The Pennsylvania State University

The Graduate School

THE USE OF MACHINE LEARNING AND GENETIC ALGORITHMS FOR CLUTTER CLASSIFICATION AND RADARCOM SIGNAL DESIGN

A Thesis in

Electrical Engineering

by

Richard Washington

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

December 2021

The thesis of Richard Washington was reviewed and approved by the following:

Ram M. Narayanan
Professor of Electrical Engineering
Thesis Advisor

Julio V. Urbina
Associate Professor of Electrical Engineering

Dmitriy S. Garmatyuk
Miami University, Ohio
Special Signatory

Saba Mudaliar
Air Force Research Laboratory
Special Signatory

Kultegin Aydin
Professor of Electrical Engineering
Head of the Department

ABSTRACT

In this thesis, an optimized orthogonal frequency-division multiplexing (OFDM) signal is created that will be used for combined radar and communications (RadarCom) applications using the many objective optimization approach. RadarCom signals provide an efficient way to perform radar and wireless communications simultaneously. One of the key objectives is to disguise the signal as clutter. In order to do this in a real use case, the clutter model for the specific environment will need to be determined. For that purpose, a clutter classification method has been developed. A variety of machine learning methods were tested for this goal. The optimization method primarily looks at the Non-dominated Sorting Genetic Algorithm II (NSGA-II), a well-known, fast sorting and elite multi objective genetic algorithm, but also compares it to other methods. While a successful way to classify clutter and improve the OFDM RadarCom signal was found, more research will need to be done for the signal to be optimized to the point of full use.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I want to thank Dr. Garmatyuk and Dr. Narayanan who have been great professors and advisors, Dr. Mudaliar for his insightful input, and Dr. Urbina for giving his time to help with the process. I also want to thank my parents and family for the support throughout the program and throughout my undergraduate experience.

Federal funds were received through the U.S. air Force. The findings and conclusions presented within are not influenced by or necessarily reflect the view of the funding agency.

Chapter 1

Introduction

There has been an effort to combine radar and communications into one system in a method that is as effective and cost efficient as possible. In certain cases, such as described in [1], there have been efforts to use radar and communication signals in an alternating manner but not simultaneously. The goal here however is to create a single waveform that can perform both duties. This has been an area of research since at least 2009 [2] when Christaan Sturm and Werner Weisbeck proposed a signal that had orthogonal sequences of phase codes and could perform both purposes. The signal they proposed was similar to orthogonal frequency-division multiplexing (OFDM) and used binary phase-shift keying (BPSK), but it was not until 2011 [3],[4] that they used OFDM explicitly. Since 2009, in large part because of computing advances and the proliferation of Software Defined Radios (SDR), joint radar communication signals have been an active area of research. OFDM specifically has received a lot of attention [5],[6],[7]. Here, we want to optimize an OFDM signal in a variety of ways, including its low probability of detection (LPD) performance. To do this, we are disguising it as clutter, and therefore need to know what the clutter looks like in the given environment, so machine learning is used for classification.

The goal of this thesis is to create an optimal signal for the purpose of radar and wireless communication, what we call a RadarCom signal. In addition to the standard radar and wireless communication qualities, the signal should also possess desirable features, such as LPD and low probability of intercept (LPI) by unauthorized platforms as privacy is always a concern. LPD

means that the signal is unlikely to be detected or noticed by unauthorized platforms. This is often done by making the signal noise like, hence the large field of noise radar [8]. However, in this thesis, we will make the signal to be clutter-like instead. This is because in many scenarios. the other platforms are aware that there are radar systems present and therefore will naturally expect to see clutter, but will not know which, if any, signals are for communication purposes. LPI means that if the signal is detected the message is less likely to be decoded or understood. This is often done by encryption [9], and we will discuss potential for this as well. Other researchers have tried to create optimal OFDM signals but rarely include all necessary parameters. For example, [10] tries to optimize pulse-sidelobe ratio (PSLR), peak to average power ratio (PAPR), and detection probability on an OFDM quadrature phase-shift keying (QPSK) signal using genetic algorithms, yet they do not do any communications and therefore do not look at the bit error rate (BER).

In order to disguise our signal as clutter, we needed to characterize the clutter in the area. A technique for doing this was shown by Brenton Bischoff in [11] where he uses deconvolution of received noise with received clutter and noise to have just the clutter returns. However, in order to make this work, the parent random distribution of the clutter returns needs to be determined, not just the clutter returns themselves. How this random distribution and its probability density function PDF will be used will be described in a later chapter. In general, for a given terrain, polarization, and frequency, there is a standard distribution that clutter forms as shown by empirical data [12]. But to narrow the specific PDF down, we show that machine learning techniques can be used to find the parent distribution from a given set of possibilities. We will be comparing Decision Trees, Linear Discriminant Analysis (LDA), K-Nearest Neighbors (KNN), and Support Vector Machines (SVM) to do this clutter classification.

Before we optimize the signal, we want to define what kind of signal we want to use and then which characteristics we want it to have. We had three ways to choose a signal type: a radar

signal that is repurposed for wireless communication such as [13], a wireless communication signal that is repurposed for radar [3][4], or creating a new signal altogether. We chose the second option, specifically choosing the OFDM signal to use as it is a commonly used communications signal; and even more so now that 5G, which uses OFDM, is being used [14],[15].

The performance measures that we chose to measure the quality of the signal was the PAPR, PSLR, the normalized root mean square error (NRMSE) power, BER, and probability of detection (PD). Why we chose these specific objectives will be discussed in Chapter 2. What will be changed in this optimization are the amplitude coefficients of each subcarrier. I chose the second Nondominated Sorting Genetic Algorithm (NSGA-II) for this task but will show some results of the third Nondominated Sorting Genetic Algorithm NSGA (NSGA-III) and Simulated Annealing to show that NSGA-II is one of the better options, if not the best.

In Chapter 2 I will discuss the fundamentals of radar and communication signals, in Chapter 3 I will discuss clutter and its past uses, in Chapter 4 I will discuss background information on Machine Learning, and in Chapter 5 I will discuss fundamentals of the genetic algorithms used. The results for clutter classification are in Chapter 6 while results of optimal signal design are in Chapter 7. Finally, concluding remarks can be found in Chapter 8.

Chapter 2

Radar and OFDM Signal Background

Radar, the process of using radio waves for detection and ranging, has been used widely since the British used it heavily in early warning systems during World War II. A radar system usually has two components. The first is a transmitter which sends an electromagnetic (EM) signal out into the environment. This EM signal can be a simple single frequency tone or something more complex, such as a linear frequency modulated (LFM) chirp. The second component is a receiver which receives the reflections of the EM signal from the environment. In some cases, there may be more than one transmitter or receiver, or in the case of passive radar, the receiver is looking at just the ambient EM waves without a dedicated transmitter. However, in most cases, using the knowledge of the original signal and the return signal one can deduce a lot about the environment, such as the presence, distance, and velocity of any targets. It can even be used to learn about environmental phenomenon, such as the wind shear and air pressure using complex knowledge of how different parts of the atmosphere affect the EM waves.

Communications with EM waves is said to begin with Guglielmo Marconi who had the first documented user of radio waves for communication in 1894 [16]. In RF communications, a transmitter sends a signal, with information encoded on it, and a receiver in another location decodes the received signal. Although the information decoding usually happens in a different place for communications, the hardware to produce and receive a signal is very similar. Therefore, if a single transmitter could be used for wireless communications and radar, then resources would be saved. Also, if one signal could be used for both , time and potentially computing resources would be saved as well.2.1 Radar Fundamentals

Of all the uses of radar, detection and ranging are the oldest and still very important, even as more complicated uses such as polarimetry have evolved. Detection and ranging are done by comparing the original signal with any received signal and is often carried out by matched filtering. Here, the original transmit signal $S_{TX}(t)$ is correlated with the received signal $S_{RX}(t)$ using (2.1):

$$u(t) = S_{RX}(t) * S_{TX}^{*}(t) = \int S_{RX}\left(t - \frac{2r}{c}\right) S_{TX}^{*}(\tau - t)\, \mathrm{d}\tau, \tag{2.1}$$

which ideally results in a delta function if the received signal is an echo of the original. In this ideal case, where there is no periodicity in the transmit signal, the peak in the delta function represents is used to find the location of the target in (2.2) [17]:

$$R = \frac{\tau_d c}{2}\ , \tag{2.2}$$

where *R* is the distance of the target, c is the speed of light, and τ_d is the time delay to receive the signal. If the correlation is done in the frequency domain, then the exact changes in frequency can be determined, and using the Doppler effect, the target's velocity can be determined. Signals other than the echoes of the transmit signal from the target will be correlated too but generally produce a much lower peak and therefore be ignored as noise. There will be echoes from other parts of the environment that produce a response. These are called clutter and will be discussed in the next Chapter.

One problem with this method is that the autocorrelation of the transmit signal used in matched filtering does not produce a perfect delta function if the signal is bandlimited. The peak has some width to it, which is inversely proportional to the bandwidth, that makes it difficult to pinpoint the exact location. In addition, there can be secondary lower peaks called sidelobes. A solution to this is thresholding and ignoring peaks that are too small. In most instances, the range

of received power for when targets are present and are not present overlap, meaning a threshold might lead to some false alarms, i.e. seeing targets that aren't there. This can be resolved by increasing the threshold level, but this will reduce the probability of detection. It is up to the user to pick the optimal threshold.

To make choosing a threshold easier, a signal's matched filter response should be as delta like as possible, with a narrow main lobe and very small side lobes. This is because sidelobes present an additional problem. When there are multiple echoes that are close enough in time and frequency, the sidelobes may overlap producing a large enough peak causing the system to see a false target and adding to the false alarm rate. For this reason, we want to use a signal with sidelobes that are as small as possible. This is done by maximizing the PSLR (2.3):

$$\text{PSLR} = 10\log_{10}\left(\frac{u(t)_{(n)}}{u(t)_{(n-1)}}\right), \tag{2.3}$$

where $u(t)_n$ is the peak of the autocorrelation function response described in (2.1) and $u(t)_{n-1}$ is the second highest peak. There are some signals that are known to have high PSLRs and are used often, such as the LFM chirp and noise radar, both of which benefit from an increase in bandwidth. An example of a LFM signal with a center frequency of 1 GHz and 5 MHZ of bandwidth is shown in Figure 3.1.

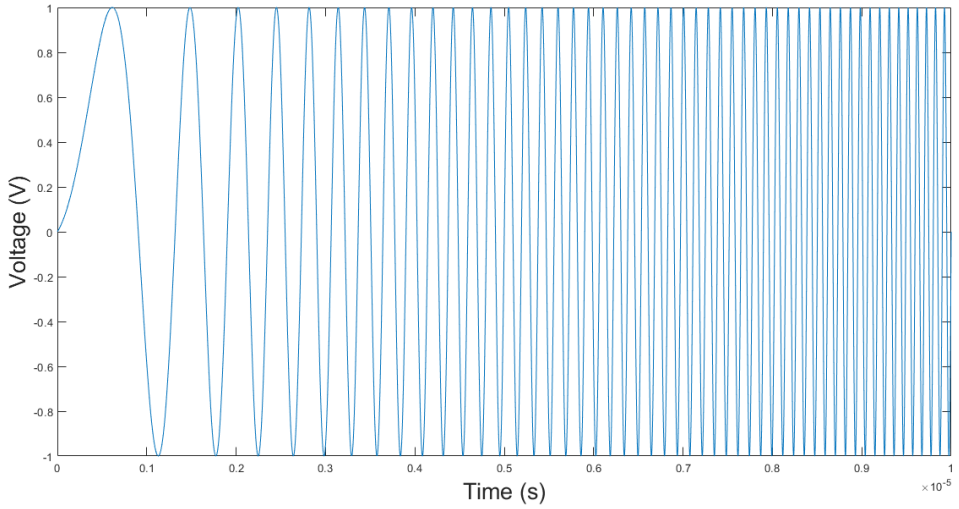


Figure 2-**1**: LFM time signal.

While an increase in bandwidth can help increase range resolution, it has other benefits as well. Since certain materials and shapes have different frequency responses, a larger sweep of frequencies can be used to identify a target based on its unique frequency responses [7]. Jammers may be present who try to drown our signal with interference [18]; while we try to avoid that with the clutter disguise, they can still pose a threat. With an increased bandwidth, the jammer would have more frequencies to jam and would likely need a wideband directional antenna, more power, or both. In the case of communications, higher bandwidth means more subcarriers can be added increasing throughput, or there can be more spacing between them decreasing the bit error rate. For these reasons we focus on using ultra-wideband (UWB) signals for our RadarCom signal.

Another constraint in radar and communication signal design is the peak to average power ratio (PAPR) (2.4):

$$\text{PAPR} = 10\log_{10}\left(\frac{s(t)_{(n)}}{s(t)_{(n-1)}}\right), \tag{2.4}$$

This stems from the hardware, specifically the power amplifier on the transmitter, and therefore the exact constraint varies by the specific radar system being used. OFDM is notorious for having

a high PAPR, since there is one position where sinusoidal waves of the same phase, but different frequencies will overlap at their peak in the time domain resulting in a higher peak and high PAPR. An example is shown in Figure 2.2.

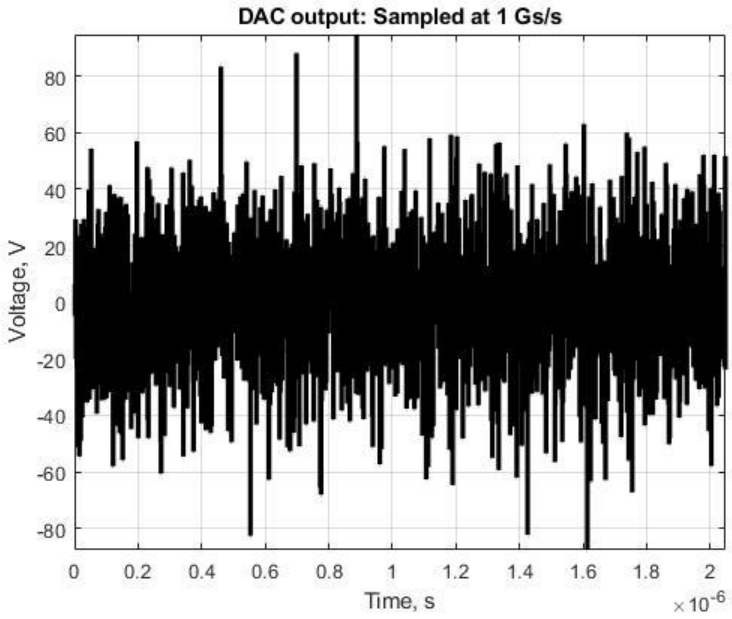


Figure 2-**2**: OFDM signal with PAPR of 7.72 dB.

