Contact Tracing for COVID-19 Using Bluetooth Low Energy

Manxi (Maggie) Shi
manxishi@gmail.com

ABSTRACT

Contact tracing in simple terms is the process of determining whom an infected person has come in contact with, and is used to attempt to slow the spread of infectious and life-threatening diseases. Here is a link to an article further describing contact tracing for COVID-19. Manual contact tracing done by public health officials is a slow and inefficient process. People are working on digital contact tracing using bluetooth low energy for it to one day replace manual contact tracing. Cell phones and other portable devices can transmit bluetooth low energy signals that can be picked up by others around them, leading to it being a very popular topic of research.

KEYWORDS

pi: Raspberry Pi 4
obstructed: wall in between two pi's
unobstructed: no wall in between two pi's
csv: comma separated values
RSSI: received signal strength indicator
Bluetooth Low Energy (BLE): classic bluetooth but with lower energy consumption
obstruction detection model: MLP Classifier used to detect whether or not the pi's had a wall between them
proximity detection model: MLP Classifier used to detect whether or not the pi's were greater than or equal to six feet apart, in this case, assuming they are unobstructed
MLP Classifier: multilayer perceptron classifier, an artificial neural network using supervised learning

I. INTRODUCTION

A. *Project Description*

This project addresses the topic of determining whether or not there is an obstruction between two devices based on RSSI values of bluetooth low energy beacons. This is relevant to the general goal of contact tracing because it helps determine if a person was possibly too close for too long to an infected person. If there is an obstruction between two devices, one can determine that there is nearly zero chance of being exposed. If there is no obstruction between two devices, going one step further to determine whether or not they were within six feet of each other can determine if someone was too close for too long. This project used two cases, no wall in between the two pi's and a wall in between the two pi's. In both scenarios, there was no movement in between the pi's to prevent disrupting the signal advertising and receiving.

B. *Background Information*

This experiment assumes that both devices are stationary and uncovered, so it does not address real world scenarios when a device is in a person's pocket, or when the person carrying the device is moving. Additionally, this experiment also assumes that there is nothing moving in between the two devices, for example no person is walking in between the two other people carrying the devices.

II. HYPOTHESIS/HYPOTHESES

Given RSSI values collected by a raspberry pi, one can determine using a machine learning algorithm whether or not there is a wall between the two raspberry pi's. Similarly, with a machine learning algorithm, one can determine if the pi's are within six feet of one another, considering that they are stationary and unobstructed. Throughout this report, the two models will be referenced using the name 'obstruction detection model' and 'proximity detection model' respectively.

III. EXPERIMENTS AND DATA COLLECTIONS

IV. TRIALS CONDUCTED

Distance between pi's (in feet)	Obstruction (1:yes 0:no)	Duration (in minutes)
1	0	10
2	0	10
…	…	…
9	0	10
10	0	10
4	1	10
5	1	10
6	1	10
7	1	10

A. *Plan and Execution*

Fourteen total data collections were conducted. Ten were unobstructed, with two pi's on the floor one foot, two

feet, three feet, up until ten feet part. Four trials were obstructed, with two pi's on the floor four, five, six, and seven feet apart with a wall between them. Each time, the pi scanned for ten minutes, with a revisit of one second. One limitation that was not able to be overcome was keeping the thickness of the wall constant. Walls were chosen to accommodate the distance between the pi's, length of the cord, and distance to an outlet. One wall had a thickness of around six feet while the other one had a thickness of around one foot. This may have caused the data to be inconsistent.

B. Data Relevance

Trials were conducted for the purpose of identifying a correlation between the standard deviation of RSSI values and whether or not the pi's had a wall between them. It was initially believed that having a wall between the two pi's would produce data with a larger standard deviation. Additionally, data was collected with the pi's placed one through ten feet apart in order to train a machine learning model to determine whether or not two devices are closer than six feet. Data from these trials proved the basic hypothesis that the further apart the pi's are, the smaller the RSSI value, but larger in magnitude since it is negative.

