Signal Detection and Digital Modulation Classification-Based Spectrum Sensing for Cognitive Radio

A Dissertation Presented

by

Curtis M. Watson

to

The Department of Electrical and Computer Engineering

in partial fulfillment of the requirements
for the degree of

Doctor of Philosophy

in the field of

Computer Engineering

Northeastern University
Boston, Massachusetts

September 2013

Curtis Watson is employed by The MITRE Corporation at 202 Burlington Road, Bedford, MA 01730. The author's affiliation with The MITRE Corporation is provided for identification purposes only, and is not intended to convey or imply MITRE's concurrence with, or support for, the positions, opinions or viewpoints expressed by that author. Approved for Public Release; Distribution Unlimited Case Number 13-2957.

NORTHEASTERN UNIVERSITY

Graduate School of Engineering

Dissertation Title:	Signal Detection and Digital Modulation Classification-Based Spectrum Sensing for Cognitive Radio
Author:	Curtis M. Watson
Department:	Electrical and Computer Engineering

Approved for Dissertation Requirement for the Doctor of Philosophy Degree

Dissertation Adviser: Prof. Waleed Meleis	Date
Dissertation Reader: Prof. Jennifer Dy	Date
Dissertation Reader: Prof. Kaushik Chowdhury	Date
Department Chair: Prof. Sheila Hemami	Date

Graduate School Notified of Acceptance:

Director of the Graduate School: Prof. Sara Wadia Fascetti	Date

Acknowledgments

I want to thank the MITRE Corporation for the opportunity to pursue my Ph.D. while I am employed as a part of the Accelerated Graduate Degree Program (AGDP), as well as the encouragement and support by my management. I could not imagine a successful completion of my research and dissertation without the benefits provided by AGDP.

I appreciate the constant encouragement by my supervisors, and in particular I want to thank Jerry Shapiro and Kevin Mauck who continually checked on my progress and motivated me to complete my dissertation.

I also want to thank Kevin Burke for his encouragement to complete my dissertation. Additionally, I enjoyed the technical discussions about digital communications, communication systems, and machine-learning & classification theory, which provided useful guidance and thoughts to complete my research.

I want to thank Matt Keffalas for many hours of discussion about machine learning and how it can be applied to communications, in particular digital modulation classification. Matt planted many seeds of ideas that helped to shape my thoughts that were applied to my research.

I want to thank Prof. Jennifer Dy and Prof. Kaushik Chowdhury for being on my committee. I appreciate the questions, suggestions, and critiques they provided which led to improvements in my research.

I want to thank Prof. Waleed Meleis for advising me through the completion of my research and dissertation. His advice helped me navigate this unique experience from which I am a better research investigator and a more polished writer. I am grateful for taking his Combinatorial Optimization course not only because it is an interesting subject, but also because it led me to seek him out to be my advisor.

Finally, I want to thank from the bottom of my heart my wife, Jaclyn for helping me complete this journey. She helped by watching the kids when I needed to do work on my research or to be on campus. She gave up weekends for me to complete my dissertation. Jaclyn has been with me from the start to the finish of my Ph.D. studies and she will enjoy with me the fruits of this labor now that it is complete. Thank you Jaclyn and I love you.

Abstract

Spectrum sensing is the process of identifying available spectrum channels for use by a cognitive radio. In many cases, a portion of the spectrum is licensed to a primary communication system, for which the users have priority access. However, many studies have shown that the licensed spectrum is vastly underutilized, which presents an opportunity for a cognitive radio to access this spectrum, and motivates the need to research spectrum sensing. In this dissertation, we describe a spectrum sensing architecture that characterizes the carrier frequency and bandwidth of all narrowband signals present in the spectrum, along with the modulation type of those signals that are located within a licensed portion of the spectrum. From this radio identification, a cognitive radio can better determine an opportunity to access the spectrum while avoiding primary users.

We describe a narrowband signal detection algorithm that takes an iterative approach to jointly estimate the carrier frequency and bandwidth of individual narrowband signals contained within a received wideband signal. Our algorithm has a number of tunable parameters, and the algorithm gives consistent performance as we varied these parameter values. Our algorithm outperforms the expected performance of an energy detection algorithm, in particular at lower signal-to-noise ratio (SNR) values. These behavioral features make our algorithm a good choice for use in our spectrum sensing architecture.

We describe a novel constellation-based digital modulation classification algorithm that uses a feature set that exploits the knowledge about how a noisy signal should behave given the structure of the constellation set used to transmit information. Our algorithm's classification accuracy outperforms a set of literature comparisons' results by an average increase of 9.8 percentage points, where the most dramatic improvement occurred at 0 dB SNR with our accuracy at 98.9% compared to 37.5% for the literature. The classifier accuracy improves using our feature set compared to the classifiers' accuracy using two feature set choices that are common in the literature by an average increase of 13.35 and 5.31 percentage points. These qualities make our algorithm well-suited for our spectrum sensing architecture.

Finally, we describe our spectrum sensing architecture that coordinates the execution of our narrowband signal detection and modulation classification algorithms to produce an spectrum activity report for a cognitive radio. This report partitions the spectrum into equally-sized cells and gives

an activity state for each cell. Our architecture detects spectrum opportunities with a probability of 99.4% compared to 87.7% and 93.8% for two other comparison approaches that use less information about the primary user's waveform. Our architecture detects "grey-space" opportunities with a probability of 96.1% compared to 49.1%. Also, the false alarm rate is significantly lower for our architecture, 13.3% compared to 46.9% and 62.7% for the two comparisons. Consequently, we conclude that a cognitive radio can achieve better spectrum utilization by using our spectrum sensing architecture that is aware of the primary user(s) waveform characteristics.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Software defined radio (SDR) is the technique of processing wireless radio communication signals completely inside a computer using software. Traditionally, radios are built in hardware to allow for real-time processing of the radio signals. In order to create a new instance of a traditional radio requires a large amount of time, effort, and cost to develop this hardware. Advances in processor technology, such as increased CPU processing speeds, and multi-core computer architectures, allows this same development for a new radio to occur, in software, on general-purpose processors [1]. By developing a radio in software, we can reduce these previously mentioned problems. Also, a SDR has the flexibility to dynamically reconfigure radio parameters, such as modulation family type, and this ability is a desirable feature for highly dynamic radio environments [2].

Cognitive radios are radios that are capable of learning their surrounding environment in order to adapt its internal parameters to improve its ability to reliably communicate. The observed environment will guide the decision making process of a cognitive radio. Determining when and where, in time and in frequency, to transmit, or sensing other radio interferers are examples of the types of decisions that can be made by a cognitive radio. A cognitive radio can also learn from previous decisions in order to improve performance in the future [3]. The flexibility of SDR to adapt its software makes it an appropriate architecture for the development of cognitive radios [4].

Licensed spectrum is an allocation of frequency bandwidth that is controlled by an individual entity. This allocation of bandwidth is called a channel of spectrum, and to gain control of a channel, the spectrum is either purchased or given by the government [5]. This individual is the primary user of the channel, and is the only user allowed to transmit on a licensed channel of spectrum. However, unlicensed spectrum is freely available bandwidth for use by anybody. The demand for wireless connectivity by modern technologies has led to an overcrowding of unlicensed spectrum. On the other hand, the licensed spectrum is vastly underutilized. For example, in Berkeley, California, an experiment found that roughly 30% of frequency band from 0 to 3 GHz was utilized [6]. Between 3 to 6 GHz, the utilization was measured to be a mere 0.5%. Numerous other studies have also

shown that for the most part the licensed spectrum is severely underutilized [3, 7–9].

The Federal Communication Commission (FCC) has taken an interest in the fact that most of the licensed spectrum is idle. The FCC is considering policy that would open the spectrum to unlicensed usage [10–12]. Cognitive radios are proposed as one method to take advantage of this more open spectrum policy [13]. These radios would determine if licensed spectrum is available for use. If the cognitive radio does not detect a primary user of channel, then the radio is free to become a secondary user of the channel [14]. A secondary user is allowed to communicate on licensed spectrum as long as the primary user is not actively transmitting. This task of monitoring for primary user activity of licensed spectrum is typically referred to as spectrum sensing.

Spectrum sensing is the process used by a cognitive radio to identify available channels of both licensed and unlicensed spectrum in order to wirelessly communicate. The frequency spectrum is a dynamic system that changes with time and location. The availability of a channel in the licensed spectrum depends on the activity of the primary user, which has priority to the licensed spectrum. The underutilization of the licensed spectrum presents an opportunity for secondary communication systems to transmit on these unused channels. This type of capability is the research motivation of spectrum sensing. For example, the IEEE 802.22 standard requires a secondary user to vacate the channel or reduce power within two seconds of a primary user becoming active [15]. Thus, the cognitive radio must quickly detect this appearance to prevent interfering with the primary user [16], which implies that a cognitive radio must constantly perform spectrum sensing. Another example of a state-of-the-art technology that requires spectrum sensing is medical devices that wish to transmit information and control wirelessly [17]. Even though spectrum sensing has been studied intensively this past decade, there remain many technical challenges [18].

Five common algorithms enable spectrum sensing: energy detection, cyclostationary detection, waveform-based sensing, radio identification, and match filtering [19–31]. Energy detection compares the received energy in frequency band to a threshold. If the threshold is exceeded, then a signal detection is asserted. Cyclostationary detectors search for periodicity that exist in modulated signals and which do not exist in background noise. Cyclostationary detectors can achieve a high probability of signal detection with a low signal-to-noise ratio (SNR). Waveform-based sensing involves detecting a known pattern or structure embedded in the received signal. Radio identification looks for characteristic values of the received signal and compares them to values expected from a primary user. Match filtering correlates the received signal with the known waveform of the primary user in order to find a match. We discuss each of these algorithm types in more detail in Section 1.2.

1.1 Statement of Work and Research Importance

The objective of this work is to design an architectural framework that allows a cognitive radio to detect the presence of a primary communication system. This framework will allow a secondary user to operate without generating excessive interference with the primary user(s) of the spectrum. The framework improves on the enabling algorithms for spectrum sensing that automatically characterize the parameters of any active signal in the spectrum, such as location, bandwidth, and modulation type.

Our research provides a number of important contributions and improvements to the field of spectrum sensing for cognitive radios. An important aspect of spectrum sensing is the ability to correctly determine spectrum availability for communication. Since a licensed user of the spectrum has priority, most signal detection systems tend to be conservative to reduce the risk of interfering with the primary user, and a user of a cognitive radio will accept a larger probability of a false positive signal detection. As a result, the user of a cognitive radio is not able to take full advantage of all available spectrum at all times.

Our work improves the reliability of the signal detection process by classifying the modulation type of any detected signal in a licensed portion of the spectrum and comparing this modulation to the known type used by the primary user. This improved reliability will mean that a cognitive radio can make better use of available spectrum for communication. In fact, for a radio to achieve a high level of cognition, further information about the waveform, such as modulation, must be exploited to effectively sense the spectrum [24].

Another important consequence of our work is that our approach can help prevent a cognitive radio from being spoofed into thinking the primary user is active by other cognitive radios trying to access spectrum for communication. If a signal is detected, then our work will provide the cognitive radio with a modulation classification on that signal. If the cognitive radio knows the type of radio a licensed user will use, then it can match that knowledge with the classified modulation type. If a match does not exist, then the cognitive radio will know it has the same rights to the spectrum as any other secondary radio. Without this extra modulation classification step, the cognitive radio must assume that any detected signal is being produced by a primary user. This type of primary user emulation can threaten the usefulness of spectrum sensing [32] and our work represents a front line detection of such an attack.

1.2 Related Work

Spectrum sensing by its nature is a very challenging problem. The environment in which we sense tends to operate in a low SNR regime along with the possibility of multipath fading [24, 25].

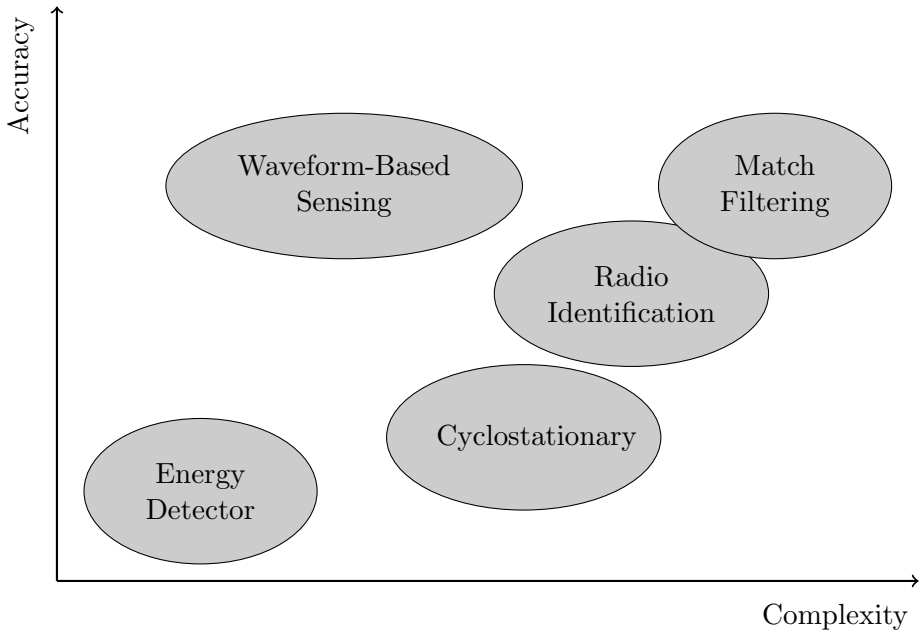


Figure 1.1: Qualitative assessment of computational complexity versus primary user detection accuracy for the five common spectrum sensing enabling algorithms where the complexity increases from left to right and the accuracy increases from bottom to top; cf. [19]

For real-time processing, the hardware requirements are demanding, such as the need for a high-sampling rate analog-to-digital converters, and efficient digital signal processing algorithms that may need FPGA implementations [19, 33, 34]. An additional challenge is that the primary user can be hidden from the spectrum sensing architecture due to fading or shadowing [35, 36].

In this dissertation, we focus on algorithmic development to enable a spectrum sensing architecture. There are five common algorithms that enable spectrum sensing: energy detection algorithms, cyclostationary algorithms, waveform-based algorithms, radio identification algorithms, and match filtering algorithms [19–31]. Each of these algorithm types have advantages and disadvantages, allowing for a collection of tradeoffs for the developer of a spectrum sensing architecture.

We qualitatively illustrate these tradeoffs in Figure 1.1 where the horizontal axis represents the algorithmic computational complexity that increases from left to right and the vertical axis represents the algorithmic accuracy in detecting an active primary user that increases from bottom to top. In the general, as we increase the algorithmic accuracy and/or complexity, we also increase the amount of knowledge required by the algorithm about the primary user. For example, in sensing the TV white space (or unused spectrum) [37–39], an energy detection algorithm could be used to sense active TV stations at a lower accuracy than an algorithm that takes advantage of supplied knowledge about the TV station channel numbers. However, in this example, the energy detection algorithm could be used in any geographical location, whereas the second algorithm would require

the active TV station channel number which change from one location to another. Next, we discuss each of these enabling algorithms.

The energy detection algorithm is a popular choice for spectrum sensing due to its low computational cost and simplicity in implementation [40–52]. Energy detection compares the received energy in frequency band to a threshold. If the threshold is exceeded, then a signal detection is asserted. Therefore an energy detection algorithm doe not require prior information about the primary user in order to detect the signal [26].

However, in order to compute the optimal threshold, we must have nearly perfect knowledge of the noise power (or variance) [53–55]. Additionally, the energy detection algorithm assumes a static environment where the noise power does not fluctuate [56] and performance degrades when there is fading and shadowing [44]. Finally, the energy detection algorithm cannot differentiate between primary and secondary users. Energy detectors are best suited to a coarse estimation due to sensitivity to noise [41]. For example, in [57], an energy detector is used as the first stage in a two-stage process to perform spectrum sensing where the output of the detector becomes the input to a cyclostationary feature detection algorithm. In general, if there is available information about the primary user's signal, such as center frequency, bandwidth, and/or modulation type, then an algorithm that utilizes that information typically can outperform the energy detection algorithm [42].

Cyclostationary feature detectors search for periodicity that exist in modulated signals and which do not exist in background noise [58, 59]. The type of transmitted signals that a spectrum sensing architecture tries to detect are stationary random process, but these signals exhibit cyclostationarity feature because the modulated signal is coupled with the sine wave carriers or other repeating codes [60]. Cyclostationary detectors can achieve a high probability of signal detection while operating in a low SNR environment by exploiting the distinct cyclostationary properties of signals [61]. This type of detection algorithm is the second most common choice of algorithm for spectrum sensing, after the energy detection algorithm [10, 21–23, 57, 62–68]. However, cyclostationary processing can tend to be computationally intense, with a complexity of $O(n^2 \log^2 n)$, where n is the number of complex-valued samples and where the performance of this algorithm requires n to be large [24]. Additionally, the cyclic frequencies calculated by this type of algorithm must be compared to a known set of cyclic frequencies that are associated with the primary users [26, 29]. These cyclic frequencies are susceptible to channel inaccuracies such as frequency and timing offsets, which can degrade the performance of a spectrum sensing architecture that uses a cyclostationary detection algorithm [24, 69].

Waveform-based sensing takes advantage of known patterns embedded into broadcast transmission of wireless systems, such as preambles, cyclic prefixes or pilot sequences [68, 70]. These patterns assist the users of this primary system with synchronization [19]. Secondary users can also utilize these known patterns to detect the presence of a primary user. Waveform-based sensing outperforms

the energy detection algorithm since it looks for a particular known pattern [71]. For example, in [72, 73], knowledge of the 802.11b packet structure was exploited to allow a cognitive radio to interoperate with a WiFi system. As another example, in [74] the pseudo-random noise sequence in the frame header of digital multimedia broadcasting-terrestrial (DMB-T) system was used to detect the primary user. However, the difficulty with using a waveform-based sensing approach is that synchronization to the primary system must be establish to reliably detect the primary user where timing and frequency offset errors can degrade the performance [67, 75].

Radio identification detection looks for various characteristics of signals operating in the spectrum, such as center frequency, bandwidth, or modulation type in order to find a match to the users of a primary system [76]. This type of identification gives a spectrum sensing architecture another dimension of information which leads to higher detection accuracy [77]. For example, the universal receiver discussed in [78] estimates the center frequency and bandwidth parameters which discriminates and identifies the standard in use. Radio identification tends to apply pattern recognition techniques as a part of the discrimination process [77, 79, 80]. However, this approach to primary user detection has not been research extensively because of the computational complexity involved and the difficulty to achieve a real-time implementation [70]. In the coming years, this type of detection algorithm will surely advance as technology progresses and improves.

Match filtering involves detecting the complete waveform of the primary user's signal in the received spectrum. This approach requires knowledge of the radio waveform in use by the primary user in order to perform a correlation and is optimal in additive white Gaussian noise [26, 81]. However, there are significant disadvantages to the match filtering approach. There are many characteristics of the primary user's waveform that must be exactly known: operating center frequency, operating bandwidth, modulation type, pulse-shaping, pilot sequence, packet format [76, 82, 83]. These characteristics in practice may vary from the published standards and if this information is inaccurate, then the matched filtering approach to primary user detection performs poorly [83]. In essence, the match filtering approach must create a receiver for the primary user's waveform in order to properly detect the primary user. To further complicate matters, an implementation of a spectrum sensing architecture that uses this approach must essentially create a receiver for all possible type of primary user system to could exist in the environment [76].

1.3 Contributions of this Research

We developed a spectrum sensing architecture to find frequency locations available in the spectrum for use by a cognitive radio. We treat the frequency spectrum as a wideband signal that contains $M \geq 0$ narrowband signals which are primary and secondary users of the spectrum. A primary user has the highest priority access to the spectrum, and our architecture must not interfere with their

access. However, our cognitive radio has equal priority access to the spectrum in use by secondary users.

The goal of this architecture is to estimate a set of parameters for each narrowband signal in order to determine which frequencies in the spectrum are available to our cognitive radio. Our system estimates the following parameters: the signal carrier frequency (or center frequency), the signal bandwidth, and the signal modulation type. We refer to this parameter set as the signal identification. We assume that we know the signal identification for primary users in the spectrum.

Our research represents a type of radio identification system that enables spectrum sensing since we try to identify further the characteristics of any detected signal in the spectrum. This type of approach has not been explored extensively because of the challenges in real-time implementation from the computational complexities of the algorithms involved [70]. The scope of our research is not concerned with the computational complexities of the algorithms we develop, but rather to show the spectrum opportunities gained by using this type of spectrum sensing.

Our research contributes to the improvement of three main components of a spectrum sensing architecture. The first contribution of our research is a narrowband signal detection algorithm which tries to locate all of the narrowband signals in the received wideband signal. The second contribution of our research is a digital modulation classification algorithm, which is used to further identify the characteristics of a detected narrowband signal that might be a primary user. Finally, the third contribution of our research is a spectrum activity report that can be utilized by a cognitive radio to opportunistically access the spectrum. We provide a summary of these contributions in this section.

Narrowband Signal Detection The first research contribution involves jointly estimating the carrier frequency and bandwidth of individual narrowband signals contained within a received wideband signal. We developed an algorithm that takes an iterative approach which operates in the frequency domain where we model the wideband signal as a mixture of M sinc functions, for which each sinc function corresponds to a single narrowband signal. In each iteration, we estimate the carrier frequency, and bandwidth of a new signal to add to our mixture model. Our algorithm has a number of tunable parameters that affect its operation, and we found consistent performance as we varied these parameter values in our tests, which creates a tradeoff-space between parameter resolution and computation-time complexity. We analytically compared and found that the performance of our algorithm outperformed the expected performance of an energy detection algorithm, in particular at lower signal-to-noise ratio (SNR) values.

Modulation Classification Our second research contribution examines classifying the modulation type of a complex baseband signal. We created a novel constellation-based digital modulation classification algorithm that uses a feature set that exploits the knowledge about how a noisy signal

should behave given the structure of the digital modulation constellation sets used to modulate the transmitted information. We use the Expectation-Maximization (EM) algorithm to cluster the received data to a discrete set of cluster points in the complex plane. After clustering, we generate statistics to form our feature vector set, from which a score is calculated using a weight vector. A genetic algorithm is used to train this weight vector. The classification rule implemented chooses the modulation that produces the smallest score. We performed a series of experiments by varying real-world conditions, such as different SNR values, and receiver inaccuracies, and found found that our classifier has a consistent performance as we varied these conditions. We performed the first comparison of modulation classification algorithms that compares accuracies achieved by classification algorithms applied to identical sets of modulation types and SNR values. We compared the classification accuracy of our algorithm against other classification algorithms reported in the literature, which showed that our algorithm outperformed the results in the literature. Finally, we compared the classification accuracy of an ensemble SVM using our feature set with two popular choices in feature sets, and showed that our feature set improved the accuracy the SNR range between 0 dB and 19 dB.

Spectrum Sensing Architecture Our third research contribution is a spectrum sensing architecture that generates an informative spectrum activity report to a cognitive radio by incorporating the knowledge of the primary user's waveform characterization. The architecture executes our narrowband signal detection algorithm on a received wideband signal and our modulation classification algorithm to produce the information needed to create the spectrum activity report. This report partitions the spectrum into equally-sized cells and gives an activity state for each of the cells, which can be open, secondary user active, or primary user active. This extra knowledge about the spectrum allows a cognitive radio to make the best possible decision about opportunistically accessing the spectrum. We perform a series of tests to evaluate our spectrum sensing architecture's ability to detect spectrum opportunities and avoid interfering with a primary user. We compare our architecture to two other approaches which is less aware of the primary user's waveform. We found that our architecture detected spectrum opportunities with a higher probability than the other two approaches, and also the primary user false alarm rate was significantly lower for our architecture. Consequently, we concluded that a cognitive radio that is aware of the primary user waveform characteristic can achieve better spectrum utilization by using our spectrum sensing architecture.

1.4 Dissertation Outline

The remainder of this dissertation is organized as follows.

In Chapter 2, we present background information. We first give an overview of a digital radio communication systems that consists of a transmitter, channel, and receiver. In addition, we

provide the mathematical equations that describe this system. Second, we give an introduction to pattern recognition, which is a topic of machine learning. We give an overview of the theory of classification and describe the different types of classifiers that are available to us.

In Chapter 3 we develop an approach to blind signal detection that jointly estimates the amplitude, bandwidth, and center frequency parameter of each narrowband signals in a received wideband signal. We assume that the received signal is a linear combination of $M \geq 0$ unknown digitally modulated transmitted narrowband signals and noise. In the frequency domain, we assume that each transmitted signal can be approximated by a sinc function and the received wideband signal is a mixture model of sinc functions. Our algorithm determines the number of transmitted signals present in the received signal by using iterative methods to find signals and adjust signal parameters to minimize the measured error and log-likelihood expression.

In Chapter 4, we describe a new constellation-based digital modulation (CBDM) classification algorithm that exploits knowledge of the shape of digital modulation constellations. The CBDM classifier uses a novel feature set, where the features incorporate the knowledge about how a noisy signal should behave given the structure of the constellation set used to modulate the transmitted information. We self-evaluate our classification algorithm along with comparing the classification accuracy against other modulation classifiers in the literature.

In Chapter 5, we detail our spectrum sensing architecture that can enhance a cognitive radio's capability to utilize the spectrum. This improvement is accomplished by identifying the characteristic parameters of each narrowband signal present in the spectrum. This identification will provide extra information for a cognitive radio to incorporate into its decision making process in an attempt to minimize its interference with a licensed user of the spectrum.

In Chapter 6, we summarize our work and discussion future improvements.

Chapter 2

Background

2.1 Digital Communications

A communications system is a set of signal processing blocks that transports information from one user to another. At a very high level, there are three main components to any communication system. The *transmitter* manipulates an electrical signal in preparation for broadcasting over a channel. This process involves modulation [84–87]. Amplitude modulation (AM) and frequency modulation (FM) are common examples of analog modulation. The electrical signal propagates through a physical medium, called the *channel*. The channel corrupts the transmitted signal. The *receiver* collects the corrupted signal from the channel. Using signal processing, the receiver attempts to recreate the original signal. This process involves demodulation [84–87]. A television or music radio broadcast is an everyday example of a communication system.

A significant limitation of an analog communication system is that the receiver must exactly match the original waveform shape in order to properly reconstruct the original signal. In contrast, a digital communication system relaxes the problem to accumulating energy to detect discrete digital information. A digital communication system is a system that converts an analog electrical signal into digital form before transmission. Digital modulation is the process of transmitting this discrete digital information. Since a digital communication accumulates energy, the digital transmissions have better immunity to noise. For this reason, digital transmissions can utilize the channel bandwidth more efficiently than the analog counterparts [84–87].

In our proposed work, we focus on digital communication systems. However, we should note that once our work is complete, our approach can be easily extended to encompass analog communication systems. Figure 2.1 illustrates our assumed representation of the transmitter-channel-receiver processing chain. We now explain mathematically and intuitively what each of the blocks model.

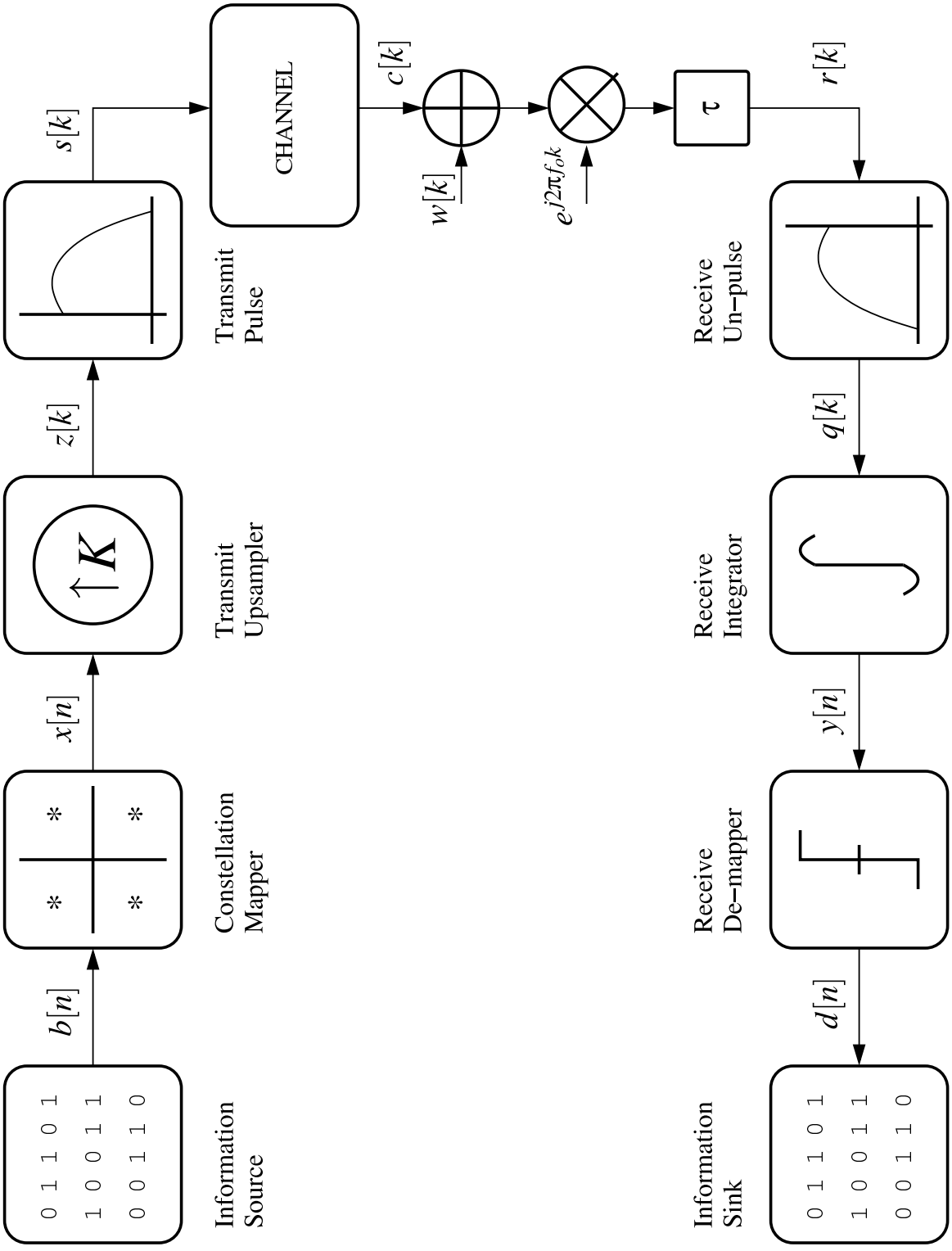


Figure 2.1: Block diagram representation of the digital communications transmitter-channel-receiver processing chain assumed in this research.

