Estimate missing data
% Least square estimation of missing data.
```

```
% Note that the system matrix is banded so the system
% equations can be solved very efficiently with a fast banded
system solver.
% By defining S and D as sparse matrices, Matlab calls a fast
% banded system solver by default.

v = -(Sc * (D' * D) * Sc') \ ( Sc * (D') * D * S' * y(k));
% v : estimated samples
% y : input matrix
% D : second order derivative matrix
% S : a matrix with values 1 at present data
% Sc: complement of S which contains 1 at places of missing data

%% Fill in unknown values
% Place the estimated samples into the signal.

x = zeros(N,1); % x variable with zeros
                % length of x=length of input matrix(N)
x(k) = y(k);    % original signal is preserved
x(~k) = v;      % Estimated values are insert at places of
missing data
figure(1);
subplot(3,1,1);plot(n, y,'k.');title('Signal with 4 missing
samples');
subplot(3,1,2);plot(n, y, 'k.', n(~k), x(~k)
,'k*');title('Estimated samples');
subplot(3,1,3);plot(n, x );title('Final signal');
```

The odd location after the even place values are given as NaN (missing values). Approximate prediction of the missing values is done using the formula and a continuous single is the output.

OUPUT:

y =

0.7342

0.1234

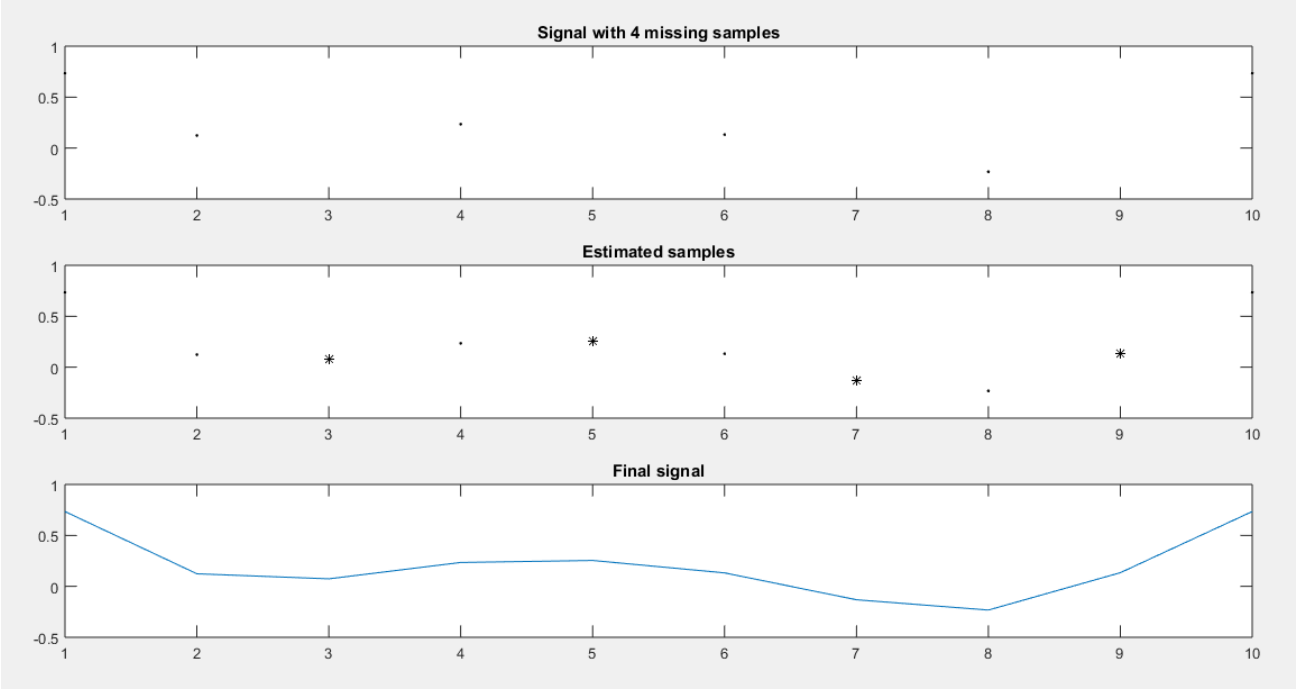
NaN

0.2345

NaN

0.1324

```
    NaN
-0.2326
    NaN
 0.7345
number_of_missing_values =
     4
```



4.

AIM:

To explore ‘speech declipping’ and to apply the missing sample estimation for the given speech signal (speech declipping) which contains missing samples (clipped speech).

CODE:

```
%% Estimation of missing data

clc;
```

```
clear all;
close all

%% Load data

load data1.txt;          % load the data from data1.txt
y = data1;               % y : data value
N = length(y);           % N : the length of the data in y
n = 1:N;                 % n : the indexes from 1 to N

%% Define matrix D
% D represents the second-order derivitive
% (2nd-order difference).
% D is defined as a sparse matrix so that Matlab
% subsequently uses fast solvers for banded systems.

e = ones(N, 1);
D = spdiags([e -2*e e], 0:2, N-2, N);

