Optimizing Regression Models for Wi-Fi Signal Location Estimation

Serhat Arslan
Stanford University
sarslan@stanford.edu

Disha Dasgupta
Stanford University
disha01@stanford.edu

1 Introduction

Virtual reality applications are becoming increasingly involved in our daily lives. Applications generally use headset/handset pairs and handset movements are generally tracked by laser sensors. Apart from the burden of pre-installment, such sensors fail to locate both headsets and handsets when some obstacle (such as the user or some furniture in the room) gets in between, severely limiting their efficacy of location estimation.

In order to solve the difficulty of headset/handset pair usage, [1] suggests the use of Wi-Fi signals as they are unaffected by obstacles. By using a machine learning approach, we can create a robust, real time estimation for location, as opposed to the more complex and computationally expensive theoretical signal processing approach.

1.1 Background

Wi-Fi communication systems can be used to measure the time-of-flight of a signal in general. Every transmitted signal has a known amplitude and phase features that can be measured by the receiver with some reductions as a function of the distance and direction to the transmitter. Eq. 1 shows the received signal where d is the distance between the transmitter and the receiver and c is the speed of light. As the distance increases, the amplitude of the signal drops, whereas the phase of the signal periodically changes.

$$g(t) = \frac{1}{d}\cos\left(2\pi f\left(t - \frac{d}{c}\right)\right) \quad (1)$$

Generally received signals are a superposition of all the reflecting signals (with different amplitudes and phases) of the original signal, which typically occur on every object in the room. Since such reflecting objects are usually randomly located, statistical approaches are required for reflection estimation. Our algorithm aims to generalize that with the proposed machine learning architecture.

In order to increase the data rates, Wi-Fi communication systems simultaneously transmit signals at multiple frequencies for each user. In our setup, the communication between antennas is achieved at 30 different frequencies. As a result, single communication between a receiver and transmitter antenna has 30*2 = 60 parameters to be collected. Coefficient of 2 in this calculation refers to the amplitude and phase of each frequency transaction.

Although amplitude reduction can be used to estimate the distance to the transmitter, the direction of the distance vector is relatively harder to compute. For this purpose, we used 3 antennas on both the transmitter and the receiver. As a result, the direction can be calculated from the relative small phase differences of the received signals from different receiving antennas. With multiple antennas, the setup (as shown of figure 1) creates 3*3 = 9 point-to-point communications with each of them carrying 60 parameters. In general, we have 540 measurement features for every measurement from our setup. Those features are used as the input of our model.

During the experiments, ground truth locations of the devices are collected with commercial laser sensors and a 3-dimensional output vector is calculated. The vector includes the

relative distance of the headset/handset pair in meters and is used as the output of our model. In other words, the location estimation problem is simplified to mapping 540 parameters into 3 parameters with linear regression or neural networks.

Figure 1: The Handset/Headset pair with the antennas located on top of each other for the experimenting purposes

2 Related Work

Indoor location estimation has been studied many times and has a wide variety of use-cases. Systems that try to make use of Wi-Fi signals generally require the deployment of new infrastructures such as antenna arrays, beacons, etc. Then the signal detection and processing for location estimation relies on amplitude and phase detection as previously mentioned. For example, [3] uses specialized access points to process the transmitted signals and [4] proposes the use of acoustic beacons for accurate signal detection. Although it is possible to increase the number of samples, most of those systems only provide accuracy within tens of centimeters, which prevents them from being used in practical situations.

Furthermore, the processing requires the estimation of the reflectors in the room for a better model of superpositioning in the signal. [5] uses the property that small changes in location slightly changes the reflecting paths, so that all paths can be statistically estimated after a duration of time. On the other hand, [6] proposes the use of stereo vison systems for precise detection of location, so that reflecting paths are easily modeled. However those techniques are computationally expensive and fail to scale for real-time applications.