2.1.1 The Transmitter

The transmission process begins with an information source. In our work, we assume that the information is a discretized version of some analog signal. If there exist only two possible discrete states, then we refer to this quantity as an information bit. More generally, we will allow M discrete states, where M is two raised to some integer power and we refer to this quantity as an information symbol. The sequence of information symbols are represented by $b[n]$, where n is a discrete time symbol index.

The constellation mapper is a processing block that maps the M possible information symbols onto M discrete points in the complex plane. This mapping is one-to-one and can be defined as f_{mod} : $[0, M) \rightarrow \mathbb{C}$. This set of M discrete points is the signal constellation and each point is a constellation point. Many digital modulation types can be visualized by looking at the (signal) constellation plot. We discuss below the modulation types we consider. The input to the constellation mapper block is a set of information symbols. The output of the constellation mapper block is a set of modulation symbols, $x[n]$, which are created using the mapping f_{mod} on the input.

The transmit upsampler and pulse filter are used to change the spectrum shape of the signal. The upsample process narrows the bandwidth of the signal transmission. If the upsample factor is K, then $K - 1$ zeros are inserted between each modulation symbol in the set of modulation symbols. The resulting sequence of upsampled modulation symbols are represented by $z[k]$, where k is a discrete time sample index. The pulse filter, g, removes high frequency images inserted [84, 88] by the upsampling process. This is accomplished by convolving the upsampled modulation symbols with the pulse filter. The resulting set of complex symbols are transmitted. These transmission symbols, $s[k]$, are the outputs of the upsample and pulse filter.

The following equations describe our transmitter model.

$$x[n] = f_{mod}(\ b[n]\ ) \tag{2.1}$$

$$z[k] = \begin{cases} x[n] & \text{for } k = nK \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

$$s[k] = (z \star g)[k] = \sum_{l} z[l]\ g[k-l] \tag{2.3}$$

The mapping of information symbols to the complex plane as determined by the modulation type is expressed in Equation 2.1. The upsampling of the modulation symbols is the simple process shown in Equation 2.2. The transmission symbols are generated by convolving the upsampled modulation symbols with the pulse filter as given in Equation 2.3. These transmission symbols are sent over a channel.

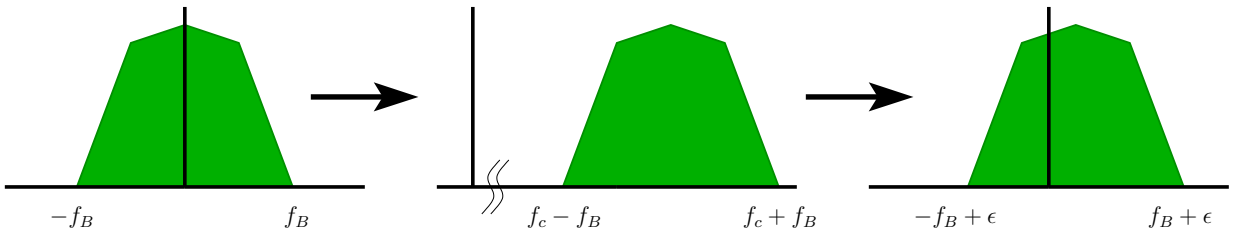


Figure 2.2: Example of a frequency shift in the received complex baseband signal due to the estimation error of the carrier frequency, f_c, by the receiver; on the left, we have the baseband signal to transmit; in the middle, we have the transmitted signal that is modulated by f_c; on the right, we have the received baseband signal that is created by underestimating the carrier frequency by an ϵ amount.

2.1.2 The Channel

The channel is the medium that the transmission symbols physically travel through to reach the receiver. Note that the signal processing blocks that we have modeled for the transmitter are working in the complex baseband domain. Baseband signals are signals that have spectral components between $[-f_B, f_B]$, for some maximum frequency f_B [84]. In a real system, before being transmitted the transmission symbols would be shifted to a passband by shifting the signal to a carrier frequency such that the spectral components lie in $[f_c - f_B, f_c + f_B]$ for some carrier frequency f_c. Note that the passband signal has the same bandwidth as the original signal.

Our channel model incorporates several components in addition to the actual physical channel. The thermal noise associated with the receiver will be a part of our channel model. We also include in our model the receiver estimation inaccuracies to frequency and timing. Frequency inaccuracy refers to inexact estimation of the carrier frequency. This error results in a frequency shift in the receiver complex baseband signal. In Figure 2.1, the frequency inaccuracy is represented as the complex multiply of $c[k]$ with $exp\{j2\pi f_o t\}$, where f_o is the inaccuracy. Timing inaccuracy refers to the uncertainty in estimating the modulation symbol boundary. In Figure 2.1, the timing inaccuracy is represented as the delay D.

An example of the error caused by the inaccurate estimation of the center frequency is illustrated in Figure 2.2. The frequency spectrum of the transmit baseband signal centered around 0 Hz is shown on the left. In the middle, we depict the passband signal that is created by modulating the transmit signal to the carrier frequency. On the right, we have the receiver baseband signal, which is not perfectly centered around 0 Hz. The amount of error, ϵ, is equal to the difference between the estimated carrier frequency and the true carrier frequency.

The following equations represent our channel model. The channel symbols, $c[k]$, are the result of sending the transmission symbols through the physical channel as shown in Equation 2.4. The re-

ceived symbols, $r[k]$, are expressed in Equation 2.5 incorporate the inaccuracies due to the receiver.

$$c[k] = \texttt{channel}(\ s[k]\ ) \tag{2.4}$$

$$r[k] = \Big(c[k+\tau] + w[k+\tau]\Big) \cdot exp\Big\{j2\pi f_0[k+\tau]\Big\} \tag{2.5}$$

Note that the thermal noise, w, is assumed to be an additive white Gaussian process. Also, the timing and frequency offsets are denoted as τ and f_o, respectively.

2.1.3 The Receiver

The receiver is the destination of the transmitted signal in a communication system. The receiver must attempt to recover the original signal from the given corrupted version. The set of processing blocks for the receiver are essentially the inverse operations of the blocks in the transmitter. The receiver unpulse filter, h, removes the effects of the transmit pulse filter. The matched filter of the transmit pulse filter is the optimal filter to use [84]. Thus, the receiver unpulse filter is the conjugated time-reversed version of the pulse filter. The output of this processing block is a sequence of unpulsed samples, $q[k]$, where k is a discrete time sample index. Next, the receivers undoes the upsample process by integrating the signal over the symbol period. For our discrete time signal, this means that we sum the K samples that correspond to one modulation symbol. The output of this block is a sequence of estimated modulation symbols, $y[n]$, where n is a discrete time symbol index. Finally, we must attempt, using the de-mapper, to produce the original information symbols. Given a particular modulation $\mathcal{M}$, we choose the constellation point, $\mu \in \mathcal{M}$, that is the closest to our estimated modulation symbol. Recall that the constellation point corresponds to an information symbol. Thus, the output of the de-mapper block is a sequence of estimated information symbols, $d[n]$.

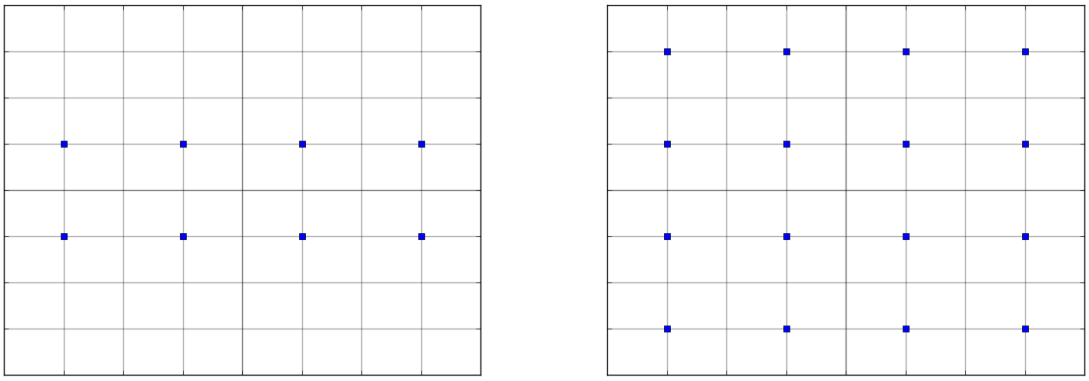
The following equations describe our receiver model.

$$q[k] = (r \star h)[k] = \sum_{l} r[l]\ h[k-l] \tag{2.6}$$

$$y[n] = \sum_{k'=0}^{K-1} q[n \cdot K + k'] \tag{2.7}$$

$$d[n] = \arg\min_{\mu \in \mathcal{M}} \big\| y[n] - \mu \big\|^2 \tag{2.8}$$

We apply the match filter of our pulse filter to the received signal in Equation 2.6. Note that $h[\xi] = g^*[-\xi]$, where * denotes the complex conjugate. We integrate the matched filter signal over a symbol period to produce our modulation symbol estimate in Equation 2.7. The estimated information symbol is set equal to the constellation point in $\mathcal{M}$ that is closest to the estimated modulation symbol, as shown in Equation 2.8. Note that this receiver process implicitly assumes



8-QAM 16-QAM

Figure 2.3: Two example QAM constellation plots.

a Gaussian noise process.

2.1.4 Problem Space Under Consideration

There exist a large number of digital modulation types and channel models. In order to successfully complete this work, we restrict the set of modulation types that we consider to those that can be represented by a signal constellation plot, and we only consider the additive white Gaussian noise channel. We will now elaborate on this reduced problem space.

Pulse amplitude modulation (PAM) conveys information in the amplitude of the transmitted signal [84]. For M-ary PAM, we define the constellation to be the set defined in Equation 2.9. In essence, PAM transmits information in one dimension of the complex plane.

$$\mathcal{M}_{PAM} = \Big\{ m \ : \ -M < m < M, \ m \text{ is odd integer} \Big\} \tag{2.9}$$

Quadrature amplitude modulation (QAM) can be viewed as using PAM in both dimensions of the complex plane [84]. The result is that the signal constellation plot looks like a grid. Figure 2.3 shows sample constellations for 8-QAM and 16-QAM.

Phase shift keying (PSK) sends information in the phase of the transmitted signal [84]. For M-ary PSK, we define the constellation to be the set of equally spaced points on the unit circle defined in Equation 2.10.

$$\mathcal{M}_{PSK} = \left\{ exp\left\{ \frac{j2\pi \cdot m}{M} \right\} \cdot m \ : \ m \in [0, M), \ m \in \mathbb{Z} \right\} \tag{2.10}$$

The additive white Gaussian noise (AWGN) channel is the most basic channel. The AWGN channel

is a zero mean complex-valued random process parameterized by the covariance matrix. Let W be a complex Gaussian random variable. For $W \sim CN(0, \Sigma)$, the probability distribution function of W is expressed in Equation 2.11.

$$p_W(w) = \frac{1}{2\pi|\Sigma|^{\frac{1}{2}}} exp\left\{-\frac{1}{2} w^* \Sigma^{-1} w\right\} \tag{2.11}$$

We will assume that the two dimensions of W are uncorrelated and have the same variance, $\Sigma = \sigma^2 I$. This assumption is reasonable, since we can decorrelate the noise [89]. In this situation, the probability distribution of W is given in Equation 2.12.

$$p_W(w) = \frac{1}{2\pi\sigma^2} exp\left\{-\frac{1}{2\sigma^2}\|w\|^2\right\} \tag{2.12}$$

The signal-to-noise ratio (SNR) is a measure of signal power to noise power. This ratio is typically expressed in decibels (dB) as shown in Equation 2.13, where P is the signal power and N_0 is the complex noise power.

$$SNR = 10 \log_{10} \left(\frac{P}{N_0}\right) \tag{2.13}$$

In our case, where the real and imaginary components of W are uncorrelated, we have $N_0 = 2\sigma^2$.

For our research we will focus on radio communication systems that would use either PAM, QAM, or PSK as the modulation type. Also, we will only consider the AWGN channel. We believe that this restricted problem space that will demonstrate the utility of our proposed work.

2.2 Pattern Recognition

Pattern recognition is an area of machine learning that develops algorithms to understand features of objects in order to classify similar objects together. An example pattern recognition problem is the task of determining the correct digit from an image of a handwritten digit [90–93]. Most pattern recognition algorithms involve a training step and a testing step. In training, we are given a representative training set of instances to use in learning a decision rule. We apply this learned rule to our test set of instances to evaluate how well the algorithm performs.

There are two types of learning procedures to perform the training step. In *supervised* learning, each instance of the training set consists of the data and the correct label for that instance [90]. Classification problems use this type of learning. During the test step, for each instance in the test set, we would present an instance to the learned classification rule. The rule would predict the class label of that instance. We can measure the accuracy of the rule by calculating the fraction of the instances whose predicted label matches the known label. In the handwritten digits example,

an instance in the training set would be the pixel image and the correct digit associated with that image. We would learn a rule to predict the digit given an image. The test step would compare the digit predictions with the truth.

In *unsupervised* learning, the instances of the training sets consist only of the raw data [94]. A common algorithm of this type is clustering. Clustering assumes a model for the data, and the training step estimates the parameters for the model that is best explained by the data. After training is complete, the result is a model for the data. During testing, we give a test instance to the model and observe the response. For example, suppose that the model was a mixture of Gaussian densities. Then, the model might give the probability that an instance is generated by each Gaussian density in the mixture model. An example of unsupervised learning is modeling the eruptions of the Old Faithful geyser at Yellowstone National Park [90].

2.2.1 Classification Theory

For a classification problem, we attempt to learn an algorithm that takes an instance and predicts the class label for that instance. Let $\Omega = \{\omega_1, \omega_2, ..., \omega_N\}$ be the set of N class labels for our problem. The data instance is represented as a D-dimensional feature vector. Let $\mathcal{X}$ be the problem feature space. We are trying to learn a function $f : \mathcal{X} \rightarrow \Omega \times \mathbb{R}$ that maps an element in the feature space to a class label and a real-valued number. The class label is the prediction on the feature vector. The real-valued number is a confidence in, or the probability of, the prediction being correct.

If we know the class conditional probability density, then the problem becomes a hypothesis test [95]. A common method to determine the best hypothesis is to determine which hypothesis is most probable [96]. The *maximum a posterior* (MAP) probability hypothesis rule for $x \in \mathcal{X}$ is shown in Equation 2.14 - 2.16.

$$\begin{aligned} \hat{\omega} &= \arg\max_{\omega \in \Omega} Pr\{\ \omega \mid x\ \} && (2.14) \\ &= \arg\max_{\omega \in \Omega} \frac{Pr\{\ x \mid \omega\ \} Pr\{\ \omega\ \}}{Pr\{\ x\ \}} && (2.15) \\ &= \arg\max_{\omega \in \Omega} Pr\{\ x \mid \omega\ \} Pr\{\ \omega\ \} && (2.16) \end{aligned}$$

The posterior probability is typically difficult to calculate. Applying Bayes' Rule, we can express the posterior probability as the product of the likelihood and prior probabilities divided by the evidence probability. Notice that the evidence probability does not depend on the class label. Thus, we do not need to include that probability in the hypothesis test as seen in Equation 2.16. In some cases, we can assume an equal prior distribution on the class labels. In these cases, the best hypothesis decision rule becomes the *maximal likely* decision, shown in Equation 2.17

$$\hat{\omega} = \arg\max_{\omega \in \Omega} Pr\{\ x \mid \omega\ \} \quad (2.17)$$

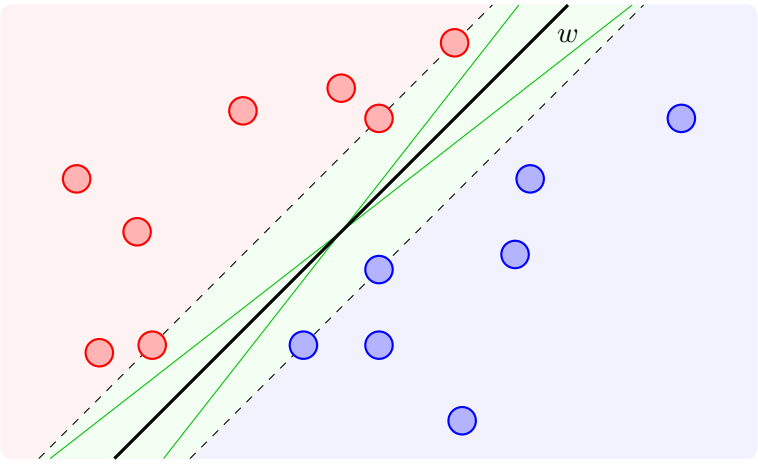


Figure 2.4: Example feature space for a binary classification problem where a support vector machine can find a seperable hyperplane to linearly discriminate the two classes. The hyperplane w shown as a black line provides that maximum margin, however other seperating hyperplanes can be found in the green region, where we depict two such hyperplanes.

2.2.2 Binary Class Label Classifiers

A classification problem with two class labels is the most basic classification problem. It is often easier to develop a two-class classification algorithm than it is to develop a classification algorithm over a larger set [97]. For our discussion, let us assume that $\Omega = \{-1, +1\}$. Given that assumption, the binary classifier is a function $f : \mathcal{X} \to \mathbb{R}$, where the sign of $f(x)$ is the predicted class label and the magnitude of $f(x)$ is the measure of confidence. The support vector machine (SVM) is an example of a popular classifier that only applies to binary class labels. In fact, a large portion of the work on the theory of learning has focused on binary classifiers [98]. We will describe a SVM classifier in more detail next.

2.2.2.1 Support Vector Machine Description

The support vector machine is a supervised learning algorithm that is a common classifier algorithm choice for binary classification problems. A SVM creates a hyperplane decision function from a training set of instances that maximizes the margin surrounding this hyperplane [90, 91, 99, 100]. We first describe the intuition behind the previous sentence, and then we discuss the mathematical theory of a SVM classifier. We assume that a training or test instance x belongs to a D-dimensional feature space, i.e. $x \in \mathcal{X} \subseteq \mathbb{R}^D$. For each x, we assume that we know the corresponding binary class label $y \in \{-1, +1\}$.

For example, consider Figure 2.4, which depicts a hypothetical set of training instances that belong to either a red or blue class in some feature space. In this example, the instances can be linearly

separated by a hyperplane w, which is represented by the black line in the figure. In other words, on this training set the classifier correctly classifies each of the training instances to the left of w to the red class and any training instance to the right of w to the blue class. However, there exist infinitely many hyperplanes that produce perfect classification on training instances by selecting a vector in the area shaded green in the figure. The green lines in Figure 2.4 illustrates two alternative possible hyperplanes that would possible zero training errors. The goal of the SVM training process is to determine the best possible separating hyperplane.

Suppose we treat the red class label as the '-1' class and the blue class label as the '+1' class in the binary classification problem. In this separable case, the red instances satisfy the equation shown in Equation 2.18, where $\langle \cdot, \cdot \rangle$ is the dot-product between two vectors and b is a scalar constant. Likewise, the blue instances satisfy the equation shown in Equation 2.19. Note that the points in the example feature space in Figure 2.4 that lie on one of the dashed lines correspond to points that satisfy either Equation 2.18 and Equation 2.19 with equality.

$$\langle x, w \rangle + b \leq -1 \qquad \text{for } y = -1 \tag{2.18}$$

$$\langle x, w \rangle + b \geq +1 \qquad \text{for } y = +1 \tag{2.19}$$

We can compactly represent both Equation 2.18 and Equation 2.19 in a single expression by incorporating the class label y as shown in Equation 2.20.

$$y\left(\langle x, w \rangle + b\right) \geq 1 \tag{2.20}$$

Again referring to Figure 2.4, the dashed lines represent the closest training instance to the hyperplane. The margin is the distance, equal to $1/\|w\|$, between the hyperplane w and the closest training instance to the hyperplane [99]. The training process of a SVM selects the hyperplane that maximizes the margin. Alternatively, the SVM selects the hyperplane that solves the optimization problem that minimizes the norm $\|w\|^2$ subject to the constraints $y_n\left(\langle x_n, w \rangle + b\right) \geq 1$ for all pairs of training instance and corresponding class label (x_n, y_n) in the training set [100].

In many cases, a hyperplane that can discriminate between our two classes with zero training errors does not exist. A more likely scenario is shown in Figure 2.5, where we shade the background red or blue to correspond to the decision region for both classes. We see that there are blue instances in the red-shaded region and also red instances in the blue-shaded region. This scenario makes the previously described optimization problem to find a hyperplane impossible to solve.

We can relax the problem and introduce slack variables in order to find the maximum soft margin, where we allow some number of training errors and these errors are penalized [90]. With this problem relaxation, an instance satisfies either Equation 2.21 or Equation 2.22 if that instance belongs to the '-1' or the '+1' classes, respectively, where ξ is the slack variable that must be

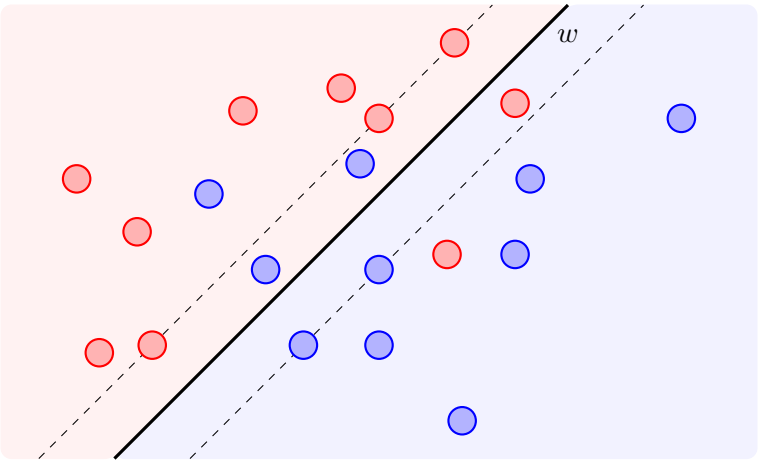


Figure 2.5: Example feature space for a binary classification problem where a support vector machine *cannot* find a seperable hyperplane to linearly discriminate the two classes for all instances.

nonnegative.

$$\langle x, w\rangle + b \leq -(1-\xi) \qquad \text{for } y = -1 \tag{2.21}$$

$$\langle x, w\rangle + b \geq +(1-\xi) \qquad \text{for } y = +1 \tag{2.22}$$

Thus, we wish to maximize the margin while minimizing the number of training errors. Equivalently, we can formulate this optimization problem by minimizing both the norm of w and the sum of the slack variables ξ_n for the N training instances as shown in Equation 2.23, subject to $y_n\left(\langle x_n, w\rangle + b\right) \geq (1-\xi_n)$ and $\xi_n \geq 0$ for all $n \in [1, N]$, where C controls the trade-off between model complexity and the number of training errors [90, 101, 102]. We see that if we increase the value of C, then we are applying a higher cost or penalty towards training errors.

$$\frac{1}{2}\|w\|^2 + C\sum_{n=1}^{N}\xi_n \tag{2.23}$$

This optimization problem is a quadratic programming problem. We introduce Lagrange multipliers to form the primal Lagrangian equations as shown in Equation 2.24, where $\alpha_n \geq 0$ and $\mu_n \geq 0$ for all n.

$$L_P = \frac{1}{2}\|w\|^2 + C\sum_{n=1}^{N}\xi_n - \sum_{n=1}^{N}\alpha_n\left\{y_n\left(\langle x_n, w\rangle + b\right) - (1-\xi_n)\right\} - \sum_{n=1}^{N}\mu_n\xi_n \tag{2.24}$$

We wish to minimize L_p and do so by setting to zero the derivative of L_p with respect to w, b, and

ξ_n for all n, which results in Equation 2.25, Equation 2.26, and Equation 2.27, respectively.

$$w = \sum_{n=1}^{N} \alpha_n y_n x_n \tag{2.25}$$

$$0 = \sum_{n=1}^{N} \alpha_n y_n \tag{2.26}$$

$$\alpha_n = C - \mu_n \tag{2.27}$$

From these results, we can create the Wolfe dual problem [90, 91, 99, 100] as shown in Equation 2.28.

$$L_D = \sum_{n=1}^{N} \alpha_n - \frac{1}{2} \sum_{m=1}^{N} \sum_{n=1}^{N} \alpha_m \alpha_n y_m y_n \langle x_m, x_n \rangle \tag{2.28}$$

We maximize L_D subject to the constraints that $0 \leq \alpha_n \leq C$ and $0 = \sum_n \alpha_n y_n$ and additionally the Karush-Kuhn-Tucker conditions provide the constraints in shown in Equation 2.29, Equation 2.30, and Equation 2.31.

$$\alpha_n \left\{ y_n \left(\langle x_n, w \rangle + b \right) - (1 - \xi_n) \right\} = 0 \tag{2.29}$$

$$\mu_n \xi_n = 0 \tag{2.30}$$

$$\left\{ y_n \left(\langle x_n, w \rangle + b \right) - (1 - \xi_n) \right\} \geq 0 \tag{2.31}$$

By solving this problem and satisfying all of these constraints, we produce the hyperplane weight vector solution, w^*, shown in Equation 2.32, which is the sum of training instances with a corresponding positive Lagrange multiplier α_n.

$$w^* = \sum_{n:\alpha_n > 0} \alpha_n y_n x_n \tag{2.32}$$

These training instances with $\alpha_n > 0$ are called the support vectors, since they exactly meet the constraint in Equation 2.31 [91].

We can classify a test instance x to the positive or negative class by determining which side of the hyperplane on which that instance exists, which is expressed in Equation 2.33 where $\hat{y}$ is the predicted class label, which is set to the sign of the dot-product between the hyperplane and the instance (plus the bias). By substituting in the expression of w^*, we see that this classification process is a function of the weighted sum of dot-products between x and each of the support

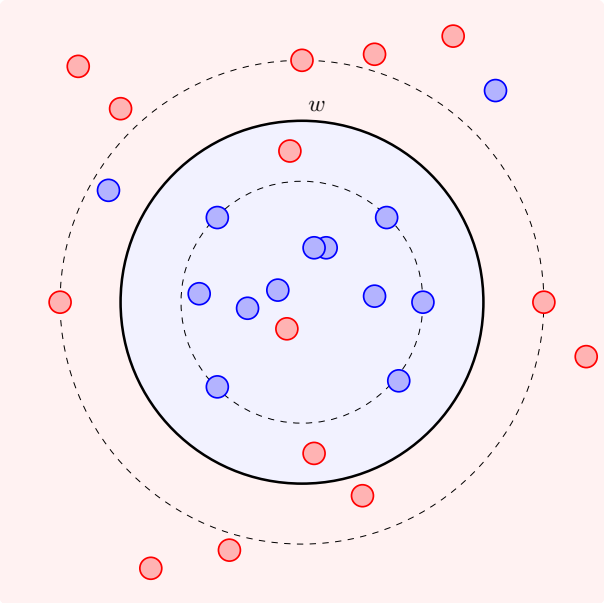


Figure 2.6: Example feature space for a binary classification problem where a support vector machine *cannot* find a seperable hyperplane in the original feature space that can linearly discriminate the two classes for all instances, but rather we need to find a mapping to another feature space that can be linearly discriminated.

vectors, as shown in Equation 2.34.

$$\hat{y} = \texttt{sign}\left(\langle w^*, x\rangle + b\right) \tag{2.33}$$

$$= \texttt{sign}\left(\sum_{n=1}^{N} \alpha_n y_n \langle x, x_n\rangle + b\right) \tag{2.34}$$

Up to this point, we have described the linear SVM classifier. However, in many cases, a hyperplane that satisfies the constraints of the described optimization cannot be found from the instances in the original feature space, and a nonlinear approach must be taken. Such an example is shown in Figure 2.6, where a nonlinear SVM must be used because a hyperplane clearly cannot separate these instances in a linear fashion.

The idea behind a nonlinear SVM is to project the instances into a higher-dimensional feature space, where in this new feature space a linearly separable hyperplane can be found (possibly with slack variables) for the two classes [91, 103, 104]. With this in mind, we would like to change the dot-product $\langle x, x_n\rangle$ in Equation 2.34 to $\langle \phi(x), \phi(x_n)\rangle$, where ϕ is the mapping to the higher-dimensional feature space. The problem with this approach is that the dot-product in the higher-dimensional feature space might be have an infinite number of dimensions and is computationally impossible. However this problem can be avoided by using the following "kernel-trick" [90, 91, 99–104].

Let the kernel, k, be a mapping $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ such that $k(x_1, x_2) = \langle \phi(x_1), \phi(x_2) \rangle$. With this kernel, we can express the classification decision rule for a nonlinear SVM by Equation 2.35.

$$\hat{y} = \texttt{sign} \left(\sum_{n=1}^{N} \alpha_n y_n k(x, x_n) + b \right) \tag{2.35}$$

This "kernel-trick" allows us to operate in the higher-dimensional feature space without actually mapping the instances into that space. However, for this trick to work, we must restrict ourselves to kernels that satisfy Mercer's Theorem.

Mercer's Theorem states that a kernel k can be expressed as a dot-product of ϕ, i.e. $k(x_1, x_2) = \langle \phi(x_1), \phi(x_2) \rangle$, if the kernel is symmetric, as shown in Equation 2.36, and positive definite, as shown in Equation 2.37, which holds for any bounded function g that is finite [103, 104].

$$k(x_1, x_2) = k(x_2, x_1) \tag{2.36}$$

$$\int g(x)^2 dx < \infty \Rightarrow \int k(x_1, x_2) g(x_1) g(x_2) dx_1 dx_2 \geq 0 \tag{2.37}$$

The following is a list of common kernels:

Kernel	$k(x_1, x_2)$	Parameters
Linear	$\langle x_1, x_2 \rangle$	
Polynomial of degree d	$(\langle x_1, x_2 \rangle + c)^d$	c and d
Gaussian radial basis function	$exp\left\{-\frac{1}{2\sigma^2}\|x_1 - x_2\|^2\right\}$	σ^2

The SVM classifier can provide good correct classification accuracy. However, in training a SVM, we must try values of C and kernel types, along with the parameters of these kernel to find the best configuration for the classification problem.