A lot of this analysis is done at the baseband frequency. However, the baseband signal is usually complex and therefore we need to recover the real and imaginary components. This also allows the phase of the signal to be received. To do this, when the received signal is mixed with the local oscillator to be down converted, the process is done twice. Once normally and once with the local oscillator being shifted by 90 degrees. The former is called the in-phase component I and the latter is the quadrature component Q. The two are then combined according to (2.5-6):

$$\bar{S}_{RX}(n) = \sqrt{Q(n)^2 + I(n)^2} \tag{2.5}$$

$$\phi_{RX(n)} = \arctan\left({}^{I(n)}\Big/_{Q(n)}\right) \tag{2.6}$$

to get the magnitude $\bar{S}_{RX}$ and phase ϕ_{RX} respectively of the baseband signal used for processing. In communications, it is often the phase component that information is encoded onto, which leaves the amplitude component to optimize.

2.2 Reducing Sidelobes

For a simple square pulse, the matched filter response is triangular in shape and actually has no sidelobes, but a very wide main lobe. In this case, the range resolution increases as the pulse length decreases, creating smaller triangles, i.e. smaller mainlobes. This increase in pulse length correlated with an increase in the bandwidth. However, increasing the pulses length to increase resolution also means that if the energy is constant in the pulse, there will be more noise and the signal-to-noise ratio (SNR) will decrease. This is undesirable, as it makes detection in radar more difficult and decreases the bit error rate in communications. So pulse compression waveforms were created to increase bandwidth and resolution at a certain pulse length; linear frequency modulated (LFM) chirps and phase-coded are very popular within this method.

LFM chirps have a linear increase in the frequency within a pulse which means a higher bandwidth is used. The LFM chirp has minimum sidelobes at −13.2 dB [17], but with amplitude weighting can be −30 dB or even much lower [17]. Phase-coded waveforms use sub pulses at different relative phases to one another to reduce sidelobes instead of amplitude weighting. Amplitude weighting does come with the tradeoffs of both SNR loss and resolution reduction. The codes that create the lowest sidelobe have been found up to 105 sub pulses, at which the lowest sidelobes of −13.2 can be achieved. An example of the ACF of the LFM chirp and OFDM signal, described in the next subsection, is shown in Figure 2.3. Both signals have the same energy and a bandwidth of 600 MHz

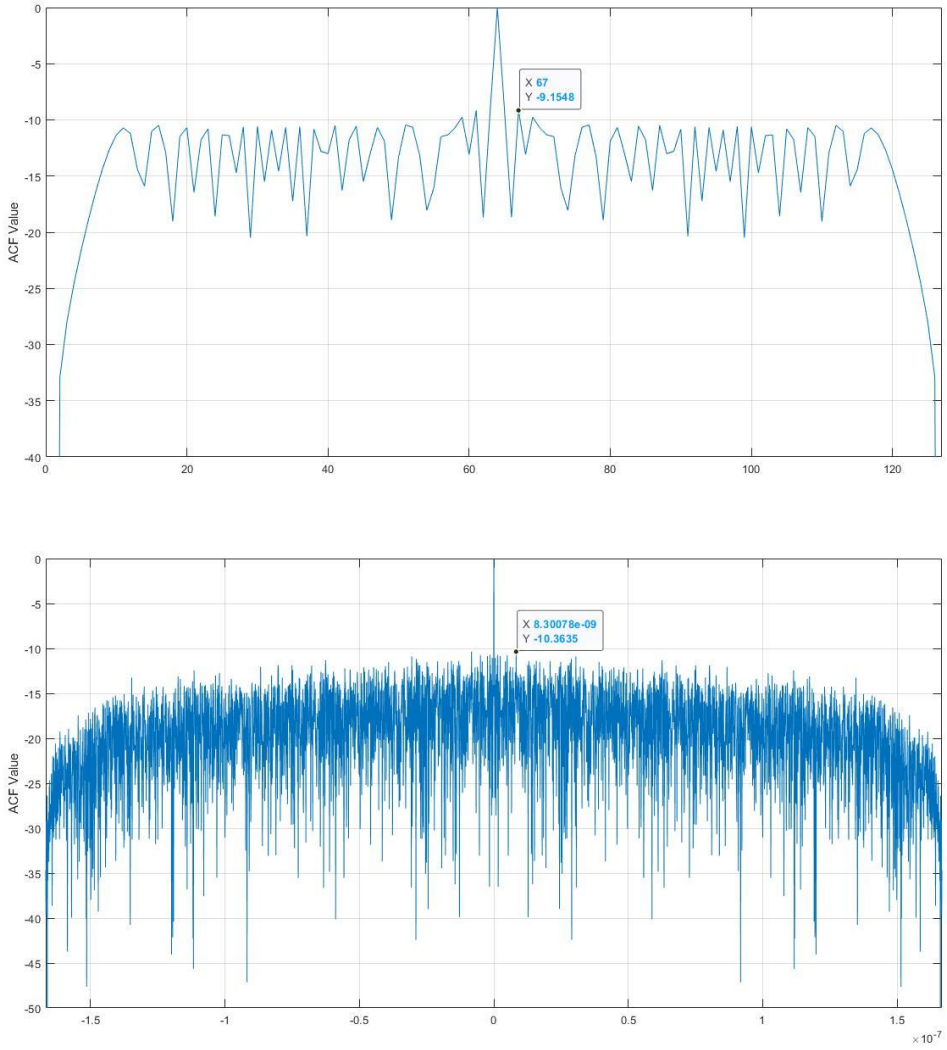


Figure 2-**3**: LFM ACF (top) and OFDM ACF (bottom).

2.3 OFDM Waveform

A common communication signal type for the transmission of digital data is the Orthogonal Frequency-Division Multiplexing (OFDM) signal. This system of transmission uses single frequency subcarriers that are orthogonal to each other, that is in the frequency domain

where one subcarrier has its peak value all other subcarriers are at zero as shown in Figure 3.2 [20]. Each subcarrier is used to transmit data that is encoded using a number of methods.

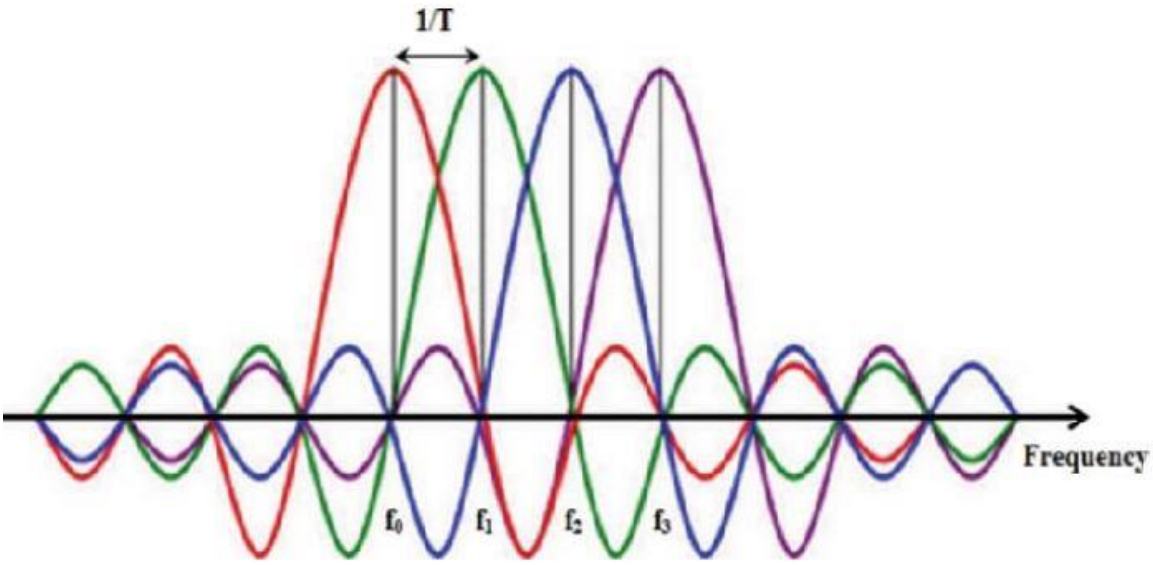


Figure 2-**4**: Frequency spectrum of OFDM.

These methods include Quadrature Phase Shift Keying (QPSK), Binary Phase Shift Keying (BPSK), Quadrature Amplitude Modulation (QAM), and many others. QPSK, BPSK and other phase shift keying methods shift the phase of each subcarrier between transmission and the number of possible equally spaced phases determines the method type, i.e. QPSK uses four phases and BPSK uses two. QAM adds amplitude modulation to create more options. While this project can work with any of the phase only modulation methods, it was decided to use QPSK specifically since it is a balance of having more throughput than BPSK, but a lower bit error rate (BER) than an 8-PSK.

Since QPSK uses four possible phases in a 360 degree space, each separated by 90 degrees from its neighbors, each subcarrier can carry two bits of information, as two bits creates four possible combinations. A higher throughput can be achieved by using a more complex

modulation scheme such as 16QAM, but this comes with the tradeoff of higher error instead caused by reduced spacing between each possible phase shift.

Throughput, the number of bits sent per second, is an important metric to define for a communications system. For OFDM communication signals the throughput is defined by (2.7):

$$\text{Throughput} = N_s \cdot B_s \cdot \text{PRF} \tag{2.7}$$

and is determined by the number of subcarriers N_s, pulse repetition frequency (PRF), and bits per subcarrier, B_s, of the modulation method used. For example if 1024 subcarriers are used with QPSK modulation and PRF is 2 kHz then a speed of 4.096 Megabits per second (Mbps) can be reached. Table 2.1 [21] gives an idea of the Mbps needed to accomplish certain tasks. It should be noted that for radar applications, a higher PRF means more ambiguous range measurements and less ambiguity in Doppler measurements. BER is another important metric and is often inversely proportional to throughput as well as SNR.

Table 2-1: Throughput needed for certain activities

Activity	**Download Speed**
Streaming Standard Definition Music	<0.5 Mbps
Browsing email, social media, and the web	1 Mbps
Streaming SD Videos	3-4 Mbps
Streaming HD Videos	5-8 Mbps
Streaming 4K Videos	15-25 Mbps
Video Calls	6 Mbps

Chapter 3

Clutter Background Information

Understanding what the radar returns means is often made more difficult by interference whether it is background noise, nearby communication transmitters, other radars, or even a radar's own echoes. These echoes from objects that are not the target, such as the terrain itself, can often make detection more difficult by reducing the Signal-to-Clutter-plus-Interference Ratio (SCIR). A visualization of clutter is provided in Figure 3-1 [22] where the satellite radar is measuring the subsurface at its nadir, but is receiving other undesired returns from the surface as well. There are ways to work around this problem, but often this relies on the clutter being characterized. For this reason, characterizing the PDF that describes clutter has been an important topic for radar research. Previous authors have used experimental data from decades of research to characterize the radar clutter distribution for many different frequency bands, terrains, angles, and polarizations [12].

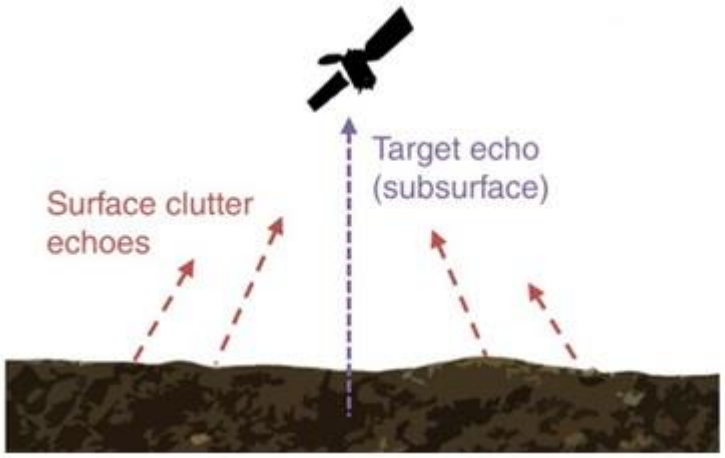


Figure 3-**1**: Visualization of clutter.

3.1 Clutter Applications

Since clutter returns vary based on the environment, it can actually be helpful to analyze and classify the distribution for various applications that need to know the environment. Clutter has been used to identify the difference between sea and land; since the two terrains have different roughness and different reflection coefficients, their clutter returns are different as well. In other cases, clutter not only causes problems for target detection, but also for meteorological radars that try to estimate weather phenomena such as rainfall. The authors in [23] tried to handle this particular issue. Clutter analysis can also help make location measurements affected by anomalous propagation, or when radar beams are bent by Earth's atmosphere. Machine learning has been used to classify clutter before, such as in [24]. Using the polarization, amplitude, phase, and frequency along with the Random Forest machine learning method, they achieved a probability of detection at 96.44% and a false alarm rate of 3.66%. In that paper, only the

presence of clutter was a concern, which it could be suppressed, but here it is desired to classify the actual distribution.

As one can see. clutter is usually an undesirable part of radar returns, but there can be some use for it. Here, we want to disguise our signal as clutter. A variety of machine learning methods will be explored to determine which method and using what approach works best.

3.2 Thesis Goals for Determining Clutter

If we are to disguise the signal as clutter, then we must understand what clutter is present in that region. Since clutter has a general distribution for certain terrain and radar setups, we can have an idea of which clutter distributions to train our machine learning model for. Additionally, since noise, or other forms of interference, can also be present we need to make sure that the model is tested for various amounts of Clutter-to-Interference Ratios (CIR). The results of applying various machine learning models are discussed in Chapter 6.

3.3 Determining the Clutter used in OFDM Optimization

In Chapter 7, we will discuss the results of the OFDM signal optimization including how well it is disguised as clutter. However, before creating a signal that looks like clutter, we need to know what clutter distribution the signal needs to look like. We also need to define what “looks like” means. In this case, there are three domains of concern. There is the frequency spectrum magnitude, frequency spectrum phase, and time domain power. In order to see how closely we fit the clutter, we used the normalized root mean squared error (NRMSE) against the distribution of choice as the metric.

In this case, a Weibull distribution with a scale parameter of 0.5 and a shape parameter of 3 was chosen to model the time domain. The frequency spectrum magnitude and frequency domain were then determined empirically by taking the Fourier Transform of a random vector from that distribution and then seeing what distribution best fit the results. A Rayleigh distribution with a scale factor of 345 best fit the frequency magnitude and a uniform distribution from [0 2π] best fit the phase distribution, which is in alignment with the literature [25].

Now that we have the distributions to compare to, we still need to know what NRMSE values are considered acceptable. First, we introduce a new parameter called NRMSE Power defined as the mean of the square of the three NRMSEs (3.1):

$$\text{NRMSE Power} = \frac{1}{3}\sum_{i=1}^{3} \text{NRMSE}_i^2 \quad . \tag{3.1}$$

Then, the NRMSEs of 200 different pairs of 100,000 long random vectors in each of three domains were found. This was done in order to create a baseline to compare our results to. The mean NRMSE Power was 0.4841 with a minimum of 0.2183 and a maximum of 0.9505. The standard deviation was 0.1397, so if we want to see results within one standard deviation of the mean, that would mean getting an OFDM signal with an NRMSE power between 0.3414 and 0.6208.

