

RSSI-Based Supervised Learning for Uncooperative Direction-Finding

Tathagata Mukherjee², Michael Duckett¹, Piyush Kumar¹, Jared Devin Paquet⁴, Daniel Rodriguez¹, Mallory Haulcomb¹, Kevin George², and Eduardo Pasilliao³

¹ CompGeom Inc., 3748 Biltmore Ave., Tallahassee, FL 32311
{michael, piyush, mallory}@compgeom.com

² Intelligent Robotics Inc., 3697 Longfellow Road, Tallahassee, FL 32311
{tathagata, kevin}@intelligentrobotics.org

³ Munitions Directorate, AFRL, 101 West Eglin Blvd, Eglin AFB, FL 32542
eduardo.pasilliao@us.af.mil

⁴ REEF, 1350 N. Poquito Rd, Shalimar, FL 32579, USA.
jaredpaquet@ufl.edu

Abstract. This paper studies supervised learning algorithms for the problem of *uncooperative direction finding* of a radio emitter using the received signal strength indicator (RSSI) from a rotating and uncharacterized antenna. Radio Direction Finding (RDF) is the task of finding the direction of a radio frequency emitter from which the received signal was transmitted, using a single receiver. We study the accuracy of radio direction finding for the 2.4 GHz WiFi band, and restrict ourselves to applying supervised learning algorithms for RSSI information analysis. We designed and built a hardware prototype for data acquisition using off-the-shelf hardware. During the course of our experiments, we collected more than three million RSSI values. We show that we can reliably predict the bearing of the transmitter with an error bounded by 11 degrees, in both indoor and outdoor environments. We do not explicitly model the multi-path, that inevitably arises in such situations and hence one of the major challenges that we faced in this work is that of automatically compensating for the multi-path and hence the associated noise in the acquired data.

Keywords: Data mining, radio direction finding, software defined radio, regression, GNURadio, feature engineering

1 Introduction

One of the primary problems in sensor networks is that of node localization [4, 9, 21]. For most systems, GPS is the primary means for localizing the network. But for systems where GPS is denied, another approach must be used. A way to achieve this is through the use of special nodes that can localize themselves without GPS, called anchors. Anchors can act as reference points through which other nodes may be localized. One step in localizing a node is to find the direction of an anchor

with respect to the node. This problem is known as Radio Direction Finding (RDF or DF). Besides sensor networks, RDF has applications in diverse areas such as emergency services, radio navigation, localization of illegal, secret or hostile transmitters, avalanche rescue, wildlife tracking, indoor position estimation, tracking tagged animals, reconnaissance and sports [1, 13, 15, 16, 23, 25] and has been studied extensively both for military [8] and civilian [6] use.

Direction finding has also been studied extensively in academia. One of the most commonly used algorithms for radio direction finding using the signal received at an antenna array is called MUSIC [22]. MUSIC and related algorithms are based on the assumption that the signal of interest is Gaussian and hence they use second order statistics of the received signal for determining the direction of the emitters. Porat et al. [18] study this problem and propose the MMUSIC algorithm for radio direction finding. Their algorithm is based on the Eigen decomposition of a matrix of the fourth order cumulants. Another commonly used algorithm for determining the direction of several emitters is called the ESPRIT algorithm [19]. The algorithm is based on the idea of having *doublets* of sensors.

Recently, researchers have used both unsupervised and supervised learning algorithms for direction finding. Graefenstein et al. [10] used a robot with a custom built rotating directional antenna with fully characterized radiation pattern for collecting the RSSI values. These RSSI values were normalized and iteratively rotated and cross-correlated with the known radiation pattern for the antenna. The angle with the highest cross-correlation score was reported as the most probable angle of the transmitter. Zhuo et al. [27] used support vector machines with a known antenna model for classifying the directionality of the emitter at 3 GHz. Ito et al. [14] studied the related problem of estimating the orientation of a terminal based on its received signal strength. They measured the divergence of the signal strength distribution using Kulback-Leibler Divergence [17] and estimated the orientation of the receiver. In a related work, Satoh et al. [20] used directional antennas to sense the 2.4 GHz channel and applied Bayesian learning on the sensed data to localize the transmitters.