2.2.3 Multiple Class Label Classifiers

Many classification problems involve more than two class labels. These problems are referred to as multiclass or multi-label problem. Many learning algorithm can naturally handle more than two class labels. A naïve Bayes classifier, and more generally a Bayesian network, can represent a joint probability distribution [90]. Thus, they can handle any number of class labels. A decision tree is another example of a classification algorithm that can handle the multiclass problem [105].

Alternatively, multiclass problems can be decomposed into a set of binary classification problems [97]. A more natural way to handle an N label problem ($N > 2$) is to decompose the problem into a set of binary classification problems. A multiclass classification problem is decomposed by remapping the class labels to the binary set, creating a larger ensemble classifier. Thus, the task

becomes developing the appropriate label remapping for each class onto the binary classifiers. The ensemble classifier must combine the outputs of the underlying classifiers in a way that produces a prediction from the multiclass set. Ensemble classifiers tend to have better accuracy than the individual classifiers that make up the ensemble [106]. Two popular approaches to creating ensemble classifiers are one-versus-all (OVA) and one-versus-one (OVO) [90].

For N class labels, an OVA ensemble consists of N classifiers. The n^{th} classifier is trained to return +1 one when presented an instance of class label n and -1 otherwise. Given a test instance, each classifier produces an output value on the real line. The sign of the output is the label. The magnitude of the output value is a measure of confidence on that label. The decision rule for OVA is to select the class associated with the classifier that produces the largest positive value. An OVO ensemble creates a classifier for every possible pair of class labels. For a problem with N class labels, there are $\binom{N}{2}$ classifiers in an OVO ensemble. For each pair of class labels, one class is trained to a positive value. The other class is trained to a negative value. The remaining other class labels are not used in training. In OVO, the ensemble classifier will select the class label by a majority vote from all the base classifiers.

Error-correcting output codes (ECOC) can also be used to build ensemble classifiers where each class label corresponds to an unique binary string of length D [107]. Thus, the ensemble has D binary classifiers. During the training phase, the expected output of the d^{th} classifier equals the bit value in the d^{th} position of the associated binary string for the class label of the training instance. The classification of a test instance is the class for which its associated binary string is the nearest in Hamming distance to the string produced by the binary classifiers.

Dietterich extended the binary string idea, by proposing to select error-correcting codewords as the unique binary strings [105]. The set of error-correcting codewords has a minimum Hamming distance, d_{min}, that separates all pairs of codewords. The number of bit classification errors we can correct is $\lfloor \frac{d_{min}-1}{2} \rfloor$. In essence, the ensemble can compensate for individual misclassifications up to a point. Allwein later added an "ignore" element to the binary set from which the unique strings are created [97]. With the addition of this element, the unique strings that represent a class label are no longer binary. However, the individual classifiers for the ensemble are still binary classifiers. This allows us to describe the previously mentioned OVA and OVO ensembles in the ECOC framework. However, to the best of our knowledge, there does not exist a proof to show that the two ensembles approaches always produce the same results. We present these proofs in Appendix A.

Chapter 3

Narrowband Signal Detection Algorithm

In this chapter, we present a novel approach to blind signal detection that jointly estimates the amplitude, bandwidth, and center frequency of narrowband signals in a received wideband signal. This parameter estimation allows us to more accurately describe our received signal in order to better determine the total number of signals present. We use iterative methods to find signals and adjust signal parameters to minimize the measured error and log-likelihood expression.

Our narrowband signal detection (NSD) algorithm has a probability of detection of 67.3%, 90.1%, 93.9%, and 94.3% for SNR values of 0 dB, 5 dB, 10 dB, and 15 dB, respectively, while producing a probability of false alarm of 89.5%, 57.8%, 42.6%, and 37.2% for SNR values of 0 dB, 5 dB, 10 dB, and 15 dB, respectively. Our experiments show that our NSD algorithm outperforms the commonly used energy detection algorithm.

3.1 Model

In our model, we assume that the received signal is a linear combination of $M \geq 0$ unknown digitally modulated transmitted narrowband signals and noise. In the frequency domain, we assume that each transmitted signal can be approximated by a sinc function and the received wideband signal is a mixture model of sinc functions. Our algorithm determines the number of transmitted signals present in the received signal, and for each signal detected, we estimate the amplitude, bandwidth, and center frequency. Next, we define the received signal in the context of our model.

We assume that our received signal, r, is the summation of $M \geq 0$ unknown transmitted signals x_m that is corrupted by complex Gaussian noise, $w = w_I + jw_Q$, where w_I is the in-phase dimension of the noise, and w_Q is the quadrature phase dimension of the noise. The received signal is a discrete

time domain sampled signal, which we use k as the sample index. We assume independence between each noise sample, and identically distributed, where $w_I[k],\ w_Q[k] \overset{\text{iid}}{\sim} N(0, \sigma^2)$ for all k. The received signal r is defined in Equation 3.1.

$$r[k] = \sum_{m=1}^{M} x_m[k] + w[k] \tag{3.1}$$

The algorithm we developed operates in the frequency domain, which requires us to use the discrete Fourier transform (DFT) on the time domain received wideband signal, where the response spans the frequency spectrum from 0 to 2π radians. We let L be the DFT size, which also indicates the number of frequency bins contained in our frequency domain signals, and implies that a single frequency bin has a frequency resolution of $\frac{2\pi}{L}$.

The frequency response of the received signal, R, is the result of a L-point DFT operation on the time domain received signal. Note that we time-average our DFT to allow the ensemble average transmitted message to be seen. The time-averaged DFT of the received signal is defined in Equation 3.2, where ℓ is frequency bin index, and T is the number of L sample blocks we average.

$$R[\ell] = \frac{1}{T} \sum_{t=0}^{T-1} \underbrace{\sum_{k=0}^{L-1} r[t \cdot L + k]\ exp\left\{\frac{-j2\pi k\ell}{L}\right\}}_{L\text{-point DFT of } t^{th} \text{ block}} \tag{3.2}$$

Let the noise-free signal be the summation of the M time domain transmitted signals, $\sum x_m$. We define $A[\ell]$ to be the time-averaged amplitude of the DFT response of this noise-free signal, as shown in Equation 3.3. We define $V[\ell]$ be the amplitude of $R[\ell]$, as shown in Equation 3.4, which includes the noise.

$$A[\ell] = \left| \frac{1}{T} \sum_{t=0}^{T-1} \sum_{k=0}^{L-1} \left(\sum_{m=1}^{M} x_m[t \cdot L + k] \right)\ exp\left\{\frac{-j2\pi k\ell}{L}\right\} \right| \tag{3.3}$$

$$\begin{aligned} V[\ell] &= \Big|R[\ell]\Big| \\ &= \left| \frac{1}{T} \sum_{t=0}^{T-1} \sum_{k=0}^{L-1} \left(\sum_{m=1}^{M} x_m[t \cdot L + k] + w[t \cdot L + k] \right)\ exp\left\{\frac{-j2\pi k\ell}{L}\right\} \right| \end{aligned} \tag{3.4}$$

Note that we use lower case variables to represent time domain signals and upper case variables to represent frequency domain signals. In addition, when the meaning is obvious, we use a subscript to a variable to indicate an indexing, *e.g.* $R_\ell \leftrightarrow R[\ell]$.

We model the frequency domain amplitude of the noise-free signal, A, as mixture of absolute value sinc functions, where each sinc function in the mixture represents a transmitted signal. For each transmitted signal, in the ideal case, the transmitter uses a rectangular pulse shape filter. With

this ideal assumption, we expect to see the shape of the absolute value of a sinc function, because the sinc function is the Fourier transform pair of the rectangular pulse [84].

Each frequency bin of the received signal, R_ℓ, is a complex Gaussian random variable. If $M > 0$, then R_ℓ will have a nonzero mean due to the summation of transmitted signals. A Rician distribution describes the amplitude of a nonzero mean complex Gaussian random variable. By Equation 3.4, if $M > 0$, we see that V_ℓ is a Rician random variable. With this fact, we can define the log-likelihood function of our received signal given the estimated noise variance, σ^2, and the estimated noise-free amplitude, A_ℓ, for each frequency bin $\ell = 0, 1, ..., L-1$.

The log-likelihood function of the frequency response amplitude of the received signal given our parameter set is expressed in Equation 3.5, where the parameter set $\theta = \left\{A_0, A_1, ..., A_{L-1}, \sigma^2\right\}$.

$$\log Pr\{\ V \mid \theta\ \} = \sum_{\ell=0}^{L-1} \frac{1}{2} \log\left(\frac{V_\ell}{\sigma^2}\right) - \frac{1}{2\sigma^2}\left(V_\ell - A_\ell\right)^2 - \frac{1}{2}\log\left(2\pi A_\ell\right) \tag{3.5}$$

This log-likelihood function can serve as a measure of how well we are estimating our parameter set. In addition, this equation will be the objective function for optimizations we describe below. First, for completeness, we present the derivation of Equation 3.5.

3.1.1 Log-Likelihood Equation Derivation

We intend to develop an algorithm that takes in a received wideband signal containing an unknown number of transmitting narrowband signals. Our algorithm works on the frequency response amplitude of the L-point DFT of the received signal, which we previously defined as V_ℓ, for $\ell = 0, 1, ..., L-1$. Thus, our input data set is $V = \{\ V_0, V_1, ..., V_{L-1}\ \}$. The log-likelihood function of receiving the data set V given our parameters θ is the log-conditional joint probability of V given θ.

By the chain rule, the conditional joint probability $Pr\{\ V \mid \theta\ \}$ is expressed in Equation 3.6.

$$\begin{aligned} Pr\{\ V \mid \theta\ \} &= Pr\{\ V_0 \mid \theta\ \} \prod_{\ell=1}^{L-1} Pr\{\ V_\ell \mid \theta, V_0, ..., V_{\ell-1}\ \} && (3.6) \\ &\leq \prod_{\ell=0}^{L-1} Pr\{\ V_\ell \mid \theta\ \} && (3.7) \end{aligned}$$

If we assume that the amplitudes of each frequency bin are independent, then we can bound the probability as seen in Equation 3.7. Note that this assumption is clearly incorrect by using the sinc approximation in our model. However, there are several justifications for this bound. First, it is analytically difficult to describe the joint probability distribution function (pdf) of L correlated Rician random variables. In fact, recent papers in the literature describe the joint pdf for at most

three Rician random variables with constraints on the correlation matrix [108]. Second, since the algorithm we develop to maximize the log-likelihood function relies on iterative optimization, we expect that an upper bound on the probability to be sufficient.

The log-likelihood function is the logarithm of the conditional joint probability, as shown in Equation 3.8. We approximate the modified Bessel function of the first kind as order zero $I_0(z)$ with $exp(z)/\sqrt{2\pi z}$ and $\log I_0(z)$ as $z - \frac{1}{2}\log(2\pi z)$. We will now derive the log-likelihood function.

$$\begin{aligned}
\log\ Pr\{\ V \mid \theta\ \} &\leq \log \prod_{\ell=0}^{L-1} Pr\{\ V_\ell \mid \theta\ \} && (3.8)\\
&= \sum_{\ell=0}^{L-1} \log\ Pr\{\ V_\ell \mid \theta\ \} && (3.9)\\
&= \sum_{\ell=0}^{L-1} \log\left[\ \frac{V_\ell}{\sigma^2} exp\left\{-\frac{1}{2\sigma^2}\left(V_\ell^2 + A_\ell^2\right)\right\} \cdot I_0\left(\frac{V_\ell \cdot A_\ell}{\sigma^2}\right)\ \right] && (3.10)\\
&= \sum_{\ell=0}^{L-1} \log\left(\ \frac{V_\ell}{\sigma^2}\ \right) - \frac{1}{2\sigma^2}\left(V_\ell^2 + A_\ell^2\right) + \log\left[\ I_0\left(\frac{V_\ell \cdot A_\ell}{\sigma^2}\right)\ \right] && (3.11)\\
&= \sum_{\ell=0}^{L-1} \frac{1}{2}\log\left(\ \frac{V_\ell}{\sigma^2}\ \right) - \frac{1}{2\sigma^2}\left(V_\ell - A_\ell\right)^2 - \frac{1}{2}\log\left(2\pi \cdot A_\ell\right) && (3.12)
\end{aligned}$$

In Equation 3.8, we start with the bounded relationship for the conditional joint probability. We use the properties of logarithms to produce Equation 3.9. In Equation 3.10, we substitute in the Rician distribution expression for $Pr(V_\ell|\theta)$. We again use the properties of logarithms to produce Equation 3.11. We substitute in our approximation for the Bessel function which with algebraic manipulation yields Equation 3.12, our final expression for the log-likelihood function.

In order to calculate the log-likelihood expression, we need to know the noise variance, σ^2, and the noise-free signal frequency response amplitude, A_ℓ, at each frequency bin $\ell = 0, 1, ..., L-1$. We will derive the maximum likely estimate for the noise variance, for which the objective function $\mathcal{J}$ to maximize is the log-likelihood function.

$$\begin{aligned}
\frac{\partial}{\partial\sigma^2}\mathcal{J} &= \frac{\partial}{\partial\sigma^2}\sum_{\ell=0}^{L-1} \frac{1}{2}\log\left(\ \frac{V_\ell}{\sigma^2}\ \right) - \frac{1}{2\sigma^2}\left(V_\ell - A_\ell\right)^2 - \frac{1}{2}\log\left(2\pi \cdot A_\ell\right) && (3.13)\\
&= \sum_{\ell=0}^{L-1} -\frac{1}{2\sigma^2} + \frac{1}{2}\left(\frac{1}{\sigma^2}\right)^2 \left(V_\ell - A_\ell\right)^2 && (3.14)
\end{aligned}$$

We start the maximization by expressing the objective function with the log-likelihood function, as shown in Equation 3.13. We take the derivative of our objective function with respect to σ^2, as

shown in in Equation 3.14. We set the derivative to zero and rearrange terms, which results in the maximum likely estimate for σ^2 is shown in Equation 3.15.

$$\sigma^2 = \frac{1}{L}\sum_{\ell=0}^{L-1}(V_\ell - A_\ell)^2. \tag{3.15}$$

By definition, A_ℓ is the amplitude, in the frequency domain, of the noise-free signal.

$$\begin{aligned} A[\ell] &= \left|\sum_{k=0}^{L-1}\left(\sum_{m=1}^{M} x_m[k]\right)\ exp\left\{\frac{-j2\pi k\ell}{L}\right\}\right| && (3.16)\\ &= \left|\sum_{m=1}^{M} X_m[\ell]\right| && (3.17)\\ &= \left[\ \left(\sum_{m=1}^{M} X_m^*[\ell]\right)\left(\sum_{m=1}^{M} X_m[\ell]\right)\ \right]^{\frac{1}{2}} && (3.18)\\ &= \left[\ \sum_{m=1}^{M}\|X_m[\ell]\|^2 + \sum_{m=1}^{M}\sum_{\substack{j=1\\ j\neq m}}^{M} X_m^*[\ell]X_j[\ell]\ \right]^{\frac{1}{2}} && (3.19) \end{aligned}$$

Equation 3.16 repeats the definition of A_ℓ from Equation 3.3. The DFT is a linear operation. Therefore, in Equation 3.17, we can expression A_ℓ as the amplitude of the summation of X_m for all m, where X_m is the DFT of the signal x_m. In Equations 3.18 and 3.19, we algebraically manipulate the equation.

Equation 3.19 gives an exact expression for the amplitude A_ℓ, where $*$ denotes the complex conjugate. However, to simplify the expression we ignore the cross-product terms between the different signals. With this simplification, we approximate A_ℓ by the square root of the summation of the amplitudes of all M known signals, which is seen in Equation 3.20.

$$A_\ell \approx \sqrt{\sum_{m=1}^{M}\|X_m[\ell]\|^2} \tag{3.20}$$

3.2 Signal Detection Algorithm

We assume that the receiver receives a signal that contains an unknown number of transmitted signals. We wish to estimate the number of transmitted signals, M, the amplitude, α_m, bandwidth, B_m, and center frequency, f_m, of each transmitted signal, for $m = 1, 2, ..., M$. Figure 3.1 illustrates this set of parameters for an ideal transmitted signal x_m, where the vertical axis is the amplitude,

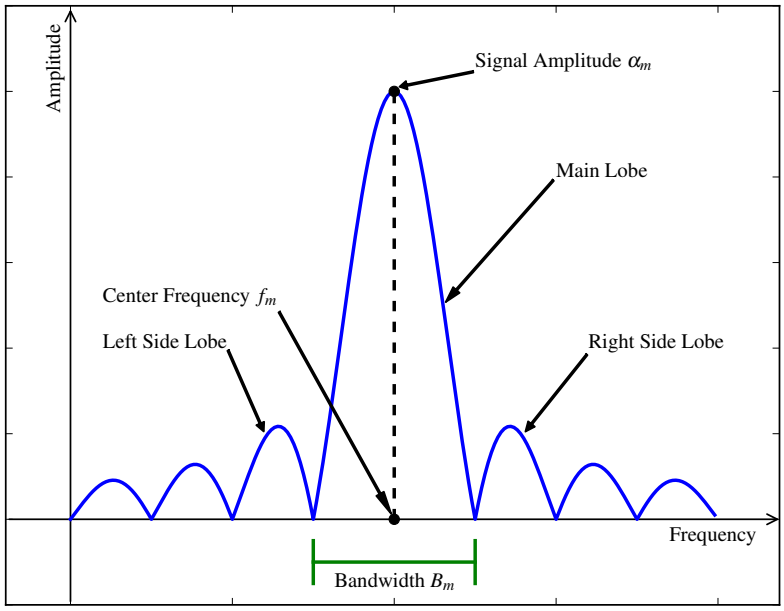


Figure 3.1: Illustration of the relationships between an ideal transmitted signal and our model parameters, where the amplitude is the peak of the signal located at the center frequency, and the bandwidth is equal to the amount of frequency contained in the main lobe of the signal.

and the horizontal axis is the frequency. The blue line in Figure 3.1 represents the amplitude response of a sinc function, which is the Fourier transform of the ideal rectangular pulse. We indicate with arrows the locations of the center frequency, and the signal amplitude. We define the signal bandwidth as the amount of frequency contained within the main lobe of the sinc function, which is indicated in Figure 3.1 by the green marker, and an arrow points to the main lobe. We also indicate the right and left side lobes with arrows, which are apart of the parameter estimation discussed later.

Figure 3.2 illustrates an example scenario that we use for purposes of explanation throughout this section. The wideband signal contains three separate narrowband signals. The parameters for signal x_1 are $\alpha_1 = 2A$, $B_1 = \frac{2}{5}\pi$, and $f_1 = \frac{15}{16}\pi$. The parameters for signal x_2 are $\alpha_2 = A$, $B_2 = \frac{2}{5}\pi$, and $f_2 = \frac{\pi}{4}$. The parameters for signal x_3 are $\alpha_3 = A$, $B_3 = \frac{2}{5}\pi$, and $f_1 = \frac{11}{8}\pi$. These signals were synthetically generated at very high SNR of 100 dB. The blue line in Figure 3.2 shows the amplitude response of the DFT of the received signal.

Energy detection is a common approach for blind signal detection [109]. In this approach, any frequency bin with an amplitude value larger than a threshold is considered a signal *e.g.* the red line in Figure 3.2. Suppose in our example there are L' DFT bins that are above our threshold. Clearly we would detect the three signals present. However, we would incorrectly assert that $L'-3$ frequency bins are also a signal. This results in a false alarm rate of $\frac{L'-3}{L'}$. It should be noted that

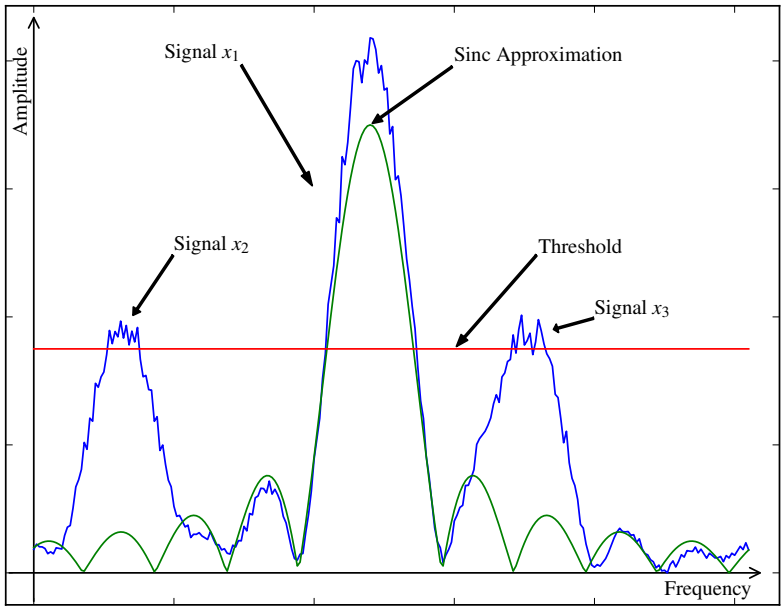


Figure 3.2: Example of a received signal that contains three transmitted signals with an overlay of the sinc approximation for signal x_1.

the energy detector approach is the implementation of the maximum likelihood decision rule for detecting purely complex exponential signals in noise [110]. As a result, this approach is commonly used for signal detection.

Since we are interested in signals that contain information, we develop a signal detection algorithm that detects digitally modulated signals, and not just purely complex exponential signals. Digitally modulated signals have a pulse function that is defined over the symbol duration. In the ideal case, a rectangular pulse is used, which corresponds to a sinc function in the frequency domain. In Figure 3.2, the absolute value of the sinc function is plotted in green, where $\text{sinc}(x) = sin(\pi x)/(\pi x)$.

Our approach uses the sinc function as an approximation for an unknown signal, which is used to estimate the center frequency and bandwidth. We define the sinc approximation for the transmitted signal, x_m, in Equation 3.21, where ℓ is the DFT frequency bin index, L is the number DFT bins, ℓ_m is the center frequency offset for x_m, and K_m is the number of samples per symbol for x_m.

$$S_\ell(K_m, \ell_m) = \left| \text{ sinc} \left(\frac{K_m}{L} \left[\ell - \ell_m \right] \right) \right| \tag{3.21}$$

In our model, we also scale the approximation by α_m, which is one of the parameters that we are estimating. Note that the L-point DFT produces a periodic signal with a period of L, and the quantity $[\ell - \ell_m]$ in Equation 3.21 is truly $[\ell - \ell_m]$ modulo L. The center frequency offset, ℓ_m,

is a function of the center frequency and the frequency bin resolution of the DFT, as shown in Equation 3.22. The number of samples per symbol (NSPS), K_m, controls the main lobe size of the sinc approximation, and is inversely proportional to the bandwidth, as shown in Equation 3.23. Using the sinc approximation, we estimate the offset and NSPS directly, which indirectly gives us our estimates of the center frequency, and bandwidth, respectively, using the relationships in Equations 3.22 and 3.23.

$$\ell_m = f_m\left(\frac{L}{2\pi}\right) \tag{3.22}$$

$$K_m = \frac{4\pi}{B_m} \tag{3.23}$$

We assume that our received signal is the linear combination of M unknown sinc approximations. Using this assumption, we believe that we can more accurately model the amplitude frequency response of the received signal.

Algorithm 1 gives an overview of our signal detection routine FIND-SIGNAL-SET. The input to the

Algorithm 1 Find Signal Set – Input: Signal V – Output: Signal Set S

```
FIND-SIGNAL-SET( V )
1  S ← ∅
2  notdone ← True
3  while notdone
4        do S ← FIND-NEW-SIGNAL( V, S )
5           S ← ADJUST-PARAMETERS( V, S )
6           S ← MERGE-SIGNALS( S )
7           notdone ← FINISH?( V, S )

8  return S
```

routine is the amplitude frequency response, V, of the received signal. The output from the routine is a parameter set, S, that describes the detected transmitted signals. First, we try to find a single new signal to add to our set S. Second, we adjust the parameters of all the signals in S. Third, we merge signals together, if the log-likelihood function value increases as a result of the merge. Finally, we determine whether or not to continue with another iteration.

3.2.1 Subroutine: FIND-NEW-SIGNAL(V, S)

This subroutine contains four steps. We first create an interference cancellation signal from all the previously detected signals in S. Second, we create a residual signal by subtracting the interference

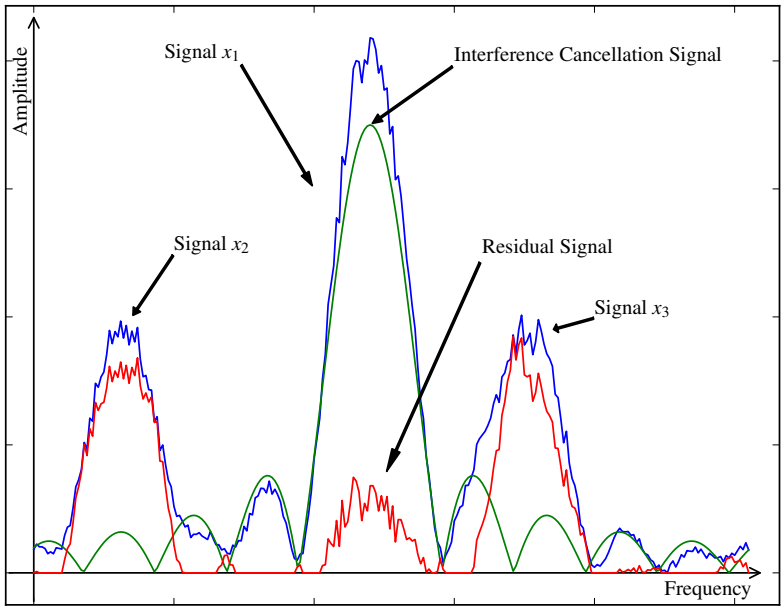


Figure 3.3: Example residual (red) signal created by removing the interference cancellation (green) signal from the (blue) received signal.

signal from the input signal V, where Figure 3.3 illustrates this process. Suppose we have already estimated signal x_1. In Figure 3.3, the blue line is the plot of V, the green line is our interference cancellation signal containing signal x_1, and the red line is our residual signal. The third step is to find a new signal from the residual signal. We complete this step by finding the parameters α_m, K_m, and ℓ_m for a sinc approximation for a new signal that best fits the residual, by minimizing the squared error between the residual signal and the sinc approximation. The final step of this subroutine is to determine whether or not to add this new signal to signal set.

3.2.2 Subroutine: Adjust-Parameters(V, S)

This subroutine tries to optimize the parameters in the signal set S in three different ways. In each iteration a single optimization method is applied. The subroutine cycles over all optimizations as the overall algorithm iterates. We describe the specific update procedures in Section 3.3.

The first method tries to improve the fit of the signals in S by re-estimating the parameters. We order the signals in our set by decreasing amplitude. Thus, we re-estimate the parameters of the strongest signals first. To start the re-estimation of a particular signal, we create the residual signal by canceling all the other signals in S from the input V. Next, we rerun the Find-New-Signal subroutine with this residual signal. However, we initialize the subroutine with the parameters of the particular signal of interest. This procedure is repeated for all signals in S.

The second method tries to improve the value of the log-likelihood function of the received signal given the parameters. Here we randomly select a signal in S and repeat the re-estimation step described in the first method. We recompute the log-likelihood value using the new parameters for this signal. We only allow the changes to be kept if the log-likelihood value improves. Otherwise, we ignore the changes. We repeat this process of randomly selecting signals until we reach a local optimum.

The third method also tries to improve the value of the log-likelihood function of the received signal. However, this method randomly selects a single parameter from $\{\alpha_m, K_m, \ell_m\}$ of a randomly selected signal in S. For that parameter, we calculate the gradient of the error function to update the parameter value. Again, we recompute the log-likelihood value using the new parameter. If the value improves, then we keep the update. Otherwise, we ignore the parameter update. Again, we repeat this process until we reach a local optimum.

3.2.3 Subroutine: Merge-Signals(S)

This subroutine determines whether or not we should merge two signals together. It is possible that, through the process of re-estimating signal parameters, two signals in S could essentially be representing the same transmitted signal. If two signals have center frequencies that are near to each other, then we create a merged signal. The merged signal is created by averaging each of the parameters of the two signals, and this merged signal is added to the signal set S.

At this point, we must decide the fate of the two original signals. Keep in mind that we wish to reduce the false alarm rate, without reducing the probability of detection. If the signals are really close to each other, then we remove the two original signals from S. However, if the two signals are close enough for use to create a merged signal, but not too close, then we decided to allow these original signals to remain in the set S. This case might initially increase the false alarm rate, but, however, we expect that our algorithm will eventually flush out the erroneous signals.

3.2.4 Subroutine: Finish?(V, S)

This subroutine tells the algorithm when to stop searching for new signals.

3.3 Update Method Derivation

The algorithm that we have described has three parameter updates: Δ_α, Δ_K, and Δ_ℓ, for the amplitude, NSPS, and offset, respectively. Each update adjusts a single parameter at a time, and is based on the (interference canceled) residual signal and the sinc approximation of a single transmitted signal based on the current parameter estimates for that signal. Suppose that our

parameter set S contains M signals and we want to update a parameter for signal x_m. The residual signal is created by subtracting from the received signal the sinc approximations of the other $M-1$ signals in S that are not the signal x_m. If our model is correct and the parameter estimation is perfect, then the residual signal will consist of the signal x_m in addition to noise.

3.3.1 Amplitude Update

The amplitude update is a function of the residual signal and the sinc approximation for the signal x_m. We adjust the amplitude our sinc approximation to closely match the residual signal, by minimizing the weighted squared error between the two signals to calculate the adjustment to the amplitude. We use a weighted squared error, because the signal energy is not uniformly distributed. For a sinc function, more than 90% of the signal energy is contained in the main lobe, and that percentage increases to over 95% when we include the two immediate side lobes. Therefore, we place more importance on correctly matching the amplitude of the sinc approximation to the residual signal in the region closer to the main lobe.

