EE 301: Digital Signal Processing

Term Project Report

RECEIVED SIGNAL STRENGTH-BASED SENSOR LOCALIZATION IN SPATIALLY CORRELATED SHADOWING

Group Members:

Anurag Prabhakar	11140
Ashish Lavania	11161
Atri Bhattacharya	11171
Chirag Jain	11226
Isha Agrawal	11320

Instructor: Dr. Rajesh M. Hegde

Mentor: Sudhir Kumar

Acknowledgements

We would like to thank Prof. R. Hegde for giving us the opportunity to further explore the world of Digital Signal Processing by working on this Term Paper and hence, getting an experience of a real life application of our theoretical knowledge. We would also like to acknowledge the invaluable assistance of Prof. K. Rajawat, who took time out of his busy schedule to help us overcome seemingly unsurmountable barriers that stopped us all too often.

Finally, we would like to thank our guide Sudhir Kumar for his assistance and guidance through every stage of the implementation process.

Introduction

Source localization is extensively used the field of Wireless Sensor Networks (WSN) and Cellular Systems. The data obtained through sensor measurements cannot be interpreted meaningfully with the information of source location. Installation of GPS to the source would be quite expensive and hence impractical. Sensor localization could be done through various sensor measurements like Time of Arrival (TOA), Time Difference of Arrival (TDOA), Received Signal Strength (RSS), etc. In this paper, RSS has been used as it is practically simple and quite inexpensive.

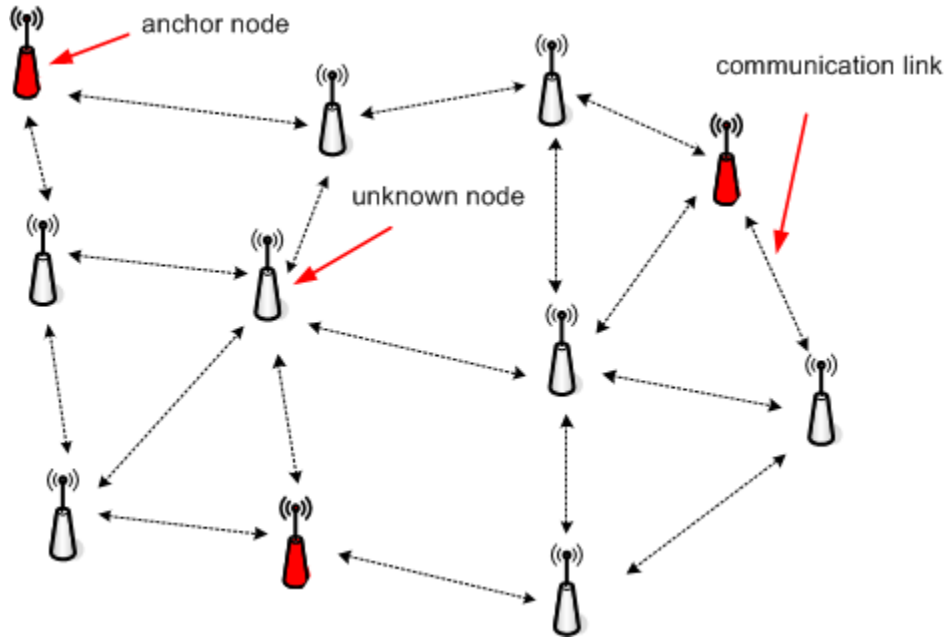
ML estimator is one of the most interesting optimizing methods used previously for sensor localization. It achieves CRLB (Cramer-Rao Lower Bound) for sufficiently high SNR. It can easily be seen that the cost function is highly non-linear and convex. Also, ML estimator has no closed form solution and is solved using iterative algorithms. The problem with such approach is that if the initial guess is not taken correctly, function could converge to a local minima or a saddle point which could lead to high estimation error. Sub-optimal estimators are a solution to this problem. Convex relaxation also helps in obtaining a solution.

Another method called the Linear Least Squares (LLS) has also been used. It produces a closed form solution but is not as accurate as the ML estimator; erroneous especially in case of limited anchor nodes.

SDP and SOCP are also used for sensor localization. In this paper, as pairwise distances are not available, localization is done only on the basis of RSS (Received Signal Strength). The RSS measurements have been assumed to be correlated. This is actually the case when we deal with indoor environments. It has been observed that sensors could be localized more accurately if signals are considered correlated; i.e. better performance is observed for the case of correlated shadowing. Semi definite programming (SDP) approach has been employed by converting the corresponding non-convex ML estimator into a convex one.