Complex algorithms for location estimation are suggested to be replaced by neural networks in [1] as a very unique perspective for the proposed problem. Since the used machine learning model would learn any reflectance models automatically, better precision levels are achieved. The virtual reality devices can easily communicate with any Wi-Fi transmitters and use trained weights to perform linear computations on the device itself on real-time too. In this report, we would like to investigate the capabilities of such neural networks to learn the behavior of the Wi-Fi signals in the room.

3 Dataset

We collected our own dataset with the setup proposed by [2]. With about 10 hours of collection, we managed to collect nearly 555,000 measurements. We collected the data in batches of 100 seconds, with different scenarios of movements between the devices. Table 1 shows the movement trends in every batch, which are generally indicative of a user's regular movements.

Table 1: Data collection trends for different batches

Batch #	**Headset**	**Handset**		**Batch #**	**Headset**	**Handset**
1	Up-Down	Up-Down		8	Back-Forward	Side-to-Side

2	8 shape (x=0 plane)	8 shape (z=0 plane)		9	Circle (x=0 plane)	Circle (z=0 plane)
3	Side-to-Side	Side-to-Side		10	8 shape	stationary
4	Back-Forward	Back-Forward		11	Cross shape (fast)	stationary
5	Up-Down	Back-Forward		12	Cross shape (slow)	stationary
6	Back-Forward	Up-Down		13-33	random	stationary
7	Side-to-Side	Back-Forward				

After combining the data from all the batches, we used the built-in methods from Python's scikit-learn library to split the data into training and test groups with a distribution of 75%-25% for training and testing, respectively. As a result, we used 415,883 samples for training and 138,628 samples for testing.

4 Methods

As we are estimating a continuous value, we decided to test our setup with both Linear Regression and Neural Network models.

4.1 Linear Regression

Linear Regression is one of the fundamental models of machine learning algorithms. It finds a set of linear combination coefficients that map the input to an output. The calculation of output is given in Eq. 2. $h_\Theta(x^i)$ denotes the predicted value of the output for the input $x^i$and Θ denotes the weights of the model.

$$h_\Theta(x^i) = \Theta_0 + \sum_{j=1}^{n} \Theta_j x_j^{(i)} \tag{2}$$

The cost (error) of the model is calculated as shown on Eq. 3 and the minimum cost is calculated iteratively with the gradient descent algorithm described in Eq. 4

$$J(\Theta) = \frac{1}{m}\sum_{i=1}^{m}\left(h_\Theta\left(x^{(i)}\right) - y^{(i)}\right)^2 \tag{3}$$

$$\Theta_j := \Theta_j - \alpha\left[\frac{1}{m}\sum_{i=1}^{m}\left(h_\Theta\left(x^{(i)}\right) - y^{(i)}\right)x_j^{(i)} + \frac{\lambda}{m}\Theta_j\right] \tag{4}$$

4.2 Preprocessing of the Dataset

While we used all 540 features for linear regression, since it was not computationally intensive, we determined that training multiple neural networks with all 540 features could be. Thus, we performed Principle Component Analysis (PCA) on the data before feeding the data into the neural networks. From the analysis, we determined that the input could be represented with ~99.5% accuracy with only 90 input features. Figure 2 shows the variance captured with respect to the number of features included after PCA.

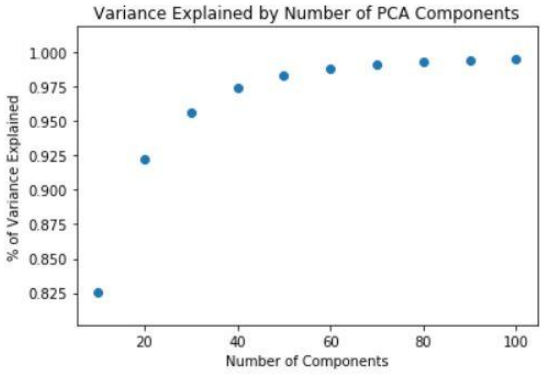


Figure 2: A 540-dimensional input vector is reduced to 90-dimensional vector after PCA with variance loss less than 0.5%