C. Examples

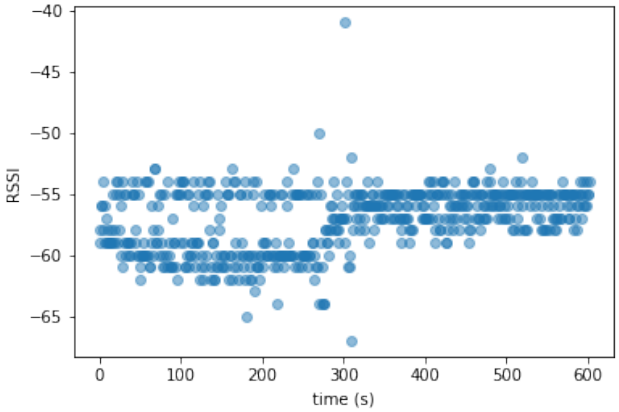

Figure 1: scatter plot of around 600 RSSI values taken at 5 feet with no obstruction

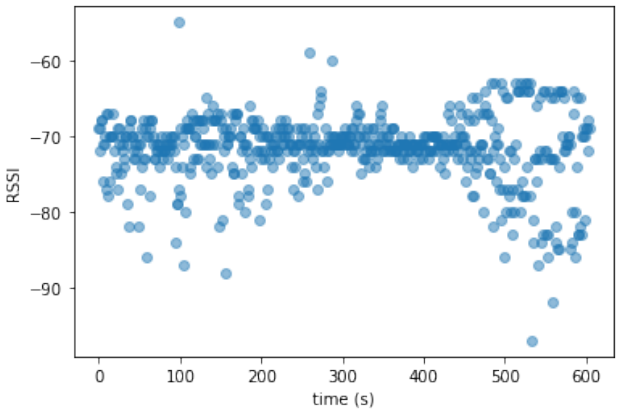

Figure 2: scatter plot of around 600 RSSI values taken at 7 feet with no obstruction

These scatter plots alone, Fig. 1 and Fig. 2, do not give much information about whether or not there was an obstruction in between two pi's or if they were less than six feet apart. The mean and standard deviation of RSSI values are also not immediately clear, so using a normal PDF graph, or normal probability density function, which illustrates the relative likelihood of obtaining a certain value in a random test. The x value of the highest y point on the graph is the mean, and also the value with the highest probability of being obtained. The more 'flat' this curve is, the larger the standard deviation is. Fig. 3 shows a normal PDF graph of RSSI values from trials 5 feet and 7 feet apart with no obstruction.

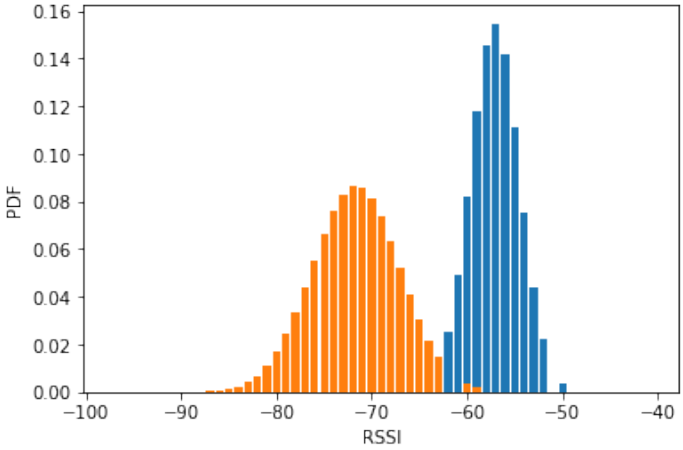

Figure 3: PDF graph of RSSI for trials: no obstruction 5 feet apart (blue) and no obstruction 7 feet apart (orange)

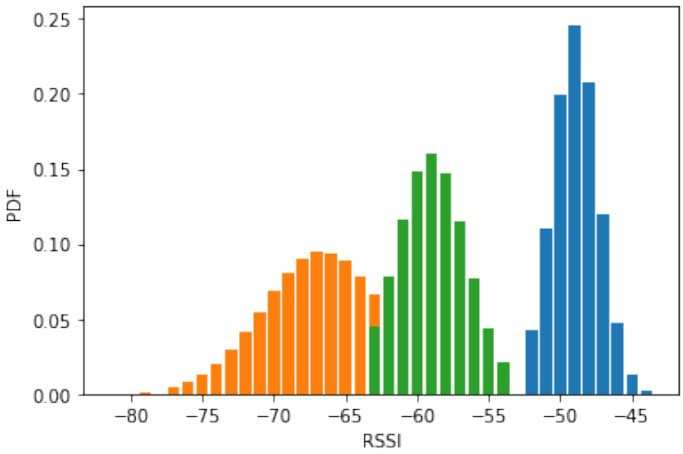

Figure 4: PDF graph of RSSI for trials: no obstruction 2 feet apart (blue), no obstruction 4 feet apart (blue), and no obstruction 6 feet apart (orange)

	mean	std	distance	obstruction	proximity
0	-61.998782	6.373927	4.0	1.0	NaN
1	-57.855288	1.458603	5.0	1.0	NaN
2	-58.877770	2.538168	6.0	1.0	NaN
3	-68.467343	3.232136	7.0	1.0	NaN
4	-43.598391	3.442809	1.0	0.0	1.0
5	-49.605283	1.640860	2.0	0.0	1.0
6	-53.301217	3.575819	3.0	0.0	1.0
7	-67.753965	4.255935	4.0	0.0	1.0
8	-57.969813	2.622117	5.0	0.0	1.0
9	-59.995698	2.515659	6.0	0.0	0.0
10	-72.593401	4.651372	7.0	0.0	0.0
11	-69.976957	2.165694	8.0	0.0	0.0
12	-75.942024	5.730364	9.0	0.0	0.0
13	-68.182564	1.901151	10.0	0.0	0.0