Since the signals will tend to have a higher NRMSE Power, 0.6208 is the primary marker of success for this objective. In other words, other actors in the area may be scanning incoming signals and classifying them as clutter or something to be investigated further. They may be doing this in the time domain, frequency domain, or both. So if the NRMSE power with the reference clutter is below 0.6208 then it would likely be classified as clutter and if it is above it would

likely be investigated further by the opposing actors. This is how we will measure our signal's LPI capabilities.

Chapter 4

Machine Learning

The field of machine learning (ML) has been around since computers were widely used and research and application has been growing a lot in the last decade, especially since the breakthrough of Google's AlexNet in 2012 [26]. With it becoming widespread and easier to use, it is natural to want to find a way of using it for this task. Since ML has been heavily applied to classification, clutter classification was the clear choice of where to use it.

There are a variety of ML algorithms categories, which can be separated into supervised and unsupervised algorithms. Supervised means there is training data with the correct answers known that is used to calibrate the algorithm. Unsupervised algorithms are applied directly to the dataset without any training. Whether speaking about the training data or test data, the data has two parts: its features and its true labels. For training data, that true label is known, while for test data, it is not known but instead trying to be found. The feature can be anything that describes that point, or it can be a combination of other features.

Overfitting can be a large issue when creating almost any type of ML model. Overfitting is when a model fits the training data very well but does not generalize to data it has not seen yet. This usually occurs when a model becomes too complex and either fits any noise in the training data or there is not enough training data to accurately model the underlying probability distribution. There are a few ways to deal with this issue. We can stop building the model early,

in the case of decision trees or neural networks, we can remove nodes, or we can use some of the training data as a validation data set to estimate its generalizability.

4.1 Decision Trees

Decision trees are a "process of partitioning the attribute space into disjoint regions until each region contains records of the same class" [27], (here they use attribute instead of feature and records instead of data points). Sometimes the regions cannot be partitioned into regions where every point is the same class, but we settle for a case where each region is labeled with the majority's class and that majority is as dominant as possible. This algorithm, like the rest, is a nonparametric method meaning the type of probability distributions that the features and class labels follow does not need to be known.

The tree part of "decision tree" is made up of nodes and paths between them, as shown in Figure 4-1 from [27]. There are root nodes where the process begins, internal nodes where a feature is chosen to separate the data, and leaf nodes which contain the class label. The process of creating nodes is an iterative one in which a new node is made and then determined to be either a leaf or an internal node based. If the group all belongs to one class or the stopping condition is met, then it is a leaf node. If it is an internal node, then the proper feature must be chosen to separate the data at that point. The proper feature is determined by the information gain it can create.

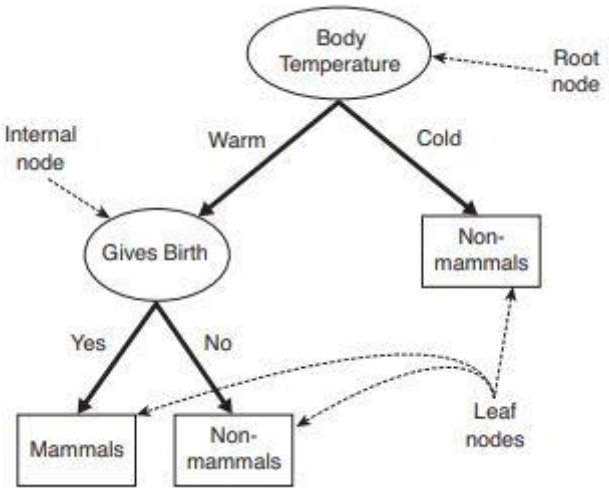


Figure 4-**1**: Example decision tree [27].

Purity can be measured by the Entropy, GINI, or Classification Error:

$$\text{Entropy} = -\sum_{i=1}^{N} \frac{n_i}{N} \log_2\left(\frac{n_i}{N}\right) \tag{4.1}$$

$$\text{GINI} = 1 - \sum_{i=1}^{N} \left(\frac{n_i}{N}\right)^2 \tag{4.2}$$

$$\text{Classification Error} = 1 - \max_i \frac{n_i}{N} \tag{4.3}$$

The goal is to maximize the purity so that the resulting partitions have a homogeneous set of class labels as possible. All three produce a similar ranking of attributes and the choice of which to use tends not to have a significant effect on the result. The Entropy is used the most often however since it can be used as, $I(\cdot)$, to find the Information Gain. which is then combined with the Split Information to create the Gain Ratio:

$$\text{Information Gain} = I(\text{parent}) - \sum_{j=1}^{M} \frac{n_j}{N} I(\text{new node}_j) \tag{4.4}$$

$$\text{Gain Ratio} = \frac{\text{Information Gain}}{\text{Split Ratio}} = \frac{\text{Information Gain}}{-\sum_{j=1}^{M} \frac{n_j}{N} \log_2(\frac{n_j}{N})} \tag{4.5}$$

where N is the total number of data points, M is the number of splits or new nodes, and n_j is the number of data points in each split.

This is done because using information gain alone can partition the data into groups that have few samples and lose generality to new data leading to overfitting. Therefore, the Split Information, a function of how many splits is made at that node based on the feature chosen, is used to discourage making too many splits. Using these measures of purity, information gain, and the number of splits made we can create new nodes of the decision tree until the stopping criterion is met.

4.2 Support Vector Machine

One reason the Support Vector Machine (SVM), which solves classification and regression problems, is used often is because it corresponds to a convex optimization problem. This means that the local solutions are also the global optimum ones [28]. However, it does not give posterior probabilities, but the Relevance Support Machine can be used for that need. The SVM is a binary classification method so when there are multiple classes the problem is reduced to a series of binary classification problems. We will not go into detail about how this combination is done as it is outside the scope of this thesis, but we will go through the math behind the two-class separable case. Then, we will briefly discuss its generalization to the non-separable case.

The formulation of an SVM problem with the assumption that the two classes are completely separated, meaning there exists some boundary that can separate the space into two homogeneous regions (shown in Figure 4-2 in red), begins with (4.5):

$$y(x) = w^T \phi(x) + b \tag{4.5}$$

where $y(x)$ is the output that corresponds to either class 1 or −1, x is the data point whose features are represented as a vector, $\phi()$ is a transformation, b is the bias, and w is the weight vector.

The margin is the distance between the distance between the decision boundary and the closest point of that margin's class as depicted in Figure 4-2. In order to have a maximum margin, we must be able to measure the margin's size with some distance function. The distance from point x_n to the decision boundary is (4.6):

$$\frac{t_n y(x_n)}{\|w\|} = \frac{t_n (w^T \phi(x) + b)}{\|w\|} \tag{4.6}$$

where t_n is the actual class of x_n, so to create a maximum margin, we have the problem (4.7):

$$\arg\max_{w,b} \frac{1}{\|w\|} \min_n t_n (w^T \phi(x) + b). \tag{4.7}$$

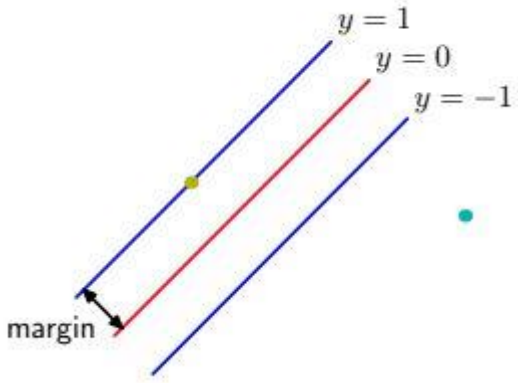


Figure 4-**2**: Visualization of SVM margin.

By creating a constraint where the closest point to the decision boundary has a distance of one (this is our starting support vector), replacing $\|w\|$ with $\|w\|^2$. which has the same argument of the minimum, and adding the factor $\frac{1}{2}$, we have a much easier optimization problem in (4.8):

$$\arg\min_{w,b} \frac{1}{2}\|w\|^2. \tag{4.8}$$

We solve the constrained optimization problem with the Lagrangian method where we add the non-negative Lagrangian multipliers a_n setting up (4.9):

$$L(w,b,a) = \frac{1}{2}\|w\|^2 - \sum_{n=1}^{N} a_n\{t_n(w^T\phi(x) + b) - 1\} \tag{4.9}$$

Solving this Lagrangian produces the constraints (4.10) and (4.11):

$$w = \sum_{n=1}^{N} a_n t_n \phi(x_n) \tag{4.10}$$

$$0 = \sum_{n=1}^{N} a_n t_n \, , \tag{4.11}$$

and substituting these constraints. we get the dual Lagrangian problem (4.12):

$$\tilde{L}(a) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m t_n t_m k(x_n, x_m) \tag{4.12}$$

In (4.12), $k(x_n, x_m)$ refers to a kernel function which is described later in this section. Since the Karush-Kuhn-Tucker (KKT) conditions hold according to [28], we know that either $a_n = 0$ or $t_0 y(x_n) = 1$. Points satisfying the latter case are the final support vectors.

Substituting (4.10) into (4.5) gives us the classifier (4.13):

$$y(x) = \sum_{n=1}^{N} a_n t_n k(x, x_n) + b \tag{4.13}$$

Once values of a_n are found the classifier (5.9) can be solved to find the values of b, which is averaged over all support vectors which make up the set, S, giving (4.14):

$$b = \frac{1}{N_S}\sum_{n\in S}[t_n - \sum_{m\in S} a_m t_m k(x_n, x_m)] \,. \tag{4.14}$$

In the case of non-separable data, slack variables are introduced. The slack variable is zero if correctly classified, greater than one if misclassified, and between zero and one if correctly classified but inside the margin. We also add C to control how soft the margin is, that is to say how heterogeneous we allow the regions to be.

This means instead of (4.8) we have (4.15) to optimize, and instead of the Lagrangian in (4.9) we have the one in (4.16):

$$\arg\min_{w,b} C\sum_{n=1}^{N}\xi_n + \frac{1}{2}\|w\|^2. \tag{4.15}$$

$$L(w,b,a) = \frac{1}{2}\|w\|^2 + C\sum_{n=1}^{N}\xi_n - \sum_{n=1}^{N} a_n\{t_n y(x_n) - 1 + \xi_n\} - \sum_{n=1}^{N}\mu_n\xi_n \tag{4.16}$$

Where in (4.9) both a_n and μ_n are Lagrangian multipliers, but here a_n is a non-negative number less than or equal to C. This difference in the Lagrangian multipliers is the primary difference between the separable and non-separable cases. After solving the Lagrangian though, we do still get the same dual Lagrangian and (4.10) is still a result so the classifier (4.13) stays the same.

The dual Lagrangian (5.10) is a quadratic programming problem that can be difficult to solve, but have the benefit of having any local maximum be the global one as well. A quadratic programming problem can be solved using multiple methods, but one very popular one is the *sequential minimal optimization* (SMO) [29] approach. Once all a_n are found, they will either equal zero and the corresponding data point is discarded, or it is greater than zero and satisfies (4.17), meaning it is a support vector.

$$t_n y(x_n) = 1 - \xi_n \tag{4.17}$$

Finally, with information we now have *b* can be found using (4.18):

$$b = \frac{1}{N_M}\sum_{n\in M}[t_n - \sum_{m\in S} a_m t_m k(x_n, x_m)] \, . \tag{4.18}$$

where *M* is the set of data points where the corresponding a_n is between 0 and *C*, or in other words, is the set of support vectors.

There are a variety of kernels that can be used in an SVM. These kernels map the data to a higher dimensional space where it may be easier to separate. Kernels can do this without explicitly transforming the data which saves computational time. Common kernels include the Gaussian, linear, and sigmoid kernels which have certain cases they are the best for. Their equations are shown below:

$$\text{Gaussian: } k(x_i, x_j) = e^{-(\|x_i - x_j\|_2^{\,2}/\alpha)}, \text{ for } \alpha > 0 \tag{4.19}$$

$$\text{Linear: } k(x_i, x_j) = x_i^T x_j \tag{4.20}$$

$$\text{Sigmoid: } k(x_i, x_j) = \tanh(\alpha x_i^T x_j + \beta) \tag{4.21}$$

4.3 Discriminant Analysis

In the case of the Linear Discriminant Analysis (LDA) approach, we assume that the data can be separated by a linear boundary. This can be described by (4.22) [28]:

$$y(x) = w^T x + w_0 \, , \tag{4.22}$$

where x is the data point and if $y(x)$ is positive it is one class and the other class if it is negative. In (4.4) w is the weight vector while w_0 is the bias term. This two-class classifier can be generalized to K classes by using a matrix of K of these equations as shown in (4.23):

$$y_k(x) = {w_k}^T x + w_{0,k} \, . \tag{4.23}$$

In this case, the class corresponding to the largest element in $\boldsymbol{y}$ is the class x is classified as. If we add a dummy variable 1 to x, it allows us to combine w_k and $w_{0,k}$ into one weight vector, which then become the weight matrix W, with the equation (4.24):

$$y(x) = W^T x. \tag{4.24}$$

The problem of course comes down to finding the matrix W which is a $N \times C$ matrix where N is the length of x_n and C is the number of classes.

There are a few ways to find the matrix W. One involves using the error function (4.25):

$$E(W) = \frac{1}{2}\text{Trace}\{(XW - T)^T(XW - T)\} \, , \tag{4.25}$$

where T is a matrix with rows t_n , of length C, corresponding to x_n and where each element $t_{n,i}$ is 1 if x_n is in class i and otherwise 0. When its derivative is set to zero and we rearrange terms, we get (4.26):

$$W = (X^T X)^{-1} X^T T \, . \tag{4.26}$$

Alternatively, we could use Fisher's Linear Discriminant which reduces the dimensionality and attempts to maximize the between class variance while minimizing the within class variance. This approach leads to the ratio (4.27):

$$J(w) = \frac{w^T S_B w}{w^T S_w w} \, , \tag{4.27}$$

which when differentiated with respect to *w* gives us (4.28):

$$w \propto S_w^{-1}(m_2 - m_1)\,, \tag{4.28}$$

where m_i is the mean vector of each class. Here only direction matters, any magnitude can be used. Once *W* is found using either method it is used in (4.24) to project the points onto a 1-D space as illustrated by Figure 4-3, once this is done the final step is to choose a threshold is chosen that minimizes the error.

