Lab: Fitting Propagation Models from Ray Tracing Data

Ray tracing is a widely-used method for predicting wireless coverage in complex indoor and outdoor environments. Ray tracers take a 3D model of some region along with locations of transmitters and can predict the path characteristics from the transmitter locations to specified receiver locations. Ray tracing requires certain assumptions (such as building materials to make the material), so they may not be exact. Nevertheless, they provide an excellent approximation and are often used by cellular carriers to select sites for deploying cells. In this lab, we will use ray tracing outputs to generate a collection of path data from which we can fit analytic models. Ray tracing can provide much more data than would be possible with time-consuming real measurements.

In going through this data, you will learn to:

- Load and describe ray tracing data produced from a commercial ray tracing tool
- Compute the omni-directional path loss from the ray tracing data
- Determine if links are in outage (no path), LOS or NLOS
- Visualize the path loss and link state as a function of distance
- Fit simple models for the path loss and link state using machine learning tools

Submission: Complete all the sections marked `TODO`, and run the cells to make sure your scipt is working. When you are satisfied with the results, print the file to PDF and submit the PDF. Do not submit the MATLAB source code.

Data from Remcom

The data is in this lab was created by NYU MS student Sravan Chintareddy and Research Scientist Marco Mezzavilla. They used a widely-used and powerful commercial ray tracer from Remcom. Remcom is one of the best ray tracing tools in the industry. Although we will illustrate the concepts at 1.9 GHz carrier, the Remcomm tool is particularly excellent for mmWave studies. Remcom has generously provided their software to NYU for purpose of generating the data for this lab and other research.

The data was in this lab comes from a simulation of a section of Reston, VA. In this simulation, a number of transmitters were placed in the area in location similar to possible micro-cellular sites. The receivers were placed on street levels similar to pedestrians (UEs). We can plot the area and the sites with the following.

```
A = imread('map.png');
imshow(A, 'InitialMagnification', 40);
```

Loading the data

We can load the data with the following command. This will create three variables in your workspace.

- txpos: A trx x 3 array of the positions of the transmitters
- rxpos: An nrx x 3 array of the positions of the receivers
- pathTable: A table of all the paths

```
load pathData;

% TODO:  Find nrx, ntx and npath
```

```
% TODO:  Plot the locations  of the TX and RX on the x-y plane.
% You can ignore the z coordinate of both in the plot.  Use different
% markers (e.g. 'o' and 's' for the TX and RXs).
```

Determine omni-directional path loss and minimum path delay

The table, pathTable, has one row for each path found in the ray tracer. Each path has a TXID and RXID and key statistics like the RX power and delay. Due to multi-path, a (TXID,RXID) pair may have more than one path.

```
% TODO:  Use the head command to print the first few rows of the path
% table.

% TODO:  Loop through the paths and create the following arrays,
% such that for each RXID i and TXID j:
%    pathExists(i,j) = 1 if there exists at least one path
%    totRx(i,j) = total RX power in linear scale
%    minDly(i,j) = minimum path delay (taken from the toa_sec column)



% TODO:  For each link (i,j), compute the omni-directional path loss,
% which is defined as the txPowdBm - total received power (dBm).
% In this dataset, txPowdBm = 36
txPowdBm = 36;
```

Plot the path loss vs. distance

Just to get an idea of the path losses, we plot the omni directional path losses as a function of distance and compare against the FSPL.

```
% TODO:  Using the arrays rxpos and txpos, compute
%   dist(i,j) = distance between RX i and TX j in meters.
% Do this without for loops.

% At this point, you should have nrx x ntx matrices such as dist,
% plomni, minDly and pathExists.  For the subsequent analysis, it is
% useful to convert these to nrx*ntx x 1 vectors.
%
% TODO:  Convert dist, plomni, minDly and pathExists to vectors
% dist1, plomni1, ...

% TODO:  Compute the free-space path loss for 100 points from dmin to dmax.
% Use the fspl() command.
dmin = 10;
dmax = 500;
fc = 1.9e9;     % Carrier frequency

% TODO:  Create a scatter plot of plomni1 vs. dist1 on the links
% for which there exists a path.  On the same plot, plot the FSPL.
% Use semilogx to put the x axis in log scale.  Label the axes.
% Add a legend.

% TODO:  Plot the path losses
```