Problem Statement

There is a network of sensors, some of whose positions are known (called anchor nodes). We wish to estimate the position of the remaining nodes based upon the strength of received signal at the anchor nodes.



An example of a network with anchor nodes highlighted

Algorithm

- Received power in dB-m at i^{th} anchor node:

$$P_i = P_0 - 10\beta \log_{10} \frac{d_i}{d_0} + n_i, \qquad i = 1, 2, \ldots, M, \qquad (1)$$

- Covariance matrix of shadowing terms assuming spatially correlated shadowing:

$$[\mathbf{Q}]_{ij} = \begin{cases} \sigma_{dB}^2, & \text{if } i = j, \\ \rho_{ij}\sigma_{dB}^2, & \text{if } i \neq j, \end{cases} \qquad (2)$$

σ dB is the standard deviation of the shadowing. ρij is the correlation coefficient between the ith and the jth links.

- Initial non-convex minimization problem:

$$\hat{\mathbf{x}}_{\mathrm{ML}} = \arg\min_{\mathbf{x}\in\mathbb{R}^2} \big(\mathbf{p} - \mathbf{g}(\mathbf{x})\big)^T \mathbf{Q}^{-1} \big(\mathbf{p} - \mathbf{g}(\mathbf{x})\big), \qquad (3)$$

where,

$$\mathbf{p} = [P_1, P_2, \ldots, P_M]^T$$

and

$$[\mathbf{g}(\mathbf{x})]_i = P_0 - 10\beta \log_{10} d_i. \qquad (4)$$

- Rearranging (1) gives

$$\log_{10} d_i + \frac{P_i - P_0}{10\beta} = \frac{n_i}{10\beta}. \qquad (5)$$

- (5) after taking power of 10 gives

$$\lambda_i d_i = 10^{n_i/10\beta}, \qquad (6)$$

where

$$\lambda_i = 10^{(P_i - P_0)/10\beta}$$

- Using the first order Taylor series approximation gives

$$\lambda_i d_i = 1 + \frac{\ln 10}{10\beta} n_i. \qquad (7)$$

- Rearranging (7) gives

$$\lambda_i d_i - 1 = \epsilon_i, \qquad (8)$$

where,

$$\epsilon_i = (\ln 10/10\beta) n_i$$

- (8) rearranged gives

$$\mathbf{Ld} - \mathbf{1}_M = \boldsymbol{\epsilon}, \tag{9}$$

where,

$$\begin{aligned} \mathbf{L} &= \operatorname{diag}\{\lambda_1, \lambda_2, \ldots, \lambda_M\} \\ \mathbf{d} &= [d_1, d_2, \ldots, d_M]^T \\ \boldsymbol{\epsilon} &= [\epsilon_1, \epsilon_2, \ldots, \epsilon_M]^T \end{aligned}$$

- The minimization now boils down to

$$\hat{\mathbf{x}} = \underset{\mathbf{x}\in\mathbb{R}^2}{\arg\min}\ \boldsymbol{\epsilon}^T \mathbf{W} \boldsymbol{\epsilon}, \tag{10}$$

where

$$\mathbf{W} = \left(\mathrm{E}[\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T]\right)^{-1} = (10\beta)^2/(\ln 10)^2 \mathbf{Q}^{-1}. \tag{11}$$

- The nonlinear and nonconvex cost function of (10) can be written as

$$\begin{aligned} \boldsymbol{\epsilon}^T \mathbf{W} \boldsymbol{\epsilon} &= \operatorname{Trace}\left\{\mathbf{W}\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T\right\}, \\ &= \operatorname{Trace}\left\{\mathbf{W}(\mathbf{Ld} - \mathbf{1}_M)(\mathbf{Ld} - \mathbf{1}_M)^T\right\}, \\ &= \operatorname{Trace}\left\{\mathbf{W}(\mathbf{LDL}^T - 2\mathbf{Ld}\mathbf{1}_M^T + \mathbf{1}_M\mathbf{1}_M^T)\right\}, \end{aligned} \tag{12}$$