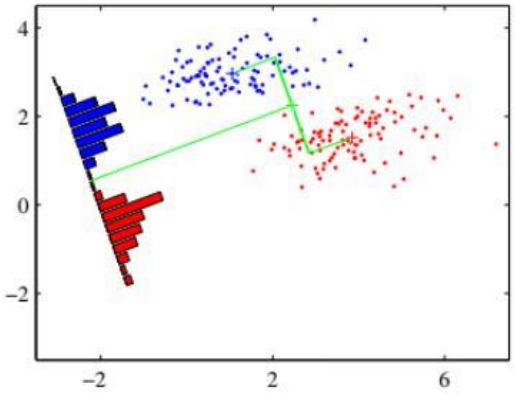


Figure 4-**3**: Visualization of data projection using Fisher's linear discriminant.

To understand Quadratic Discriminant Analysis (QDA), we must look at Discriminant Analysis from a probabilistic generative standpoint. Here, we show the model as (4.29-31) [28]:

$$p(C_i|x) = \frac{p(x|C_i)p(C_i)}{\sum_j p(x|C_j)p(C_j)} = \frac{\exp\,(a_i(x))}{\sum_j \exp\,(a_j(x))}\,, \tag{4.29}$$

$$a_i(x) = \ln\,(p(x|C_i)p(C_i))\,, \tag{4.30}$$

$$p(x|C_i) = \frac{1}{(2\pi)^{\frac{D}{2}}}\frac{1}{\sqrt{|\Sigma|}}\exp\,(-\frac{1}{2}(x-\mu_i)^T\Sigma^{-1}(x-\mu_i))\,. \tag{4.31}$$

Note that in (4.31), *D* is the dimensionality of the problem or the number of features, Σ is the covariance matrix of the classes, and μ_i is the mean vector of each class. Here we see that we want to find the class that is most likely given datapoint *x* based on the rules of conditional

probabilities. This comes down to a maximization problem of (4.32) across *i* [30]. As stated in both [30] and [28], when we assume the covariance matrix is the same for all classes, the quadratic term "$-\frac{1}{2}x^T\Sigma^{-1}x$" is cancelled and we are left with a linear term and LDA otherwise the model is QDA. QDA can create quadratic boundaries in the feature space while LDA can only create linear ones, so ideally QDA will perform better.

$$-\frac{1}{2}(x-\mu_i)^T\Sigma^{-1}(x-\mu_i)+\ln(p(C_i))=x^T\Sigma^{-1}\mu_i-\frac{1}{2}\mu_i^T\Sigma^{-1}\mu_i-\frac{1}{2}x^T\Sigma^{-1}x+\ln(p(C_i)) \quad (4.32)$$

4.4 K-Nearest Neighbors

The K-Nearest Neighbors (KNN) approach is perhaps the simplest method of machine learning. Before classification an integer, *K*, is chosen. The new data point is classified as the same class as the plurality of class labels found in the *K* nearest training data points in that feature space. With this method, the performance relies heavily on having an accurate representation of the underlying phenomenon be given by the training data set. KNN can be augmented by using a kernel method to transform the data to a higher dimensional space where they will be classified better.

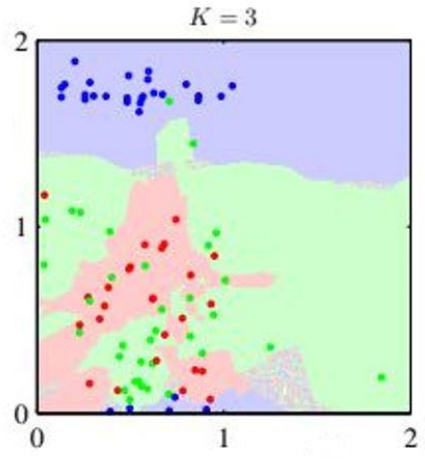


Figure 4-**4**: Example of KNN.

An example is shown in Figure 4-4 where data points are shown as dots and their colors represent their class. The shaded regions are what a new data point would be classified as based on *K=3*, that is the majority vote winner at that point. In the center bottom, we can see a region where there is no winner; here a class could be chosen by random or using some other method. Although KNN is simple, it requires all training data to be kept for classification unlike other approaches; therefore, classification time can be longer as distances need to be computed to all training data points and more memory is needed.

In order to make KNN more effective there are variations. However, most of which change the distance metric and the distance weight. The distance metric is how the distance is measured and in the basic case is Euclidian. In the cosine distance metric, the data points are treated like vectors and the cosine of the angle between them is taken, where one minus that cosine is the cosine distance. There is also the cubic metric distance that is equivalent to the Minkowski distance where $p = 3$. The distance weight concerns how important each of the k "votes" are and in most cases it is equal. The votes importance can also fall off with the inverse of the distance, or as is used in this paper the inverse of the distance squared. This means within the k closest points to the data point being classified, far ones have less input in the classification.

Chapter 5

NSGA-II and Other Heuristics Background

Genetic Algorithms (GAs) are a part of a larger set of heuristics called Evolutionary Algorithms that take inspiration from evolution and natural selection. Natural selection is defined by Merriam-Webster as, "a natural process that results in the survival and reproductive success of individuals or groups best adjusted to their environment and that leads to the perpetuation of genetic qualities best suited to that particular environment". In the case of GAs the "particular environment" is described as some set of functions that we want to maximize or minimize. The "individuals" are defined by their "chromosome" which is a set of numbers that are the input to those functions. This method is often used in multiple objective optimization (two or three objectives) or many objective optimization (four or more objectives) when the search space is too large and other methods are ineffective or too slow.

GAs can be described as looping between two main parts: measuring the best of a population and creating new offspring from the best of that population. The difference between various GAs is in how the "best" individuals that move to the next generation are defined and how the offspring are generated.

In 2002, Debs developed the Nondominated Sorting Genetic Algorithm-II (NSGA-II) [31] which was an innovation from his 1995 NSGA-I [32], and it has been widely used since. The NSGA-III was developed in 2014 [33], but does not always perform better and will be discussed in a later section. For NSGA-II, the population is separated into fronts; in each front every sample is dominated by the same number of other samples. For one sample, $\mathbf{x_1}$, to dominate another, $\mathbf{x_2}$, it must be better in one of **M** objective and not worse in any, as described by (5.1-2):

$$\forall\, i \in [0, M] \quad \{f_i^1(x_1) \leq f_i^2(x_2)\} \tag{5.1}$$

and,

$$\exists\, i \in [0, M] \quad \{f_i^1(x_1) < f_i^2(x_2)\}. \tag{5.2}$$

The best front then, called the pareto front, is the one where the samples are not dominated by any other. In the NSGA-II, both the parent and offspring samples are used when sorting into fronts and creating the next parent set. This is done to ensure elitism, that is if one of the best samples is in the parent generation, it can still move to the next generation. The original pseudocode of the algorithm from [5] is provided in Figure 5-1, but a summary of how it works is provided below.

To create the next parent population, a predefined number of the total population (previous parent and offspring) is selected by including the best front, then the next front, and the next until that predefined number is met. If an entire front cannot be added without exceeding that number, then the crowded comparison operator is used to find the remaining samples. This operator picks samples that are the most different from the ones already chosen using a crowding function. Then the parents go through a binary tournament, two are chosen randomly and the best is mutated, to create the next generation. Mutation then occurs first by crossover which is when two selected parents swap sections of their DNA. Mutation then happens where a random place on the DNA is altered. Once this is done the offspring can be combined with the parent population to once again be sorted into fronts.

```
fast-non-dominated-sort(P)
for each p ∈ P
   S_p = ∅
   n_p = 0
   for each q ∈ P
      if (p ≺ q) then                    If p dominates q
         S_p = S_p ∪ {q}                 Add q to the set of solutions dominated by p
      else if (q ≺ p) then
         n_p = n_p + 1                   Increment the domination counter of p
   if n_p = 0 then                       p belongs to the first front
      p_rank = 1
      F_1 = F_1 ∪ {p}
i = 1                                    Initialize the front counter
while F_i ≠ ∅
   Q = ∅                                 Used to store the members of the next front
   for each p ∈ F_i
      for each q ∈ S_p
         n_q = n_q − 1
         if n_q = 0 then                 q belongs to the next front
            q_rank = i + 1
            Q = Q ∪ {q}
   i = i + 1
   F_i = Q
```

Figure 5-**1**: Pseudocode of NSGA-II.

One dilemma in the NSGA-II is that sorting is done by determining the amount of other individuals that dominate a certain individual. This process does not allow for the prioritization of one objective over another which can be very important in many applications. One sample can only dominate another or not dominate, “dominating more” is ill-defined. [34] in 2009 and a group that includes the creator of the NSGA-II, Kalynamoy Debs, in 2021 [35] proposed solutions to this problem but do not create degrees of priority only levels. That is to say, one cannot choose objective A is 50% more important than objective B; only that is more important. This may be a desired quality to have because this can allow a user to control how much objective B suffers on behalf of objective A’s improvement.

Using an additional change to a recent modification to the NSGA-II, this problem can be fixed, however. In 2020 researchers, proposed a modification to the NSGA-II algorithm to address dominant resistant solutions. Dominant resistant solutions are ones that perform so well in one objective that they can survive and reproduce in the algorithm even if they are deficient in other areas. As an example is when the problem has three objectives and a solution has a score of (.1, .1, 100), the solution will be hard to dominate because it scores well in two objectives and so it will persist even though it does poorly in the third objective. What they proposed is an adjustment to the objective functions where the new objective function includes the mean of all the objective functions so poor performance in one category is carried to all:

$$f_i^{\text{New}}(x) = (1-\alpha) f_i^{Old}(x) + \alpha \frac{1}{M} \sum_{j=1}^{N} f(x)_j^{\text{Old}} \,. \tag{5.3}$$

$$f_i^{\text{New}}(x) = (1-\alpha) \gamma_i f_i^{\text{Old}}(x) + \alpha \frac{1}{M} \sum_{j=1}^{N} \gamma_j f(x)_j^{\text{Old}} \,. \tag{5.4}$$

What we propose then is using the prioritization coefficients on the old objectives as shown in (5.4). With this addition, since every old objective affects all new objectives, prioritization on that old objective can affect all new objectives as well. Previously the addition of a prioritization weight would have no effect on sorting, since increasing the gap in one objective does not affect if a solution dominates any another solution. For example, if we have two objectives and two solutions: solution A gives the results (5, 10) and solution B gives the results (8, 9) then neither solution dominates the other. Giving the second objective a priority weight of ten makes the results (5, 100) and (8, 90) which does not affect the domination. However, trying this using (5.4) with α equal to 0.5, we can get the results (28.75, 76.25) and (23.5, 69.5), and now solution B dominates solution A. If the second objective is only twice as important, then (5.4) gives (8.75, 16.25) and (10.5, 15.5). In this case, the domination situation again does not

change because objective 2's priority is not enough to overcome the large disparity in objective 1. This shows that this method gives the user not only the ability to prioritize but the ability to control the degree of that prioritization.

5.1 NSGA-III and Simulated Annealing

There were two other methods we wanted to use to compare the NSGA-II too. The NSGA-III, which was designed specifically to handle many objective problems, and Simulated Annealing (SA). SA is a Monte-Carlo and metallurgy based heuristic designed in the 1970's and 1980's for various optimization problems.

The main difference between NSGA-II and NSGA-III is that in the NSGA-III, there are reference points in the objective space and the algorithm tries to create results that lie near those reference points. This is done to ensure diversity of solutions and probe more options, which NSGA-II also does but with its crowding function instead, which has shown to often be less effective and more computationally expensive. The reference points can be auto generated, or user created. For an *M*-objective problems with a parameter *p* chosen representing how finely the space is to be searched is set, *H* points are needed [33] calculated by (5.5):

$$H = \binom{M+p-1}{p}. \tag{5.5}$$

In [36], it was shown that the NSGA-III does not always outperform the NSGA-II. In fact in cases where the objectives were maximized, and the solution set was spread out, the NSGA-II performed better. It was also shown that the number of objectives did not consistently predict which would perform better.

SA is a relatively simple process, starting with a random solution *A* and a potential replacement solution *B*. If *B* is better than *A* based off a user defined score then *B* replaces *A*. However, if *A* is better than *B*, *B* might still replace *A* with some probability *P* based on the

current “temperature", *T*. With every iteration the “temperature” cools and the probability *P* is reduced leading to a steady-state solution. According to [37] the probability *P* is decided according to (5.6):

$$P = e^{(B+A)/T} . \tag{5.6}$$

The algorithm can be varied to in order to make it more efficient, usually by choosing *B* to be something that is less than completely random, such as creating it based off of *A*.

Chapter 6

Clutter Classification Results

In our paper [17],, we looked at how machine learning could be used to decipher the underlying distribution of clutter. The simulated clutter was a random vector from one of nine random distributions chosen and had noise added to some of the elements to simulate interference. Since the assumed method of gathering this data would be through OFDM signals, it may not be the case that all subcarriers have this interference added, therefore in the simulation, only 80% of the subcarriers/elements had interference added to them. Since the data vectors were a combination of a continuous distribution and a mixed distribution, the usual Goodness-of-Fit models may not work as well. This is the reason machine learning is useful.

Here we do something similar but with some variation to see if the results remain constant or improve. We still use statistical characteristics for classification features, specifically the first four moments: mean, variance, skewness, and kurtosis. In the original paper, the interference was modeled as a constant, but in the updated results the Gaussian distribution used to create the noise and was modeled according to (6.1):

$$f(I) = N(\mu_m, 0.2)/\sqrt{\text{CIR}} \text{ if } \boldsymbol{u} \geq 0.2 \text{ ; } 0 \text{ otherwise,} \tag{6.1}$$

where $\boldsymbol{u}$ is a uniformly distributed variable from [0 1], μ_m is the mean for the distribution it is being added to, and CIR is the Clutter-to-Interference Ratio in numeric units. This is done to make the interference more realistic. Another difference is that more and different distributions were used; fifteen distributions were used instead of nine. The Weibull, Rayleigh, and Log-Normal distributions were still used, but with new parameters chosen to make the distributions

closer to one another in the same family. The parameters are shown in Table 2. The purpose of this is to test how accurate the algorithms can be when the distributions are closer to one another.