Classify points

In many analyses of propagation models, it is useful to classify links as being in LOS, NLOS or outage. Outage means there is no path.

```
% TODO:  Create a vector Ilink of size nrx*ntx x 1 where
% linkState(i) = losLink = 1:   If the link has a LOS path
% linkState(i) = nlosLink = 2:   If the link has only NLOS paths
% linkState = nrx*ntx;
losLink = 0;
nlosLink = 1;
outage = 2;


% TODO:  Print the fraction of the links in each of the three states
```

Plot the path loss vs. distance for the NLOS and LOS links

To get an idea for the variation of the path loss vs. distance, we will now plot the omni path loss vs. distance separately for the LOS and NLOS points. You should see that the LOS points are close to the FSPL, but the NLOS points have much higher path loss.

```
% TODO:  Create a scatter plot of the omni path loss vs. distance
% using different markers for LOS and NLOS points.  On the same graph,
% plot the FSPL.  Label the axes and add a legend.
```

Linear fit for the path loss model

We will now fit a simple linear model of the form,

```
plomni = a + b*10*log10(dist) + xi,  xi ~ N(0, sig^2)
```

MATLAB has some basic tools for performing simple model fitting like this. The tools are not as good as sklearn in python, but they are OK. In this case, you can read about the fitlm() command to find the coefficients (a,b) and sig for the LOS and NLOS models. For sig, take the RMSE as the estimate, which is the root mean squared error.

```
% TODO:  Fit linear models for the LOS and NLOS cases
% Print the parametrs (a,b,sig) for each model.


% TODO:  Plot the path loss vs. distance for the points for the LOS and
% NLOS points as before along with the lines for the linear predicted
% average path loss, a + b*10*log10(dist).
```

Plotting the link state

The final part of the modeling is to understand the probability of a link state as a function of the distance. To visualize this, divide the distances into bins with bin i being

```
[(i-1)*binwid, i*binwid],  i = 1,...,nbins
```

We will create an array nbins x 3 arrays:

```
cnt(i,j) = number links whose distance is in bin i and linkState = j
cntnorm(i,j)
         = cnt(i,j) / sum( cnt(i,:) )
         = fraction of links whose distance is in bin i and linkState = j
```

```
nbins = 10;
binwid = 50;
binlim = [0,nbins*binwid];
bincenter = ((0:nbins-1)' + 0.5)*binwid;

% TODO:  Compute cnt and cntnorm as above.  You may use the histcounts
% function.


% TODO:  Plot cntnorm vs. bincenter using the bar() command with the
% 'stacked' option.  Label the axes and add a legend
```

Predicting the link state

We conclude by fitting a simple model for the probability. We will use a simple multi-class logistic model where, for each link i, the relative probability that linkState(i) == j is given by:

```
log P(linkState(i) == j)/P(linkState(i) == 0)
    = -B(1,j)*dist1(i) - B(2,j)
```

for j=1,2. Here, B is the matrix of coefficients of the model. So the probability that a link is in a state decays exponentially with distnace. 3GPP uses a slightly different model, but we use this model to make this simple.

Fitting logistic models is discussed in the ML class. Here, we will use the MATLAB mnrfit routine.

```
% TODO:  Use the mnrfit() method to find the coefficients B.  You will need
% to set the response variable to y = linkState + 1 since it expects class
% labels starting at 1.


% TODO:  Use the mnrval() method to predict the probabilties of each class
% as a function of distance.


% TODO:  Plot the probabilities as a function of the distance.
% Label your graph.
```

Compare the link state

Finally, to compare the predicted probabilties with the measured valued, plot the probabilities as a function of the distance on top of the bar graph in a way to see if they are aligned. You should see a good fit