The weighted error function between the residual signal V and our sinc approximation is shown in Equation 3.24, where ℓ is the frequency bin index, w_ℓ is the weight value for the ℓ^{th} frequency bin, and recall that $S_\ell(K_m, \ell_m) = \left| sinc\left(\frac{K_m}{L}\left[\ell - \ell_m\right]\right)\right|$ is the sinc approximation.

$$E = \sum_{\ell=0}^{L-1} w_\ell \left[V_\ell - \Delta_\alpha S_\ell(K_m, \ell_m)\right]^2 \tag{3.24}$$

As mentioned, we place more importance on minimizing the error for the indices corresponding to the main lobe and two immediate side lobes of our sinc approximation. For these indices,we set w_ℓ to ten, and give the remaining weights a value of one. This weighting places an order of magnitude of relative importance between the two weight regions.

The derivative of the error function with respect to Δ_α is shown in Equation 3.25.

$$\frac{\partial}{\partial \Delta_\alpha} E \;=\; \sum_{\ell=0}^{L-1} w_\ell \cdot 2\left[V_\ell - \Delta_\alpha \cdot S_\ell(K_m, \ell_m)\right](-1) \cdot S_\ell(K_m, \ell_m) \tag{3.25}$$

We set Equation 3.25 to zero and perform algebraic manipulation to produce our expression for Δ_α, shown in Equation 3.26.

$$\Delta_\alpha \;=\; \frac{\sum_{\ell=0}^{L-1} w_\ell \cdot V_\ell \cdot S_\ell(K_m, \ell_m)}{\sum_{\ell=0}^{L-1} w_\ell \cdot S_\ell(K_m, \ell_m)^2} \tag{3.26}$$

The equation for Δ_α makes intuitive sense. The numerator is a weighted inner product between the residual signal and the sinc approximation. The denominator calculates the weighted energy

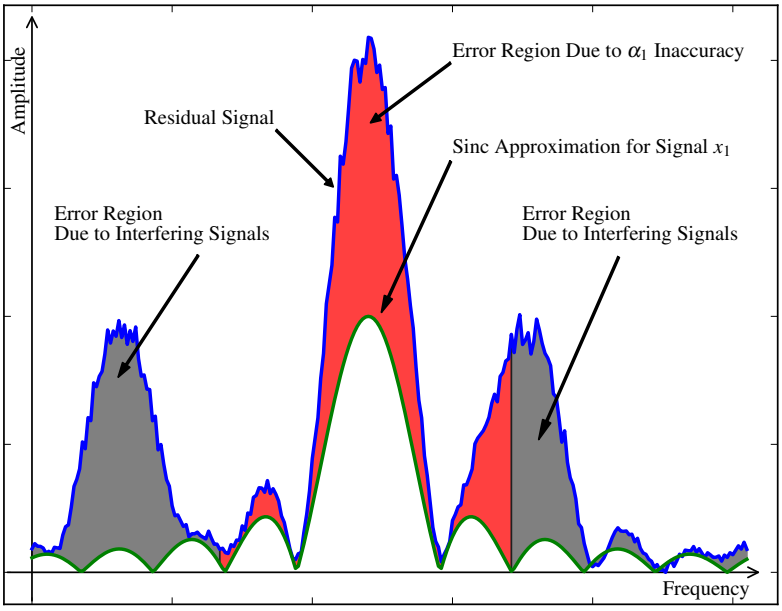


Figure 3.4: Example error regions for Δ_α calculation to adjust the amplitude value.

of the sinc approximation. The ratio of the two values give us the best estimate of an amplitude.

We smooth our amplitude update with our current amplitude value to produce our new amplitude estimate, $\hat{\alpha}_m$, which also prevents drastic change to this parameter in each iteration, as seen in Equation 3.27. We use the smoothing value $\lambda = 0.25$ for our implementation.

$$\hat{\alpha}_m = (1 - \lambda)\alpha_m + \lambda\Delta_\alpha \tag{3.27}$$

Figure 3.4 illustrates the intuition behind this parameter update. The residual signal and our sinc approximation using the current amplitude estimate are show in Figure 3.4 by the blue and green lines, respectively. The area shaded in red in Figure 3.4 corresponds to the error around the main lobe and the two immediate side lobes, and the remaining error is shaded in grey. We see that the amplitude estimate needs to be increased to reduce the amount of red. However, we notice that there are two yet-to-be-detected signals in the residual signal on both sides of x_1 in Figure 3.4. The far side lobes, in the grey shaded area, contain a large amount of error, which is not due to our signal of interest in the center of the figure. Thus, we place a larger importance on the red shaded area than the grey area. Suppose that we perfectly estimate the amplitude of the center signal, which reduces the red area error to nearly zero, then there will still remain a significant amount of grey area error. If we did not weight the error, then the grey area error would cause us to incorrectly increase our amplitude estimate. In other words, we weight the error to attenuate

the impact by the other signals in the residual signal on our amplitude estimation.

3.3.2 NSPS Update

The NSPS update is also a function of the residual signal and the sinc approximation for the signal of interest. For this update, we consider how well the main lobe of our sinc approximation matches the residual signal. Recall that the size of the main lobe is inherently defined by the bandwidth, and that the bandwidth is inversely proportional to the NSPS. We calculate an update to the NSPS that minimizes the squared error over the indices of the main lobe. We do not want any inaccuracy in our estimate of the amplitude to bias our NSPS estimation. Therefore, we clip the residual signal to the peak value of our amplitude estimate. We use a quadratic function to approximate the main lobe to simplify the derivation of our estimate.

Before deriving for our NSPS estimate update, we define the following items. The set $\mathcal{L}$ contains the indices ℓ that correspond to the main lobe of the sinc approximation for the signal x_m, which is defined in Equation 3.28. We approximate the main lobe by a quadratic function, where the support of this approximation is $\mathcal{L}$, and is expressed by Equation 3.29 with $C_m = \alpha_m \left(\frac{\Delta_K}{L}\right)^2$. For a given V_ℓ, we can calculate ζ_ℓ such that $V_\ell = Q_{\zeta_\ell}(\alpha_m, \ell_m)$, which projects V_ℓ onto the quadratic function. With algebraic manipulation, we can express ζ_ℓ in Equation 3.30, where $\mu = +1$ if $\ell > \ell_m$, otherwise $\mu = -1$.

$$\mathcal{L} = \left\{\ell : -\pi < \frac{\pi\Delta_K}{L}[\ell - \ell_m] < \pi\right\} \tag{3.28}$$

$$Q_\ell(\alpha_m, \ell_m) = \alpha_m - C_m(\ell - \ell_m)^2 \tag{3.29}$$

$$\zeta_\ell = \ell_m + \mu\Delta_K^{-1} L\sqrt{\frac{\alpha_m - V_\ell}{\alpha_m}} \tag{3.30}$$

We have two error functions that we attempt to reduce. The first error function, E_V, measures the squared difference between the amplitude of residual signal V and our quadratic function in the range of indices that correspond to the domain of the main lobe, shown in Equation 3.31. The second error function, E_H, measures the squared difference between horizontal position of the residual signal V and the projection onto our quadratic function in the range of indices that correspond to the domain of the main lobe, shown in Equation 3.32.

$$E_V = \sum_{\ell\in\mathcal{L}} [V_\ell - Q_\ell(\alpha_m, \ell_m)]^2 \tag{3.31}$$

$$\begin{aligned} E_H &= \sum_{\ell\in\mathcal{L}} [\ell - \zeta_\ell]^2 \\ &= \sum_{\ell\in\mathcal{L}} \left[\ell - \left(\ell_m + \mu\Delta_K^{-1} L\sqrt{\frac{\alpha_m - V_\ell}{\alpha_m}}\right)\right]^2 \end{aligned} \tag{3.32}$$

Ideally, both error functions should contain the same amount of error, but we expect to see variations in error value due to inaccuracies in our model and of our parameter estimates. For the remainder this derivation, we assume that V_ℓ is actually a clipped version of V_ℓ, where we saturate the value of V_ℓ to α_m, if V_ℓ is greater than α_m.

We start by taking the derivative of the first error function with respect to Δ_K in Equation 3.33. The error function is easier to work with as a function of C_m instead of Δ_K. By the chain rule of derivatives, we see in Equation 3.33 that we can minimize the error function with respect to C_m as well, since $\frac{\partial C_m}{\partial \Delta_K}$ is a constant positive value. In Equations 3.34 and 3.35, we perform the partial derivative, and the appropriate algebraic manipulation.

$$\frac{\partial}{\partial \Delta_K} E_V = \underbrace{\frac{\partial C_m}{\partial \Delta_K}}_{>0} \frac{\partial}{\partial C_m} E_V \tag{3.33}$$

$$\frac{\partial}{\partial C_m} E_V = \sum_{\ell \in \mathcal{L}} -2 \cdot [V_\ell - Q_\ell(\alpha_m, \ell_m)] \cdot \frac{\partial}{\partial C_m} Q_\ell(\alpha_m, \ell_m) \tag{3.34}$$

$$= \sum_{\ell \in \mathcal{L}} 2 \cdot \left[V_\ell - \left(\alpha_m - C_m(\ell - \ell_m)^2\right)\right] \cdot (\ell - \ell_m)^2 \tag{3.35}$$

By setting the derivative in Equation 3.35 to zero, and substituting the relationship between Δ_K and C_m, we can manipulate the terms to produce our NSPS estimate, $\Delta_{K,V}$, which is based on the first error function E_V, and is shown in Equation 3.36.

$$\Delta_{K,V} = \sqrt{\frac{\sum_{\ell \in \mathcal{L}} (\alpha_m - V_\ell) \cdot (\ell - \ell_m)^2}{\sum_{\ell \in \mathcal{L}} \alpha_m L^{-2} \cdot (\ell - \ell_m)^4}} \tag{3.36}$$

Next, we take the derivative of the second error function with respect to Δ_K in Equation 3.37. In Equations 3.38 and 3.39, we perform the partial derivative, and the appropriate algebraic manipulation, where we define $\Gamma_\ell = L\sqrt{\frac{\alpha_m - V_\ell}{\alpha_m}}$.

$$\frac{\partial}{\partial \Delta_K} E_H = \frac{\partial}{\partial \Delta_K} \sum_{\ell \in \mathcal{L}} \left[\ell - \left(\ell_m + \mu \Delta_K^{-1} L \sqrt{\frac{\alpha_m - V_\ell}{\alpha_m}}\right)\right]^2 \tag{3.37}$$

$$= \sum_{\ell \in \mathcal{L}} -2 \cdot \left[\ell - \left(\ell_m + \mu \Delta_K^{-1} \Gamma_\ell\right)\right] \frac{\partial}{\partial \Delta_K} \left(\ell_m + \mu \Delta_K^{-1} \Gamma_\ell\right) \tag{3.38}$$

$$= \sum_{\ell \in \mathcal{L}} -2 \cdot \left[\ell - \left(\ell_m + \mu \Delta_K^{-1} \Gamma_\ell\right)\right] \left(-\mu \Delta_K^{-2} \Gamma_\ell\right) \tag{3.39}$$

By setting the derivative in Equation 3.39 to zero, we can manipulate the terms to produce our NSPS

estimate, $\Delta_{K,H}$, which is based on the second error function E_H, and is shown in Equation 3.40.

$$\Delta_{K,H} = \frac{\sum_{\ell \in \mathcal{L}} \Gamma_\ell^2}{\sum_{\ell \in \mathcal{L}} (\ell - \ell_m) \cdot \mu \Gamma_\ell^2} \tag{3.40}$$

We produce our NSPS update by a convex combination of $\Delta_{K,V}$ and $\Delta_{K,H}$, as shown in Equation 3.41, where β is the convex factor between 0 and 1. If β is close to 0, then our NSPS update is influenced more by $\Delta_{K,V}$, whereas if β is close to 1, then our NSPS update is influenced more by $\Delta_{K,H}$. We need to experimentally determine this value. We again smooth our NSPS update with our current NSPS value to produce our new NSPS estimate, $\hat{K}_m$, as shown in Equation 3.42. We use the smoothing value $\lambda = 0.25$ for our implementation.

$$\Delta_K = (1 - \beta)\Delta_{K,V} + \beta \Delta_{K,H} \tag{3.41}$$

$$\hat{K}_m = (1 - \lambda) K_m + \lambda \Delta_K \tag{3.42}$$

3.3.3 Offset Update

We adjust the signal offset by using an approach inspired by the process of balancing the torque being applied to a lever. We use the error signal computed from our current estimate of the signal parameters to adjust the signal offset. We wish to derive the increment offset Δ_ℓ associated with the parameter ℓ_m given the current estimates of α_m and K_m from a residual signal. This update procedure is slightly different than the previous procedures in that we are computing an incremental change to the value. In contrast, we previously computed the new value of a parameter explicitly.

Given the current offset value, we divide the main lobe into left and right sections, i.e. indices greater than the offset ℓ_m, and indices less than the offset, respectively. Recall that the set $\mathcal{L}$ was previously defined to be the support set of the main lobe in our sinc approximation. We let the sets $\mathcal{L}_L$ and $\mathcal{L}_R$ each be a subset of $\mathcal{L}$, as shown in Equations 3.43 and 3.44, respectively.

$$\mathcal{L}_L = \left\{ \ell : \ell \in \mathcal{L}, \ell < \ell_m \right\} \tag{3.43}$$

$$\mathcal{L}_R = \left\{ \ell : \ell \in \mathcal{L}, \ell \geq \ell_m \right\} \tag{3.44}$$

We see that $\mathcal{L}_L$ and $\mathcal{L}_R$ are the sets of indices that contain the indices that are smaller or larger, respectively, than our center index ℓ_m.

This approach is inspired by the process of balancing the torque applied to a lever. We let the offset ℓ_m represent the location of the fulcrum of our lever. We calculate the center of mass on both sides of the fulcrum. From these two centers of mass, we can calculate the torque produced.

The left mass M_L is the summation of the difference between the clipped residual signal and the clipped sinc function approximation, as seen in Equation 3.45. We normalize M_L by the cardinality

of $\mathcal{L}$. The clip value μ_L is chosen to be the minimum of the maximum residual signal value and the maximum sinc function approximation, as seen in Equation 3.46.

$$M_L = \frac{1}{|\mathcal{L}|} \sum_{\ell \in \mathcal{L}_L} \lceil V_\ell \rceil^{\mu_L} - \lceil \alpha_m S_\ell(K_m, \ell_m) \rceil^{\mu_L} \tag{3.45}$$

$$\mu_L = \min \left\{ \max_{\ell \in \mathcal{L}_L} V_\ell, \max_{\ell \in \mathcal{L}_L} \alpha_m S_\ell(K_m, \ell_m) \right\} \tag{3.46}$$

Note that $\lceil z \rceil^\mu$ equals z when less than μ, and otherwise equals μ. Again, we use this clipping operation to reduce the effect of inaccurate amplitude estimation on our parameter estimation of the offset.

Similarly, we define the right mass M_R with the associated right clip value μ_R in Equations 3.47 and 3.48, respectively.

$$M_R = \frac{1}{|\mathcal{L}|} \sum_{\ell \in \mathcal{L}_R} \lceil V_\ell \rceil^{\mu_R} - \lceil \alpha_m S_\ell(K_m, \ell_m) \rceil^{\mu_R} \tag{3.47}$$

$$\mu_R = \min \left\{ \max_{\ell \in \mathcal{L}_R} V_\ell, \max_{\ell \in \mathcal{L}_R} \alpha_m S_\ell(K_m, \ell_m) \right\} \tag{3.48}$$

Continuing with the balancing lever approach, both the left and right masses apply a torque on the lever about the fulcrum. If a clockwise torque is produced, then we need to shift the center index to the right. Likewise, if a counterclockwise torque is produced, then we need to shift the center index to the left. We adjust the offset by an amount that would produce an equilibrium between the torques created by the two centers of mass.

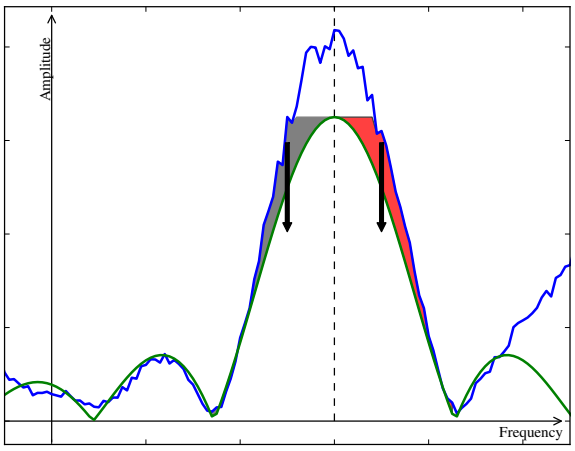
We allow the magnitude of the incremental offset to be no larger than one. We define `clip` function in Equation 3.49. The torque is calculated by clipping the subtraction of the right mass from the left mass, as shown in Equation 3.50. We generate our new estimate of the offset by adding a portion of the calculated increment to the current offset estimate, as seen in Equation 3.51. We use $\lambda = 0.25$ for our implementation.

$$\texttt{clip}(z) = \begin{cases} +1, & z \geq 1 \\ z, & -1 \leq z \leq 1 \\ -1, & z \leq -1 \end{cases} \tag{3.49}$$

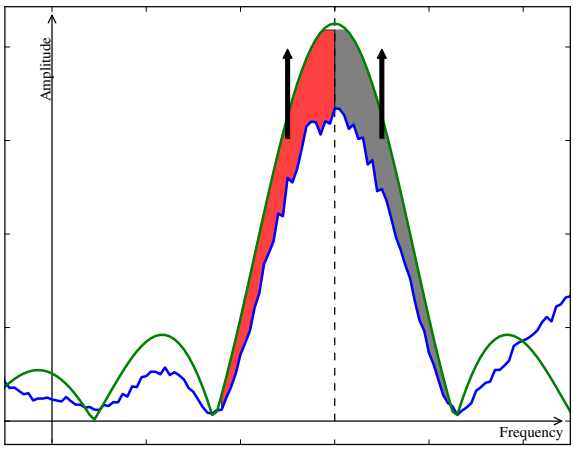
$$\Delta_\ell = \texttt{clip}(M_L - M_R) \tag{3.50}$$

$$\hat{\ell}_m = \ell_m + \lambda \Delta_\ell \tag{3.51}$$

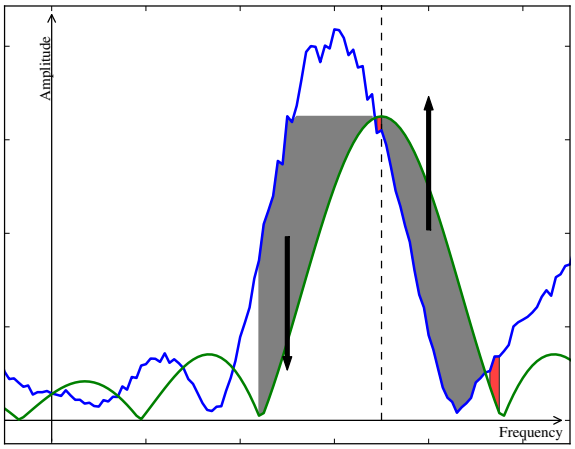
Figure 3.5 illustrates four possible cases for estimating our incremental offset. The residual signal, and our sinc approximation are depicted by the blue, and green lines, respectively, in Figure 3.5. The dotted black line is the location of our current estimate of the center index ℓ_m. In Figure 3.5,



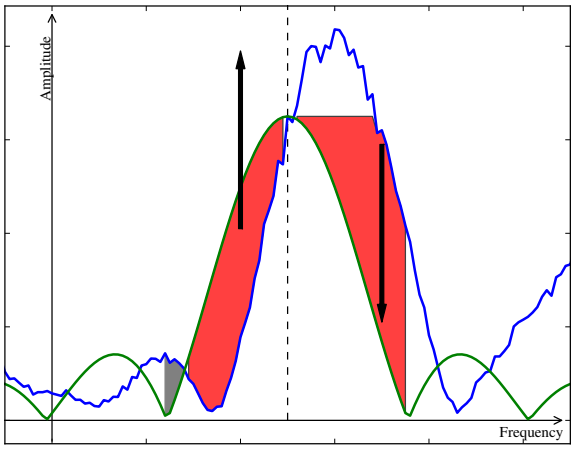
(a) Case 1: Sinc approximation below data



(b) Case 2: Sinc approximation above data



(c) Case 3: Sinc approximation with large right shift



(d) Case 4: Sinc approximation with large left shift

Figure 3.5: Example cases for determining the incremental offset Δ_ℓ to update the center frequency.

the red shaded areas create a clockwise torque, and the grey shaded areas create a counterclockwise torque.

The first case occurs when the sinc approximation is below the residual signal, as depicted in Figure 3.5a. We see that offset ℓ_m is nearly correct and the torques produced about the fulcrum both pull downward. As a result, we expect the calculated incremental offset to be fairly small.

In the second case, the sinc approximation is now above the residual signal, as in Figure 3.5b. We see again that offset ℓ_m is nearly correct, but the torques produced about the fulcrum both pull upward. We also expect the calculated incremental offset in this case to be fairly small.

Figure 3.5c shows the third case where our offset ℓ_m has a large shift to the right, which causes mostly counterclockwise torques. In this case, we expect the incremental offset to shift our offset

to the left. Similarly, Figure 3.5d shows the fourth case where our offset ℓ_m has a large shift to the left, which causes mostly clockwise torques. In this case, we expect the incremental offset to shift our offset to the right.

3.4 Experiments and Analysis

We present two experiments and one analytical evaluation to quantify our NSD algorithm's ability at detecting a narrowband signal and estimating the parameters associated with that detection. In the first experiment, we allow the NSD algorithm to iterate 100 times before stopping while varying the DFT size and the β-parameter which is important to the estimation step. In the second experiment, we vary the maximum allowable detections and the number of algorithmic iterations to see the effect on the probability of detection and the probability of false alarm. The computation time of the NSD algorithm is directly proportional to these two parameters, so if we can reduce the value of these parameters without degrading the algorithm's performance, then we can reduce the computation time. Finally, we analytically evaluate and compare our NSD algorithm to the energy detection algorithm.

3.4.1 Initial Experiments

The test cases assess our parameter estimation of our algorithm, by varying the values of the SNR value from the set $\{1 \text{ dB}, 10 \text{ dB}, 100 \text{ dB}\}$, the DFT size from the set $\{256, 512\}$, and the β value used in the NSPS update from the set $\{0.00, 0.25, 0.50, 0.75, 1.00\}$. In all test cases, we use an input received signal that is created by linearly combining three signals of interest, which we have used as an example throughout this section. Recall that the three signals have he following parameters. The parameters for signal x_1 are $\alpha_1 = 2A$, $B_1 = \frac{2}{5}\pi$, and $f_1 = \frac{15}{16}\pi$. The parameters for signal x_2 are $\alpha_2 = A$, $B_2 = \frac{2}{5}\pi$, and $f_2 = \frac{\pi}{4}$. The parameters for signal x_3 are $\alpha_3 = A$, $B_3 = \frac{2}{5}\pi$, and $f_1 = \frac{11}{8}\pi$. Note that the number of samples per symbol, K_m, and offset, ℓ_m, for each signal is a function of the DFT size, where $K_m = \frac{4\pi}{B_m}$ and $\ell_m = f_m\left(\frac{L}{2\pi}\right)$. For all test cases, the value of K_m is 10 for all signals. For the test cases where the DFT size equals 256, the value of ℓ_1, ℓ_2, and ℓ_3 is 120, 32, 176, respectively, and for the test cases where the DFT size equals 512, the value of ℓ_1, ℓ_2, and ℓ_3 is 240, 64, 352, respectively.

Table 3.1 is a super-table of tables that lists the results of our tests with the DFT size equal 256, where the row of the super-table is the β value used, the column of the super-table is the SNR value used. An element in this super-table contains the results of the test case identified by the row and column, as a table of amplitude, NSPS, and offset for each signal detected by our algorithm.

We notice that we detect no more than five signals in any given test case, which is most likely due to the combination of the inaccuracy of our model, incomplete merging of signals, and the lack of

Test		SNR = 1 dB			SNR = 10 dB			SNR = 100 dB		
Case	β	Amp.	NSPS	Offset	Amp.	NSPS	Offset	Amp.	NSPS	Offset
		22.64	5.65	34.60	34.23	11.55	120.30	60.98	13.73	119.31
		20.48	11.79	121.47	31.53	11.46	119.94	33.06	13.28	31.55
256-1	0.00	19.82	13.10	118.66	28.84	11.37	119.59	15.37	3.80	134.79
		17.56	9.35	176.30	23.42	14.04	31.43	14.34	3.75	134.07
		13.38	5.31	113.17	21.55	12.81	175.69			
		38.24	8.72	119.44	40.54	9.88	121.13	72.35	9.37	120.02
		19.50	9.33	28.43	23.92	47.72	111.77	30.76	10.84	31.64
256-2	0.25	18.46	24.83	171.81	19.74	10.43	31.51	29.46	11.77	176.48
		16.56	27.35	183.60	19.67	11.96	176.41			
		14.10	45.73	86.57	19.09	11.97	119.18			
		37.35	6.96	118.19	49.13	8.11	119.67	63.51	7.84	119.81
		22.25	128.00	27.03	17.56	9.81	31.37	24.84	8.48	176.05
256-3	0.50	21.57	25.25	36.19	17.23	10.92	177.58	19.68	7.52	32.45
		15.86	117.11	23.18						
		15.12	128.00	179.12						
		36.94	6.59	118.20	45.16	7.21	119.63	35.39	9.26	120.00
		28.54	128.00	34.20	14.31	7.72	176.28	33.42	9.15	120.06
256-4	0.75	25.29	128.00	27.21	11.87	6.46	30.42	26.30	8.18	175.59
		19.28	128.00	180.25				19.50	7.20	32.00
		17.79	128.00	23.12						
		36.44	6.41	118.14	24.54	8.86	120.00	50.03	6.86	119.43
		34.07	128.00	35.13	23.63	8.89	119.78	37.23	8.40	119.88
256-5	1.00	25.67	128.00	27.24	17.34	8.80	31.91	24.43	9.94	120.33
		18.75	128.00	180.24	16.86	8.76	176.11	20.96	6.32	174.07
		18.04	128.00	23.14				20.05	7.42	32.14

Table 3.1: Parameter estimate test results using 256-DFT and varying β and SNR.

Test		SNR = 1 dB			SNR = 10 dB			SNR = 100 dB		
Case	β	Amp.	NSPS	Offset	Amp.	NSPS	Offset	Amp.	NSPS	Offset
		41.73	3.65	256.89	45.62	13.17	237.91	95.80	13.05	239.15
		38.29	5.90	77.58	44.46	13.04	240.03	50.88	13.42	63.29
512-1	0.00				34.86	13.46	63.53	28.49	3.84	278.63
					19.22	3.79	317.24	18.01	3.66	267.96
					17.64	3.88	253.82	15.73	3.68	267.00
		48.41	165.06	1.41	80.77	9.57	240.33	102.61	9.67	240.65
		47.91	4.03	248.46	32.32	13.33	353.13	44.52	10.55	63.53
512-2	0.25	35.09	28.36	54.61	21.59	256.00	32.17	41.77	11.33	352.71
		25.88	256.00	506.37	15.60	90.09	225.16	18.84	72.23	226.50
		19.59	228.59	6.11	14.14	220.82	28.19	8.75	256.00	210.25
		65.89	7.40	237.44	70.70	8.06	240.16	91.91	8.14	240.17
		40.13	256.00	54.36	27.61	9.10	63.38	40.20	9.22	63.73
512-3	0.50	37.99	256.00	58.37	27.34	10.72	354.20	37.41	9.40	352.00
		35.47	256.00	51.17	25.81	256.00	224.31	18.38	178.48	220.33
		32.01	256.00	44.14	15.24	210.24	220.15			
		65.51	7.28	237.41	66.03	7.09	239.43	74.87	7.31	239.42
		51.93	256.00	1.44	23.30	7.51	352.62	36.08	8.02	64.87
512-4	0.75	42.38	256.00	54.38	21.39	6.35	61.74	32.58	10.24	240.47
		35.74	256.00	51.18				31.28	6.72	350.01
		23.35	256.00	7.07						
		65.20	7.19	237.34	43.21	7.81	239.44	50.53	7.57	239.76
		44.73	256.00	1.71	35.38	8.46	240.26	49.00	8.03	240.28
512-5	1.00	41.83	256.00	54.39	26.89	7.70	353.14	34.61	6.73	350.52
		35.48	256.00	51.22	25.66	7.53	63.73	28.06	6.29	64.26
		25.07	256.00	8.02						

Table 3.2: Parameter estimate test results using 512-DFT and varying β and SNR.

a true stopping mechanism. The amplitude estimates are relatively proportional, but seem highly sensitive to the SNR value. We also notice that we find all three offsets in all but one test cases, and where the accuracy is within 2 frequency bins for the 10 dB, and 100 dB SNR cases, and within 6 frequency bins for the 0 dB SNR cases.

As expected, we see that the NSPS estimates are sensitive to the value of β. Our NSPS estimation is not very successful when the SNR value was 1 dB, which is understandable, since 1 dB of signal-to-noise ratio is a hard test case. The estimation works a little better the closer β is to 0.00. With 10 dB of SNR, we see that we overestimate NSPS when β is 0.00 or 0.25, and underestimate NSPS when β is 0.75 or 1.00, and when β is 0.50, we have a balance of underestimating and overestimating. With 100 dB of SNR, we see that we overestimate NSPS when β is 0.00, and underestimate NSPS when β is 0.50, 0.75, or 1.00, and when β is 0.25, we have a balance of underestimating and overestimating.

Table 3.2 is a super-table of tables that lists the results of our tests with the DFT size equal 512, where the row of the super-table is the β value used, the column of the super-table is the SNR

value used. Again, an element in this super-table contains the results of the test case identified by the row and column, as a table of amplitude, NSPS, and offset for each signal detected by our algorithm.

We notice that we detect no more than five signals in any given test case. Again, we see that the amplitude estimates are relatively proportional, but seem highly sensitive to the SNR value. For SNR values of 10 dB and 100 dB, we find all three offsets with $\beta \neq 0.00$, and where the accuracy is within 4 frequency bins. For SNR values of 10 dB and 100 dB, and $\beta = 0.00$, we see that detect the signal with offsets 64 and 240, but not the signal with offset equal to 352.

The NSPS estimation for the test cases with the DFT size equal to 512 correlates with the estimation for the test cases with the DFT size equal to 256. Our NSPS estimation again is not very successful when the SNR value was 1 dB. With 10 dB of SNR, we see that we overestimate NSPS when β is 0.00 or 0.25, and underestimate NSPS when β is 0.75 or 1.00, and when β is 0.50, we have a balance of underestimating and overestimating. With 100 dB of SNR, we see that we overestimate NSPS when β is 0.00, and underestimate NSPS when β is 0.50, 0.75, or 1.00, and when β is 0.25, we have a balance of underestimating and overestimating.