Figure 5: section of pandas DataFrame containing cleaned data

V. Analysis and Algorithm

A. Description

When looking at the raw data, it was noticed that two devices always appeared in the "address" column of the csv file. However, only one was the wanted address of the receiver raspberry pi. A function was created to drop the unwanted rows and reset all indices. The detection algorithm is centered around using a supervised neural network, multilayer perceptron classifier, MLP Classifier, from the python library scikit-learn. This model utilizes supervised learning, meaning it requires training data with inputs paired with correct outputs. After the model learns from this training data, it is given test data to test its performance. In this experiment, two MLP Classifiers were used, one obstruction detection model and one proximity detection model. The first model was given an input of standard deviation and mean of RSSI values and expected to output 0 for unobstructed and 1 for obstructed. The second model was given an input of simply the mean RSSI's and expected to output 0 for six feet or further and 1 for closer than six feet. Both of these are binary classification problems, meaning there are only two categorical outputs.

To prepare the data for analysis, first a function was created to extract the mean and standard deviation of the RSSI's for each trial. Then, all the resulting dataframes were concatenated into a large dataframe containing means and standard deviations. Except there was one issue, the models did not yet know the output paired to all these inputs. To test the hypothesis that a machine learning algorithm could detect obstruction between two pi's, the dataframe had to contain '0' values for no obstruction and '1' values for obstruction. An extra column was added to the dataframe containing values for this purpose. Similarly, in order for the proximity detection model to predict whether or not the pi's were closer than six feet, it would need to be trained on data labeled with their respective distances. Another column with distances for each respective trial was added to the dataframe. For each distance, if it was greater than or equal to six feet, a '0' was added to a new column labeled 'proximity', and a '1' was added to this column if the distance was less than six feet. Since the proximity detection model is a binary classifier, meaning it outputs a 1 for too close and 0 for safe. Because machine learning models require very large amounts of data to successfully produce accurate results, the values coming from fourteen trials were added a hundred times to the final dataframe, each time slightly randomized using python's 'random' library. Since this experiment's approach took only two final values from each data collection of ten minutes, it was simply not possible to get a thousand or more trials, which is why this randomizing approach was used. The final dataframe had 1,800 rows, each with five columns: mean RSSI, standard deviation of RSSI, distance between the pi's, value of obstruction, and value of proximity.

Once the data was cleaned and ready to be analyzed, the data was split into 30/70 training data and testing data. Then, the obstruction detection model and proximity detection model, both MLP Classifiers, were trained using the training data containing input values and their paired output values. After giving the models testing inputs and storing the predicted output values, the accuracy score was calculated for both training data and testing data. Some additional tools to visualize the results were used, including an ROC curve and confusion matrix, Fig. 6 and Fig. 7.

B. Results and Examples

The accuracy on training data for the obstruction detection model was 88.96% and 88.7% on testing data. The accuracy for the proximity detection model was 89.14% on training data and 92% on testing data. Both these models did not overfit, as shown by the very similar accuracies for both the training and testing data. Overfitting models have a very high accuracy on training data, but a very low accuracy on testing data. This is because the model is trained too specifically to the training data and not generalized enough to real world data. However, the very similar accuracies for both training and testing data raises some reason for further investigation. Both models were also fairly good at reducing the number of false negatives and false positives, the obstruction detection model with 24 and 21 respectively and the proximity detection model with 24 and 0 respectively. A false positive in the case of the obstruction model means that the model wrongly determined there was an obstructor between the two devices when really there was not. This increases risk because a person does not believe he/she has been exposed, therefore will not properly quarantine and/or seek treatment. A false negative in the case of the proximity model means that the model wrongly determines the people were more than six feet apart, when in reality they were too close. This also puts people at higher risk because they falsely believe they were not exposed. For the obstruction detection model and the proximity detection model, the area under the ROC curves were 95 and 97 respectively.