In this paper we use an uncalibrated receiver (directional) to sense the 2.4 GHz channel and record the resulting RSSI values from a directional as well as an omni-directional source. We then use feature engineering along with machine learning and data mining techniques (see Figure 1) to learn the bearing information for the transmitter. Note that this is a basic ingredient of a DF system. Just replicating our single receiver with multiple receivers arranged in a known topology, can be used to determine the actual location of the transmitter. Hence pushing the boundary of this problem will help DF system designers incorporate our methods in their design. Moreover, such a system based only on learning algorithms would make them more accessible to people irrespective of their academic leaning. We describe the data acquisition system next.

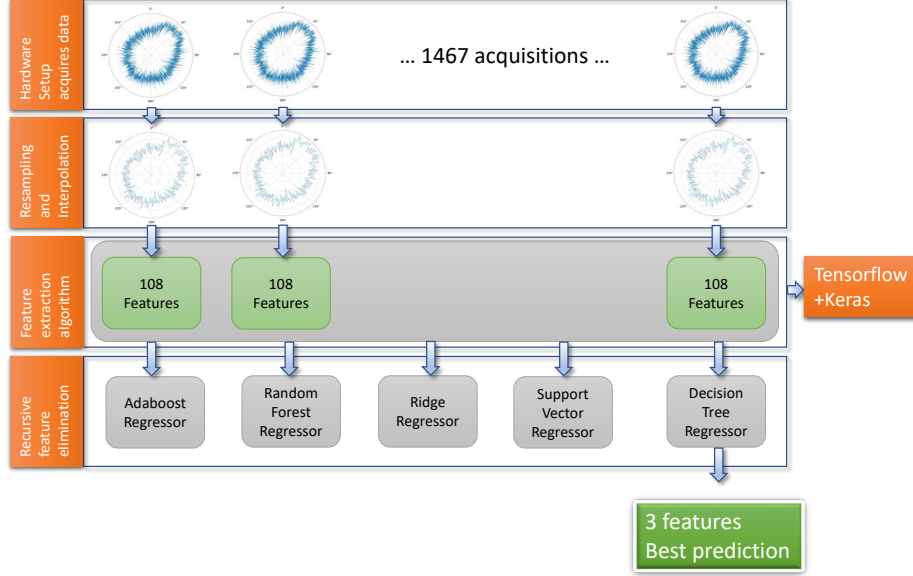


Fig. 1: Data processing system for learning algorithm

2 Data Acquisition System

The data collection system is driven by an Intel NUC (Intel i7 powered) running Ubuntu 16.04. For sensing the medium, we use an uncharacterized COTS 2.4 GHz WiFi Yagi antenna. This antenna is used both as a transmitter as well as a receiver. For the receiver, the antenna is mounted on a tripod and attached to a motor so that it can be rotated as it scans the medium. We also use a TP-Link 2.4GHz 15dBi gain WiFi omni-directional antenna for transmission to ensure that the system is agnostic to the type of antenna being used for transmission. For both transmission and reception the antenna is connected to an Ettus USRP B210 software defined radio. To make the system portable and capable of being used anywhere we power the system with a 12V6Ah deep cycle LiFePO battery. A Nexus 5X smart-phone is used to acquire compass data from its on-board sensors and this data is used for calibrating the direction of the antenna at the start of each experiment.

There are two main components for our setup: the receiver and the transmitter. For our tests, we placed the receiver at the origin of the reference frame. The transmitter was positioned at various locations around the receiver. The transmitter was programmed to transmit at 2.4 GHz. and the receiver was used to sense the medium at that frequency as it rotated about its axis. Our experiments were conducted both indoors and outdoors.