Based on the results, in Tables 3.1 and 3.2, we believe that the best β value to use is 0.50, which signifies that our NSPS update relies equally on the estimate produced by the two types of error functions. We noticed a balance between overestimating and underestimating the number of samples per symbol when $\beta = 0.50$, and intuitively make sense, because we do not have a justification for a preference in error functions to bias the NSPS update.

3.4.2 Parameter Refinement Experiments

This next experiment attempts to select the the best choice for two parameters in our NSD algorithm. The first parameter is the maximum number of signal detections, M_D, that we allow our algorithm to create. The second parameter is the number of iterations, N_I, to run our algorithm before we stop. We must appropriately select the values for both of these parameters in order to have maximal performance while preventing unnecessary computations by the algorithm.

We test our algorithm at four SNR values, 0 dB, 5 dB, 10 dB, and 15 dB, to evaluate the performance as we increase the signal strength with respect to the noise. At each SNR value, we generate 1000 wideband signals each containing a single narrowband signal that is randomly located within the wideband signal, and we add white Gaussian noise where the variance to set to yield the specified SNR value. Note that we use our knowledge of the true location of the narrowband signal in order to calculate the performance. We run our algorithm for 500 iterations with a specific value for the maximum number of signal detections. We let M_D take on the values 5, 10, 20, 50, and 500, so we run this experiment 5 times.

We record the signal detections after each iteration of the algorithm from which we calculate the probability of detection and the probability of false alarm. Recall that a detection produced by our algorithm is the joint estimate of both the signal center frequency and the signal bandwidth. Also, recall that the center frequency is with respect to the DFT bin and DFT size. These two facts results in occasions when our detection algorithm picks a center frequency that is slightly off, which is to say that the estimated center frequency is technically wrong, but, in combination with the estimated bandwidth, the estimated center frequency is for all practical purposes correct. With this in mind, for each detection, if the frequency offset is within a single DFT bin of the true value then we mark the detection as a true detection.

On the j^{th} iteration of our algorithm when applied to the k^{th} test signal, we let $N_D(j;k)$ be total number detections for that iteration. We let $N_T(j;k)$ be total number of true detections. We define the probability of detection for the j^{th} iteration when applied to the k^{th} test signal, $P_D(j;k)$, and the probability of false alarm for the j^{th} iteration when applied to the k^{th} test signal, $F_A(j;k)$ in Equation 3.52 and Equation 3.53, respectively. Note that it is possible for our algorithm to produce more than a single detection that is considered to be a true detection, i.e. $N_T(j;k) > 1 \ \forall j$. This situation can occur because our algorithm can return non-integer frequency bin offsets.

$$P_D(j;k) = \begin{cases} 1 & \text{if } N_T(j;k) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{3.52}$$

$$F_A(j;k) = \frac{N_D(j;k) - N_T(j;k)}{N_D(j;k)} \tag{3.53}$$

We average our probability measures over the 1000 test signals to give estimates of the probability of detection for the j^{th} iteration, and the probability of false alarm for the j^{th} iteration, as shown in Equation 3.54 and Equation 3.55, respectively. As the SNR value increases, we expect the probability of detection to also increase and the probability of false alarm to decrease.

$$P_D(j) = \frac{1}{1000} \sum_{k=1}^{1000} P_D(j;k) \tag{3.54}$$

$$F_A(j) = \frac{1}{1000} \sum_{k=1}^{1000} F_A(j;k) \tag{3.55}$$

For these experiments, we run our algorithm with a DFT size of 256. This size along with the fact that we run our algorithm for a maximum of 500 iterations makes our experiment with $M_D = 500$ essentially the "infinite" case, in the sense that we expect that the performance in this experimental configuration yields the asymptotic results that would be produced if we allowed an infinite number of signal detections.

We first show in Figure 3.6 our narrowband detection algorithm performance when we allow our

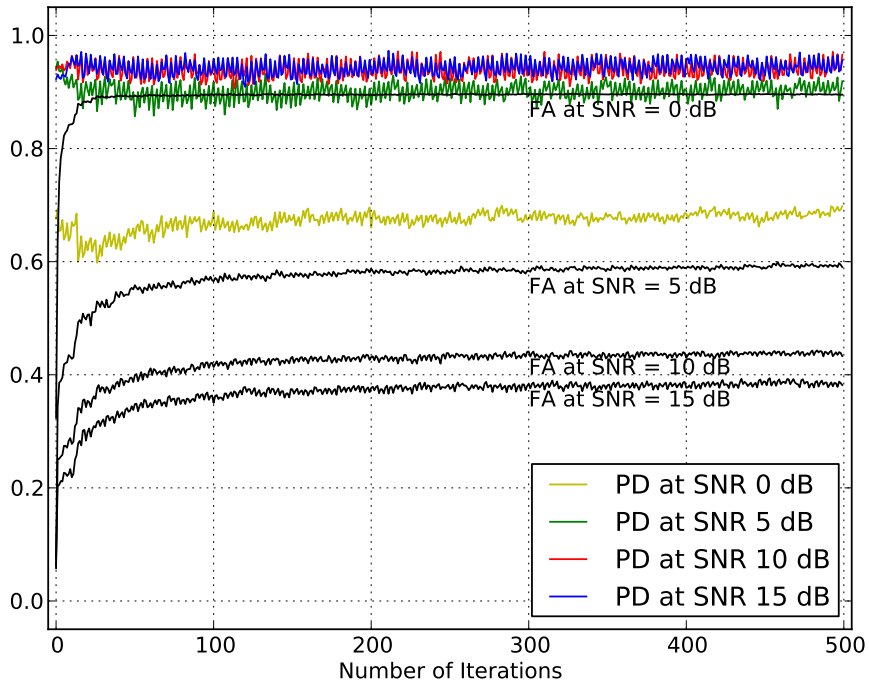


Figure 3.6: The narrowband detection algorithm's probability of detection and probability of false alarm performance versus the maximum number of algorithmic iterations at SNR values of 0 dB, 5 dB, 10 dB, and 15 dB where the maximum number of signals that can be detected is set to 500.

algorithm to detect a maximum of 500 signals where the vertical axis represents a probability and the horizontal axis is the number of iterations of the algorithm. We plot the probability of detection when operating our algorithm on a signal at 0 dB, 5 dB, 10 dB, or 15 dB SNR with yellow, green, red, or blue line, respectively. The probability of false alarm is plotted using a black line where we annotate the figure to mark which curve is associated with operating on a signal at 0 dB, 5 dB, 10 dB, or 15 dB SNR.

At a fixed SNR value, we see that P_D, averaged over the number of iterations, is fairly constant: $67.3\% \pm 1.8\%$ at 0 dB SNR, $90.1\% \pm 1.5\%$ at 5 dB SNR, $93.9\% \pm 1.6\%$ at 10 dB SNR, and $94.3\% \pm 1.4\%$ at 15 dB SNR. Similarly, at a fixed SNR value, we see that F_A, averaged over the number of iterations, is also fairly constant after a few initial iterations: $89.2\% \pm 3.0\%$ at 0 dB SNR, $57.2\% \pm 3.9\%$ at 5 dB SNR, $42.0\% \pm 3.4\%$ at 10 dB SNR, and $36.6\% \pm 3.3\%$ at 15 dB SNR The algorithm's probability of detection and false alarm rate matches our intuition about its behavior as we vary the SNR value.

Note that if we ignore the false alarm rate prior to the 25^{th} iteration then the average performance is: $89.5\% \pm 0.2\%$ at 0 dB SNR, $57.8\% \pm 1.7\%$ at 5 dB SNR, $42.6\% \pm 1.7\%$ at 10 dB SNR, and $37.2\% \pm 1.7\%$ at 15 dB SNR. In this case, the average false alarm rate does not change much, but the standard deviation is reduced. The reason for this "ramp-up" period in false alarm rate is due

to the fact that in a single iteration of our algorithm we can only possibly add a single detection. As a result, during the first few iterations the algorithm is more likely to detect the true signals rather than the false detections due to noise, lowering the false alarm rate.

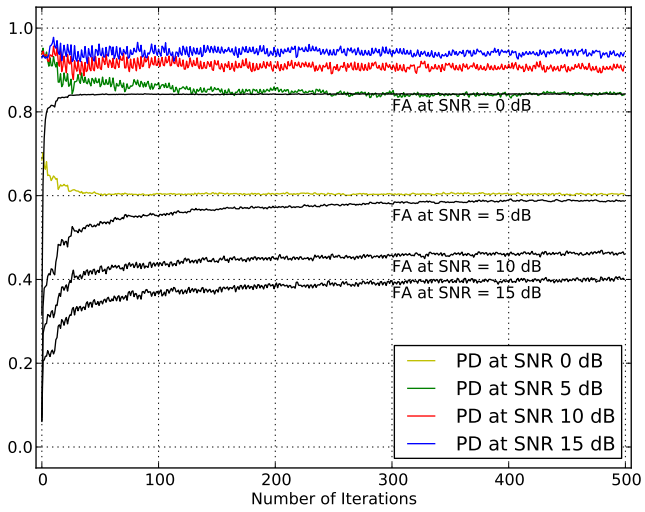
In Figure 3.7, we show the results of the experiment where we vary the maximum number of allowed detections where we tested 5, 10, 20, and 50 allowable detections. Again, each figure plots the probability of detection when applying our algorithm to a signal at 0 dB, 5 dB, 10 dB, or 15 dB SNR with the yellow, green, red, or blue lines, respectively, and the probability of false alarm when applying our algorithm using black lines where we annotate the figure to mark which curve is associated with operating on a signal at 0 dB, 5 dB, 10 dB, and 15 dB SNR.

We notice that the four plots in Figure 3.7 are very similar. Qualitatively, from the plots we see that the behavior of our algorithm is the same regardless of the number of maximum signal detections. Quantitatively, we see that the average performance agrees with the qualitative assessment. In Table 3.3, we list the P_D and F_A performances averaged over the total number of iterations for our four cases where we allow 5, 10, 20, and 50 maximum signal detections in our algorithm.

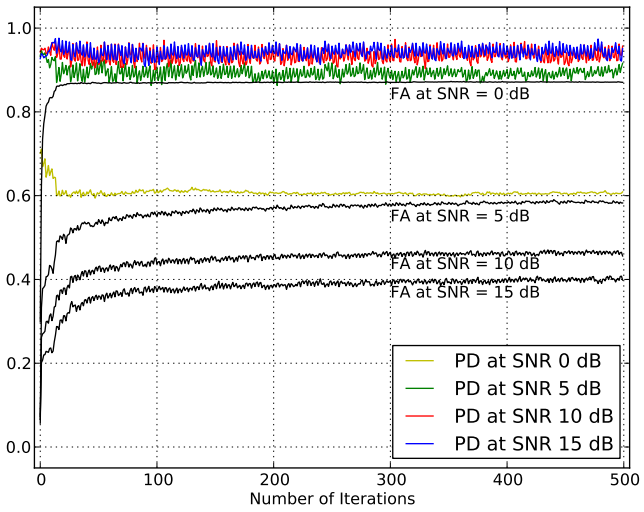
The probability of detection statistics are in Table 3.3a. We see that P_D equals approximately 90%, 93%, and 94% for 5 dB, 10 dB, and 15 dB, respectively, regardless of the number of maximum signal detections. However, at 0 dB SNR, the $P_D \approx 60\%$ for $M_D = 5$ and $M_D = 10$ and $P_D \approx 65\%$ for $M_D = 20$ and $M_D = 50$. For this SNR value, noise power is equal to the signal power, which means that our detection algorithm could get deceived more easily. In this case the improvement in P_D is likely due to the fact that when we allow more detections to be made by our algorithm there is a better chance that we find the true signal along with many false detection from the noise, which would result in an increased value of P_D.

The probability of false alarm statistics are in Table 3.3b. We see that F_A equals approximately 57%, 44%, and 38% for 5 dB, 10 dB, and 15 dB, respectively, regardless of the number of maximum signal detections. Again, we see that the value of M_D affects the results at 0 dB SNR where the $F_A \approx 84\%$ for $M_D = 5$, and $P_D \approx 87\%$ otherwise. Similar to the results when P_D equals 0 dB SNR, the lower value of F_A at $M_D = 5$ is likely due to the fact that we are preventing many detections from occurring, i.e. at most 5. If our algorithm detects the true signal, then at worst for $M_D = 5$ the false alarm rate would be $\frac{4}{5}$ or 80%.

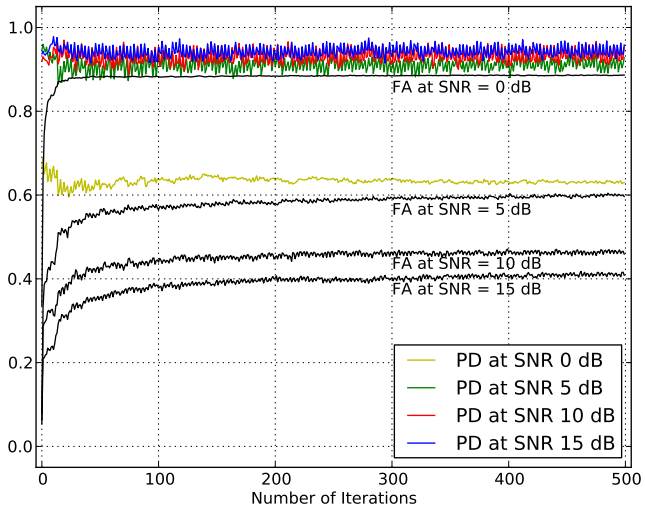
Based on the results from this experiment we have a better understanding of the relationship between the NSD performance, the maximum number of signal detections, and the number of iterations before stopping the algorithm. With respect to the maximum number of signal detections, we saw that the NSD algorithm's probability of detection and probability of false alarm were very consistent regardless in the choice of this parameter. The computational complexity of the algorithm is proportional to the number of detections that the algorithm is processing, so if we reduce the maximum number of detections, by 5 or 10, then we can also reduce the processing time. However,



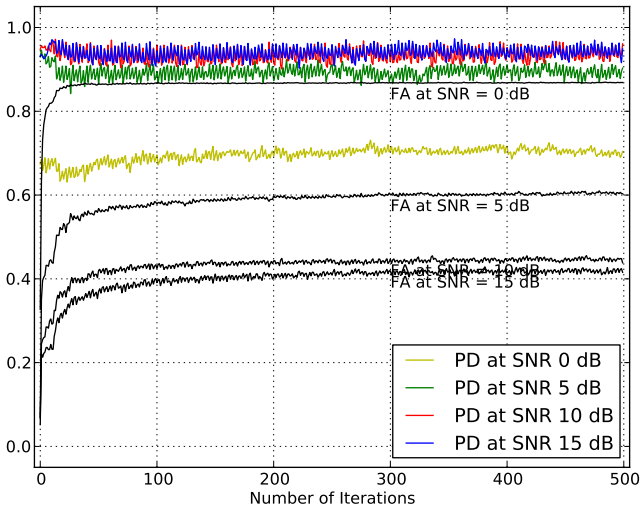
(a) 5 maximum signal detections



(b) 10 maximum signal detections



(c) 20 maximum signal detections



(d) 50 maximum signal detections

Figure 3.7: The narrowband detection algorithm's probability of detection and probability of false alarm performance versus the maximum number of algorithmic iterations at SNR values of 0 dB, 5 dB, 10 dB, and 15 dB where each subplot is the result of a different maximum number of signals that can be detected.

SNR	Maximum Signal Detections 5	10	20	50
0 dB	60.6%	60.8%	63.4%	69.7%
5 dB	85.2%	89.3%	91.1%	89.3%
10 dB	90.9%	93.5%	93.1%	93.4%
15 dB	94.2%	94.2%	94.3%	94.0%

(a) Probability of Detection

SNR	Maximum Signal Detections 5	10	20	50
0 dB	83.9%	86.7%	88.1%	86.4%
5 dB	56.6%	56.4%	57.7%	58.5%
10 dB	44.4%	44.6%	44.8%	43.1%
15 dB	37.7%	38.0%	38.8%	39.9%

(b) Probability of False Alarm

Table 3.3: The average performance for probability of detection and probability of false alarm averaged over the number of iterations at 0 dB, 5 dB, 10 dB and 15 dB SNR as a function of the maximum number of signal detections.

our test only considered received signals with a single transmitting signal. Thus, the trade-off between computation processing and maximum number of detections should be a function of signal density, i.e. the expect number of signals present. If we know the signal density, D, then we might allow $M_D = 5 \cdot D$. On the other hand, if we do not know D then we might select a number around 20 or 30 for M_D to allow a greater number of signal detections to be found, since we saw that the choice of M_D does not affect performance at the cost of extra performance.

We potentially have two modes of operation when selecting the number of iterations. We saw that P_D is fairly constant over the complete range of number of iterations that we tested. From the figures, it appears that there is a point at which F_A is constant that occurs around the 25^{th} iteration. If we want to operate with a constant F_A then we should iterate more that 25 times. However, if we want a reduce F_A but with a high variability then we could iterate less than 25 times. The computational complexity of the algorithm is also proportional to the number of iterations. This trade-off between the number of iterations and the variance of the F_A is best evaluated at a spectrum sensing architectural level. On the one hand, an increase in processing time to detect the presence of signals reduces the amount of time available to transmit our own data. However, an increase in uncertainty about the accuracy of our detections could lead the spectrum sensing architecture to believe there are more signals present than there truly are, which reduces the amount of spectrum available to transmit our own data.

3.4.3 Comparison to the Energy Detection Algorithm

The energy detection algorithm [40–52, 111–116] is a common approach to detecting narrowband center frequencies from a received wideband signal. The appeal of this algorithm is the simplicity of its implementation. The algorithm performs a DFT on the received signal and then looks for DFT bins that contain energy greater than a detection threshold. Note that the energy detection algorithm only estimates the center frequency of a narrowband signal. Afterwards, post-processing must be completed to estimate the signal bandwidth.

Another attractive characteristic of the energy detection algorithm is that we can use it to analytically evaluate the probability of detection and probability of false alarm. These probabilities can easily be calculated for a single DFT bin. In order to fairly compare the performance of our NSD algorithm with the energy detector, we must be able to calculate the probability of detection and false alarm over the complete spectrum of the received signal. The remainder of this section derives these probabilities and evaluates the performance of the algorithm.

We begin our analysis of the energy detection algorithm by deriving the probability of detection and the probability of false alarm from the perspective of a single DFT bin. We assume that at most one user is transmitting at any time. For a single DFT bin, we have a binary hypothesis decision to determine if a signal is present in that bin. The hypothesis $\mathcal{H}_0$ is the case when a signal is not present and thus we expect that the frequency response in the ℓ^{th} DFT bin is due to noise-only, as shown in Equation 3.56. The hypothesis $\mathcal{H}_1$ is the case when a signal is present and thus we expect that the frequency response in the ℓ^{th} DFT bin is due to the signal-plus-noise, as shown in Equation 3.57. Recall that $R[\ell]$, $X[\ell]$, and $W[\ell]$ are the frequency responses of the received signal, transmitted signal, and noise signal, respectively, at the ℓ^{th} DFT bin.

$$\mathcal{H}_0 \quad : \quad R[\ell] = W[\ell] \tag{3.56}$$

$$\mathcal{H}_1 \quad : \quad R[\ell] = X[\ell] + W[\ell] \tag{3.57}$$

As the name suggests, the energy detection algorithm measures the energy in a DFT bin, compares the probability of that measurement under both hypothesis $\mathcal{H}_0$ and hypothesis $\mathcal{H}_1$, and then selects the hypothesis with the larger probability. The energy, power and amplitude of the signal in a DFT bin are all related in a linear fashion, so we can derive the energy detection process with respect to any of those quantities. This relationship allows us to define the probability distribution functions with respect to the amplitude.

Recall that the value of each DFT bin is a complex Gaussian random variable where under the hypothesis $\mathcal{H}_0$ the amplitude of R_ℓ is a Rayleigh distribution and under the hypothesis $\mathcal{H}_1$ the amplitude of R_ℓ is a Rician distribution. We let ρ be the the random variable for the DFT bin amplitude. The probability distribution function for ρ under hypothesis $\mathcal{H}_0$ and hypothesis $\mathcal{H}_1$ is shown in Equations 3.58 and 3.59, respectively, where σ^2 is the single dimension noise variance and P is the transmit power from which we define the SNR as $10\log_{10}\frac{P}{2\sigma^2}$. Note that these two

hypothesis distributions are valid for any DFT bin.

$$p_\rho(\rho|\mathcal{H}_0) = \frac{\rho}{\sigma^2} exp\left\{-\frac{1}{2\sigma^2}\rho^2\right\} \tag{3.58}$$

$$p_\rho(\rho|\mathcal{H}_1) = \frac{\rho}{\sigma^2} exp\left\{-\frac{1}{2\sigma^2}(P+\rho^2)\right\} I_0\left(\frac{\rho\sqrt{P}}{\sigma^2}\right) \tag{3.59}$$

Figure 3.8 shows the probability distribution function for the amplitude when operating at SNR values of 15 dB, 10 dB, 5 dB, and 0 dB. In each plot, we depict the amplitude distribution when there is no signal present, $p_\rho(\rho|\mathcal{H}_0)$ with a blue line and we depict the amplitude distribution when there is a signal present, $p_\rho(\rho|\mathcal{H}_1)$ with a green line. The figure shows how the overlap between the two distributions increases as the SNR value decreases.

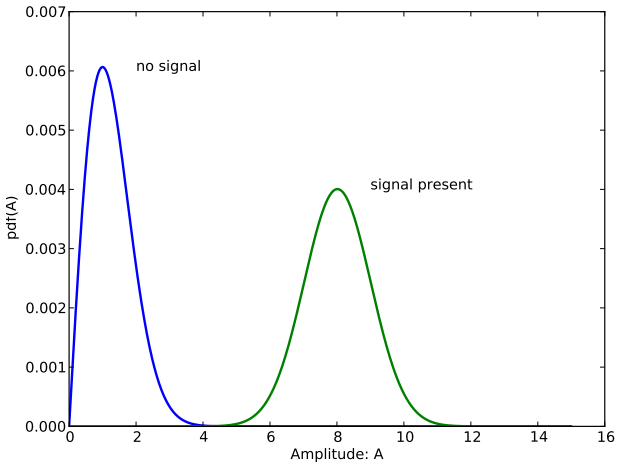
As shown in Equation 3.60, the energy detector makes the decision by comparing the likelihood ratio to a threshold. If the ratio is greater than the threshold then we decide that we are observing hypothesis $\mathcal{H}_1$, otherwise then we decide that we are observing hypothesis $\mathcal{H}_0$.

$$\begin{array}{ccc} & \mathcal{H}_1 & \\ \frac{p_\rho(\rho|\mathcal{H}_1)}{p_\rho(\rho|\mathcal{H}_0)} & \gtrless & \text{detection} \\ & \mathcal{H}_0 & \text{threshold} \end{array} \tag{3.60}$$

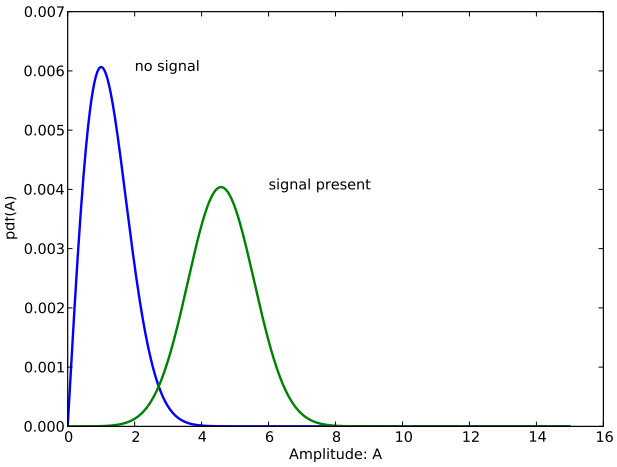
In Figure 3.9, we illustrate the effect of the detection threshold choice in Equation 3.60 on the probability of detection. We suppose that the detection threshold has a value of 2. In the figure, we shaded in green the portion of the distribution $p_\rho(\rho|\mathcal{H}_1)$ that corresponds to detecting the signal. The integration of this green shaded area equals the the probability of detection. We see that if the detection threshold is fixed and the SNR decreases, then our probability of detection also decreases. Conversely, we could fix our desired probability of detection, but to do so we must recalculate the threshold as a function of the SNR value.

Similarly, in Figure 3.10, we illustrate the effect of our threshold choice in Equation 3.60 on the probability of false alarm. Again, suppose the detection threshold has a value of 2. In the figure, we shaded in blue the portion of the distribution $p_\rho(\rho|\mathcal{H}_0)$ that corresponds to incorrectly claiming a signal is present. The integration of this blue shaded area equals the the probability of false alarm. Notice that for a particular threshold, the probability of detection and the probability of false alarm do not necessarily sum to 1 because these two quantities are the integration on two different distribution functions.

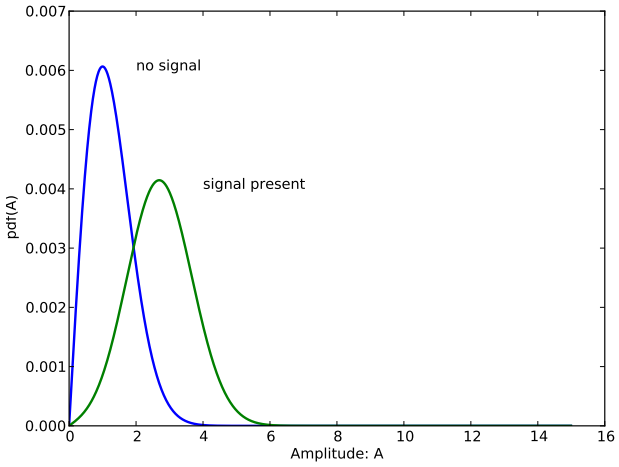
The analysis so far is derived with respect to a single DFT bin to yield the local probability of detection and probability of false alarm, which we denote by $P_D^{(\ell)}$ and $F_A^{(\ell)}$, respectively. However these local probabilities cannot be used in a fair comparison to our NSD algorithm since this algorithm operates on the whole spectrum not just a single DFT bin. To make a fair comparison,



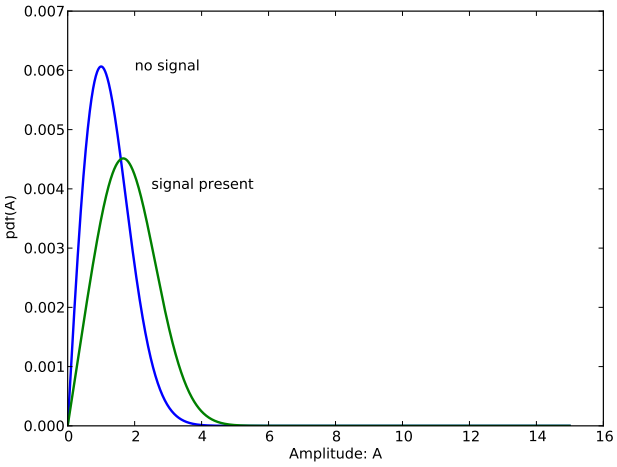
(a) SNR = 15 dB



(b) SNR = 10 dB

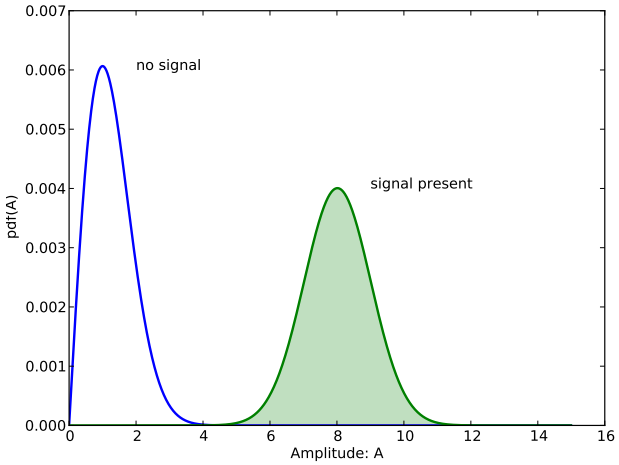


(c) SNR = 5 dB

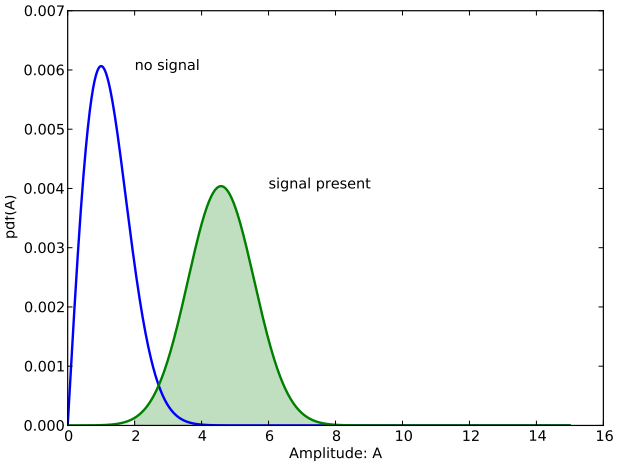


(d) SNR = 0 dB

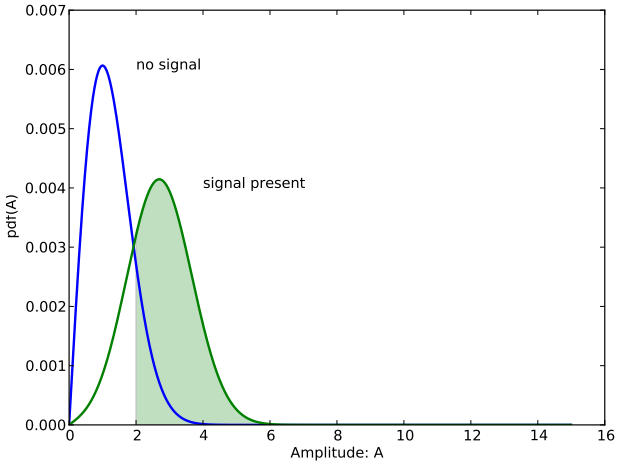
Figure 3.8: Probability distribution functions of the amplitude under hypothesis $\mathcal{H}_0$ and hypothesis $\mathcal{H}_1$ when operating at SNR values of 15 dB, 10 dB, 5 dB, and 0 dB where the blue line represents the distribution for $\mathcal{H}_0$ which corresponds to Equation 3.58 and the green line represents the distribution for $\mathcal{H}_1$ which corresponds to Equation 3.59.