%% Define matrices S and Sc

k = isfinite(y);  % k  : logical vector.1 where elements are
finite valve.0 where elements are infinite(NaN)
S = speye(N);     % S  : sparse matrix with size N.
S(~k, :) = [];    % S  : sampling matrix in which samples are
missing removing that row
Sc = speye(N);    % Sc : sparse matrix with size N.
Sc(k, :) = [];    % the rows at which values are present are
omitted
number_of_missing_values = sum(~k)% count the number of missing
data

%% Estimate missing data
% Least square estimation of missing data.
% Note that the system matrix is banded so the system
% equations can be solved very efficiently with a fast banded
system solver.
% By defining S and D as sparse matrices, Matlab calls a fast
% banded system solver by default.

v = -(Sc * (D' * D) * Sc') \ ( Sc * (D') * D * S' * y(k));
% v : estimated samples
% y : input matrix
% D : second order derivative matrix
% S : a matrix with values 1 at present data
% Sc: complement of S which contains 1 at places of missing data
```

```
%% Fill in unknown values
% Place the estimated samples into the signal.

x = zeros(N,1); % x variable with zeros
                % length of x=length of input matrix(N)
x(k) = y(k);    % original signal is preserved
x(~k) = v;      % Estimated values are insert at places of
missing data

figure(1);
subplot(3,1,1);plot(n, y);title('Signal with 139 missing
samples');
subplot(3,1,2);plot(n, y, 'k', n(~k), x(~k)
,'k.');title('Estimated samples');
subplot(3,1,3);plot(n, x );title('Final signal');
```

Speech clipping happens if the signal is not properly normalized while audio processing or when the signal is amplified beyond the range of the capability of the input voltage can produce.

The signal beyond the capability of the amplifier is simply clipped. Reducing the signal level will avoid the signal clipping. The information at the peaks is lost when signal is chipped.

Estimating the values at the signal clipped regions to estimate the original signal is called as Speech signal declipping.

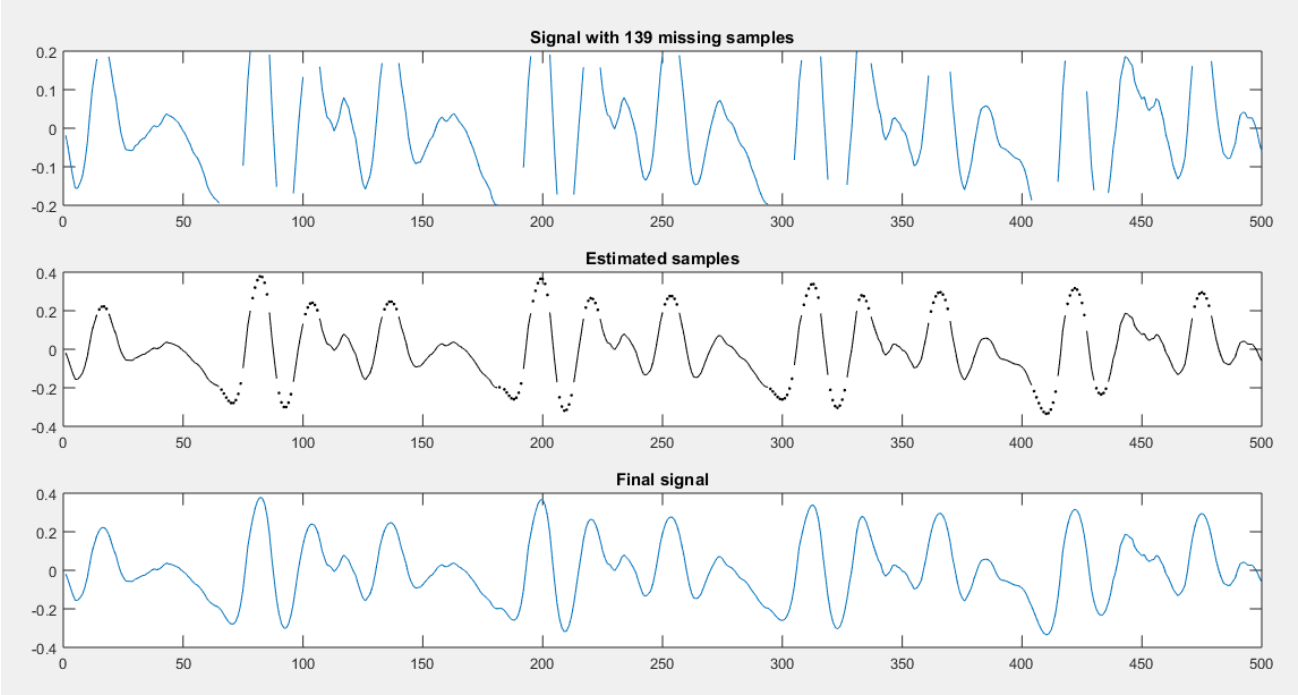
An ideal declipping algorithm will recover the original unclipped signal.[1]

Speech declipping data given to the algorithm which calculated the approximate values of the missing values.

OUPUT:

number_of_missing_values =

139



REFERENCE:

[1] Harvilla, Mark J. / Stern, Richard M. (2014): "Least squares signal declipping for robust speech recognition", In *INTERSPEECH-2014*, 2073-2077.

[2] Selesnick I. Least squares with examples in signal processing. Connexions. 2013 Apr 17.