For our analysis we consider one full rotation of the receiver as the smallest unit of data. Each full rotation is processed, normalized and considered as a unique data point that is associated with a given bearing to the transmitter. For each

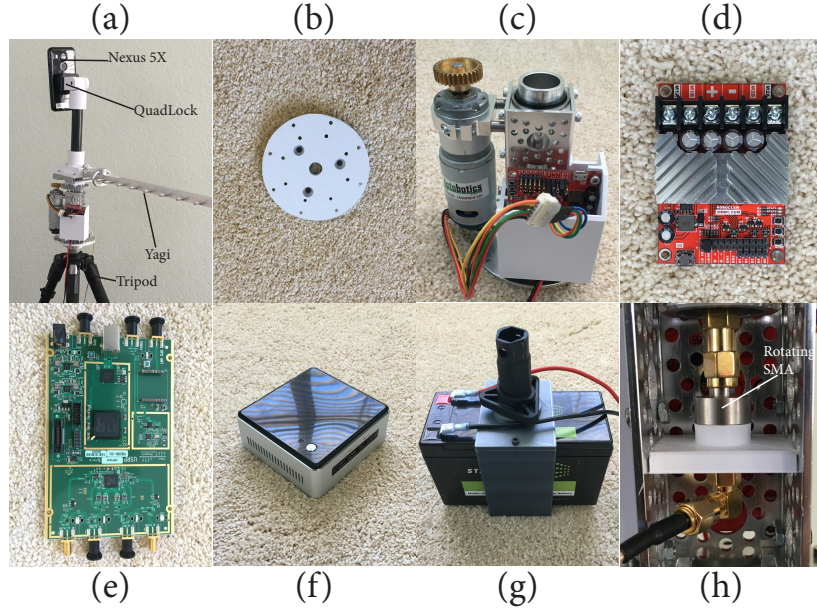


Fig. 2: (a): The full yagi setup, (b): plate adapter, (c): Pan Gear system composed of motor and motor controller mounted on standing bracket, (d): motor controller, (e): B210 Software Defined Radio, (f): NUC compact computer, (g): StarkPower lithium ion battery and holder, (h): chain of connectors from B210 to antenna including a rotating SMA adapter located in standing bracket

experiment we collected several rotations at a time, with the transmitter being fixed at a given bearing with respect to the receiver, by letting the acquisition system operate for a certain amount of time. We call each experiment a run and each run consists of several rotations.

There are two important aspects of the receiver that need to be controlled: the rotation of the yagi antenna and the sampling rate of the SDR. The rotation API has two important functions that define the phases of a run: first, *finding north* and aligning the antenna to this direction, so that every time the angles are recorded with respect to this direction; and second, the actual *rotation*, that makes the antenna move and at the same time uses the Ettus B210 for recording the spectrum. In the first phase the yagi is aligned to the magnetic north using the compass of the smart phone that we used in our system. In the second phase the yagi starts to rotate at a constant angular velocity. While rotating, the encoder readings are used to determine the angle of the antenna with respect to the magnetic north, and the RSSI values are recorded with respect to these angles. It should be noted that the angles from the compass are not used because the encoder readings are more accurate and frequent. The end of each rotation is determined based on the angles obtained from the encoder values.