4.3 Neural Networks

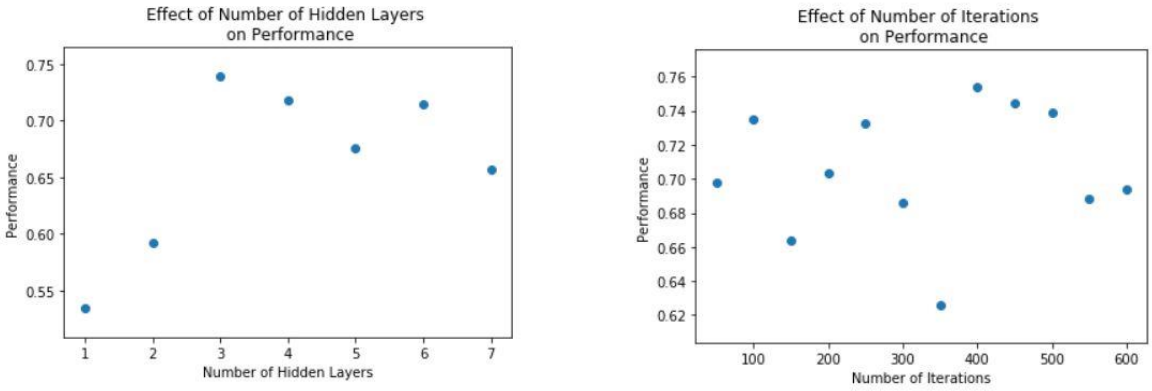
Neural networks are more complicated architectures that are successful at estimating non-linear hypothesis functions. We tried multiple neural networks with different depths to tune for the best model for our problem. We fed in a 90-dimensional input vector and multiplied that with the first hidden layer weights of the model to produce the activations of that layer. Then a bias term was added to the layer as another activation node and fed to the second layer weights. This process was iterated until the output layer. For each neural network, we determined cost using the Eq. 5.

$$J(\Theta) = \frac{1}{m}\sum_{i=1}^{m}\left(h_\Theta\left(x^{(i)}\right) - y^{(i)}\right)^2 + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_l-1}\left(\Theta_{i,j}{}^{(l)}\right)^2 \quad (5)$$

To minimize the cost function above, we used backpropagation instead of gradient descent.

5 Results

To determine the parameters that optimize our neural network's performance, we tested different hidden layer depths, different regularization terms, and different numbers of iterations. We used 60 activation nodes on each hidden layer (roughly the two thirds of the size of input layer) based on a standardized rule described by [7]. The performance of the model for different depths, regularizations, and iterations is shown in Figure 3. On the analysis for neural network depth, we observed that the performance of the model decreases after 3 hidden layers, which may indicate overfitting. Our analysis also shows that optimal performance is reached after 400 iterations although the results are varied due to the random initialization of the weights. Since the location estimator is a non-linear function, our linear regression model could not estimate the output as well as the neural network with optimal parameters could.



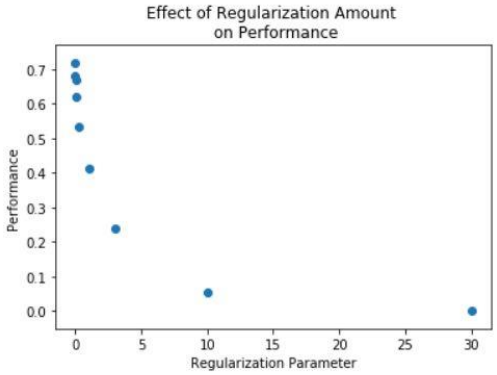


Figure 3: Model performance with different depths, regularization, and iterations models