There were 18,000 random vectors created 100 instances of each of the 15 distributions at 12 different CIR levels. The CIR ranged from 0 dB to 10 dB with increments of 1 and also included ∞ dB (a case with no interference). As was done in the previous paper, two other features were included: a marker for which of the distribution families it came from and a marker for the CIR range the vector has. The three CIR groups were: [0 dB 3 dB], [4 dB 7 dB], and [8 dB ∞ dB]. However, in a real scenario, both of these features will themselves need to be determined, so we will also test to see if when instead of these additional features being hand set, they are themselves determined by machine learning themselves and they are still useful. The description of the fifteen distributions used are below in Table 6-1 and the equations for the PDFs of Weibull, Rayleigh, and Log-Normal distributions respectively are in (6.2-4):

$$W(\lambda,k)=\frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1}e^{-\left(x/\lambda\right)^{k}};x\geq 0, \tag{6.2}$$

$$R(\sigma)=\frac{x}{\sigma^{2}}e^{-\left(x/\sigma\sqrt{2}\right)^{2}} \tag{6.3}$$

$$L(\mu,\sigma)=\frac{1}{x\sigma^{2}\sqrt{2\pi}}e^{-\left(\left(\ln(x)-\mu\right)/\sigma^{2}\sqrt{2}\right)^{2}}, \tag{6.4}$$

Table 6-1: List of Distributions

Weibull	**Rayleigh**	**Log-Normal**
W(1.25,1)	R(1)	L(0.35,0.2)
W(1.5,1)	R(1.05)	L(0.4,0.2)
W(1.25,1.2)	R(1)	L(0.35,0.25)
W(1.5,1.2)	R(1.15)	L(0.4,0.25)
W(1.5,2)	R(2)	L(0.374, 0.225)

6.1 Testing Features and Algorithms

In the following. four tables the clutter classification accuracy results, in percentages, of a variety of machine learning methods are shown with either one of, both, or neither of the additional features. It should be noted LDA and QDA could not be used on finding the distribution family because the class names were constant.

Table 6-**2**: Four Moments Only

ML Algorithm	Acc.	ML Algorithm	Acc.	ML Algorithm	Acc.
KNN-1	96.2	**KNN-10 Cubic**	96.7	**Fine Tree**	75.4
KNN-10	96.8	**KNN-10 Weighted**	96.9	**Medium Tree**	52.4
KNN-100 Coarse	93.3	**LDA**	60.0	**Coarse Tree**	31.6
KNN-10 Cosine	92.1	**QDA**	80.4		
SVM 1v1	79.1	**SVM 1vAll**	32.5		

Table 6-**3**: Four Moments and Distribution Family

ML Algorithm	Acc.	ML Algorithm	Acc.	ML Algorithm	Acc.
KNN-1	98.8	**KNN-100 Cosine**	97.7	**Fine Tree**	80.8
KNN-10	98.8	**KNN-10 Cubic**	98.7	**Medium Tree**	61.6
KNN-100 Coarse	95.8	**KNN-10 Weighted**	99.0	**Coarse Tree**	30.7
SVM 1v1	85.3	**SVM 1vAll**	58.6		

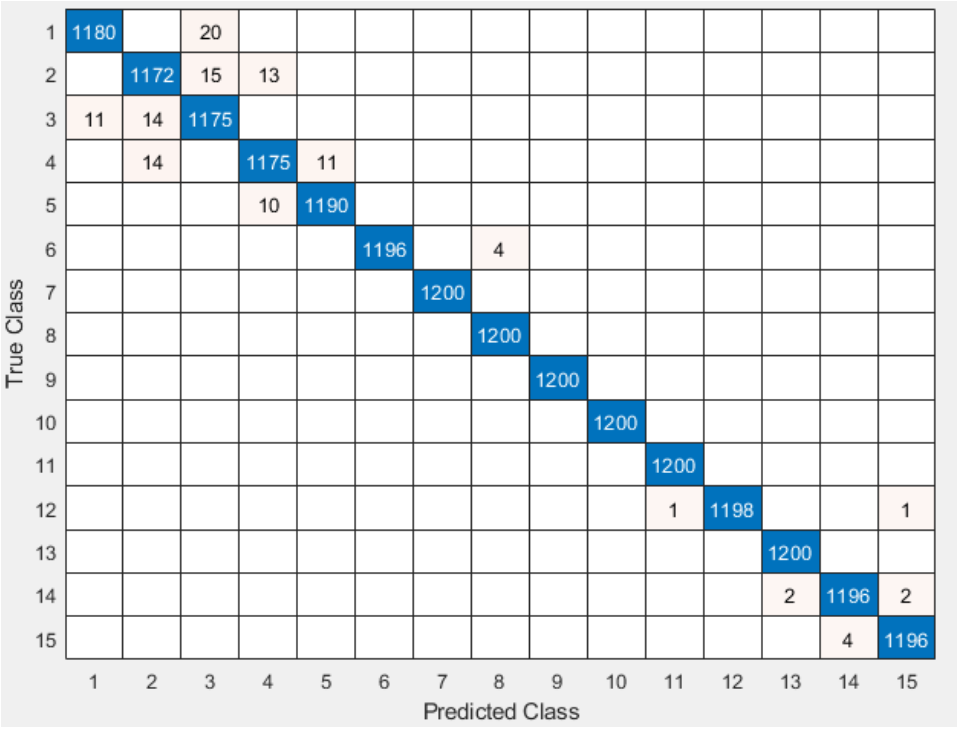
Table 6-**4**: Four Moments and CIR Levels

ML Algorithm	Acc.	ML Algorithm	Acc.	ML Algorithm	Acc.
KNN-1	96.6	**KNN-10 Cubic**	97.1	**Fine Tree**	76.0
KNN-10	97.2	**KNN-10 Weighted**	97.3	**Medium Tree**	52.4
KNN-100 Coarse	94.3	**LDA**	61.6	**Coarse Tree**	31.6
KNN-100 Cosine	96.5	**QDA**	83.6		
SVM 1v1	80.9	**SVM 1vAll**	43.8		

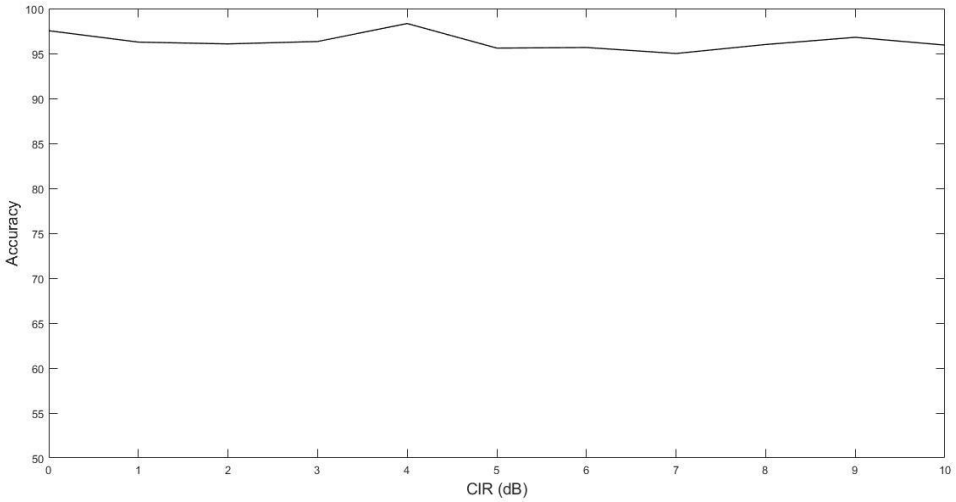
Table 6-**5**: Four Moments and Both CIR Levels and Distribution Family

ML Algorithm	Acc.	ML Algorithm	Acc.	ML Algorithm	Acc.
KNN-1	99.1	**KNN-100 Cosine**	99.0	**Fine Tree**	81.4
KNN-10	99.1	**KNN-10 Cubic**	99.0	**Medium Tree**	61.7
KNN-100 Coarse	96.7	**KNN-10 Weighted**	99.3	**Coarse Tree**	30.7
SVM 1v1	87.9	**SVM 1vAll**	58.6		

We see here that only the KNN method was able to have accuracy above 90% with several variations of it achieving 99% or better. The KNN-10 Weighted worked the best with 99.3% accuracy when the two additional features were used. At least one member of the other three method families were able to reach 80%. As for the additional features, they did seem to be helpful, although the distribution family was more helpful the two together performed the best. The confusion matrix is shown below in Figure 6-1a. Here classes 1-5 are Rayleigh distributions, classes 6-10 are the Weibull distributions, and classes 11-15 are the Log-Normal distributions.



(a)



(b)

Figure 6-1: (a) Weighted KNN confusion matrix, (b) CIR vs. accuracy.

From Figure 6-1(a), one can see that the algorithm had the most difficulty with the Rayleigh distribution and was very accurate with the Weibull distribution. We can also see that when there were misclassifications, it was within the same distribuion family. This is good because this way, the OFDM signal created based on a misclassification will still be similar to the clutter in the area. From Figure 6-1(b), we can see how the accuracy changed as CIR increased, this was tested on a completly different set of 18,000 vectors with the exact same parameters as the previous one. The accuracy remains stable with the maximum of 99.67% occuring at 3 places, 0 dB, 9 dB, and 10 dB. At ∞ dB, the accuracy was 100% which makes sense since the lack of interference makes the task easier. The low point of 98.53% occurred at 4 dB.

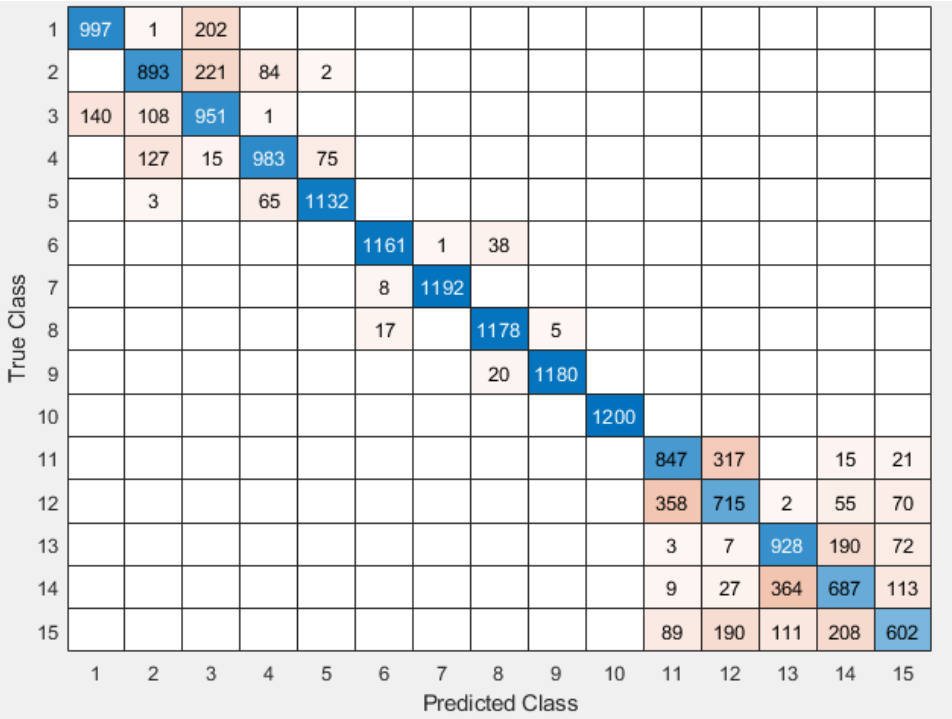


Figure 6-2: Fine Tree confusion matrix.

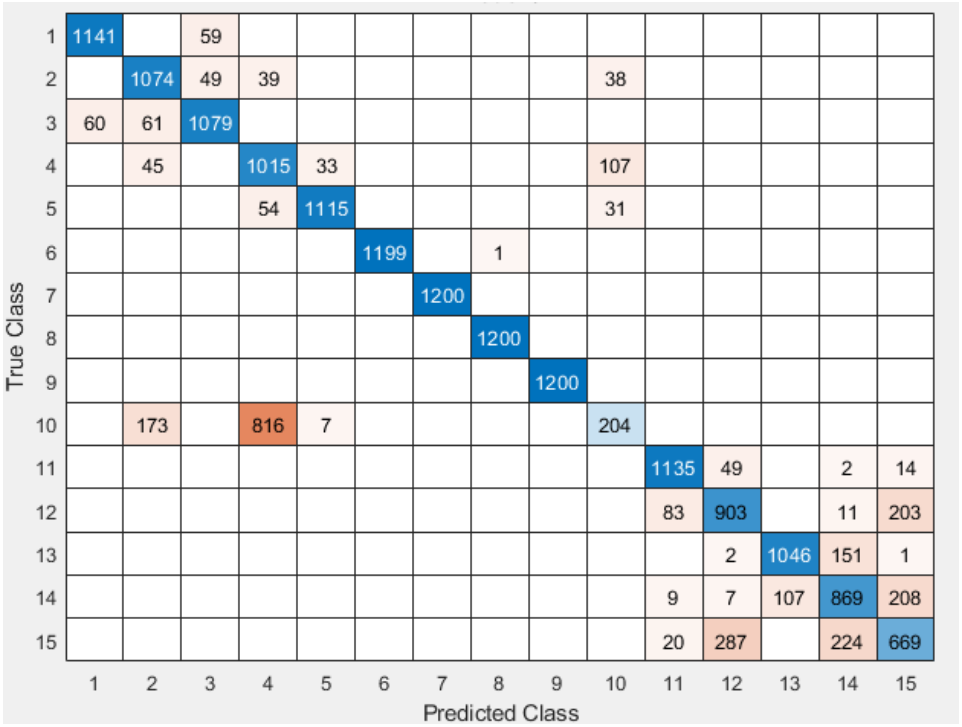


Figure 6-3: QDA confusion matrix.

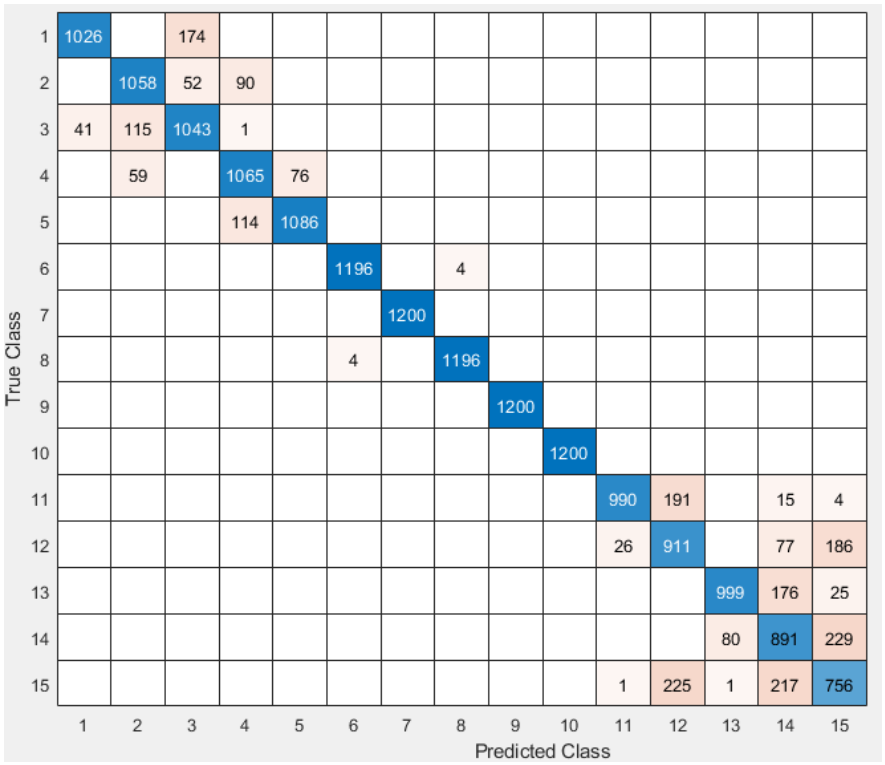


Figure 6-4: SVM One vs. One confusion matrix.