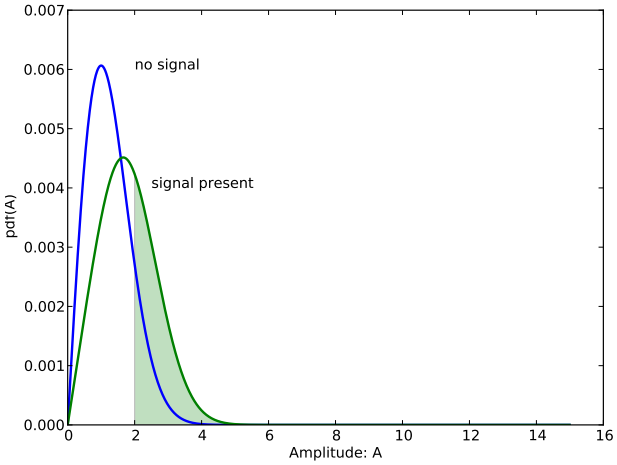
(a) SNR = 15 dB



(b) SNR = 10 dB

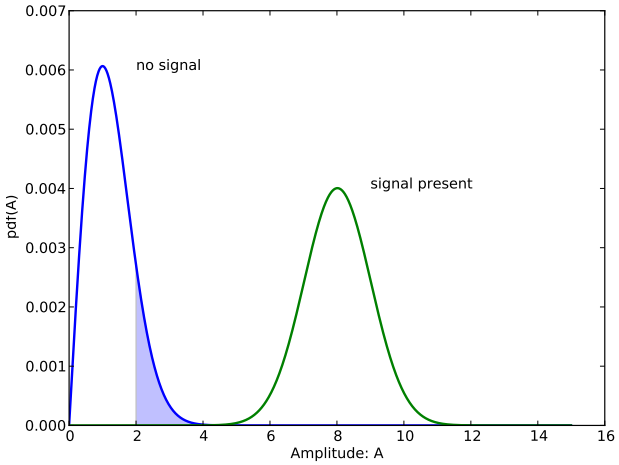


(c) SNR = 5 dB

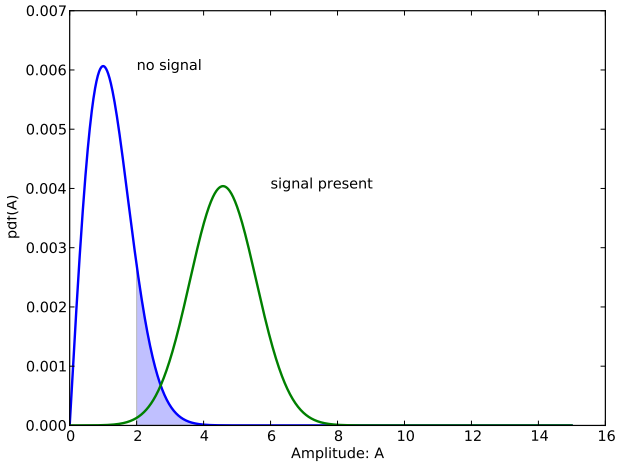


(d) SNR = 0 dB

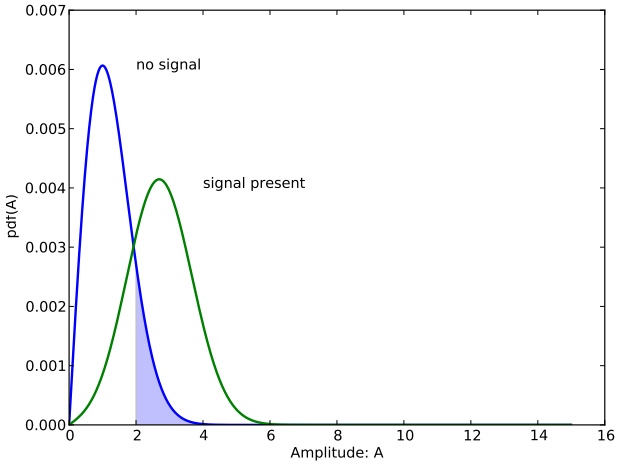
Figure 3.9: Visualization of the probability of detection when the ratio test threshold τ corresponds to an amplitude of 2 at SNR values of 15 dB, 10 dB, 5 dB, and 0 dB where P_D equals the distribution function integration of the green-shaded area.



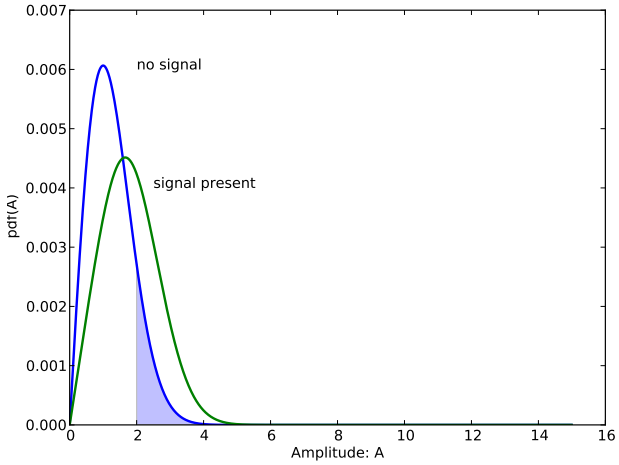
(a) SNR = 15 dB



(b) SNR = 10 dB



(c) SNR = 5 dB



(d) SNR = 0 dB

Figure 3.10: Visualization of the probability of false alarm when the ratio test threshold τ corresponds to an amplitude of 2 at SNR values of 15 dB, 10 dB, 5 dB, and 0 dB where F_A equals the distribution function integration of the blue-shaded area.

we continue our analysis by deriving the global probability of detection and probability of false alarm for the energy detection algorithm where these global probabilities are a function of the local probabilities.

Recall that in Section 3.4.2 we defined a truth detection event of the transmitting signal when the detection algorithm produces a center frequency for a detection that is within a single DFT bin from the true value. Then, the probability of detection is 1 if the truth detection event occurs, otherwise the probability is 0. We defined the probability of false alarm as the ratio of the number of detections that are not considered a truth detection event to the total number of detections.

Assuming that we use a DFT with L bins, we can derive the global probability of detection and probability of false alarm for the energy detector from the local $P_D^{(\ell)}$ and $F_A^{(\ell)}$. Again, if we assume that a single transmitting signal is present, the energy detector expects there to be one DFT bin where the amplitude is distributed by hypothesis $\mathcal{H}_1$ and the remaining $L-1$ bins have an amplitude that is distributed by hypothesis $\mathcal{H}_0$.

A truth detection event occurs in the energy detector when the amplitude surpasses the detection threshold at the DFT bin corresponding to the true center frequency of the transmitting signal or either neighboring DFT bins on either side. Then, the probability of detection is 1 minus the probability that this truth detection event did not occur, as shown in Equation 3.61, where the probability that the truth detection event does not occur equals the probability of a miss detection in the true DFT bin multiplied by the product of the probability of two true negative detections in the neighboring DFT bins in which these probability quantities are functions of $P_D^{(\ell)}$ and $F_A^{(\ell)}$.

$$P_D = 1 - \underbrace{\left(1 - P_D^{(\ell)}\right)}_{\substack{\text{miss} \\ \text{detection}}} \times \underbrace{\left(1 - F_A^{(\ell)}\right)^2}_{\substack{\text{two} \quad \text{true} \\ \text{negatives}}} \tag{3.61}$$

The probability of false alarm is more challenging to express. This probability is a function of two random variables: N_D and N_F, where N_D is the number of detections returned by the energy detector in the true DFT bin or the two neighboring DFT bins and N_F is the number of detections returned by the energy detector outside of the previous three DFT bins. The random variable N_D can take four values 0, 1, 2, and 3, and has the distribution as shown in Equation 3.62 where we let $p_D = P_D^{(\ell)}$, $p_F = F_A^{(\ell)}$, and $q_i = 1 - p_i$ for $i \in \{D, F\}$. The random variable N_F is related to the $L-3$ DFT bins that do not count towards a true detection event and can be modeled as counting the number of heads in the Bernoulli process of $L-3$ biased coin flips where the probability of heads is equal to $F_A^{(\ell)}$ and is distributed as a binomial random variable, as shown in Equation 3.63. Thus, the false alarm ratio is $\frac{N_F}{N_D+N_F}$. We can estimate the probability of false alarm with the

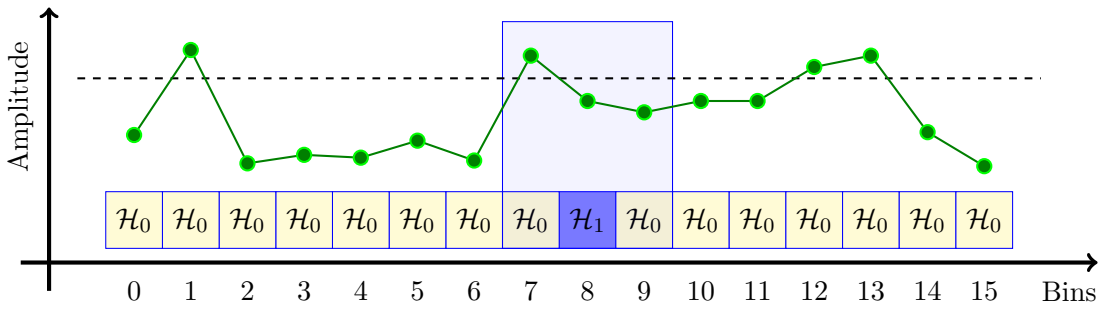


Figure 3.11: Hypothetical example when a signal is transmitted at a center frequency corresponding to the DFT bin-8 where the green line depicts the amplitude of the received signal and the dashed line represents the detection threshold and we see that $N_D = 1$ and $N_F = 3$.

expected value of $\frac{N_F}{N_D+N_F}$.

$$Pr\{\ N_D = n\ \} = \begin{cases} q_D \times q_F \times q_F & \text{for} \quad n = 0 \\ (2 \cdot q_D \times p_F \times q_F) + (q_F \times q_F \times p_D) & \text{for} \quad n = 1 \\ (2 \cdot p_D \times p_F \times q_F) + (p_F \times p_F \times q_D) & \text{for} \quad n = 2 \\ p_D \times p_F \times p_F & \text{for} \quad n = 3 \end{cases} \tag{3.62}$$

$$Pr\{\ N_F = n\ \} = \binom{L-3}{n} (p_F)^n \ (q_F)^{L-3-n} \tag{3.63}$$

In Figure 3.11, we illustrate by example the intuition of the energy detector and the method to compute N_D and N_F and also P_D and F_A. The vertical axis is the amplitude calculated by the energy detection algorithm and the horizontal axis is the frequency spectrum indexed by DFT bin where in our example we suppose there are 16 bins, i.e. $L = 16$. Suppose that there is a single transmitting signal located at DFT bin-8 and we plot the amplitude of our hypothetical received signal with the green line. We show our detection threshold as a dashed black line. We indicate for each DFT bin which hypothesis yields the correct amplitude distribution for that bin, i.e. $\mathcal{H}_1$ for DFT bin-8 and $\mathcal{H}_0$ for the remaining bins. We shade with a blue box the set of DFT bins in which a detection triggers a true detection event, DFT bin-7, 8, and 9. In this example, we see there are four DFT bin amplitudes greater than the detection threshold, however only one detection falls into the true detection event zone and thus $N_D = 1$ and $N_F = 3$, which corresponds to $P_D = 1$ and $F_A = \frac{3}{4}$.

We can compare this energy detector capability with the measured performance of our narrowband detection algorithm reported in Section 3.4.2, which is tabulated in Table 3.4. The first column in the table lists the SNR value. The second and third columns contain the measured P_D and F_A of our NSD algorithm, respectively. The fourth column reports the F_A of the energy detector when

SNR	P_D(NSD)	F_A(NSD)	F_A(ED) when we fix P_D(ED) = P_D(NSD)	P_D(ED) when we fix F_A(ED) = F_A(NSD)
0 dB	0.673	0.891	0.983	0.007
5 dB	0.901	0.572	0.977	0.129
10 dB	0.939	0.421	0.707	0.875
15 dB	0.943	0.367	0.000	1.000

Table 3.4: The energy detection algorithm's probability of detection and probability of false alarm performance comparison to the NSD measured performance.

we fixed the energy detector's P_D to equal the P_D for the NSD at the particular SNR. For example, in the first row where the SNR is 0 dB, we report the NSD to have $P_D = 0.673$ and so we set the detection threshold such that the energy detector also has a $P_D = 0.673$. With this detection threshold, we find that the corresponding F_A for the energy detector is 0.983, which is the value in the fourth column of the first row. Likewise, the fifth column reports the P_D of the energy detector when we fixed the energy detector's F_A to equal the F_A for the NSD at the particular SNR.

We can compare the probability of detection between our NSD algorithm and the energy detection algorithm by looking at the second and fifth columns in Table 3.4. We see that the P_D for our algorithm is significantly better than the energy detector at the lower SNR values: 67.3% compared to 0.7% at 0 dB SNR and 90.1% compared to 12.9% at 5 dB SNR. Even at 10 dB SNR, the P_D for our NSD algorithm outperforms the energy detection algorithm: 93.9% compared to 87.5%. At 15 dB SNR, our NSD algorithm has a $P_D = 94.3\%$ whereas the energy detector has a perfect probability of detection.

We can compare the probability of false alarm between our NSD algorithm and the energy detection algorithm by looking at the third and fourth columns in the Table 3.4. Again, we see that the F_A for our algorithm is better that of the energy detector. At 0 dB SNR, the F_A for our NSD algorithm outperforms the energy detection algorithm: 89.1% compared to 98.3%. Our NSD algorithm significantly outperforms the energy detection algorithm at 5 dB and 10 dB SNR: 57.2% compared to 97.7% and 42.1% compared to 70.7%, respectively. However, again at 15 dB SNR, our NSD algorithm underperforms with respect to the energy detector with $F_A = 36.7\%$, whereas the energy detector does not produce a false alarm in this situation.

Overall our NSD algorithm provides a better probability of detection and false alarm than the energy detector, especially at lower SNR values. This improved performance is due the fact that our NSD algorithm attempts to model the received signal as a mixture model of sinc functions. However the trade-off for this performance gain is an increase in computational complexity compared to the simplicity of the energy detection algorithm.

We have seen that at high SNR values the energy detection algorithm can give better P_D and F_A values than our algorithm, such as the 15 dB case in our experiments. This result is due to the

differences in algorithmic approach between the two techniques. The energy detector simply sets a detection threshold and recall from Figures 3.9 and 3.10 that the two hypothesis distributions become further separated as the SNR value increases. In the presence of high SNR, this relationship allows the energy detection algorithm to easily set the detection threshold such that the probability of detection is nearly 100% and the probability of false alarm is nearly 0%. In contrast, our NSD algorithm is an iterative algorithm that tries to add a new narrowband signal to the mixture model each iteration. If the algorithm has a good model of the received signal but there remain more iterations to process, the algorithm unknowingly adds unnecessary additional signals to the model. This effect inherently causes the NSD to produce false alarm detections even at high SNR values.

3.5 Discussion

In our first experiment, we showed that our NSD algorithm performs very well at detecting narrowband signals from a received wideband signal. Our initial tests showed that the algorithm estimation of the center frequency and bandwidth works well regardless of the DFT size used by the algorithm. This test shows that we can trade frequency resolution for computation-time complexity without sacrificing accuracy.

In our second experiment, we examined the performance of the NSD algorithm as we vary the maximum number of detections, M_D, and the number of algorithmic iterations, N_I. We found that the probability of detection, P_D, and probability of false alarm, F_A, maintains a consistent averaged performance value regardless of the value of M_D. When we evaluated the effect of N_I, again, we saw a consistent averaged performance value as we varied N_I. However, we noticed that there was a initial period when $N_I < 25$ where the variance on the F_A is higher than the variance on the F_A when $N_I > 25$.

Finally, we analytically compared the performance of our NSD algorithm with the expected performance of an energy detection algorithm. With the exception of 15 dB SNR, when we fix the P_D of of the energy detection algorithm to match the P_D of our NSD algorithm, the NSD had a lower F_A value than the energy detection. Likewise, except at 15 dB SNR, when we fix the F_A of of the energy detection algorithm to match the F_A of our NSD algorithm, the NSD had a higher P_D value than the energy detection. At higher SNR values the energy detection algorithm outperforms our NSD algorithm. A possible improvement would be to detect when we are we are operating in a high SNR environment and then switch to the energy detection algorithm, otherwise use our NSD algorithm.

We use the following parameterization of the NSD algorithm in our spectrum sensing architecture. We use a 256-sized DFT with $M_D = 5 \cdot D$, where D is the expected number of signals present, and $N_I = 50$. We chose these parameter values based on the results of the experiments presented.

Future work could investigate the effect of the parameters in the NSD on the resulting bandwidth estimations. By reducing the DFT size, the maximum number of detections allowed, or the number of iterations to run the algorithm, we can reduce the computation time necessary to implement this algorithm without greatly affecting the accuracy of the center frequency estimation.

Another improvement to study is a better method to insert a new signal detection in our mixture model. This feature would prevent the NSD algorithm from adding a new detection on every iteration, which we saw increased the likelihood of a higher probability of false alarm. Reinforcement learning could be used to learn a better rule for adding new signals.

Finally, we can improve the stopping mechanism of our algorithm. We showed that we can iterate the algorithm for a number of iterations and then stop where this approach yields reasonable performance. However, we potentially could create and train a classifier to predict whether or not there exists more signal in the residual signal, which would allow the algorithm to stop sooner and save computation time.

Chapter 4

Modulation Classification

Digital modulation is the process of mapping a sequence of digital information onto M discrete values in the complex plane, where this set of M values is referred to as the constellation set [84]. Given a noisy received signal, automatic modulation classification (AMC) is the problem of determining the type of modulation used to transmit information. Intelligent radio communication systems require efficient AMC in order to handle the variety of signals produced by flexible SDR communication systems [117]. Spectrum sensing applications also benefit from AMC in order to better recognize the primary users of the spectrum [19].

In this chapter, we describe a novel digital modulation classification process that we developed based on modulation constellations. We receive complex baseband samples that were transmitted over a complex additive white Gaussian noise (AWGN) channel, which are clustered onto template modulation constellations using the Expectation-Maximization (EM) algorithm. After clustering, we generate statistics to form our feature vector set, from which a score value is calculated using a weight vector. We implemented a genetic algorithm to train the weight vector. The classification rule implemented chooses the modulation that produces the smallest score value. The novelty of our constellation-based digital modulation (CBDM) classification algorithm is that we generate a unique feature set that incorporates knowledge about how a noisy signal should behave given the structure of the constellation set used to modulate the transmitted information.

We show that our CBDM classification algorithm along with features that we develop for this algorithm is fairly robust to non-ideal conditions, such as frequency offset, phase offset, and symbol boundary misalignment, due to inaccuracies caused by real-world receivers. We also show that our algorithm is superior to many of the modulation classification algorithms present in the literature. We perform three evaluations to reach this conclusion.

In the first evaluation, we vary the parameters of our classification algorithms and consider non-ideal conditions, such as frequency offset, and symbol boundary misalignment. We find that our

classifier approach does a better job at classifying the modulation-family compared to classifying the modulation-type. We see that the weighted average error of our classifier decreases in value as we increase the stride value. We observe that using individual weight vectors for each modulation template worked best, and also that using more generation cycles used in the genetic algorithm yields better weight vectors. Finally, we find that our classifier performed equally well in non-ideal test scenarios.

In the second evaluation, we perform the first comparison of modulation classification algorithms that compares accuracies achieved by classification algorithms applied to identical sets of modulation types and SNR values. First, we perform this comparison of our CBDM classification algorithm to six classifiers from the literature and find that our algorithm consistently outperforms the results reported. For example, at 0 dB SNR, the correct modulation classification accuracy of our algorithm averages 23.8 percentage points better than the results in the literature, where the most dramatic increase was from 37.5% to 98.3%. Second, we show that our algorithm had a slight degradation in accuracy when using a larger class label set, but yet still maintains comparable performance to the first experiment. This outcome is even more impressive because the class label set contains a significantly larger number of modulations. This scalability of a modulation classification algorithm to a larger class label set has not been previously discussed in the literature.

In the third evaluation, we directly compare the performance of an ensemble SVM classifier using our developed CBDM feature set against the the temporal and cumulant feature sets. To the best of our knowledge, this is the first direct comparison of feature sets in modulation classification on the *same* data set. On average over a large range of SNR values, the classifier using our constellation-based features outperforms the classifier using the temporal features by 5.31 percentage points, and also outperforms the classifier using the cumulant features by 13.35 percentage points.

The remainder of this chapter is organized as follows. In Section 4.1, we review the modulation classification work that is already in the literature. In Section 4.2, we describe the complex baseband model used to represent our signals. In Section 4.3, we outline the steps to use the EM algorithm to cluster our received samples to a modulation constellation template. In Section 4.4, our classification approach is described. In Section 4.5, we describe the simulation experiments that we performed and report the results. In Section 4.6, we comment on the performance of our classification algorithm and suggest future improvements.

4.1 Literature Review

Many pattern recognition approaches have been explored to classify modulation. A decision tree using second-moment temporal statistics to differentiate between analog and digital modulations was one of the first attempts to use pattern recognition to identify modulation type [118]. The

literature contains a wealth of pattern recognition approaches to classify the modulation of a signal [119–150]. In general, there are four considerations in developing a modulation classifier: the feature set, the classification algorithm, the modulations to include in the class label set, and the simulation's channel type and/or receiver impairments. These four items have been explored in varying depth in the literature.

The training and testing sets of modulated data can be large in size. For example, a single data instance of 1024 complex baseband samples that is stored with 32-bit floating-point precision creates a 8 kilobyte file. A good feature set choice should reduce the data size, and also maintain the necessary information to accurately discriminate between the different modulation types. There are many ways to create a feature set for the modulation classification problem. There are two common feature sets that are popular in the literature, the temporal feature set, and the higher-order and cumulant statistic feature set.

The temporal feature set was initially used with a decision tree in [118], and became the first standard choice of features for many later papers. This feature set calculates temporal statistics of the received signal, such as the mean and variance of the signal amplitude, phase and frequency. In [119], an artificial neural network was implemented using this standard set of features, and tested on a large set of modulation types, including both analog and digital were tested. However, the digital modulation types test only had symbols with one or two bits. The accuracy reported was greater than 97% at SNR values of 10 dB and 20 dB. In [120], five classifiers using temporal statistic features were evaluated, two decision tree classifiers, a minimum-distance classifier, and two neural network classifiers. All five classifiers were reported to have a correct classification performance over 95% at 10 dB SNR. In [121], a classifier that uses temporal statistics was created to discriminate between 2-FSK, 4-FSK, and 2-PSK with the purpose to ultimately identify the Link-4A communication protocol. The author reports that the algorithm can identify the protocol at SNR values of -10 dB or greater.

Another popular feature set choice calculates higher-order moments and cumulants features of the received signal. In [122], the author used a decision tree to differentiate a set of PAM, PSK, and QAM modulations, and reported an accuracy performance of 85%, 90%, and 95% at SNR values of 6 dB, 8 dB, and 10 dB, respectively. In [123], the author used a modulation set of 2-ASK, 4-ASK, 8-ASK, 4-PSK, 8-PSK, and 16-QAM, and signals were created at 10 dB and 15 dB. A support vector machine (SVM) was used as the classifier with performance above 93% for all types. In [124], a SVM was trained on a large set of digital modulations, and the author tried many different kernels for the SVM using a particle swarm optimization algorithm to find the best kernel parameters. The best result was an accuracy of 91%, 94%, 97%, 98%, and 99% for SNR values of -4 dB, 0 dB, 4 dB, 8 dB, and 12 dB, respectively. In [125], a hierarchical classification scheme based on higher-order statistics was developed. The experiments considered discriminated between two and four modulation types over a large range of SNR values. The performance reported was

over 90% for SNR values greater than 10 dB.

In some cases, a modulation classifier was developed that combined these two popular feature sets. In [126], a multilayer perceptron neural network classification algorithm was developed that use cumulant and temporal statistics as features. A genetic algorithm was used to selection which subset of features are the best. The performance reported was 99% at 0 dB SNR and 93% at -5 dB. In [127], an SVM classification algorithm was developed that use cumulant and temporal statistics as features. This work tried to identify both analog and digital modulations. The performance reported was 86% at 0 dB SNR and over 93% for SNR values of 5 dB, 10 dB, 20 dB, and 30 dB.

The appeal of these two popular feature sets is the simplicity to implement, and the ability to theoretically pre-calculate the expected feature values. However, the problem is that these features are typically derived using the additive white Gaussian noise (AWGN) channel, which makes these features susceptible to other channel types and receiver impairments.

Many of the modulation classification attempts use a standard machine-learning type of classification algorithm, such as the decision tree, neural network, or SVM. One reason is the ease to train these algorithms on classification problems by providing a set of training examples. However, a decision tree needs a good representation of the feature space distribution, which requires a very large training set, a neural network can overfit, and a SVM has many parameter configurations, which requires a search to find the best configuration. These issues lead towards work with other types of classification algorithms.

In [128], fuzzy clustering was used to generate a probability density function of the complex samples, and the resulting density function was used to classify the signal. The author ran three tests, which only had two or three modulations under consideration, and the performance reach 90% at low SNR. In [129], the Wavelet transform is used to de-noise the samples, followed by subtractive clustering to determine the number of peaks in their constellation. The authors considered only the QAM modulation type, and experimented with two, four, five, and six bits per symbol. The performance reported was 72%, 100%, and 100% for SNR values 0 dB, 5 dB, and 10 dB, respectively. A nearest-neighbor classifier was tried in [130], and the author reports on small set of modulations with only one or two bits per symbol, and tested over a number of SNR value for which 3 dB was the lowest, which performed at an accuracy of at least 95%.

In [131], a genetic algorithm was used to cluster the received samples in the complex plane, and then a hierarchical clustering algorithm further merged the clusters. In [132], the previous work was improved upon by using a fuzzy clustering algorithm, and also a Hamming neural network. The author experimented with PSK and QAM modulations over a SNR value range from 0 dB to 20 dB, and reports 100% accuracy at 0 dB, 8 dB, and 15 dB for 4-PSK, 8-PSK, and 16-PSK, respectively, and 100% accuracy at 0 dB, 3 dB, 10 dB, and 17 dB for 4-QAM, 16-QAM, 64-QAM, and 256-QAM, respectively. In [133], a classifier is developed that uses genetic programming to

select the best higher-order features. A k-nearest neighbor classification is used to make the label selection. The author considered BPSK, QPSK, 16-QAM, and 64-QAM. The performance reported was 89.3%, 99.8%, and 99.9% at 4 dB, 12 dB, and 20 dB, respectively.

In [134], a maximum likelihood classifier was developed based on underlying modulation constellations. The experiments only had to differentiate between two to four modulations, and the performance reported 95% accuracy at SNR values between 6 dB to 10 dB, which depended on which test was performed. In [135], a generalized likelihood ratio test is created to discriminate between BPSK and QPSK modulations. The experiments in this paper operated at low SNR values, -10 dB to 0 dB. The performance reported ranged from 75% to 95%. In [136], a discrete likelihood ratio test based on rapid estimation capable to identify most M-ary linear digital modulations in real-time is developed. The author experiments with tests to discriminate between two types of PSK signals as well as tests to discriminate between a PSK and QAM signal. This classifier outperforms the classical likelihood ratio test classifier.

In [137], an algorithm was developed to recognize 2-PSK, 4-PSK, and 8-PSK by modeling the expected distribution of the received signal due to the constellation diagrams. The author reports 95% correct classification for BPSK at 0 dB SNR, QSPK at 4 dB SNR, and 8-PSK also at 4 dB SNR. In [138], a classification algorithm was developed based on the Kolmogorov-Smirnov test to classify QAM. The algorithm calculates a statistic from the cumulative distribution function and compares it with a reference value. The author states that this approach offers a lower complexity classifier compared to cumulant-based techniques with better performance.

The variety of classification algorithms discussed have the tendency to specialize on a limited set of modulations. The problem in this case is that the algorithm may have been tailored to this specialty, and may not extend well to other modulation types. For example, if an algorithm is developed to classify QAM modulation types only, then that algorithm may not perform well at classifying PSK modulation types.

There are many modulation types that exist for radio communications. The selection of which modulation types to use as the class label set for the classifier varies from author to author. This variety in modulation class label sets creates a significant challenge to directly compare different modulation classification approaches because there does not exist a common testbench data set in the literature. The simplicity to simulate and synthesize data examples of different modulation types is one reason for the lack of a common testbench. Another problem with the lack of a common testbench is that many examples in the literature use a class label set that contains only modulation types that transmit 1 or 2 bits per symbol. In this situation, if the modulation family is determined, for example PSK, then it is simple to differentiate between BPSK, which is 1 bit per symbol, and QPSK, which is 2 bits per symbol. By limiting the modulation set to 1 or 2 bits per symbol, the classification task is simpler. A third problem with the lack of a common testbench is that many examples in the literature use small class label sets, such as 2, 3, or 4 modulation

types. The difficulty of classifying a small number of class labels is much easier than larger class label set. Additionally, the literature does not remark on the ability to scale to larger class label sets for these algorithms that experimented with smaller sets.