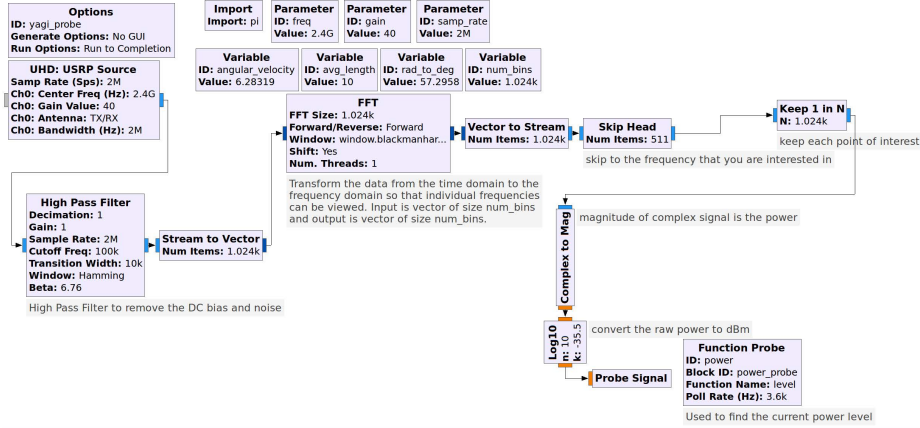


Fig. 3: GNURadio Companion flow graph

In order to record the RSSI, we created a GNURadio Companion flow graph [2]. Our flow graph gets the I/Q values from the B210 (*UHD: USRP Source*) at a sample rate and center frequency of 2 MHz and 2.4 GHz respectively. We run the values through a high-pass filter to alleviate the DC bias. The data is then chunked into vectors of size 1024 (*Stream to Vector*), which is then passed through a Fast-Fourier Transform (*FFT*) and then flattened out (*Vector to Stream*). This converts the data from the time-domain to the frequency-domain. The details of the flow-graph is shown in Figure 3.

3 Data Analysis

Now we are ready to describe the algorithms used for processing the data. Our approach has three phases: the feature engineering phase takes the raw data and maps it to a feature space; the learning phase uses this representation of the data in the feature space to learn a model for predicting the direction of the transmitter; and finally we use a cross validation/testing phase to test the learned model on previously unseen data. We start with feature engineering.

3.1 Feature Engineering

As mentioned before, our data consists of a series of *runs*. Each run consists of several rotations and each rotation is vector of (angle, power) tuples. The length of this vector is dependent on the total time of a rotation (fixed for each run) and speed at which the SDR samples the spectrum, which varies. Typically, each rotation has around 2200 tuples. In order to use this raw data for further analysis we transformed each rotation into a vector of fixed dimension, namely $k = 360$. We achieved this by simulating a continuous mapping from angles to powers based on the raw data for a single rotation and by reconstructing the

vector using this mapping for k evenly spaced points within the range 0 to 2π . The new set of rotation vectors denoted by R is a subset of \mathbb{R}^k . For our analysis, we let $k = 360$ because each run is representative of a sampling from a circle.

During the analysis of our data, we noticed a drift in one of the features (*moving average max value* which is defined below). This led us to believe that the encoder measurements were changing with time during a run (across rotations). Plotting each run separately revealed a linear trend with high correlation (Figure 5). In order to correct the drift, we computed the least squares regression for the most prominent runs (runs which displayed very high correlation), averaged the slopes of the resulting lines, and used this value to negate the drift. The negation step was done on the raw data for each run because at the start of each run, the encoder is reset to zero. Once a run is corrected it can be split into rotations. Since each rotation vector can be viewed as a time-series, we use time series feature extraction techniques to map the data into a high dimensional feature space. Feature extraction from time series data is a well studied problem, and we use the techniques described by Christ et al. [5] to map the data into the feature space. In all there were 86 features that were extracted using the algorithm. In addition to the features extracted using this method, we also added a few others based on the idea of a moving average [12].

More precisely, we use the *moving average max value*, which is the index (angle) in the rotation vector where the max power is observed after applying a moving average filter. The filter takes a parameter d , the size of the moving average which for a given angle is computed by summing the RSSI values corresponding to the preceding d angles, the angle itself and the succeeding d angles. Finally this sum is divided by the total number of points $(2d + 1)$, which is always odd. We use the *moving average max value* with filter sizes ranging from 3 to 45, using every other integer. This gives an additional 22 features, which brings the total to 108 features.

3.2 Learning Algorithms

Note that we want to predict the bearing (direction) of the transmitter with respect to the receiver for each rotation. As the bearing is a continuous variable, we formulate this as a regression problem. We use several regressors for predicting the bearing: 1) *SVR*: *Support vector regression* is a type of regressor that uses an ϵ -insensitive loss function to determine the error [17]. We used the RBF kernel and `GridSearchCV` for optimizing the parameters with cross validation