6.2 Testing Final Combination

Here, we want to see which method is the best at determing the distribution family and the CIR level. We can see from Table 6-6 that while most methods performed well, KNN was again the best. KNN-Weighted had one less misclassification than KNN-Cubic, so it was used in the final test. It was also the best and therefore used to determine the CIR level.

Table 6-**6**: Predicting Distribution Family

ML Algorithm	Acc.	ML Algorithm	Acc.	ML Algorithm	Acc.
KNN-1	97.5	**KNN-10 Cubic**	98.1	**Fine Tree**	96.7
KNN-10	98.0	**KNN-10 Weighted**	98.1	**Medium Tree**	94.7
KNN-100 Coarse	96.7	**LDA**	92.5	**Coarse Tree**	93.1
KNN-10 Cosine	96.3	**QDA**	93.0		
SVM 1v1	93.9	**SVM 1vAll**	93.4		

Table 6-**7**: Predicting CIR Level

ML Algorithm	Acc.	ML Algorithm	Acc.	ML Algorithm	Acc.
KNN-1	99.5	**KNN-10 Cubic**	99.5	**Fine Tree**	97.6
KNN-10	99.5	**KNN-10 Weighted**	99.6	**Medium Tree**	87.3
KNN-100 Coarse	98.6	**LDA**	75.8	**Coarse Tree**	76.0
KNN-10 Cosine	98.7	**QDA**	74.4		
SVM 1v1	79.5	**SVM 1vAll**	70.3		

In the final tes,t KNN-Weighted was used in all three parts: to determine the distribution family from the first four moments, to determine the CIR level from the first four moments, and to determine the final distribution from all six features. It performed with a 97.1% accuracy. While it is lower than the 99.3% achieved when the additional features were hand set, it is higher than any method could achieve with the four moments alone, which was 96.8%. The Accuracy vs. CIR was also tested, the lowest accuray was 96% at 6 dB and with no interference, it was 98.7%. The Accuracy vs. CIR of KNN-Weighted with the four moments alone is pictured in Figure 6-5 and is quite similar with a lower minimum of 95.67%. Therefore, these two extra

features can be said to give a slight edge and that the KNN-Weighted method is the best classifier for this method of clutter classification.

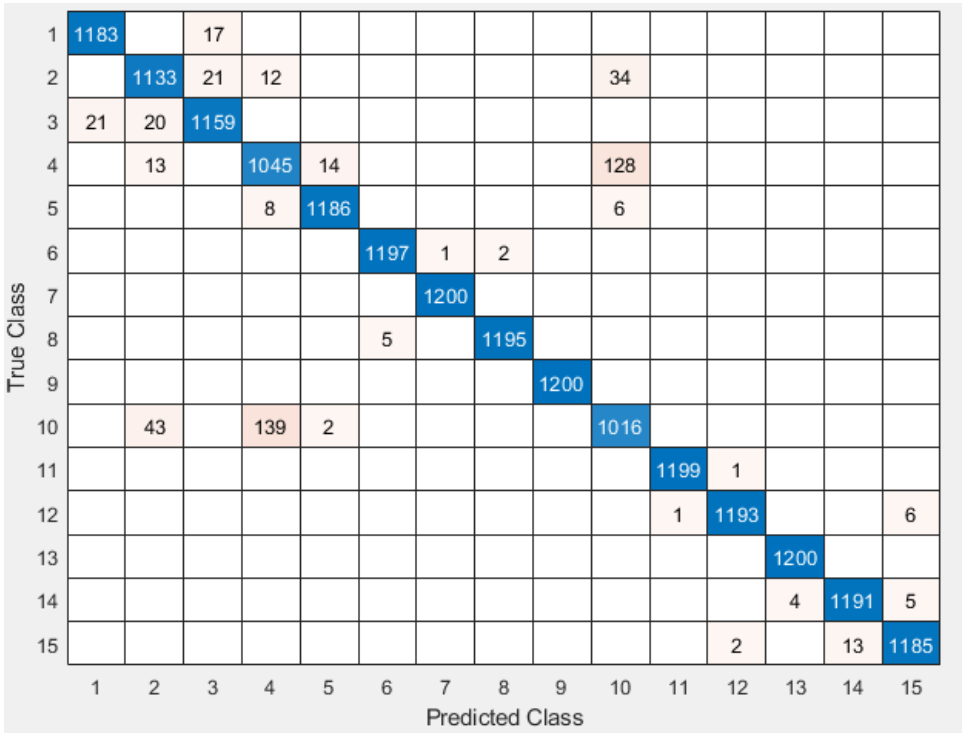


Figure 6-5: Final classification confusion matrix.

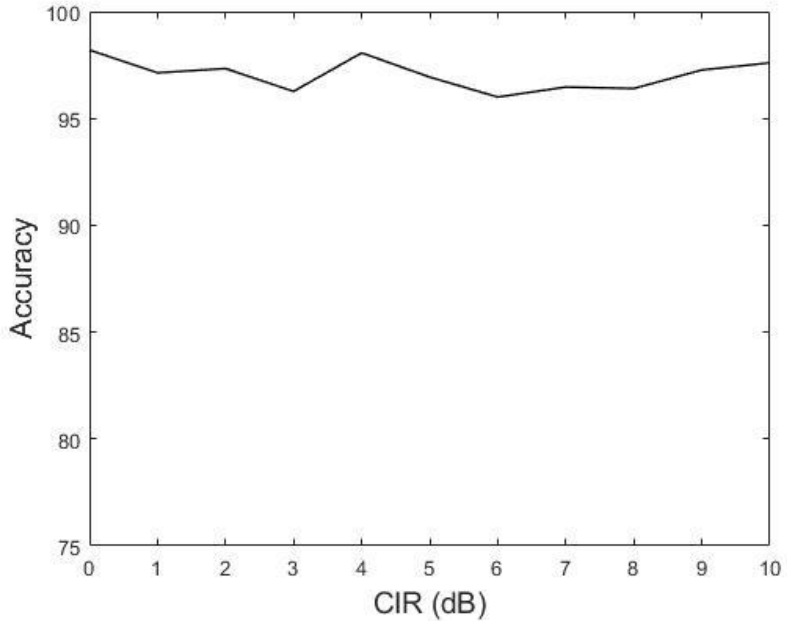


Figure 6-6: CIR vs. Accuracy of final classification.

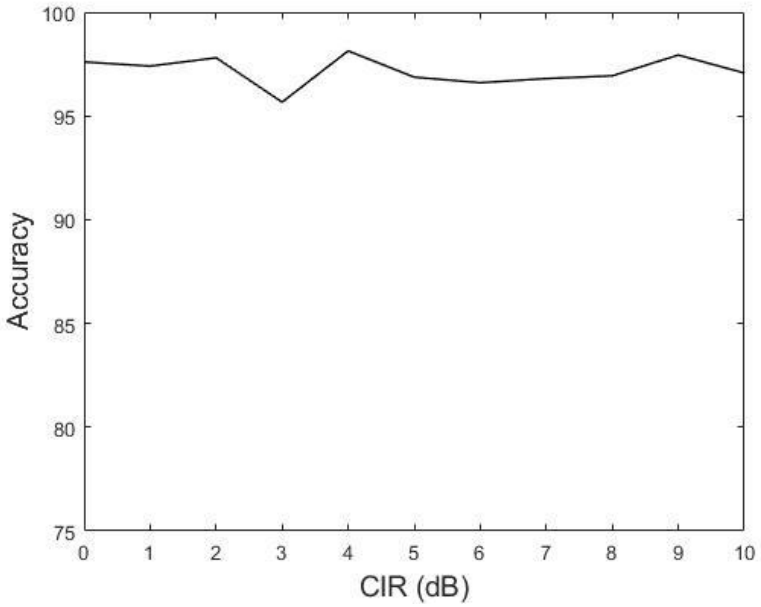


Figure 6-7: CIR vs. Accuracy of KNN-Weighted four moments alone.

6.3 Goodness of Fit

The Goodness-of-Fit in MATLAB was tested to see how well it could choose which two vectors came from the same family. At 1 dB, CIR a random vector of each vector was measured using the NRMSE to a random vector from each of the other 14. Only twice did the lowest NRMSE score come from a vector from the same family or even the second closest family, when testing the L(.35,.2) and L(.4,.2) distributions. In fact, it predicted one of those two in every case. Therefore, we can see comparing random vectors using this method is not very effective compared to machine learning.

Chapter 7

Optimization Results

7.1 Optimization Results

Here, we display and discuss the results of optimizing an OFDM signal across five objectives, or a subset of them. Those five are the probability of detection, PSLR, PAPR, BER and NRMSE power. The probability of detection and NRMSE power were measured at 5 dB SNR. In this case we are using QPSK, but on only a tenth of the 1024 of the subcarrier's phases, so that the rest can be optimized for the other objectives, and especially the NRMSE power with respect to the clutter model. This does of course of the drawback of decreasing throughput by a factor of 10.

In order to test the NSGA-II, we first wanted to look at what occurred over 1000 generation of its implementation with each generation having a population of 200. The three important parameters were α which determined to what extent the final objectives depended on its specific original objective compared to the mean of the original objectives, η_c, and η_m which were the probabilities of crossover and pointwise mutation respectively. α ranged from 0 to 1, 0 meaning the final objective is the same as the original, to 1 where all final objectives are replaced with the mean objective functions. The probabilities of crossover and mutation range from 0 to 100 percent.

In the following results, α was chosen as 0.25, which was determined to be low enough to strike a balance between being low enough to allow for effective prioritization but not too low as to completely override the importance of the original objectives. Both η_c and η_m were set to 25% to start. The results are shown below in Figures 7-1 to 7.5 for three different use cases

described in Table 7-1. It also includes the baseline numbers from 200 randomly generated signals with no optimization.

Table 7-1: Priority Weights and Baseline

	Prob. Of Det.	**PAPR**	**PSLR**	**BER**	**NRMSE Power**
Radar Focused	20	15	15	0	0
Comms. Focused	0	0	0	10	40
All Equal	10	10	10	10	10
Baseline	94.05%	9.46 dB	11.70 dB	3.52%	.87

In all three cases, there are a total of 50 priority "points" that have the same meaning as γ_i in (5.4). The All Equal Case is rather self-explanatory, here we want to maximize all objectives equally. In the Radar Case, where the user wants to create the best radar signal possible, the probability of detection is the most important, while the PAPR and PSLR are not as important and BER and NRMSE Power are not considered at all. Finally, in the Communications Case, there is a high priority for covertness, therefore NRME Power is prioritized heavily and BER is also to be optimized.

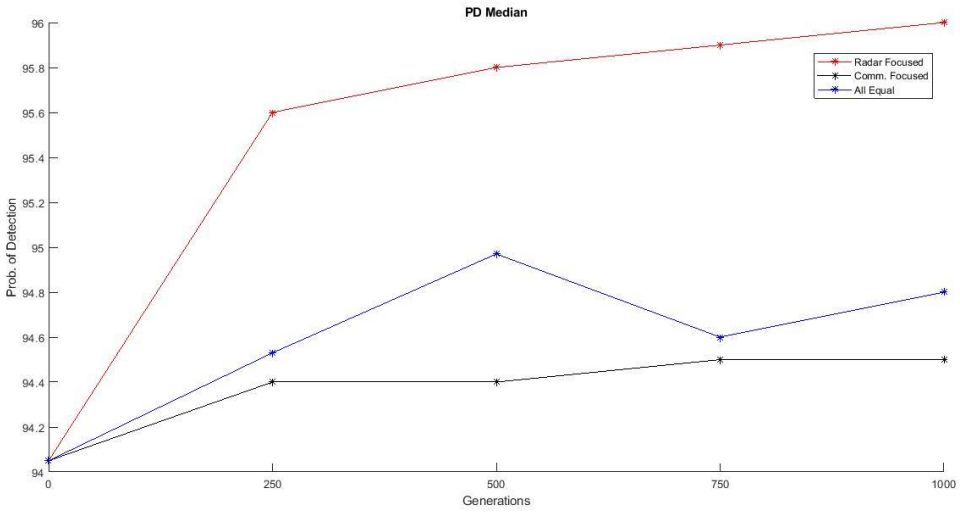


Figure 7-1: Probability of Detection results after 1000 generations.

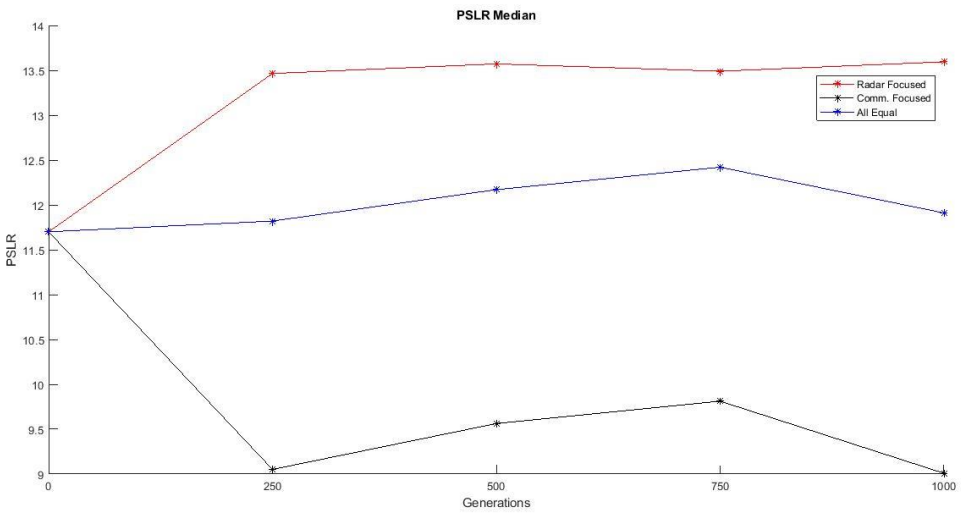


Figure 7-2: PSLR results after 1000 generations.