where,

$$\mathbf{D} = \mathbf{d}\mathbf{d}^T$$

- The diagonal elements of the matrix D are

$$[\mathbf{D}]_{ii} = d_i^2 = \begin{bmatrix} \mathbf{y}_i \\ -1 \end{bmatrix}^T \begin{bmatrix} \mathbf{I}_2 & \mathbf{x} \\ \mathbf{x}^T & z \end{bmatrix} \begin{bmatrix} \mathbf{y}_i \\ -1 \end{bmatrix}, \tag{13}$$

where,

$$z = \mathbf{x}^T\mathbf{x}.$$

- We need to relax the non-affine terms as follows

$$\begin{aligned} z = \mathbf{x}^T\mathbf{x} \ &\Rightarrow \ \begin{bmatrix} \mathbf{I}_2 & \mathbf{x} \\ \mathbf{x}^T & z \end{bmatrix} \succeq \mathbf{0}_3, \\ \mathbf{D} = \mathbf{d}\mathbf{d}^T \ &\Rightarrow \ \begin{bmatrix} \mathbf{D} & \mathbf{d} \\ \mathbf{d}^T & 1 \end{bmatrix} \succeq \mathbf{0}_{M+1} \end{aligned}$$

- This leads to the final minimization problem which is an SDP minimization problem by relaxing the non-affine terms in (10)

$$\begin{aligned}
&\underset{\mathbf{x},z,\mathbf{d},\mathbf{D}}{\text{minimize}} \quad \text{Trace}\left\{\mathbf{W}(\mathbf{L}\mathbf{D}\mathbf{L}^T - 2\mathbf{L}\mathbf{d}\mathbf{1}_M^T)\right\} \\
&\text{subject to } [\mathbf{D}]_{ii} = \begin{bmatrix}\mathbf{y}_i \\ -1\end{bmatrix}^T \begin{bmatrix}\mathbf{I}_2 & \mathbf{x} \\ \mathbf{x}^T & z\end{bmatrix} \begin{bmatrix}\mathbf{y}_i \\ -1\end{bmatrix}, \\
&\qquad\qquad \begin{bmatrix}\mathbf{D} & \mathbf{d} \\ \mathbf{d}^T & 1\end{bmatrix} \succeq \mathbf{0}_{M+1}, \quad \begin{bmatrix}\mathbf{I}_2 & \mathbf{x} \\ \mathbf{x}^T & z\end{bmatrix} \succeq \mathbf{0}_3. \qquad (14)
\end{aligned}$$

MATLAB Codes

Main_const_shadowing

```
%main file with a fixed value of shadowing, sigma is a scalar
sigma=4; N=120;
%takes less than a minute to run
error=sdp(sigma,N) ;
%sdp estimator
[a,b]=ecdf(error);
%getting the CDF of the localization error
plot(b,'-ro')
xlabel('localization error')
ylabel('CDF')
hold
%ML estimator file, kept in same directory
error_ml=ml2(sigma,N)
[a,b]=ecdf(error_ml);
plot(b,'-.b');
```

Main_variable_shadowing

```
%main file for varying shadowing, i.e. sigma as vector
%takes several minutes to run
sigma=1:5:41;
N=40;
rmse=zeros(1,length(sigma));
%rmse initialized to (1 x length(sigma)) to store root mean square values
for k=1:length(sigma)
error=sdp(sigma(k),N)
%solve sdp for variable shadowing
rmse(k)=norm(error);
%filling rmse values corresponding to values of shadowing
end
plot(sigma,rmse,'-ro')
xlabel('shadowing(DB)')
ylabel('RMSE')
hold
for k=1:length(sigma)
error=ml2(sigma(k),N)
%solve for ml2 to get error vector and resultant rmse values
rmse(k)=norm(error);
end
plot(sigma,rmse,'-.b')
%plot on same graph
```