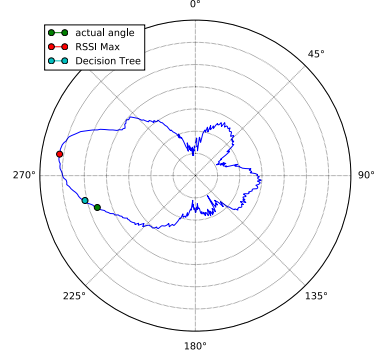


Fig. 4: Example rotation with markers for the actual angle and two predicted angles using the max RSSI and Decision Tree methods

2) *KRR: Kernel ridge regression* is similar to SVR but uses a squared-error loss function [17]. Again, we use the RBF kernel with `GridSearchCV` to get the optimal parameter 3) *Decision Tree* [3]: we used a max depth of 4 for our model and finally 4) *AdaBoost with Decision Tree* [7] [26]: short for *adaptive boosting*, uses another learning algorithm as a “weak learner”. We used a decision tree with max depth 4 as our “weak learner”.

Although we have a total of 108 features, not all of them will be important for prediction purposes. As a result, we try two different approaches for selecting the most useful features: 1) the first one ranks each feature through a scoring function, and 2) the second prunes features at each iteration and is called *recursive feature extraction with cross-validation* [11]. For the first we use the function *SelectKBest*, and for the later we used *RFECV*, both implemented in ScikitLearn.

We also use neural networks for the prediction task. We used Keras for our experiments, which is a high-level neural networks API, written in Python and capable of running on top of TensorFlow. We used the Sequential model in Keras, which is a linear stack of layers. The results on our dataset are described in section 4.

4 Experiments and Results

In this section we present the results of our experiments. In total, we collected 1467 *rotations* (after drift correction) at 76 unique angles (an example of a rotation reduced to 360 vertices can be seen in Figure 4). After reducing each rotation to 360 power values, we ran the dataset through the feature extractor, which produced 108 total features. We tried out several regressors, namely, (SVR, KR, DT, and AB) and strategies:- (moving average max value without learning, moving average max value with learning, SelectKBest, RFECV, and neural networks). The objective for this set of tests was to find the predictor that yielded the lowest mean absolute error (MAE), which is the average of the absolute value of each error from that test.

4.1 Regressors

We used the data from the feature selection phase to test a few regressors. For each regressor, we split the data (50% train, 50% test), trained and tested the model, and calculated the MAE. We ran this 100 times for each regressor and took the overall average to show which regressor preformed the best with all the features. The results from these tests are in Table 1.

Table 1: The average error for the each regressors over 100 runs with 50-50 split

	SVR	KRR	DT	AB
Avg. Error	26.4°	55.2°	16.2°	22.1°

From the results we can see that *decision trees* give the lowest MAE compared to the other regressors. We also noticed that *decision trees* ran the train/test

cycle much faster than any other regressor. Based on these results, we decided to use the *decision tree* regressor for the rest of our tests.

4.2 Moving Average Max Value

One of the first attempts for formulating a reliable predictor was to use the *moving average max value* (MAMV). We considered using this feature by itself as a naive approach. We predict as follows: whichever index the *moving average max value* falls on is the predicted angle (in degrees). For our tests, we used a moving average with size 41 (MAMV-41), which was ranked the best using **SelectKBest**, for smoothing the angle data. Since no training was required, we used all the rotations to calculate the MAE. As seen in Table 2 the MAE was 57.1° . Figure 6 shows the errors for each rotation, marking the inside and outside rotations, as well.

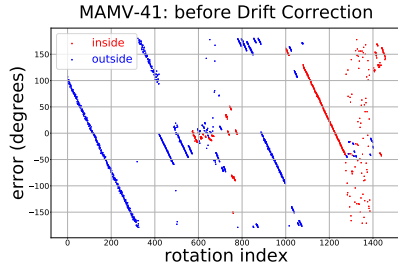


Fig. 5: Errors before drift correction. Lines are runs

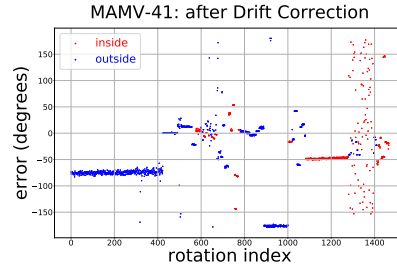


Fig. 6: Errors after drift correction. Lines are runs

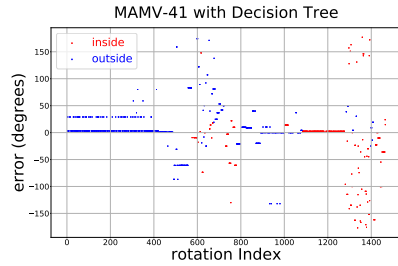


Fig. 7: Errors for MAMV-41 with Decision Tree learning. Even/odd for test/train split