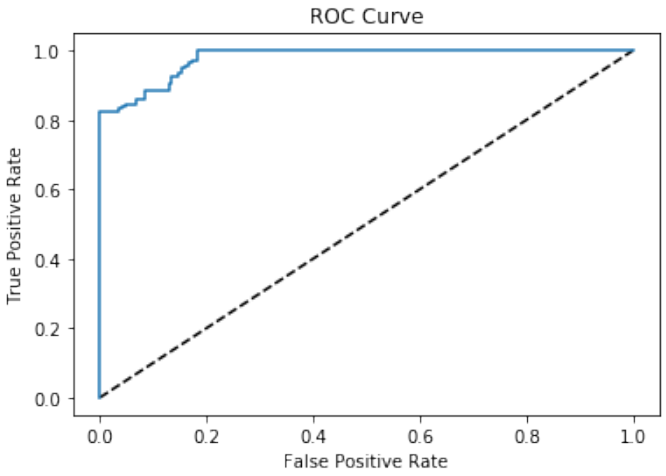

Figure 6: ROC curve for proximity detection model using MLP Classifier, AUC = 0.95

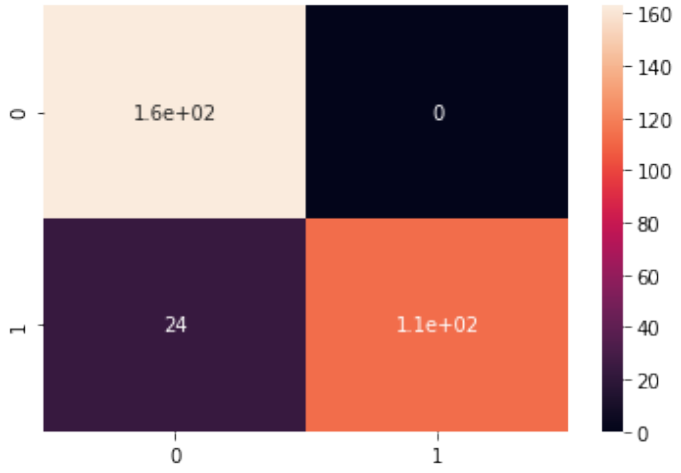

Figure 7: confusion matrix for proximity detection model using MLP Classifier

VI. Conclusions

A. *Hypothesis Evaluation*

The hypothesis was proven true because two machine learning models, specifically MLP Classifiers, were able to determine with high accuracy if there was a wall between the pi's and if they were too close, for unobstructed pi's.

B. *Noteworthy Conclusions*

There is relatively no correlation between the presence of an obstruction and the increase of standard deviation of RSSI values. It was initially believed that having a wall between the pi's would cause the RSSI values collected to be more spread out. However, data shows that that is not necessarily the case.

VII. Next Steps

After performing a few extra tests where there was constant movement between the pi's, it was determined that the RSSI values were too ambiguous to reliably determine proximity. However, in the real world, people rarely stand still for long periods without any moving object or person in between them. Thus, this experiment cannot be necessarily applied to the real world. This experiment assumed two devices to be completely stationary as well as uncovered. In order to make it more practically applicable, more trials should be conducted with movement in between the two pi's or moving the pi's during data collection. Additionally, more trials should be conducted with device(s) covered with fabrics or set in purses or backpacks to emulate real world situations. Using only bluetooth-based contact tracing may not be the most feasible and effective method of contact tracing due to the lack of information plain RSSI values give, but using bluetooth in combination with other methods such as radar can be very powerful in advancing the development of contact tracing for infectious diseases.

References

1. GitHub PiPACT Repository - Manxi Shi
https://github.com/maggies1105/PiPACT
2. COVID-19 Contact Tracing CDC
https://www.cdc.gov/coronavirus/2019-ncov/daily-life-coping/contact-tracing.html
3. Bluetooth Low Energy
https://en.wikipedia.org/wiki/Bluetooth_Low_Energy
4. Understanding AUC-ROC Curve
https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5
5. Probability Density Functions
https://www.investopedia.com/terms/p/pdf.asp
6. PACT: Private Automated Contact Tracing
https://pact.mit.edu
7. Using BLE Signal Strength Estimation to Facilitate Contact Tracing for COVID-19
https://arxiv.org/pdf/2006.15711.pdf
8. Apple | Google Privacy Preserving Contact Tracing
https://blog.google/documents/57/Overview_of_COVID-19_-Contact_Tracing_Using_BLE.pdf
8. Bluetooth Contact Tracing
https://www.technologyreview.com/2020/04/22/1000353/bluetooth-contact-tracing-needs-bigger-better-data/
9. MLP: Multilayer Perceptron
https://en.wikipedia.org/wiki/Multilayer_perceptron
10. Classification in Supervised Machine Learning
https://medium.com/@categitau/in-one-of-my-previous-posts-i-introduced-machine-learning-and-talked-about-the-two-most-common-c1ac6e18df16