SDP

```
%%The following code implements the paper titled "RECEIVED SIGNAL STRENGTH-
BASED SENSOR LOCALIZATION IN SPATIALLY CORRELATED SHADOWING" by vaghefi,
buehrer. Most studies for RSS (received signal strength) for sensor
localization assume that the shadowing components are uncorrelated. However,
here we assume that the shadowing is spatially correlated. Under this
condition, it can be shown that the localization accuracy can be improved if
the correlation among links is taken into consideration. The localization
problem is formulated as an SDP minimization and solved using the cvx for
 MatLab. Just run to see results.
function [error]=sdp(sigma,N)
%%%%(initializations)
m=4;                                                    %no. of anchors
%N=10;                                                  % N = number of test
cases
error=zeros(1,N);                                       % localization error
vector stored the estimation error for each of the N cases
ml=zeros(1,N);
X=rand(N,2)-.50;                                        % X is a set of N
random points in the 2D plane
%%%% for-loop to estimate location for each of the test cases
for z = 1:1:N
    echo off                                            %allowing the
commands to be viewed as they execute, for debugging
    %cvx_pause(false);
   % clc
     clearvars -except z error X  sigma  N
      y=.5*[1,-1, 1, -1 0 .25 .1 .2;                                %4
anchors
       1, 1, -1, -1 0 .5 -.1 -.2];
    x1=X(z,:);                                          %randomly generated
sensor point, to be estimated later
                                                        %taking the z-th
point for estimation of location
    p0=-40;                                             % p0 is the power at
unit length (d0=1)
    beta=4;                                             % beta = path loss
exponent, taken here to be 4dB
    d0=1;
    %sigma=4;
    rho=.8;
        %%%%(initial calculations for L,lambda,d,P all calculated assuming
known sensor x1)
    % Shadowing modeled as not-independant identically distributed Gaussian
random
    % variables, with the covariance matrix q. For iid (independant
    % identically distributed Gaussian random variables, q=sigma^2 * eye(4)

    p=p0*ones(1,8);
    d1=([x1' x1' x1' x1' x1' x1' x1' x1'] - y)'*([x1' x1' x1' x1' x1' x1' x1'
x1'] - y); d1=sqrt((diag(d1))');  % d1 =Vector of distance of sensor in
question from each of the anchor nodes
```

```
    p=p-10*beta*log10(d1/d0);
% p = Vector of Recieved Signal Strenghts at each of the anchor nodes
    lambda=10.^((p-p0*ones(1,8))/10*beta);
% lambda is a user-defined variable, defined as 10^((P-P0)/10*beta).
    L=diag(lambda);

    q=ones(8);
%Filling the q matrix
    for i=1:8
        for j=1:8
           if(i==j) q(i,j)=sigma^2;
           else q(i,j)=rho*sigma^2;
           end
        end
    end

    W=((10*beta)^2 / (log(10))^2 )*inv(q);                          % The W is a
weighting matrix which is proportional to the inverse of the covariance
matrix

    %%
    %%%% calculation of d vector, D matrix and final problem formulation in
cvx(http://cvxr.com/cvx/doc/CVX.pdf) assuming x as a variable

    %cvx_begin quiet sdp
    cvx begin sdp
    variables x(1,2);
    variable pp(1,1);
    variable D(8,8);
    variable d(1,8);
    p=W*L*D*L';
%p and q are cost functions to be minimized
    q=2*d'*ones(1,8)*W*L;
    minimize (trace(p) - trace(q))
%W- m x m; L- m x m; D - m x m; d - m x 1, m=no. of anchors
  % size(D)
   %  size(d)
    subject to
%constraints
    [eye(2) x'; x pp] == semidefinite(3) ;

      for j=1:8
%(eqn #13 in paper
       D(j,j)>=[y(1,j),y(2,j),-1]*[eye(2) x';  x  pp]*[y(1,j),y(2,j),-1]'; %D
matrix modeled to help making minimization problem linear and convex
      end

    [D d'; d 1] == semidefinite(9) ;
    cvx end
    error(z)=norm(x-x1);
%storing the estimation error due to the z-th point
end
%cvx_quiet(s_quiet);  %to avoid getting cvx internal messages on screen
%cvx_pause(s_pause);
```

ML

```
%%(initializations)
% N = number of test cases
% error vector stored the estimation error for each of the N cases
% X is a set of N random points in the 2D plane
function [error]=ml2(sigma,N)
error=zeros(1,N);
ml=zeros(1,N);
X=rand(N,2)-.50;

% for-loop to estimate location for each of the test cases
for z = 1:1:N
    echo off
    clc
    clearvars -except z error X sigma

    y=.5*[1,-1, 1, -1 0 .25 .1 .2; ;
       1, 1, -1, -1 0 .5 -.1 -.2];
    x1=X(z,:);

    %taking the z-th point for estimation of location
    p0=-40; beta=4;d0=1;
    %sigma=4;
    rho=.5;

    %%
    %(initial calculations for L,lambda,d,P all calculated assuming known x1)
    % p0 is the power at unit length, taken to be 1m here
    % p = Vector of Recieved Signal Strenghts at each of the anchor nodes
    % d = Vector of distance of sensor in question from each of the anchor
nodes
    % beta = path loss exponent, taken here to be 4dB
    % lambda is variable defined as 10^((P-P0)/10*beta).
    % q models the shadowing terms(shadowing is taken to be iid Gaussian
    % random variables). It is the covariance matrix of the same.
    %
    p=p0*ones(1,8);
    d1=([x1' x1' x1' x1' x1' x1' x1' x1'] - y)'*([x1' x1' x1' x1' x1' x1' x1'
x1'] - y); d1=sqrt((diag(d1))');
    p=p-10*beta*log10(d1/d0);
    lambda=10.^((p-p0*ones(1,8))/10*beta);
    L=diag(lambda);
    %%
    q=ones(8);
    for i=1:8
        for j=1:8
           if(i==j) q(i,j)=sigma^2;
           else q(i,j)=rho*sigma^2;
           end
        end
    end
```