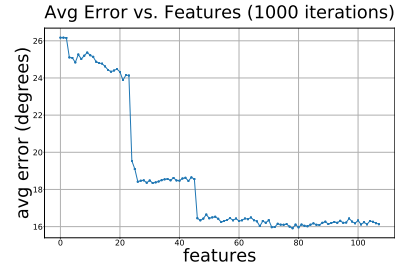


Fig. 8: MAE vs. ranked features from **SelectKBest** using Decision tree over 1000 runs

Our next step was to use the *decision trees* with the MAMV-41 feature. We applied a 50/50 train/test split in the data and calculated the MAE for each run. We averaged and reported the MAE for all runs. The average MAE over 1000 runs was 25.9° (Table 2). Figure 7 shows a graph of errors for the train/test split

where the odd index rotations were for training and the even index rotations were for testing.

4.3 SelectKBest

As mentioned before, we used **SelectKBest** to rank all the features. In order to get stable rankings, we ran this 100 times and averaged those ranks. Once we had the ranked list of features, we created a “feature profile” by iteratively adding the next best feature, running train/test with the *decision tree* regressor for that set of features, and recording the MAE. We repeated this process 1000 times and the results are shown in (Figure 8). It is to be noted that the error does not change considerably for a large number of consecutive features but there are steep drops in the error around certain features. This is because many features are similar and using them for the prediction task does not change the error significantly. The first plateau consists mostly of MAMV features since they were ranked the best among all the other features. The first major drop is at ranked feature 24, which marks the start of the set of continuous wavelet transform coefficients (CWT). The second major drop is caused by the addition of the 2nd coefficient from Welch’s transformation [24] (2-WT) at ranked feature 46. Beyond that, no significant decrease in MAE is achieved by the inclusion of another feature. The best average MAE over the whole profile is 15.7° at 78 features (Table 2).

4.4 RFECV and Neural Network

We ran **RFECV** with a decision tree regressor using a different random state every time. **RFECV** consistently returned three features: MAMV-23, MAMV-41, and 2-WT. Using these three features, we trained and tested on the data with a 50/50 split 10000 times with the decision tree regressor. The average MAE was 11.0° (Table 2). Between **RFECV** and **SelectKBest**, there are four unique features which stand out among the rest. To be thorough, we found the average MAE for all groups of three and four from these four features. None of them were better than the original three features from **RFECV**.

Table 2: Comparison of average error among predictor methods

	MAMV-41	MAMV-41(DT)	SelectKBest	RFECV	Neural Net
Avg. MAE	±57.1°	±25.9°	±15.7°	±11.0°	±15.7°

For the neural network approach, we used all the features which were produced from the feature selection phase. We settled on a $108 \Rightarrow 32 \Rightarrow 4 \Rightarrow 1$ layering. The average MAE over 100 runs with a 50/50 train/test split was 15.7° (Table 2). In order to show how the neural network stacked against feature selection, we performed an experiment showing each method’s performance versus a range of train/test splits. Figure 9 shows that **RFECV** with its three features performed better than our neural network at all splits.