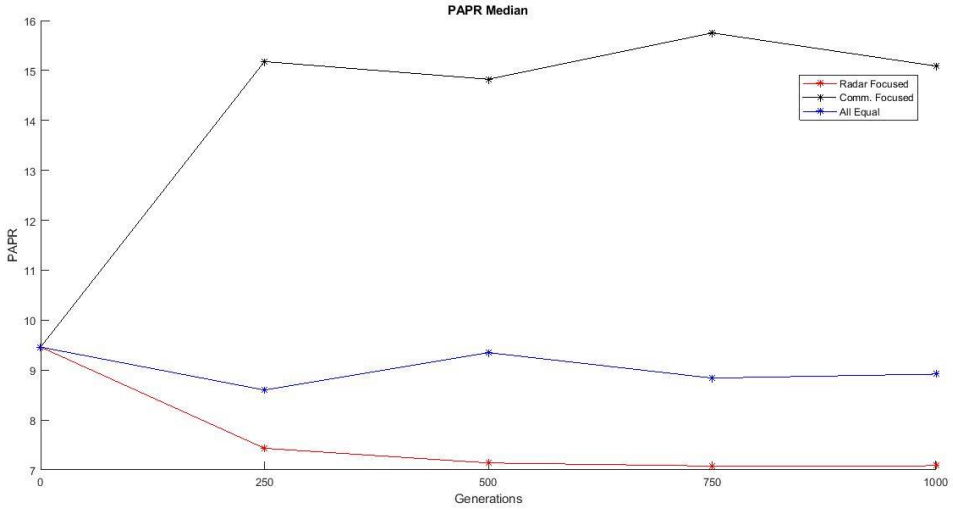


Figure 7-3: PAPR results after 1000 generations.

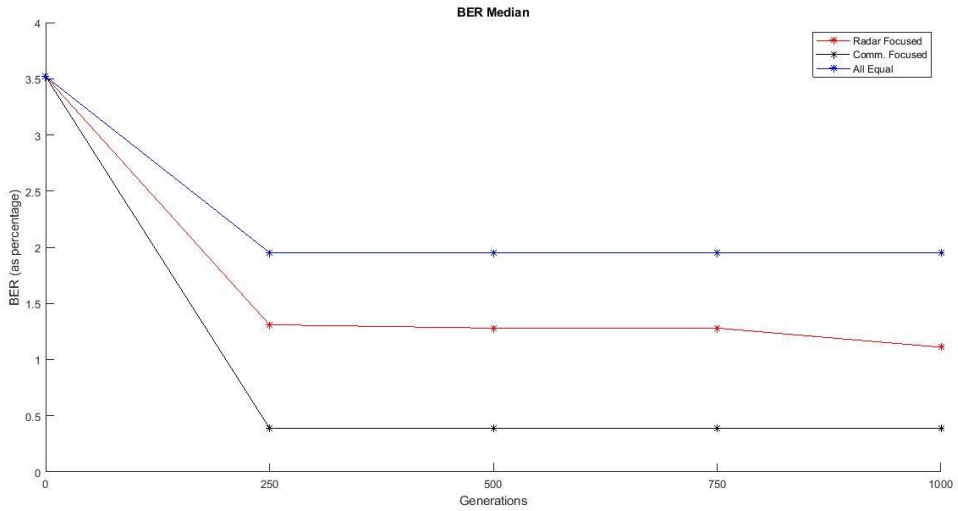


Figure 7-4: BER results after 1000 generations.

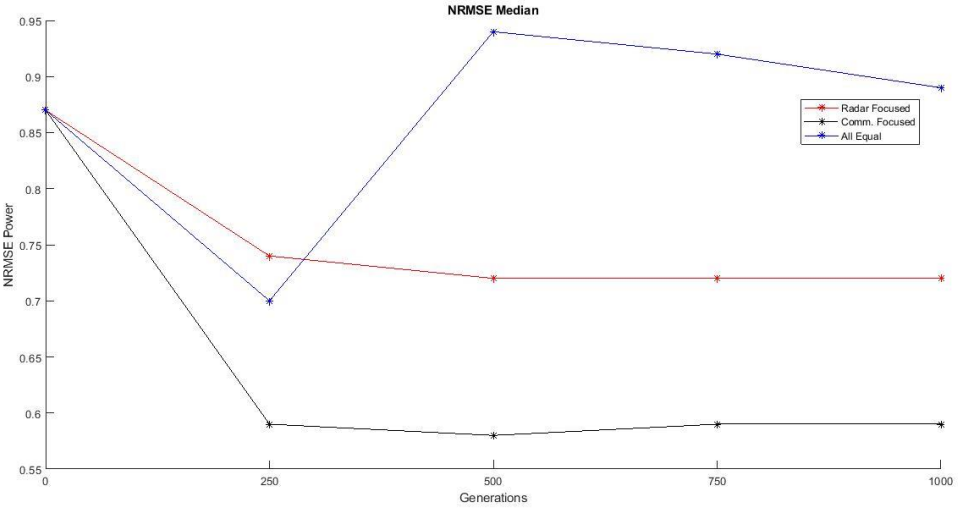


Figure 7-5: NRMSE results after 1000 generations.

One of the things that stands out from these graphs is that there is eventually a plateau in performance. The most likely cause of this is that as the algorithm tries to improve in one category, it comes at the expense of another. For instance, in the All Equal Case between generation 750 and 1000, the NRMSE power decreases which is desired, but the PSLR moves in

the wrong direction. An example of the opposite occurs between generation 250 and 500, the NRMSE power increases and the PSLR does slightly increase as desired. In the Communication Focused Case, the NRMSE power and BER seem to reach their natural plateaus as there is not much room for improvement. In the Radar Focused Case, the probability of detection actually does not plateau. The PSLR and PAPR did plateau, but this can be at least in part explained as having less priority than the probability of detection. To get a deeper look into the performance, we look into what happens in the first 300 generations, with the same parameters being used.

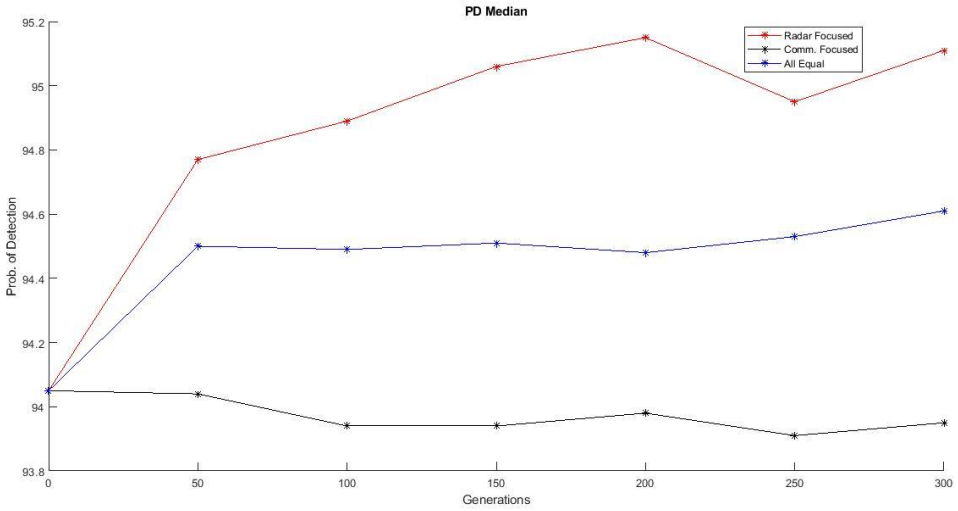


Figure 7-6: Probability of Detection results after 300 generations.

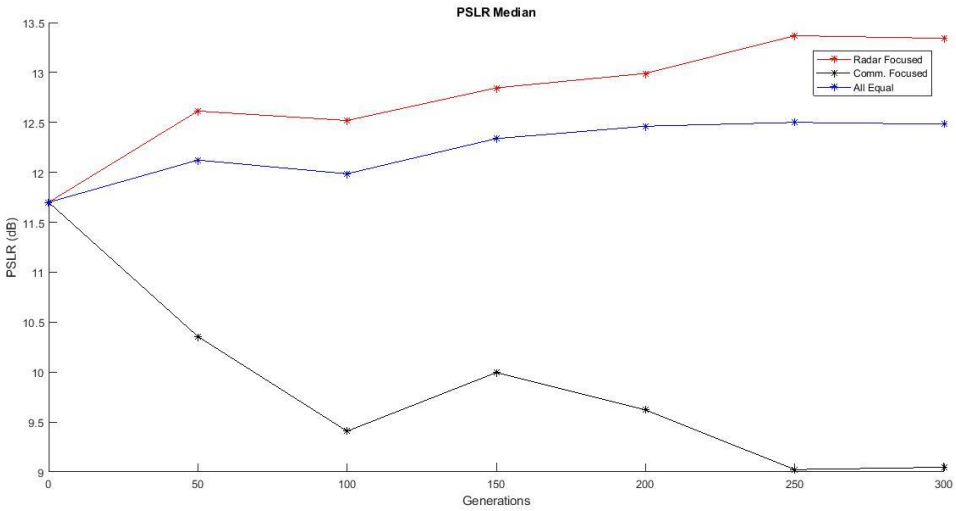


Figure 7-7: PSLR results after 300 generations.

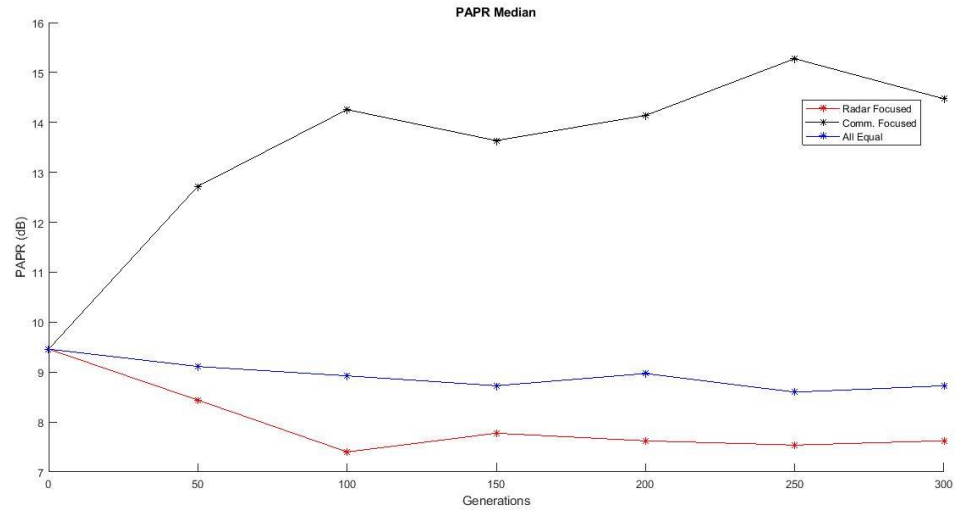


Figure 7-8: PAPR results after 300 generations.

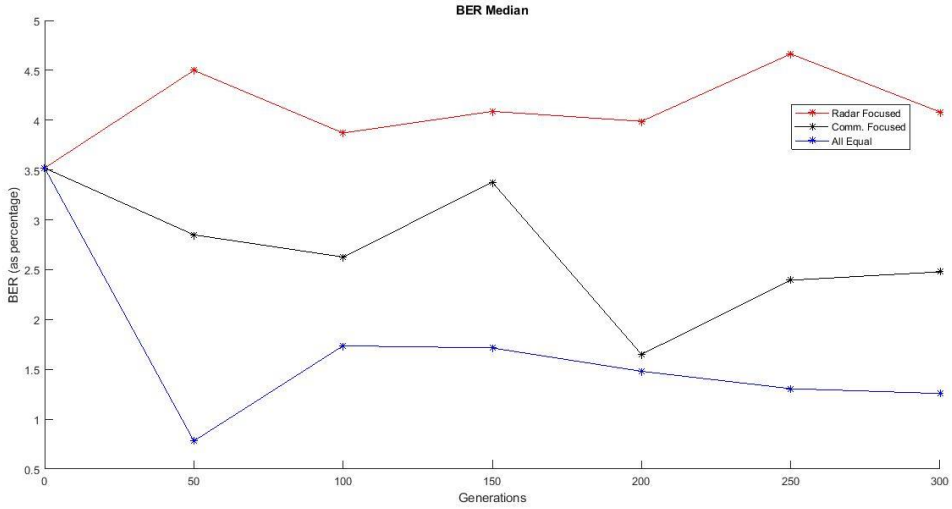


Figure 7-9: BER results after 300 generations.

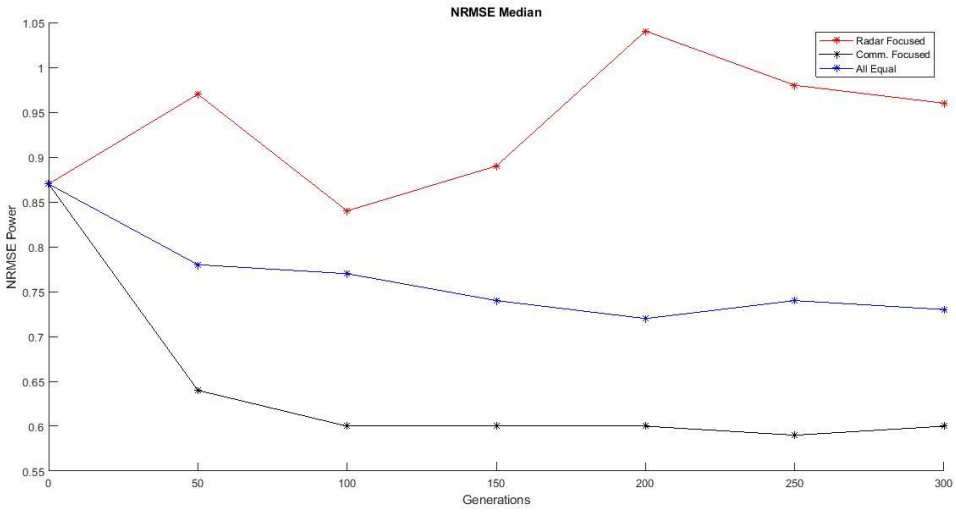


Figure 7-10: NRMSE results after 300 generations.

In this case, we see more gradual changes in the results, but in some cases most of the improvement happens early. For instance, both the NRMSE Power in the Communications Case and the PAPR in the radar case see most of their improvement in the first 100 generations. Overall though, it seems 300 generations is an appropriate length to run the algorithm in most cases. In Figure 7-10, we see that in the Communications Case, the NRMSE power does reach below 0.6208, meaning there is a good chance the signal would be mistaken for clutter. Meanwhile, the All Equal Case does not quite meet that threshold and in fact actually increases after 300 generations likely in an attempt to improve other objectives. It does reach a lower BER than the Communications Case in the 300 generations though, reaching 1.26% compared to 2.48%.

In the Radar Case, the probability of detection increased 1.13% of the maximum 6.33%, PSLR increased 14%, and PAPR increased 19.4%. In the All Equal Case, probability of detection increased 60%, PSLR increased 8.3%, and PAPR increased 7.8%. We see here that since the Radar Case had more priority weight in these three cases, it performed better.