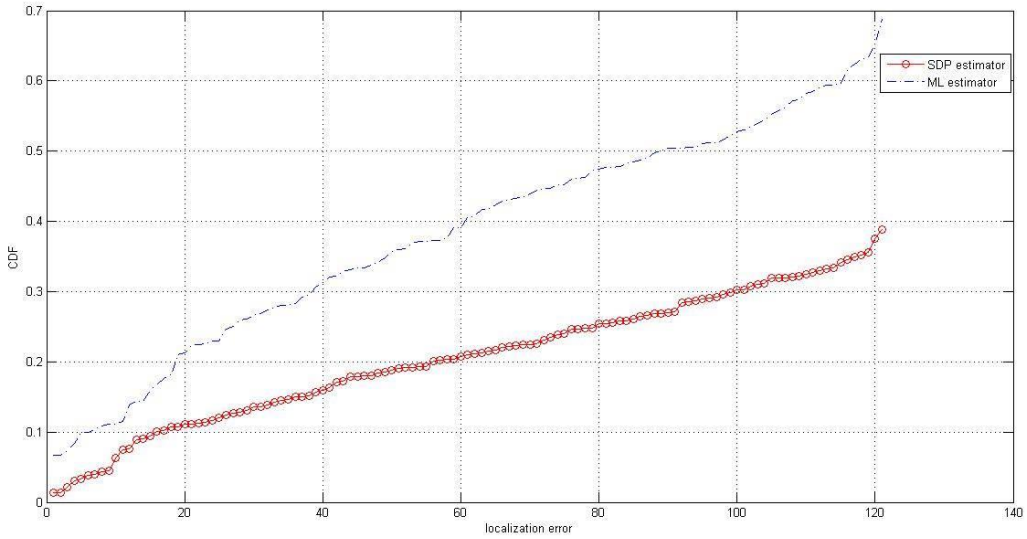
```
    % The W is a weighting matrix which is proportional to the inverse of
the covariance matrix
    W=((10*beta)^2 / (log(10))^2 )*inv(q);
    x=sym('x',[1,2]);
    for k=1:8
       tmp=x' ;
       tmp=tmp - y(:,k);
       d(k)=(tmp(1)^2+tmp(2)^2)^0.5;  %d is convex, 4x1 vector of distances
between x and y(i)
    end
    for j=1:8
       D(j,j)=[y(1,j),y(2,j),-1]*[eye(2) x';  x  x*x']*[y(1,j),y(2,j),-1]';
    end
    p=W*L*D*L';
    q=2*d'*ones(1,8)*W*L;
    F=trace(p)+trace(q);
    %% calculation of d vector, D matrix and final problem formulation in
cvx(http://cvxr.com/cvx/doc/CVX.pdf) assuming x as a variable
    x=fminunc(@(x)F,0);

    error(z)=norm(x-x1); %storing the estimation error due to the z-th
point
end
```

Results and Inferences

We obtained the following graphs from simulating the algorithm with anchor nodes at

- (1,1)
- (1,-1)
- (-1,1)
- (-1,-1)
- (0,0)
- (0.25,0.5)
- (0.1,-0.1)
- (0.2,-0.2)

We encountered problems with the Disciplined Programming paradigm with the CVX solver, to ensure every expression inside cvx is convex. We figured out that the non-diagonal entries of D matrix do not affect the optimal values and hence replaced D=d*d' with corresponding Schur components.
We obtained the following graph for the Cumulative Distribution Function (CDF) of the error of various random test cases.



We have obtained the following graph for the Root Mean Square of the Error (RMSE) for different coefficients of shadowing.