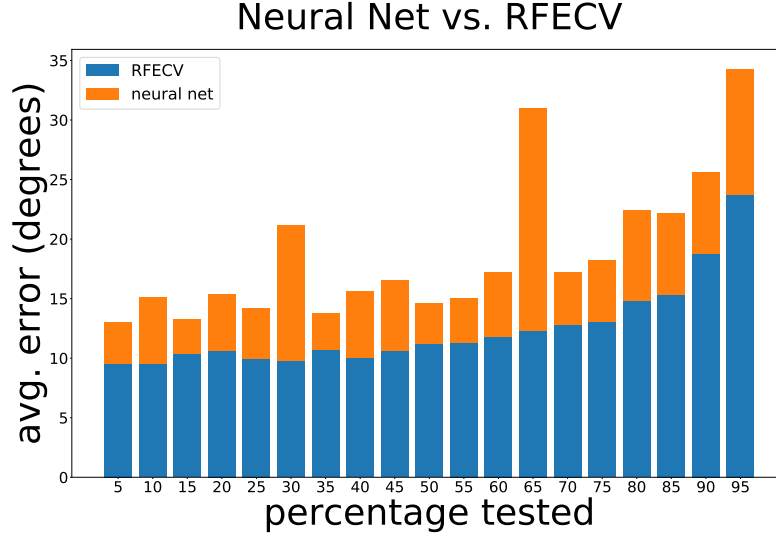


Fig. 9: Neural net vs. RFECV performance. The x-axis represents the percentage of the dataset tested with the other partition being used for training (for example 5% tested means 95% of the dataset was used for training).

4.5 Discussion

From our results in Table 2, we determined that RFECV was the best feature selector. The amount of time it takes to filter out significant features is comparable to SelectKBest, but RFECV produces fewer features which leads to lower training and testing times. Figure 9 shows that RFECV beats neural networks consistently for a range of train/test splits. There are a couple of possible reasons why RFECV performed better. The SelectKBest strategy ranks each feature independent of the other features, which means similar features will have similar ranks. As features are iteratively added, many consecutive features will be redundant. This is evident in Figure 8 where the addition of similar features cause very little change in MAE creating plateaus in the plot. Our SelectKBest method was, in a way, good at finding some prominent features (where massive drops in MAE occurred), but not in the way we intended whereas RFECV was better at ranking diverse features.

5 Conclusion and Future work

The main contribution of this paper is to show that using pure data mining techniques with the RSSI values, one can achieve good accuracies in direction-finding using COTS directional receivers. There are several directions that can be pursued for future work: 1) How accurately can cell phones locations be analyzed with the current setup? 2) Can we minimize the total size of our receive system?

3) How well does this system work for different frequencies and ranges around them? 4) When used in a distributed setting, how much accuracy can one achieve for localization, given k receivers operating at the same time (assuming the distances between them is known). In the journal version of this paper, we will show how to theoretically solve this problem, but extensive experimental results are lacking. In the near future, we plan to pursue some of these questions using our existing hardware setup.