7.2 Changing Parameters

In this section, we show how the parameters affects the performance. First, in Tables 7-2 and 7-3, there are four different crossover and mutation parameters tested with α still equal to 0.25 and over 300 generations. The Communications Case is shown first, arguments for either 25/50 or 50/25 could be made based on whether the significant increase in the less important BER is preferred over the mild improvement in the more important NRMSE power. In the next table are the results for the Radar Case; here again, a case could be made for setting the parameters at 25/50 or 50/50.

Table 7-2: Varied Mutation Parameters in the Communications Case

η_c/η_m	**Prob. Of Det.**	**PAPR**	**PSLR**	**BER**	**NRMSE Power**
25/25	93.95%	14.47	9.05	2.48%	0.5952
25/50	93.99%	14.56	9.63	2.25%	0.5861
50/25	93.97%	14.48	9.61	1.55%	0.5998
50/50	94.04%	14.49	9.91	2.64%	0.6039

Table 7-3: Varied Mutation Parameters in the Radar Case

η_c/η_m	**Prob. Of Det.**	**PAPR**	**PSLR**	**BER**	**NRMSE Power**
25/25	95.11%	7.62	13.34	4.08%	0.9580
25/50	95.08%	7.59	13.35	3.69%	0.9919
50/25	95.28%	7.84	13.04	4.43%	1.003
50/50	95.30%	7.78	13.02	4.34%	0.94

Next, we tried a different α of 0.8 which did help both in the Radar Case and Communications Case, as shown in Tables 7-4 and 7-5. A value of 0.8 does perform better in these cases. While no one set of parameters brought overwhelming increases in performance, in future work, an exhaustive search could be done with a starting point of $\alpha = 0.8$, $\eta_c = 0.25$, and $\eta_m = 0.50$.

Table 7-**4**: Varied α Parameter in the Radar Case

α	**Prob. Of Det.**	**PAPR**	**PSLR**	**BER**	**NRMSE Power**
0.25	95.11%	7.62	13.34	4.08%	0.9580
0.80	95.47%	7.39	13.48	4.79%	0.9385

Table 7-**5**: Varied α Parameter in the Communications Case

α	**Prob. Of Det.**	**PAPR**	**PSLR**	**BER**	**NRMSE Power**
0.25	93.95%	14.47	9.05	2.48%	0.5952
0.80	93.94%	14.76	10.36	.86%	0.5943

7.3 NSGA-III and Simulated Annealing

Here, we show the results of the NSGA-III, first in Table 7-6, where we tested the same three cases with the same parameters as described in Section 7.1. There were five divisions on each objective in this case and $m = 0.02$.

Here, we see very poor performance in all cases in Table 7-6. The Radar Case and Communications Case have the opposite performance compared to what one would expect. The NRMSE power was highest in the Communications Case and Radar Focused Case had the lowest probability of detection. After a significant amount of troubleshooting, the conclusion was that the priority weight system did not work in the NSGA-III case, although the exact reason was not determined. The solution was to not implement priority weights and run the original NSGA-III with the five objectives and a variety of parameters.

Table 7-6: NSGA-III Results

	Prob. Of Det.	**PAPR**	**PSLR**	**BER**	**NRMSE Power**
All Equal	94.00%	10.01	11.17	0.39%	0.7089
Comms Focused	94.00%	10.66	8.85	3.52%	0.8910
Radar Focused	93.95%	9.53	11.68	3.13%	0.7737

Table 7- 7: NSGA-III Results with No Prioritization $m = 0.02$

$m = 0.2$	**Prob. Of Det.**	**PAPR**	**PSLR**	**BER**	**NRMSE Power**
0.25/0.25	94.40%	10.37	12.07	2.73%	0.6680
0.25/0.50	94.30%	9.33	12.42	2.73%	0.7239
0.50/0.25	94.40%	9.54	12.50	2.34%	0.6967
0.50/0.50	94.30%	9.11	3.15	2.73%	1.07

Table 7-8: NSGA-III Results with No Prioritization $m = 0.2$

$m = 0.2$	Prob. Of Det.	PAPR	PSLR	BER	NRMSE Power
0.25/0.25	94.40%	10.37	12.07	2.73%	0.6680
0.25/0.50	94.40%	9.90	12.34	2.73%	0.6586
0.50/0.25	94.30%	9.98	12.73	2.34%	0.6631
0.50/0.50	94.50%	9.53	10.93	2.34%	0.7836

Here we can see that overall $m = 0.2$ is performs better and that 0.50/0.50 is the best for probability of detection and 0.25/0.50 is the best for NRMSE power, but 0.50/0.25 is the best overall. In the NSGA-II All Equal Case for 300 generations, the probability of detection at 94.61%, PAPR at 8.73 dB, and BER at 1.26% were better than the NSGA-III in all cases, while the NRMSE power of 0.73 and PSLR of 12.48 dB were worse in most cases. Overall, it does seem that the NSGA-II performs better, and it has the advantage of working with the priority weights.

Simulated Annealing was also tested, but there were two general problems. One in which it was too “hot”, meaning that even if random solutions were not better than the starting point, they replaced them dramatically reducing performance. The second was when it was “too cold” and no new substitutions could be found. This likely stems from the fact that it was random subcarriers being created and as mentioned in the section about Simulated Annealing, it can be helpful to create the new population that is similar to the original instead of a completely random one. However, that is too close to the method of genetic algorithms to be tested in this thesis. A fusion of the two is a potential concept to explore in the future, however.

Chapter 8

Conclusion

8.1 Concluding Remarks

In this work, a model for creating an OFDM signal that can perform in radar and communications applications while working in a hostile environment was developed. The model included using QPSK as the communications scheme and using the amplitude of the subcarriers, as well as many of their phases to help create a clutter like model. In order for this to be effective, we needed to know the type of clutter in the environment. In order to achieve these goals, a variety of computer science tools were applied from Machine Learning and Genetic Algorithms.

From Machine Learning, multiple variations of decision trees, discriminant analysis, SVMs, and KNNs were applied. In the end, it was clear that KNN, and specifically the Weighted KNN was the best at this task. We also confirmed that along with the first four moments of the random vector, using machine learning to find the additional two features of CIR and distribution family was also helpful in overall classification. The overall classification rate was 97.1% and was significantly better than using a goodness-of-fit measure like NRMSE.

Once the clutter model was determined, it was attempted to optimize the signal across five objectives. They were PSLR, PAPR, probability of detection, BER, and NRMSE power. The NSGA-II outperformed the NSGA-III in this task, Simulated Annealing did not perform better than random generation, although there are possibly better implementations out there. In the end, we were able to improve all five objectives at once or choose a subset to improve. However, there is plenty of room of improvement for the magnitude of the optimization. Finally, a way to implement priority levels in the NSGA-II beyond just discrete levels was developed

8.2 Future Work

If this work were to continue, it is suggested to focus more on the signal optimization than clutter classification at this point. Another suggestion is to optimize over the phases of subcarriers not used. Otherwise, it is recommended to try other optimization algorithms such as a generative adversarial network or other variants of the SA. In addition. hardware testing is recommended to compare results with simulations, both for signal optimization and clutter classification. The one in Miami University research lab would be a good place to start as it is already a ultrawideband radar. One other change that could be investigated is the noise added to our signals may not always be additive white gaussian noise (AWGN) and so other noise models may be considered to make the optimization model more realistic.

References

[1] J. Moghaddasi and K. Wu, "Multifunctional Transceiver for Future Radar Sensing and Radio Communicating Data-Fusion Platform," in *IEEE Access*, vol. 4, pp. 818-838, 2016

[2] C. Sturm and W. Wiesbeck, "Joint integration of Digital Beam-forming Radar with communication," *2009 IET International Radar Conference*, 2009

[3] Y. L. Sit, C. Sturm, and T. Zwick, "Interference cancellation for dynamic range Improvement in an OFDM joint radar and communication system," *2011 8th European Radar Conference*, 2011, pp. 333-336.

[4] Y. L. Sit, C. Sturm, and T. Zwick, "Doppler estimation in an OFDM joint radar and communication system," *2011 German Microwave Conference*, 2011, pp. 1-4.

[5] S. Kafshgari and R. Mohseni, "The effect of target fluctuation on the OFDM radar detection performance," *2012 20th Telecommunications Forum (TELFOR)*, 2012, pp. 827-830

[6] S. Zhu, R. Yang, X. Li, J. Zuo, D. Li, and Y. Ding, "An Optimizing Method of OFDM Radar Communication and Jamming Shared Waveform Based on Improved Greedy Algorithm," in *IEEE Access*, vol. 8, pp. 186462-186473, 2020

[7] D. Garmatyuk, M. Simms, and S. Mudaliar, “UWB multicarrier radar target scene identification with 2-D diversity utilization and GLRT refinement,” *IEEE Sensors Lett.*, vol. 3, no. 12, Dec. 2019.

[8] C. J. Pici and R. M. Narayanan, "Multifunctional Radar and Communications Waveform Using Chaos," *NAECON 2018 - IEEE National Aerospace and Electronics Conference*, 2018, pp. 568-572.

[9] D. Kellett, D. Garmatyuk, S. Mudaliar, N. Condict, and I. Qualls, “Random sequence encoded waveforms for covert asynchronous communications and radar,” *IET Radar, Sonar and Navig*., vol. 13, no. 10, pp. 1713 – 1720, Oct. 2019

[10] G. Lellouch, A. K. Mishra and M. Inggs, "Design of OFDM radar pulses using genetic algorithm based techniques," in *IEEE Transactions on Aerospace and Electronic Systems*, vol. 52, no. 4, pp. 1953-1966, August 2016

[11] Y. L. Sit, C. Sturm, and T. Zwick, "Interference cancellation for dynamic range Improvement in an OFDM joint radar and communication system," *2011 8th European Radar Conference*, 2011, pp. 333-336.

[12] F. Ulaby and M. Dobson, *Handbook of Radar Scattering Statistics for Terrain.* Norwood, MA: Artech House, 2019.

[13] J. G. Metcalf, C. Sahin, S. D. Blunt, and M. Rangaswamy, "Analysis of symbol-design strategies for intrapulse radar-embedded communications," in *IEEE Transactions on Aerospace and Electronic Systems*, vol. 51, no. 4, pp. 2914-2931, Oct. 2015

[14] R. Ayadi, I. Kammoun, and M. Siala, "Bridging the Gap Between CP-OFDM and ZP-OFDM for the Provision of Ultra-Low Latency Services in 5G," in *IEEE Systems Journal*, vol. 14, no. 1, pp. 603-613, March 2020

[15] H. Lin, "Flexible Configured OFDM for 5G Air Interface," in *IEEE Access*, vol. 3, pp. 1861-1870, 2015

[16] Brown, Antony. *Great Ideas in Communications.* D. White Co., 1969, page 141

[17] M. Richards, J. Scheer, and W. Holm, *Principles of Modern Radar.* Edison, NJ: SciTech Publishing, 2010.

[18] D. J. Bachmann, R. J. Evans, and B. Moran, "Game Theoretic Analysis of Adaptive Radar Jamming," in *IEEE Transactions on Aerospace and Electronic Systems*, vol. 47, no. 2, pp. 1081-1100, April 2011.

[19] D. J. Bachmann, R. J. Evans, and B. Moran, "Game Theoretic Analysis of Adaptive Radar Jamming," in *IEEE Transactions on Aerospace and Electronic Systems*, vol. 47, no. 2, pp. 1081-1100, April 2011.

[20] Shaharyar Kamal, Cesar A. Azurdia-Meza, and Kyesan Lee,"Family of Nyquist-I Pulses to Enhance Orthogonal Frequency Division Multiplexing System Performance," *IETE Technical Review*, vol. 33, pp. 187-198, 2016.

[21] L. Pensworth, "What is Mbps?," 2020 [Online]. Available: https://dailywireless.org/internet/what-is-mbps/. [Accessed September 2021].

[22] L. Carrer and L. Bruzzone, "Solving for ambiguities in radar geophysical exploration of planetary bodies by mimicking bats echolocation," *National Communications*, vol. 8, pp. 2248, 2017.

[23] M. A. Rico-Ramirez and I. D. Cluckie, "Classification of ground clutter and anomalous propagation using dual polarization weather radar," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 46, no. 7, pp. 1892-1904, July 2008.

[24] M. Rangaswamy, D. Weiner, and A. Ozturk, "Computer generation of correlated non-Gaussian radar clutter," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 31, no. 1, pp. 106 – 116, Jan. 1995.

[25] J. A. Jackson and R. L. Moses, "Clutter model for VHF SAR imagery," *Proc. SPIE 5427, Algorithms for Synthetic Aperture Radar Imagery XI*, pp. 271-283, Sept. 2004.

[26] V. Kurama, "A Review of Popular Deep Learning Architecures: AlexNet, VGG16, and GoogleNet," 2020 [Online], Available: blog.paperspace.com/popular-deep-learning-architecures-alexnet-vgg-googlenet/. [Accessed September 2021].

[27] P. Tan, M. Steinbach, and V. Kumar, “Classification: Basic Concepts, Decision Trees, and Model Evaluation,” 2021. [Online], Available: https://www-users.cse.umn.edu/~kumar001/dmbook/ch4.pdf. [Accessed September 2021].

[28] C. Bishop, *Pattern Recognition and Machine Learning.* New York City, NY: Springer Science+Business Media, LLC, 2006.

[29] “9.2.3 Optimal Classification,” 2021. [Online], Available: https://online.stat.psu.edu/stat508/lesson/9/9.2/9.2.3 [Accessed September 2021].

[30] J. Platt, “Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines. Advances in Kernel Methods-Support Vector Learning,” Microsoft Research, Tech. Report MSR-TR-98-14, 1998.

[31] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II,” *IEEE Trans. on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[32] N. Srinivas and K. Deb, "Multiobjective function optimization using nondominated sorting genetic algorithms", *Evol. Comput.*, vol. 2, no. 3, pp. 221-248, Fall 1995.

[33] K. Deb and H. Jain, "An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box

Constraints," in *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 577-601, Aug. 2014.

[34] K. K. Mishra, A. Kumar, and A. K. Misra, "A variant of NSGA-II for solving priority based optimization problems," *2009 IEEE International Conference on Intelligent Computing and Intelligent Systems*, pp. 612-615, 2009.

[35] L. Lai, L. Fiaschi, M. Cococcioni, and K. Deb, "Solving Mixed Pareto-Lexicographic Multiobjective Optimization Problems: The Case of Priority Levels," in *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 5, pp. 971-985, Oct. 2021.

[36] H. Ishibuchi, R. Imada, Y. Setoguchi, and Y. Nojima, "Performance comparison of NSGA-II and NSGA-III on various many-objective test problems," *2016 IEEE Congress on Evolutionary Computation (CEC)*, pp. 3045-3052, 2016

[37] F. Y. Vincent, S.-W. Lin, W. Lee, and C.-J. Ting, ‘‘A simulated annealing heuristic for the capacitated location routing problem,’’ Computer. Ind. Eng., vol. 58, no. 2, pp. 288–299,