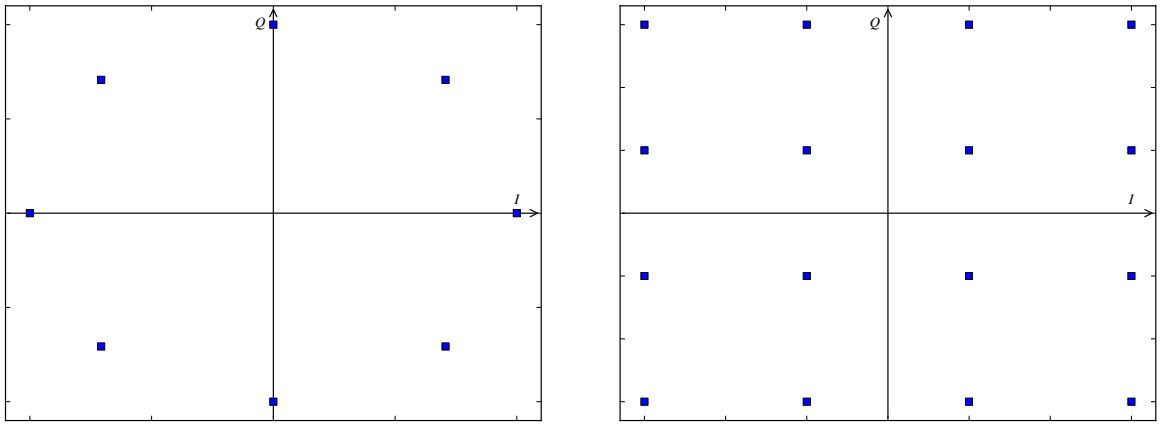
Most of the work in the literature is focus on the ideal-case receiver with only the AWGN channel. This decision is a good initial evaluation a modulation classification algorithm. However, other work in the literature tries to incorporate receiver impairments, spectral pulse-shaping filters, and also fading channels.

In [139], a classifier is developed that uses higher-order statistics and cumulants extracted from signals that are filtered by a raised cosine pulse shape. The author tried to discriminate between of two modulations at a time, where the modulations were ASK, PSK, and QAM type of modulations. The author varied the number of received symbol to understand the behavior of their algorithm. The performance reported was near 60% when using only 100 symbols, but increased to nearly 100% the number of symbols became large (> 7000). In [140], a decision tree classifier was developed that uses higher-order and cumulant statistics that is robust against the presence of a carrier frequency offset. The performance reported was 90% and 95% at 10 dB and 15 dB SNR, respectively.

In [141], cumulant statistics are used to identify digital modulations on signals that experience Rayleigh fading. The author created two classifiers. One classifier determined BPSK versus QPSK with a reported performance of 80% at 0 dB and greater with perfect channel information. One classifier discriminated between 4-QAM, 16-QAM, and 64-QAM with a reported performance of 60% at 0 dB with perfect channel information and 75% at 10 dB and greater with perfect channel information and In [142], a classifier was developed to discriminate between 8-PSK and $\frac{\pi}{4}$-shifted QPSK. This situation is challenging, because the received symbol distribution in a constellation diagram look the same for the two types of modulation. The simulation results reported that the scheme yield good classification at low SNR values in both additive white Gaussian noise and fading channels.

In [143], cyclostationary features were tested, but the author only tries to determine the modulation of a signal, in a four-tap multipath environment. The tests used SNR values in the range of 0 dB to 20 dB at 4 dB incremental steps, with performance above 85% for all SNR values. In [144], a bank of deterministic particle filters were used to jointly estimate the channel and determine the modulation scheme in used. This work consider a multi-path channel environment. The reported performance was 60%, 70%, 85%, 100%, and 100% at 5 dB, 10 dB, 15 dB, 20 dB, and 25 dB SNR, respectively. In [145], a multi-stage Gibbs algorithms for constellation classification was created to identify digital modulations for signals affected by unknown channel conditions. The algorithm demonstrated robustness to linearly distortive finite-impulse response channels.

The literature contains a large body of work on the modulation classification problem. Our work contributes to this body in a number of ways. We create a novel constellation-based digital modu-



(a) 8-PSK (b) 16-QAM

Figure 4.1: Example IQ constellations for two different modulation families.

lation classification algorithm that uses a feature set that exploits the knowledge about how a noisy signal should behave given the structure of the constellation set used to modulate the transmitted information. Our algorithm is limited to digital modulations that can be easily represented by a constellation set, however we made our class label set large with 12 class labels. We perform a direct comparison with other classifiers in the literature. We also show that our algorithm has a small performance degradation as we increase the number of class labels. We derive our algorithm with the AWGN channel in mind, but we believe the classification accuracy with not be degraded in flat-fading channels, which we plan to explore in the future.

4.2 Complex Baseband Model

The classification algorithm that we developed is specifically designed to classify digital modulations. The types of M-ary digital modulation families under consideration are PAM, QAM, and PSK, which were defined in Section 2.1.4. For a given modulation type, we let $\mathcal{M}$ denote the modulation constellation as a set of M known constellation points in the complex plane. We call a modulation constellation a template when the scaling of the constellation is unknown. Figure 4.1 depicts two sample template constellation plots for 8-PSK and 16-QAM, where the horizontal axis is the in-phase (I) component, and the vertical axis is the quadrature phase (Q) component. Appendix B contains the constellation plots for all modulation types that we consider in our evaluation.

We developed a classifier that tries to determine the modulation type used to generate the received symbols. For each modulation type, we use the EM algorithm to cluster the received symbols, where we use the modulation constellation as the cluster means. However, we are uncertain about

the scaling of the received symbols, so we use the template modulation constellation instead, and the EM algorithm estimates the appropriate scaling to use.

We assume that the digitally modulated signal is transmitted through a complex Gaussian channel, and we also assume an ideal rectangle pulse filter was used in the transmitter. Let N be the message length or the number of information symbols transmitted, and let K be the number of samples per symbol. Let τ be the symbol boundary offset, where this offset is the number of samples that the receiver mistakenly believes belongs to the previous symbol. Let f_o be the carrier frequency offset between the true carrier frequency of the transmitted signal and the estimated carrier frequency by the receiver.

We have the following relationships for our sequences based on the mathematical relationships defined in Section 2.1. The information symbols are $x[n]$ for $n = 1, 2, ..., N$, where n is the symbol index. The assumption that the pulse filter is ideal implies that the transmitted symbols are $s[k] = x[n]$ for $k = n \cdot K + k'$, where k is the sample index. The corrupted version of the transmitted symbols are the received symbols, as shown in Equation 4.1, where $v[\cdot]$ are independent identically distributed complex Gaussian random variables, and τ, and f_o, are a timing, and frequency offsets, respectively, due to the inaccuracies of the receiver. Our estimated modulation symbols $y[n]$ are defined by Equation 4.2.

$$r[k] = \Big(s[k+\tau] + v[k+\tau]\Big) \cdot exp\Big\{j2\pi f_0[k+\tau]\Big\} \tag{4.1}$$

$$y[n] = \sum_{k'=nK}^{nK+(K-1)} \Big(s[k'+\tau] + v[k'+\tau]\Big) \cdot exp\Big\{j2\pi f_0[k'+\tau]\Big\} \tag{4.2}$$

We define our data set, Y, to be the set of received complex baseband samples $y[n]$ for $n = 1, 2, ..., N$. For our initial derivation, we assume ideal conditions, i.e. $\tau = f_o = 0$, which simplifies Equation 4.2 to $y[n] = x[n] + w[n]$, where $w[n]$ is the summation of K complex Gaussian random variables, and each sample is independent and identically distributed $w[n] \overset{\text{iid}}{\sim} CN(0, N_0)$ for $n = 1, 2, ..., N$. We also assume that the in-phase and quadrature phase components of the noise are independent, i.e. $w_I[n]$, $w_Q[n] \overset{\text{iid}}{\sim} N(0, \sigma^2)$, where $w_I[n] = \Re\{w[n]\}$, $w_Q[n] = \Im\{w[n]\}$, and $\sigma^2 = \frac{N_0}{2}$.

The transmitted symbol $x[n]$ is a complex symbol from the modulation constellation template $\mathcal{M}$, where there exists $\mu_m \in \mathcal{M}$, such that $x[n] = A \cdot \mu_m$, for which A is an unknown complex scalar due to the unknown power of the transmitter and unknown phase rotation due to the channel. We assume that value of A and the set $\mathcal{M}$ are constant over our set of samples.

To simplify the notation, when the meaning is clear, we let a subscript indicate an index, *e.g.* $x[n] \leftrightarrow x_n$. Let μ_m be a given point in a modulation constellation template and let $\theta = \{A, \sigma^2\}$ be our parameters, then the probability of receiving the sample y_n given θ and μ_m is expressed in

Equation 4.3.

$$Pr\{\ y_n \mid \theta, \mu_m\ \} = \frac{1}{2\pi\sigma^2} exp\left\{-\frac{1}{2\sigma^2}||y_n - A\mu_m||^2\right\} \tag{4.3}$$

Using Equation 4.3 we can derive the EM algorithm that uses the modulation constellation templates in order to cluster the data set Y.

4.3 The EM Algorithm

We use the Expectation-Maximization (EM) algorithm to cluster our received data set with the constellation points from a single, fixed modulation type. After the EM algorithm completes, we extract features that are associated with the fixed modulation type. We repeat this process for all of the modulation constellation templates, and the classifier uses all of the features to determine the modulation type that generated the received data set. We describe our classification process in Section 4.4.

The EM algorithm finds the maximum likelihood solution to models with hidden variables [90], which we use to determine the clustering of our received samples. Recall that we previously defined a received sample, y_n, as the sum of a transmitted symbol, x_n, and a zero-mean complex Gaussian random variable, w_n, which implies that y_n is a complex Gaussian random variable with a mean equal to x_n. For the modulation constellation template $\mathcal{M}$, the transmitted symbol x_n can be one of M possible values, where $M = |\mathcal{M}|$, and the exact value is unknown due to the unknown transmit power and channel effects. However, the constellation template $\mathcal{M}$ creates a structural relationship between each constellation point in the complex plane. If we knew the variance of the noise, σ^2, and the scaling of the transmitted symbols, A, then we could cluster the received sample to the constellation point $\mu_m \in \mathcal{M}$ that is most likely responsible for generating that received sample. The EM algorithm is suitable for the task of estimating these parameters, $\theta = \left\{A, \sigma^2\right\}$, in order to perform our clustering, where the cluster means are constrained to be the scaled-version of the constellation points in the template, i.e. $A\mu_m$ for all $\mu_m \in \mathcal{M}$.

The input to the EM algorithm is the data set Y, and a modulation constellation template $\mathcal{M}$, which in the EM framework is called the *incomplete* data set. As a reminder we derive the EM algorithm using a single template, but our classifier uses the EM algorithm once for each of the modulation constellation templates. We let the random vector Z be the set of all z_{nm}, where z_{nm} is a binary random variable that equals 1 if y_n belongs to the m^{th} Gaussian random variable associated with $\mu_m \in \mathcal{M}$, and 0 otherwise. The set $\{\ Y, \mathcal{M}, Z\ \}$ is called the *complete* data set in the EM framework.

The EM algorithm is an iterative algorithm that performs an expectation and a maximization in each iteration. The expectation step, or E-step, calculates $Q\left(\theta|\theta^{(t)}\right)$, which is the expected value of the complete data log-likelihood function with respect to Z conditioned on Y [151] and shown

in Equation 4.4. The maximization step, or M-step, finds a new parameter estimate $\theta^{(t+1)}$ that maximizes $Q\left(\theta|\theta^{(t)}\right)$ using the current estimates.

$$Q\left(\theta|\theta^{(t)}\right) = E_{Z|Y}\Big[\ \log\ Pr\{\ Y, Z \mid \theta, \mathcal{M}\ \}\ \Big|\ Y\ \Big] \tag{4.4}$$

In order to calculate $Q\left(\theta|\theta^{(t)}\right)$, we need an expression $Pr\{\ Y, Z \mid \theta, \mathcal{M}\ \}$. We start by expressing the joint probability as a mixture of complex Gaussian distributions based on the constellation $\mathcal{M}$ as shown in Equation 4.5. Since the symbols in Y are assumed to be independent of each other, we use the product rule of probability to express the total probability in terms of the individual symbol, as shown in Equation 4.6. For an individual symbol y_n, only one random variable in the set $\{z_{n1}, ..., z_{nM}\}$ has the value one and the rest are zero valued. This allows us to express the conditional complete data probability, as shown in Equation 4.7, which is an expression that can be more easily manipulated. Recall that we assume an equally likely prior distribution on the constellation points, i.e. $Pr\{\ \mu_m\ \} = \frac{1}{M}$, for $m = 1, 2, ..., M$, which gives us our final expression in Equation 4.8.

$$Pr\{\ Y, Z \mid \theta, \mathcal{M}\ \} = \sum_{m=1}^{M} Pr\{\ Y, Z \mid \theta, \mu_m\ \}\, Pr\{\ \mu_m\ \} \tag{4.5}$$

$$= \sum_{m=1}^{M} \prod_{n=1}^{N} Pr\{\ y_n, Z \mid \theta, \mu_m\ \}\, Pr\{\ \mu_m\ \} \tag{4.6}$$

$$= \prod_{m=1}^{M} \prod_{n=1}^{N} [Pr\{\ y_n \mid \theta, \mu_m\ \}\, Pr\{\ \mu_m\ \}]^{z_{nm}} \tag{4.7}$$

$$= \left(\frac{1}{M}\right)^{N} \prod_{m=1}^{M} \prod_{n=1}^{N} [Pr\{\ y_n \mid \theta, \mu_m\ \}]^{z_{nm}} \tag{4.8}$$

4.3.1 EM Algorithm: E-step

The expectation step computes the expected value of Z conditioned on the data and the parameters, i.e. $E\left[z_{nm}|Y, \mathcal{M}\right]$ for all n and m. For brevity we denote $E\left[z_{nm}|Y, \mathcal{M}\right]$ by γ_{nm}, which is referred as the responsibility that the m^{th} constellation point contributes to the n^{th} observed value, and is computed as shown in Equation 4.9. Note that we assumed that each constellation point is equally likely to be transmitted, otherwise we would need to incorporate those probabilities.

$$\gamma_{nm} = \frac{Pr\{\ y_n \mid \theta, \mu_m\ \}}{\sum_{j=1}^{M} Pr\{\ y_n \mid \theta, \mu_j\ \}} \tag{4.9}$$

An intuitive explanation of this expectation step is as follows. Suppose we have two constellation

points, μ_1, and μ_2, and we have a sample y_n that is located at some point in the complex plane, which is the result of transmitting either μ_1 or μ_2. The responsibility γ_{n1} is the probability that we would receive y_n if μ_1 was transmitted, and likewise, the responsibility γ_{n2} is the probability that we would receive y_n if μ_2 was transmitted. If we computed $\gamma_{n1} = 0.95$, and $\gamma_{n2} = 0.05$, then, for us to receive y_n, we would believe that μ_1 was most likely transmitted, or most responsible. This calculation would occur when y_n is located in the complex plane very close to μ_1. However, if we computed $\gamma_{n1} = \gamma_{n2} = 0.5$, then, for us to receive y_n, we could not judge whether μ_1 or μ_2 is most responsible for us to receive y_n. This calculation would occur when y_n is located equidistant in the complex plane to both μ_1 and μ_2.

4.3.2 EM Algorithm: M-step

The maximization step finds the parameter values that maximize the expected conditional complete log-likelihood function, $Q\left(\theta|\theta^{(t)}\right)$, given the previous E-step calculation. For brevity, we let $\mathcal{J}$ be equal to $Q\left(\theta|\theta^{(t)}\right)$, which is the objective function to find the maximum log-likelihood estimates of our parameters. In Equation 4.10, we take the logarithm of Equation 4.8 to express $\mathcal{J}$. In Equation 4.11, we substitute the expression for $Pr\{\ y_n \mid \theta, \mu_m\ \}$, from Equation 4.3, and ignore `const`, which equals the quantity $-N \log M$, and does not influence the maximization. We apply the properties of logarithms to give Equation 4.12.

$$\mathcal{J} = \sum_{n=1}^{N}\sum_{m=1}^{M} E[z_{nm}] \log Pr\{\ y_n \mid \theta, \mu_m\ \} + \texttt{const} \tag{4.10}$$

$$= \sum_{n=1}^{N}\sum_{m=1}^{M} \gamma_{nm} \log \left[\frac{1}{2\pi\sigma^2} exp\left\{-\frac{1}{2\sigma^2}||y_n - A\mu_m||^2\right\}\right] \tag{4.11}$$

$$= \sum_{n=1}^{N}\sum_{m=1}^{M} \gamma_{nm} \left[-\log\left(2\pi\sigma^2\right) - \frac{1}{2\sigma^2}||y_n - A\mu_m||^2\right] \tag{4.12}$$

To estimate the noise variance, and complex amplitude, we take the derivative of the objective function with respect to σ^2, and A, as shown in Equations 4.13 and 4.14, respectively.

$$\frac{\partial \mathcal{J}}{\partial \sigma^2} = \frac{\partial}{\partial \sigma^2}\sum_{n=1}^{N}\sum_{m=1}^{M} \gamma_{nm} \left[-\frac{1}{\sigma^2} + \frac{1}{2}\frac{1}{(\sigma^2)^2}||y_n - A\mu_m||^2\right] \tag{4.13}$$

$$\frac{\partial \mathcal{J}}{\partial A} = \frac{\partial}{\partial A}\sum_{n=1}^{N}\sum_{m=1}^{M} \gamma_{nm} \frac{1}{\sigma^2}\left[\ y_n - A\mu_m\ \right]\mu_m^* \tag{4.14}$$

Setting the derivatives to zero and rearranging terms, we find the maximum likelihood estimates of σ^2 and A, in Equation 4.15 and Equation 4.16, respectively. Note that μ_m^* is the complex conjugate of μ_m.

$$\sigma^2 = \sum_{n=1}^{N}\sum_{m=1}^{M} \gamma_{nm}\ \frac{1}{2}||y_n - A\mu_m||^2 \tag{4.15}$$

$$A = \frac{\sum_{n=1}^{N}\sum_{m=1}^{M} \gamma_{nm}\ y_n\ \mu_m^*}{\sum_{n=1}^{N}\sum_{m=1}^{M} \gamma_{nm}\ \mu_m\ \mu_m^*} \tag{4.16}$$

We see in Equation 4.15 that the estimate of σ^2 is half the sample complex noise power, which intuitively matches our definition for σ^2. The estimate of A in Equation 4.16 is also a reasonable equation. The numerator is average power of the data projected onto each constellation point, the denominator is average power of the constellation points themselves, and the ratio of the numerator and denominator gives the scale factor A.

The EM algorithm cycles between the E-step and M-step while updating the responsibilities γ_{nm} and unknown parameters A and σ^2. The EM algorithm stops when the percent change of $Q\left(\theta|\theta^{(t)}\right)$ between time $t-1$ and t is less than 0.01%. After the EM algorithm completes, we cluster each symbol in Y to the constellation point that provides the most responsibility, and we define the m^{th} cluster, C_m, in Equation 4.17.

$$C_m = \{n : \gamma_{nm} > \gamma_{nk}, k \neq m, y_n \in Y\} \tag{4.17}$$

This clustering and other aspects of the EM algorithm computation are used to create our features for classification, which we discuss in Section 4.4.1.

4.4 Classification Process

We implemented a classifier that selects the modulation type with a feature vector extracted from the results of EM algorithm that produces the lowest score, where the score is the inner product between the feature vector and a trained weight vector. We define the set $\mathcal{T}$ to be the template set that contains all of the template modulation constellations, and for some template $T \in \mathcal{T}$, we let $F_T(Y)$ be the feature vector extracted from the result of the EM algorithm using the template T and the data set Y, and we let W_T be the weight vector. We discuss the feature and weight vectors below. The decision rule for the classifier is shown in Equation 4.18, and the classifier implementation is illustrated in Figure 4.9.

$$\hat{T} = \arg\min_{T\in\mathcal{T}} < F_T(Y), W_T > \tag{4.18}$$

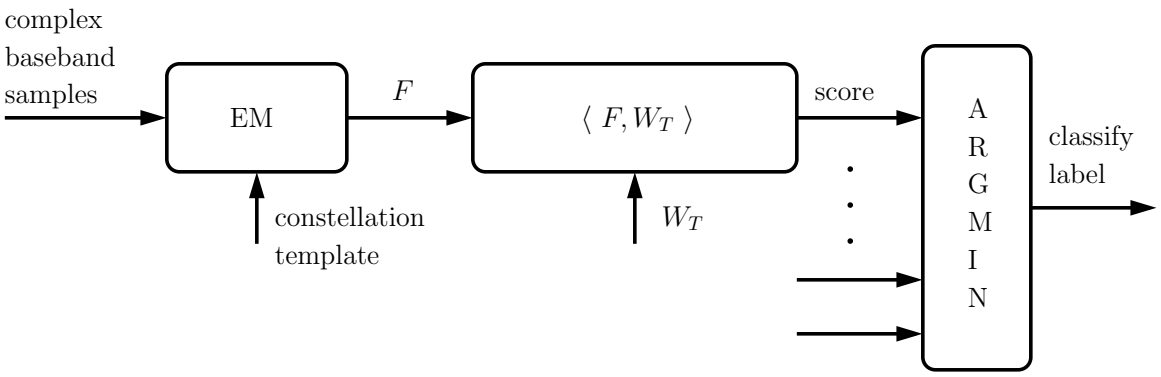


Figure 4.2: Structure of the classifier implementation.

4.4.1 Feature Vector Description

We calculate seven feature values, $F_1, F_2, ..., F_7$, from the data set Y as a function of the clustering results of the EM algorithm for each template modulation constellation, T. We denote the feature vector as $F_T(Y) = [F_1, F_2, ..., F_7]$. For the data set Y, only one template modulation constellation is the correct template, and we expect that our features will produce discriminating values between the correct and incorrect template modulation constellations.

Feature F_1 is the expected conditional complete log-likelihood function averaged over the number of samples in the data set, as shown in Equation 4.19. The EM algorithm checks that $Q\left(\theta|\theta^{(t)}\right)$ has converged to a local optimum to determine when to stop iterating. The normalization is intended to make this feature invariant to received signals of different sequence lengths.

$$F_1 = \frac{1}{N} Q\left(\theta|\theta^{(t)}\right) \tag{4.19}$$

Feature F_2 is the Shannon-Jensen divergence of the cluster distribution, as shown in Equation 4.20, which is a similarity measure between two probability distributions [89]. We assume that the transmitted symbols are distributed uniformly over the M template constellation points, which represents the cluster distribution p, where $p(m) = \frac{1}{M}$ for $m = 1, 2, ..., M$. We let the distribution $\tilde{p}$ be the estimated cluster distribution based on the EM algorithm clustering, where $\tilde{p}(m) = \frac{|C_m|}{N}$ for $m = 1, 2, ..., M$. The Shannon-Jensen divergence is based on the Kullback-Leibler divergence, as shown in Equation 4.21, where $D(x\|y) = \sum_m x(m) \log \frac{x(m)}{y(m)}$ and $\xi = \frac{1}{2}(p + \tilde{p})$.

$$F_2 = SJD\left(\ p, \tilde{p}\ \right) \tag{4.20}$$

$$= \frac{1}{2} D(p\|\xi) + \frac{1}{2} D(\xi\|\tilde{p}) \tag{4.21}$$

Feature F_3 is a measure of the model complexity, as seen in Equation 4.22, which when combined

with feature F_1 is typically referred as the Bayesian information criterion, and is commonly used to distinguish clusterings with different numbers of cluster points [152], which is our case since we compare modulations with varying constellation set sizes.

$$F_3 = \frac{1}{N}\left[M + \log\, N\right] \tag{4.22}$$

Feature F_4 is the number of small clusters produced by the EM algorithm, as shown in Equation 4.23. For each cluster C_m, we expect $\tilde{p}(m)$ to have a value close to $\frac{1}{M}$. We consider the m^{th} cluster to be small if $\tilde{p}(m) < \epsilon$, where $\epsilon = \frac{1}{8M}$, which implies that the cluster size is 8 times smaller than expected. We chose the factor 8, because it is highly unlikely that the correct template would produce a cluster with a size that is 12.5% of the expected cluster size.

$$F_4 = \left|\left\{m : \tilde{p}(m) < \epsilon\right\}\right| \tag{4.23}$$

Feature F_5 is the average minimum within-cluster distance-squared of each cluster C_m. If a cluster contains zero symbols, then we add a large penalty, P, as shown in Equation 4.24. Due to our assumption that the noise is a complex Gaussian process, we expect a cloud of symbols clustered around each constellation point, and also there should be a symbol that is very close to that constellation point. If we try to cluster using an incorrect constellation set, then the assumptions become invalid and we expect to possibly have clusterings of symbols that are far from constellation points.

$$F_5 = \frac{1}{M}\left[\sum_m \min_{n \in C_m} ||y_n - A\mu_m||^2 + \sum_{m:|C_m|=0} P\right] \tag{4.24}$$

Feature F_6 is the number of EM algorithm iterations until a convergence is reached. We expect this feature to be a low number when running the algorithm with the correct constellation, since the structure of the constellation matches that of the signal and the EM algorithm should find the local optimum quickly. Otherwise this feature should be a high number, because EM algorithm will struggle to match the incorrect constellation with the signal. In our data set, which will be discussed later, we found that the average value of this feature to be 29.48 when running the algorithm with the correct constellation, otherwise the average value of this feature was 39.63, which confirms our intuition about this feature.

$$F_6 = \text{number of EM algorithm iterations} \tag{4.25}$$

Feature F_7 estimates the SNR of the data set from A and σ^2 which are estimated by the EM algorithm, as shown in Equation 4.26, where the average energy per symbol $\bar{E}_s$ is calculated from the complex amplitude A and the template modulation constellation, and σ^2 is the noise variance.

Feature Value Mathematical Formulation

$$
\begin{aligned}
F_1 &= \frac{1}{N} Q\left(\theta|\theta^{(t)}\right) \\
F_2 &= SJD\left(\ p, \tilde{p}\ \right) \\
F_3 &= \frac{1}{N}\left[M + \log\ N\right] \\
F_4 &= \left|\left\{m : \tilde{p}(m) < \epsilon\right\}\right| \\
F_5 &= \frac{1}{M}\left[\sum_m \min_{n \in C_m} ||y_n - A\mu_m||^2 + \sum_{m:|C_m|=0} P\right] \\
F_6 &\equiv \text{number of EM algorithm iterations} \\
F_7 &= 10\ \log_{10}\ \left(\frac{\bar{E}_s}{2\sigma^2}\right)
\end{aligned}
$$

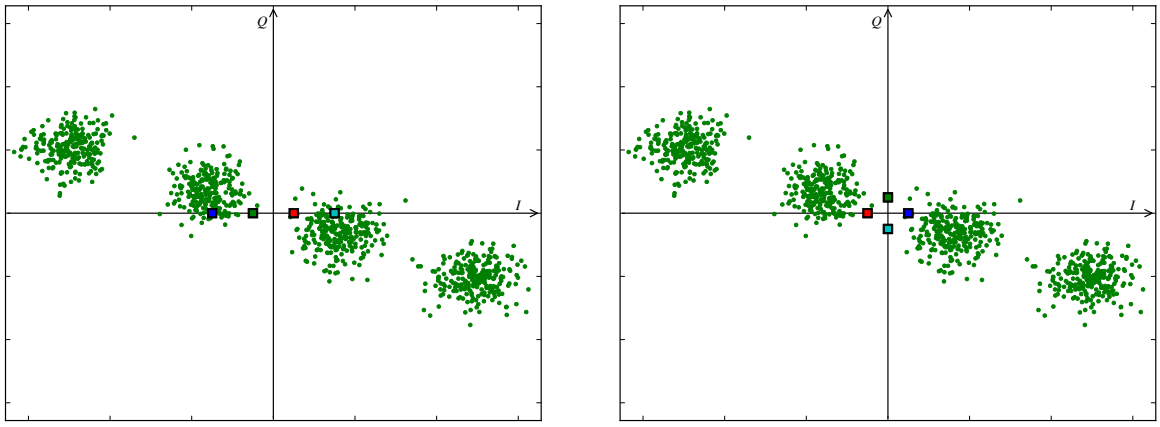
Table 4.1: Feature values produced by the EM algorithm for use in the classifier.

This feature is most useful when we have prior knowledge of the channel SNR.

$$F_7 = 10\ \log_{10}\ \left(\frac{\bar{E}_s}{2\sigma^2}\right) \tag{4.26}$$

We summarize our seven features in Table 4.1.

We expect that the log-likelihood feature, F_1, provides the most discriminative information between modulation types because by its definition this is the likelihood of the received samples given the modulation type. The feature F_2 gives an indication that the received samples are uniformly distribution about the cluster means. As we mentioned, the feature F_3 in conjunction with F_1 gives us a model complexity. In many cases, these three feature values can discriminate between the different modulation types. However, there are situations that this is not true, and we need more feature values. The features F_4 and F_5 determine how well the clustering of the received samples match the structure of the constellation template, which helps discriminate between modulation types when the first three features cannot. For example, suppose a BPSK signal is received at very low SNR. We expect the features F_1 and F_2 to have values near 1 and 0, respectively, which is the correct behavior when using the BPSK constellation template. However, when using a higher-order modulation constellation template, such as 64-QAM, we can also find the features F_1 and F_2 to have values near 1 and 0, respectively, which is due to the larger degrees-of-freedom provided by the higher-order modulation constellation template. The features F_4 and F_5 help alleviate this issue because it is unlikely for all of the cluster means of the higher-order incorrect modulation template to match the underlying BPSK structure of the received samples, even though the likelihood is



(a) 4-PAM template constellation

(b) 4-PSK template constellation

Figure 4.3: Example EM algorithm start state on two templates using received data from 4-PAM.

near to 1. Finally, the features F_6 and F_7 are values produced during the EM algorithm processing, so these values are given to us computationally free. In learning our weight vectors value during training, which is explained in Section 4.4.2, we allow a genetic algorithm to determine which feature values are the most important to providing modulation type discrimination and adjust the associated weight values accordingly.

Figures 4.3, 4.4, and 4.5, illustrate an example where we receive samples that were generated using 4-PAM with noise added, run the EM algorithm on the 4-PAM and 4-PSK templates, and qualitatively describe the feature values that are produced.

In Figure 4.3, we show the constellation plot of the received samples in green, and we notice the four distinct clusters of points. We overlay the 4-PAM template with squares in Figure 4.3a, and likewise we overlay the 4-PSK template with squares in Figure 4.3b, which represents the starting point of the EM algorithm. In both cases we expect the EM algorithm to scale the templates to better fit the data. In the 4-PAM case, we also expect the EM algorithm to apply a phase rotation to the template to align with the received data. We imagine that the EM algorithm will rotate the 4-PSK template, as well, but it is difficult to guess the amount of rotation. This uncertainty comes from the fact that the 4-PSK template forms a square, but the data generated by the 4-PAM template forms a line, and we are unsure which orientation of a square is most likely to fit a line in a probabilistic sense.