References

1. Bahl, V., Padmanabhan, V.: Radar: An in-building rf-based user location and tracking system. Institute of Electrical and Electronics Engineers, Inc. (March 2000), <https://www.microsoft.com/en-us/research/publication/radar-an-in-building-rf-based-user-location-and-tracking-system/>
2. Blossom, E.: Gnu radio: Tools for exploring the radio frequency spectrum. *Linux J.* 2004(122), 4– (Jun 2004), <http://dl.acm.org/citation.cfm?id=993247.993251>
3. Breiman, L., Friedman, J., Olshen, R., Stone, C.: *Classification and Regression Trees*. Wadsworth (1984)
4. Capkun, S., Hamdi, M., Hubaux, J.P.: Gps-free positioning in mobile ad hoc networks. *Cluster Computing* 5(2), 157–167 (2002)
5. Christ, M., Kempa-Liehr, A.W., Feindt, M.: Distributed and parallel time series feature extraction for industrial big data applications. *arXiv preprint arXiv:1610.07717* (2016)
6. Finders, D.: Introduction into theory of direction finding (2017), <http://telekomunikacije.etf.bg.ac.rs/predmeti/ot3tm2/nastava/df.pdf>, [Online; accessed 28-Feb-2017]
7. Freund, Y. and Schapire, R.E.: A decision-theoretic linearization of on-line learning and an application to boosting (1995)
8. Gething, P.: Radio Direction-finding: And the Resolution of Multicomponent Wavefields. IEE electromagnetic waves series, Peter Peregrinus (1978), <https://books.google.com/books?id=BCcIAQAIAAJ>
9. Graefenstein, J., Albert, A., Biber, P., Schilling, A.: Wireless node localization based on rssi using a rotating antenna on a mobile robot. In: 2009 6th Workshop on Positioning, Navigation and Communication. pp. 253–259 (March 2009)
10. Graefenstein, J., Albert, A., Biber, P., Schilling, A.: Wireless node localization based on rssi using a rotating antenna on a mobile robot. In: Positioning, Navigation and Communication, 2009. WPNC 2009. 6th Workshop on. pp. 253–259. IEEE (2009)
11. Guyon, I., Weston, J., Barnhill, S., Vapnik, V.: Gene selection for cancer classification using support vector machines. *Machine Learning* 46, 389–422 (2002)
12. Hamilton, J.D.: *Time series analysis*, vol. 2. Princeton university press Princeton (1994)
13. Huang, W., Xiong, Y., Li, X.Y., Lin, H., Mao, X., Yang, P., Liu, Y., Wang, X.: Swadloon: Direction finding and indoor localization using acoustic signal by shaking smartphones. *IEEE Transactions on Mobile Computing* 14(10), 2145–2157 (Oct 2015), <http://dx.doi.org/10.1109/TMC.2014.2377717>
14. Ito, S., Kawaguchi, N.: Orientation estimation method using divergence of signal strength distribution. In: Third International Conference on Networked Sensing Systems. pp. 180–187 (2006)

15. Kolster, F.A., Dunmore, F.W.: The radio direction finder and its application to navigation. Washington (1922), ISBN: 978-1-333-95286-0
16. Moell, J., Curlee, T.: Transmitter Hunting: Radio Direction Finding Simplified. TAB book, McGraw-Hill Education (1987), <https://books.google.com/books?id=RfzF2-fHJ6MC>
17. Murphy, K.P.: Machine learning: a probabilistic perspective. MIT press (2012)
18. Porat, B., Friedlander, B.: Direction finding algorithms based on high-order statistics. *IEEE Transactions on Signal Processing* 39(9), 2016–2024 (1991)
19. Roy, R., Kailath, T.: Esprit-estimation of signal parameters via rotational invariance techniques. *IEEE Transactions on acoustics, speech, and signal processing* 37(7), 984–995 (1989)
20. Satoh, H., Ito, S., Kawaguchi, N.: Position estimation of wireless access point using directional antennas. In: *International Symposium on Location-and Context-Awareness*. pp. 144–156. Springer (2005)
21. Savarese, C., Rabaey, J.M., Beutel, J.: Location in distributed ad-hoc wireless sensor networks. In: *Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on*. vol. 4, pp. 2037–2040. IEEE (2001)
22. Schmidt, R.: Multiple emitter location and signal parameter estimation. *IEEE transactions on antennas and propagation* 34(3), 276–280 (1986)
23. Ward, T., Pasilião, E.L., Shea, J.M., Wong, T.F.: Autonomous navigation to an rf source in multipath environments. In: *MILCOM 2016 - 2016 IEEE Military Communications Conference*. pp. 186–191 (Nov 2016)
24. Welch, P.: The use of fast fourier transform for the estimation of power spectra: a method based on time averaging over short, modified periodograms. *IEEE Transactions on audio and electroacoustics* 15(2), 70–73 (1967)
25. Wikipedia: Direction finding — Wikipedia, the free encyclopedia (2016), https://en.wikipedia.org/wiki/Direction_finding, [Online; accessed 20-Dec-2016]
26. Zhu, J., Zou, H., Rosset, S., Hastie, T.: Mutli-class adaboost (2009)
27. Zhuo, L., Dan, S., Yougang, G., Yaqin, S., Junjian, B., Zhiliang, T.: The distinction among electromagnetic radiation source models based on directivity with support vector machines. In: *Electromagnetic Compatibility, Tokyo (EMC'14/Tokyo), 2014 International Symposium on*. pp. 617–620. IEEE (2014)