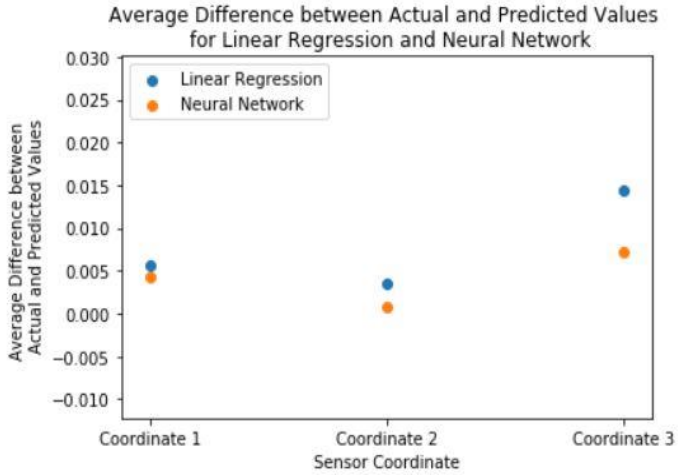


Figure 4: Comparison of linear model and optimal neural network

6 Discussion and Conclusion

We achieved the best results with a neural network that has 3 hidden layers with 60 activations each, 0.001 as the regularization parameter, and 400 iterations. We decreased the dimensions of the input from 540 to 90 through PCA then fed the 90-dimensional input vector into the neural network.

Our neural network model almost always produced estimations that were close to the ground truth within a centimeter. Such a precision in location estimation is comparable to the related work in state of the art systems and clearly promises for a commercial use in the future.

An extension to our work could focus on the generalization of the model for different communication environments with different reflective behavior. Such experiments may require better tuning for the lowest possible error rates, but would enable the commercial use of such systems for general purpose use. With a more flexible schedule, we believe the idea could even be deployed on present systems with Wi-Fi enabled antennas on them.

Contributions

Data Collection/Algorithm Design – Serhat and Disha; Initial Code – Disha; Code Editing/ Tuning – Serhat; Initial Report Write Up – Serhat. Final Report Edits/Tuning – Disha.

References

[1] Manikanta Kotaru, Alexander Anemogiannis, Samuel Joseph, and Sachin Katti. 2017. Demo: Position Tracking for Virtual Reality Using Commodity WiFi. In Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking (MobiCom '17). ACM, New York, NY, USA, 488-489. DOI: https://doi.org/10.1145/3117811.3119865

[2] Manikanta Kotaru, et. al. 2018. Real-time VR Positioning Using a Single Commodity WiFi Device. Under Review for ML Conference. Berlin, Germany.

[3] Kiran Joshi, Steven Hong, and Sachin Katti. 2013. PinPoint: localizing interfering radios. In Proceedings of the 10th USENIX conference on Networked Systems Design and Implementation (nsdi'13), Nick Feamster and Jeff Mogul (Eds.). USENIX Association, Berkeley, CA, USA, 241-254.

[4] Kaikai Liu, Xinxin Liu, and Xiaolin Li. 2013. Guoguo: enabling fine-grained indoor localization via smartphone. In Proceeding of the 11th annual international conference on Mobile systems, applications, and services (MobiSys '13). ACM, New York, NY, USA, 235-248. DOI: https://doi.org/10.1145/2462456.2464450.

[5] Manikanta Kotaru and Sachin Katti. 2018. Position Tracking for Virtual Reality Using Commodity WiFi. In Proceedings of the 10th on Wireless of the Students, by the Students, and for the Students Workshop (S3 '18). ACM, New York, NY, USA, 15-17. DOI: https://doi.org/10.1145/3264877.3264882

[6] Swarun Kumar, Stephanie Gil, Dina Katabi, and Daniela Rus. 2014. Accurate indoor localization with zero start-up cost. In Proceedings of the 20th annual international conference on Mobile computing and networking (MobiCom '14). ACM, New York, NY, USA, 483-494. DOI=http://dx.doi.org/10.1145/2639108.2639142

[7] Jeff Heaton. 2008. Introduction to Neural Networks for Java, 2nd Edition (2nd ed.). Heaton Research, Inc..