Figure 4.4 shows the clustering produce by the EM algorithm with the 4-PAM template, where the square points indicate the cluster means and the smaller points are the received samples, where the color indicates the cluster membership. As expected, the EM algorithm scales and rotates the 4-PAM template to match the received samples very well, which should cause the conditional

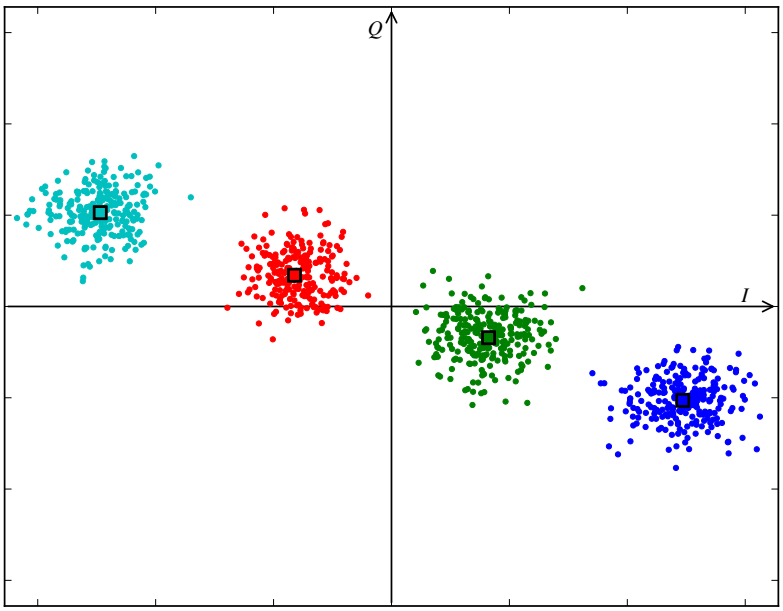


Figure 4.4: Final EM algorithm clustering result using the 4-PAM template constellation on received data generated from 4-PAM modulation.

complete log-likelihood value (F_1) to be large. The distribution of points is nearly uniform, which causes both the Shannon-Jensen divergence (F_2), and the number of small clusters (F_4) to be small. The average minimum within-cluster distance-squared (F_5) should also be small, because we see many received sample points near each cluster mean point.

Figure 4.5 shows the clustering produced by the EM algorithm with the 4-PSK template, where the square points indicate the cluster means, the smaller points are the received samples, and the color indicates the cluster membership. We see that the EM algorithm scales and rotates the 4-PSK template to align one of the template's dimensions with the received samples. Obviously, the 4-PSK template is the incorrect type, but, the EM algorithm is performing correctly since the PAM modulation uses only one dimension to transmit symbols. The conditional complete log-likelihood value for the 4-PSK template is smaller than that found with the 4-PAM template. We see that the distribution of points is uneven, i.e. all samples belong to two out of the four cluster means, which causes the Shannon-Jensen divergence to be larger than that found with the 4-PAM template. The cluster mean in the first quadrant is a cluster with 3 samples, which are encircled in grey in Figure 4.5, and the cluster mean in the third quadrant is a cluster with 1 sample, which is also encircled in grey in Figure 4.5. We expect the average minimum within-cluster distance-squared to be large because every cluster mean is far from a received sample.

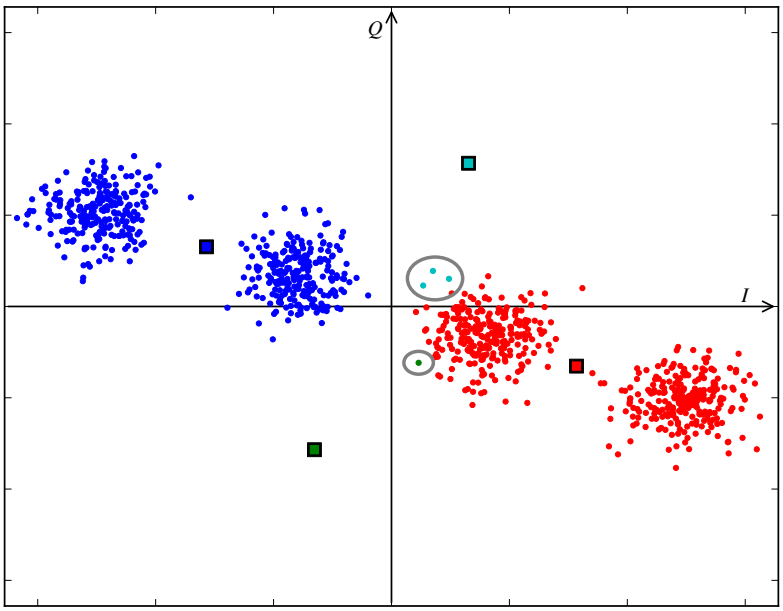


Figure 4.5: Final EM algorithm clustering result using the 4-PSK template constellation on received data generated from 4-PAM modulation.

4.4.2 Weight Vector Training

Our classifier uses supervised training to learn the weight vector, which means our training set consists of training instances consisting of a data set Y and modulation type label. We implemented a genetic algorithm search to train our weight vectors W_T for all $T \in \mathcal{T}$.

A genetic algorithm is search algorithm that uses evolutionary principles to build the best fit structure to solve the problem of the search, where this structure is called a chromosome [153]. The search begins with a population of chromosomes equal to an initial seed chromosome, which we consider as the zero generation. The algorithm iterates over a number of generation cycles to find the best chromosome.

One generation cycle performs the following five steps, which we discuss in more detail below:

1. We randomly select a chromosome subset from the population to mutate
2. We randomly select a chromosome subset from the population to mate
3. We add new chromosomes to the population
4. We evaluate the fitness of each chromosome in the population
5. We remove a number of chromosomes from the population

4.4.2.1 Chromosome Mutation

We randomly select a subset of chromosomes from the population to mutate, such that the subset contains 10% of the population. The mutation operation on a chromosome W consists of a number of random vectors, G_1, G_2, R_1, R_2, and S, of size equal to W, where each element of G_1, G_2 are i.i.d. Bernoulli$\left(\frac{1}{2}\right)$ random variables, each element of R_1, R_2 are i.i.d. Uniform$(-1, +1)$ random variables, and each element of S are i.i.d. random variables that take values of -1 or $+1$ with the probability of $+1$ equal to 0.995. We use the mutate rule, shown in Equation 4.27, to update the chromosome W, where the products in the rule are element-wise products.

$$\texttt{mutate}(W) = (1 - G_1) \cdot W + G_1 \cdot S \cdot W \cdot (1 + 0.125R_1) + 0.001 \cdot G_2 \cdot R_2 \tag{4.27}$$

The mutation randomly scales the weight vector so that it can grow or shrink in value. We see in Equation 4.27 that the value of G_1 either keeps the old W value or uses a scaled version of W. The scaled version is a function of R_1 and S, where R_1 determines the scale factor between 87.5% and 112.5% of the value of W, and S can change the sign value.

We also see in Equation 4.27 that we add a small value that depends on G_2 and R_2 to the weight vector. This additional value, which ranges from -0.001 to +0.001 prevents an all zero weight vector from remaining as such indefinitely.

4.4.2.2 Chromosome Mating

We randomly select a set of chromosome pairs from the population to mate, such that the set size is equal to 25% of the population size. Note that it is possible for a chromosome to mate more than once in a generation cycle. For each chromosome pair W_1 and W_2 selected to mate, we select a cross-over index in the chromosomes to perform the mating. We randomly select the cross-over index from the set $\{1, 2, ..., \mathcal{W}\}$, where $\mathcal{W} = |W_1| = |W_2|$, and each index is equally likely to be selected. The mating operation swaps the elements in W_1 and W_2 that are located at indices greater than or equal to the cross-over index, and all other elements remain the same. Figure 4.6 shows an example of a mating operation where the cross-over index equals 3. The orange and green elements in the figure are associated with the original mating vectors W_1 and W_2, respectively, and we can see the result of the mating operation.

4.4.2.3 Population Growth

During the population growth phase, we add new chromosomes to the population until the population size exceeds a minimum population size and the population size has increased by at least 5%.

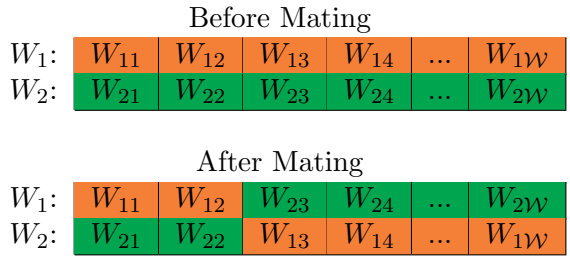


Figure 4.6: Example mating operation on weight vectors W_1 and W_2 with cross-over index equal to 3; the orange and green colored cells are the values of W_1 and W_2, respectively, before the mating operation.

There are four types of new chromosomes that we add to the population, and these types are a function of the initial seed chromosome, the best chromosome we have evolved since the zero generation, and the worst chromosome from the previous generation, which we denote by W_S, W_B, and W_W, respectively.

The first type is simply a copy of W_B, and similarly, the second type is a copy W_W. The third type is the chromosome produce by $\texttt{average}(W_S, W_B)$, where `average` is defined in Equation 4.28. Likewise, the fourth type is the chromosome produce by $\texttt{average}(W_S, W_W)$. When adding new chromosomes, we cycle over these four types of new chromosomes to create diversity in the population.

$$\texttt{average}\,(W_1, W_2) = \frac{1}{2}\big(W_1 + W_2\big) \tag{4.28}$$

4.4.2.4 Population Fitness Evaluation

The fitness evaluation of the population set allows us to create a rank ordering of the chromosomes in the set. Recall that the chromosome W represents a $C \times 7$ classifier weight vector to use in the classification decision rule, where C is the number of classes and 7 is the number of features. We use penalty functions with an associated penalty weight value to calculate the fitness of a chromosome. We have four penalty functions to evaluate each chromosome W in our population set using the training set $\mathcal{T}$. Recall that a training instance $(x, y) \in \mathcal{T}$ consists of both a feature vector x and class label y.

The goal of the genetic algorithm is to find the chromosome W that minimizes the weighted sum total of each penalty function value for that particular W. To guide the genetic algorithm search, we use four penalty functions, ϕ_p, for $p = 1, 2, 3, 4$, to calculate the fitness of a chromosome. We let λ_p be the penalty weight value associated with the penalty function ϕ_p, for $p = 1, 2, 3, 4$. We evaluate each chromosome W over the training set $\mathcal{T}$ using the fitness equation, which is weighted

sum of the penalty functions, as shown in Equation 4.29.

$$\texttt{fitness}(W) = \sum_{p=1}^{4} \lambda_p \phi_p(W, \mathcal{T}) \tag{4.29}$$

Penalty function ϕ_1 counts the number of misclassifications using the classifier weight vector W on the training set $\mathcal{T}$, as shown in Equation 4.30, where $\delta[\cdot]$ is an indicator function that returns a value of 1 when the argument is true, otherwise the function returns a value of 0, and W_i is the sub-vector of W that is associated with the i^{th} class label. By minimizing this penalty function, we intend to improve the accuracy of the classifier.

$$\phi_1(W, \mathcal{T}) = \sum_{(x,y)\in\mathcal{T}} \delta \left[y \neq \arg \min_{i\in[1,C]} W_i^T \cdot x \right] \tag{4.30}$$

Penalty function ϕ_2 produces the summation of error-magnitudes when a misclassification occurs using the classifier weight vector W on the training set $\mathcal{T}$, as shown in Equation 4.31, where the quantity $W_y^T \cdot x$ is the score value of the true class label associated with the instance x, and the quantity $W_i^T \cdot x$ is the score value of the i^{th} class label. Each error-magnitude inside the summation is a nonnegative value and has a non-zero value when a misclassification occurs. By minimizing this penalty function, we intend to find weight vectors that "barely" selects an incorrect class label for an instance when a misclassification happens, i.e. the minimum score value that is associated with an incorrect class label is slightly smaller than the score value for the correct class label.

$$\phi_2(W, \mathcal{T}) = \sum_{(x,y)\in\mathcal{T}} \underbrace{\left(W_y^T \cdot x\right) - \left(\min_{i\in[1,C]} W_i^T \cdot x \right)}_{\text{error-magnitude}} \tag{4.31}$$

Penalty function ϕ_3 is the summation of L_1-norm of the weight vector W_i for all class labels. as shown in Equation 4.32. The utility of minimizing this penalty function can be seen in conjunction with ϕ_4 and will be discussed below.

$$\phi_3(W, \mathcal{T}) \quad = \quad \sum_{i=1}^{C} \sum_{j=1}^{7} |W_{i,j}| \tag{4.32}$$

Penalty function ϕ_4 is the summation of the inverse squared-L_2-norm for all classes, as shown in Equation 4.33. By minimizing this penalty function, we are maximizing the L_2-norm of each weight vector. This maximization will prevent an all-zero weight vector, which would otherwise cause the

classifier to always select the class label associated with that all-zero weight vector.

$$\phi_4(W, \mathcal{T}) = \sum_{i=1}^{C} \left\{ \sum_{j=1}^{7} W_{i,j}^2 \right\}^{-1} \tag{4.33}$$

Notice that in ϕ_3 we are minimizing the L_1-norm of the weight vectors, while in ϕ_4 we are maximizing the L_2-norm of the same vectors. These two actions guide the genetic algorithm search to find weight vectors that place most of its weight on the most discriminate features. This effect should, also, help improve the accuracy of the classifier.

In our preliminary work, which is discussed in Section 4.5.1, we set $\lambda_1 = 10$ and $\lambda_2 = \lambda_3 = \lambda_4 = 1$. These values of penalty weights intuitively made sense because this configuration places the most importance on ϕ_1, which counts the number of misclassifications on the training set. However, the penalty functions are not constrained to return a value in the same range, therefore we must determine the appropriate scaling. Also, we do not know which penalty functions are the best at guiding the genetic algorithm search to find the most discriminate classifier weight vectors that produces a good classifier. For these reasons, we chose to investigate a better choice of appropriate values to use for each λ_p for the four penalty functions.

To determine the appropriate penalty weight values, we performed a coarse search over some possible weight value combinations. To facilitate this test, we created a collection of noisy, digitally modulated signals, where we vary the SNR between 0 dB and 19 dB. At each SNR value, and for each modulation, we created 50 received signal instances, where each instance contains 1000 symbols that are randomly selected from the constellation set of the modulation with equal probability and corrupted by additive Gaussian noise, and where the 12 modulations in our set are: BPSK, QPSK, 8-PSK, 16-PSK, 8-QAM, 16-QAM, 32-QAM, 64-QAM, 4-PAM, 8-PAM, 16-PAM, 32-PAM.

We ran a number of tests to understand how our choice of penalty weight values affects the classification performance. For each test, we selected the penalty weight values, trained, and tested our classifier using 5-fold cross-validation at each SNR value. We measured and recorded the average classification accuracy over the SNR values for the classifier using the penalty weight values. We varied the values of the penalty weights to determine the best parameterization on this limited search. Specifically, we tried values of λ_p in the set $\{0, 1, 10, 10^2, 10^3, 10^4, 10^5, 10^6\}$ for each of the four penalty functions. We have four penalty weights that initially gives us 4096 test scenarios. However, we removed the unnecessary case where $\lambda_p = 0$ for all p, and also all of the repetitive test scenarios. For example, the test scenario where $\lambda_p = 1$ for all p and the test scenario where $\lambda_p = 10$ for all p are duplicate tests since the two scenarios are scalar multiples of each other. By removing these unnecessary tests, we ultimately had 1695 test scenarios to evaluate.

In Figure 4.7, we visualize the average classification accuracy measured for the different penalty weight value parameterizations as a heat map. The heat map forms a 64-by-64 grid, where we

index each row and column position by two letters. The grid position label letters A through H map to the penalty weight values of 0, 1, 10, 10^2, 10^3, 10^4, 10^5, and 10^6, respectively. The rows of the heat map denote the penalty weight values for λ_1 and λ_2, and the columns of the heat map denote the penalty weight values for λ_3 and λ_4. For example, the row index 'BA' and column index 'BD' indicates the penalty weight values: $\lambda_1 = 1$, $\lambda_2 = 0$, $\lambda_3 = 1$, and $\lambda_4 = 10^3$. The accuracy is displayed as a color, where the closer the accuracy is to 1, then the color is darker in red-intensity and the closer the accuracy is to 0, then the color is darker in blue-intensity. As the accuracy varies between 0-1, then the color traverses the color spectrum from blue to red. Note that we fill in the locations in the heat map which represent a duplicate test with the value calculated by the test experimented. In order words, the grid location indexed by 'AAAA' has the same value and color as the grid location indexed by 'BBBB', which is due to the fact that those grid locations represent duplicate test scenarios.

Clearly, we tested a small subset of possible penalty weight possible parameterizations in $\mathbb{R}^4_+$. However, from the heat map in Figure 4.7, we see that this coarse sampling of the penalty weight values produces a fairly smooth distribution of accuracy values in that the search space. A small number of penalty weights resulted in a poor classifier in terms of correct classification accuracy, as indicated by the locations in the heat map with a color closer to blue. However, the heat map also shows that for the most part any selection of penalty weights yield a good classifier.

We can rank-order the classifier accuracies for each penalty weight value sets evaluated. In Table 4.2, we list the top-10 performing penalty weight parameterizations with the highest classifier accuracy. Note that the "H-IDX" column in the table refers to the heat map grid indexing used in Figure 4.7, which facilitates cross-referencing. We see that the top-10 parameterizations all perform over 93.6%. We notice that there does not appear to be a clear pattern to the penalty weight values of these top ranking parameterizations. The parameterization previously used with $\lambda_1 = 10$ and $\lambda_2 = \lambda_3 = \lambda_4 = 1$ experimentally resulted in an accuracy of 89.742%, which ranks 936 out of the 1695 parameterizations tested.

We acknowledge that the search performed and reported in this section is limited in scope. Based on these results, we conclude that the variation in classification accuracy for different penalty weight values is small, and it is therefore not necessary to do a more exhaustive search of penalty weight values to construct a good classifier.

4.4.2.5 Population Reduction

Population reduction is an attempt to incorporate the survival of the fittest idea in the search. After the fitness evaluation of the population set, we determine which chromosomes to remove. We protect the top 10% of the population to avoid decimating the best chromosomes, and we also protect the bottom 10% to prevent prematurely reaching a local optimum in the search. We

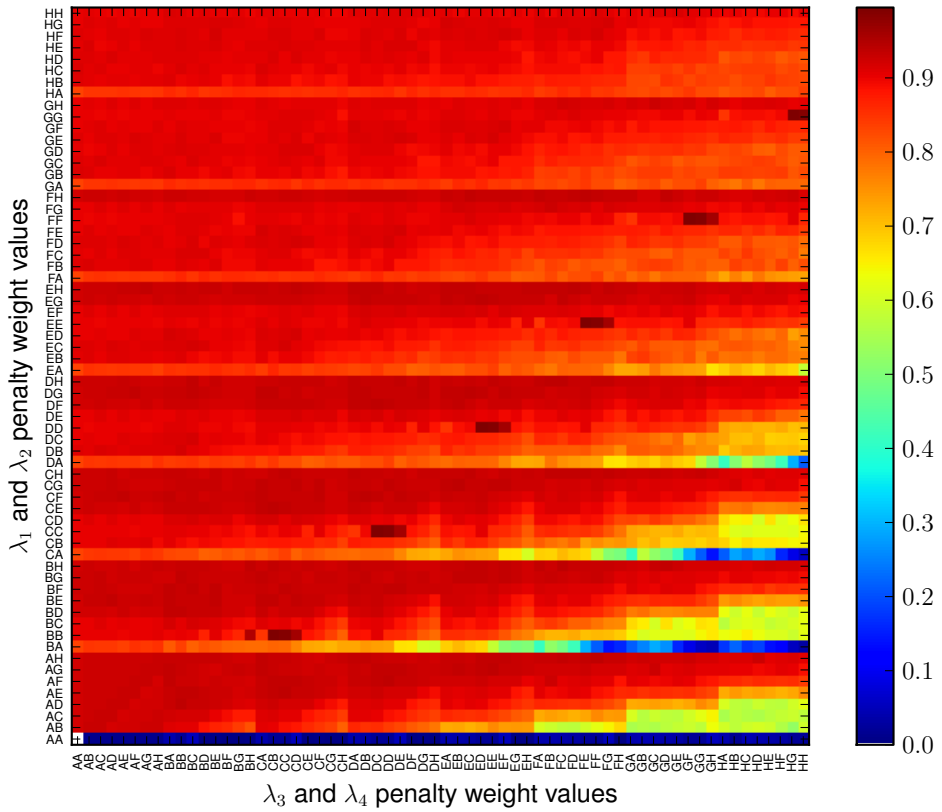


Figure 4.7: Heat map visualization of the average classification accuracy for the modulation classifier using different parameter values for the penalty weights λ_p for $p = 1, 2, 3, 4$ which guides the genetic algorithm search in the training process of the classifier. The labels A through H correspond to the penalty weight of 0, 1, 10, 10^2, 10^3, 10^4, 10^5, and 10^6, respectively.

Rank	H-IDX	λ_1	λ_2	λ_3	λ_4	Accuracy
1	BBCB	1	1	10	1	0.99500
2	BBCC	1	1	10	10	0.99083
3	BBCD	1	1	10	100	0.96271
4	BBBH	1	1	1	10^6	0.94900
5	CGEB	10	10^5	10^3	1	0.93958
6	DHFB	100	10^6	10^4	1	0.93850
7	BHFC	1	10^6	10^4	10	0.93808
8	CECB	10	10^3	10	1	0.93783
9	CFDB	10	10^4	100	1	0.93775
10	ADBB	0	100	1	1	0.93675

Table 4.2: Evaluation set top-10 performing penalty weight value parameterizations with respect to classifier accuracy.

randomly remove chromosomes from the middle 80% to reduce the size of the population set by 5%, and if needed we continue to delete chromosomes until the population size is smaller than a maximum population size.

4.5 Experiments

We designed a series of experiments to determine the quality of our modulation classification algorithm. Specifically, we first evaluate the performance of our classifier with test instances that operate in varying real-world conditions, such as different signal-to-noise ratio values, and receiver inaccuracies. The second evaluation examines the performance of our classification algorithm with respect to the performance of other classification algorithms reported in the literature. The final evaluation directly compares the classification accuracy of an ensemble SVM using our feature set with other classifiers using two popular choices in feature sets. We shall show that the approach used in our classification algorithm and our novel feature set both perform well at the task of modulation classification.

4.5.1 First Evaluation – Our Algorithm Only

Since a real-world environment is always changing, a spectrum sensing application must able to handle many different environmental conditions. In this first evaluation, we designed a set of experiments to determine the classifier accuracy operating in varying real-world conditions, such as different SNR values and receiver inaccuracies. We also wish to know the minimum number of classifiers needed to achieve a desirable performance in our overall spectrum sensing system. Many studies in the literature test modulation classifiers by varying the SNR values. However, to report on the performance, typically one classifier is trained at a particular SNR and that classifier is tested at that same SNR. This process is repeated for ν SNR values to produce a plot with ν points. This method is a reasonable way to measure the performance of our classifier, but to implement this approach into our spectrum sensing system we would need to create each of these ν classifiers to handle each ν SNR values. As ν gets large, the computational requirements on our system will grow large as well. We wish to determine if it is possible to reduce the computational requirements of our system by training fewer classifiers over a range of SNR values instead of a single classifier for each SNR value.

We let our modulation constellation template set $\mathcal{T}$ contain the following types of modulations: BPSK, QPSK, 8-PSK, 16-PSK, 4-PAM, 8-PAM, 16-PAM, 32-PAM, 8-QAM, 16-QAM, 32-QAM, 64-QAM. For reference, the constellation plots for each of these templates can be found in Appendix B. We created seven different channel condition test scenarios parameterized by the message length,

Scenario	N	K	τ	f_o
S_1	1000	10	0	0%
S_2	1000	5	0	0%
S_3	1000	10	2	0%
S_4	1000	5	1	0%
S_5	1000	10	0	5%
S_6	1000	10	0	10%
S_7	1000	10	0	20%

Table 4.3: Scenario test conditions used to generate experiments to evaluate the modulation classifier.

number of samples per symbol, symbol boundary offset, and carrier frequency offset, and the scenario is expressed as a tuple, (N, K, τ, f_o).

Table 4.3 enumerates the simulation scenarios that we tested. Let $\mathcal{S} = \{S_1, S_2, ..., S_7\}$ be the set of seven test scenarios. Test scenarios $S1$ and $S2$ are the ideal cases that differ in the number of samples per symbol. Test scenarios $S3$ and $S4$ test the misalignment of the symbol boundary. We chose a misalignment of 20%. Test scenarios $S5$, $S6$, and $S7$ test the frequency offset of 5%, 10%, and 20%, respectively. Recall from our derivation that we assumed ideal conditions for our receiver. These simulation scenarios will help us determine the robustness of our classifier to non-ideal conditions.

For each test scenario in Table 4.3, we want to determine the classification accuracy of our classifier at a number of SNR values, where this SNR set is denoted by Σ as defined in Equation 4.34.

$$\Sigma = \left\{ \ \sigma \text{ dB} \ : \ 0 \leq \sigma < 20, \ \sigma \in \mathbb{Z} \right\} \tag{4.34}$$

For each scenario, modulation type in $\mathcal{T}$, and SNR value in Σ, we created 50 signals.

The basic procedure for our experiments on a classifier is the following: select an experiment signal set, run 5-fold cross-validation, and produce a confusion matrix, which is outlined in Algorithm 2 with the procedure SIMULATE-SCENARIO.

The inputs to SIMULATE-SCENARIO are the scenario tuple S, the stride set R, and the SNR set Σ, where $S = (N, K, \tau, f_o)$. We define the scenario data set, $\mathcal{D}$, as the set that contains all instances in scenario S created with each possible SNR value. The stride set R creates a partition of the scenario data set using a stride value ρ, as defined in Equation 4.35 where $\rho \in [1, 2, 4, 10, 20]$ and $\texttt{SNR}(\cdot)$ is a function that returns the SNR value of the argument.

$$R = \left\{ \ \mathcal{D}_r \ : \ \mathcal{D}_r = \left\{ \ d \ : \ d \in \mathcal{D}, \ \rho(r) \leq \texttt{SNR}(d) < \rho(r+1) \right\}, \ r = 0, ..., \frac{|\Sigma|}{\rho} \ \right\}. \tag{4.35}$$

Algorithm 2 Procedure to simulate and test a particular scenario.

```
SIMULATE-SCENARIO( S, R, Σ )
    Input   ▷ S   : scenario tuple
    Input   ▷ R   : stride set
    Input   ▷ Σ   : SNR set
    Output  ▷ C   : confusion matrices
1   C ← zero matrix
2   for D_r ∈ R
3       do TRAIN-SET,TEST-SET ← CREATE-SETS( R, r )
4          CLASSIFIER ← TRAIN( TRAIN-SET )
5          PERFORMANCE ← TEST( CLASSIFIER,TEST-SET)
6          C ← UPDATE-CONFUSION-MATRIX( C, PERFORMANCE )
7   return C
```

The stride set that we used contains five values: 1, 2, 4, 10, and 20. The stride value determines a partitioning of the set S to create a number of training sets, and this partition will be the method used to determine the number of classifiers to create in our final system.

For example, a stride value of 1 yields 20 subsets of $\mathcal{D}$, one subset for each SNR value in Σ. For another example, a stride value of 4 yields 5 subsets of $\mathcal{D}$, where one subset contains all instances in $\mathcal{D}$ with SNR in $[0 \text{ dB}, 4 \text{ dB})$, and one subset contains all instances in $\mathcal{D}$ with SNR in $[4 \text{ dB}, 8 \text{ dB})$, etc.

The output of SIMULATE-SCENARIO is a set of confusion matrices, which compactly describes the classifier's performance. We have 12 modulation types in the template set $\mathcal{T}$, so our confusion matrix is a 12 by 12 matrix. The ij^{th} element of the confusion matrix indicates the number of instances that were classified as the j^{th} modulation type when the i^{th} modulation type is the correct label. A diagonal element of the confusion matrix expresses the number of instances correctly classified by the classifier, which implies that the trace of the confusion matrix is the total number of correctly classified test instances. The accuracy of the classifier is defined as the trace of the confusion matrix divided by the total number of test instances. Our algorithm produces a set of confusion matrices, which has one matrix associated with each subset of the stride set R.

In line 1 of SIMULATE-SCENARIO, we initialize all confusion matrices to the zero matrix. In line 2 of SIMULATE-SCENARIO, we start a for-loop over the stride set. In lines 3-6 of SIMULATE-SCENARIO, we perform a 5-fold cross-validation to train, test, and update the confusion matrices of our classifier using the previously described supervised training process. In 5-fold cross-validation, we create five subsets to use as training and testing sets from the set $\mathcal{D}_r$. We train and test our classifier five times, where on the k^{th} fold we use the k^{th} subset of $\mathcal{D}_r$ as the training set, and the remaining four other sets will represent the testing set. The classifier is trained on the training set, and the

	2-PSK	4-PSK	8-PSK	16-PSK	4-PAM	8-PAM	16-PAM	32-PAM	8-QAM	16-QAM	32-QAM	64-QAM
2-PSK	1	1	0	0	0	0	0	0	0	0	1	0
4-PSK	0	1	0	0	1	0	1	0	0	0	1	0
8-PSK	0	1	1	0	1	0	0	0	0	0	0	0
16-PSK	0	0	0	1	0	0	0	0	0	0	0	0
4-PAM	0	0	0	1	1	0	0	0	0	1	1	0
8-PAM	0	1	0	0	1	1	0	0	0	1	0	0
16-PAM	1	1	0	0	0	0	1	0	0	0	1	0
32-PAM	0	0	0	1	0	0	0	1	0	0	0	0
8-QAM	0	0	0	0	0	0	0	0	1	0	0	0
16-QAM	0	0	0	1	0	0	1	0	0	1	0	0
32-QAM	0	0	0	0	0	0	0	0	0	1	1	0
64-QAM	0	0	0	0	0	0	1	0	0	1	1	1

(a) Full confusion matrix

	PSK	PAM	QAM
PSK	6	3	2
PAM	5	5	4
QAM	1	2	7

(b) Condensed confusion matrix

Table 4.4: Example of creating a condensed confusion matrix from the full confusion matrix.

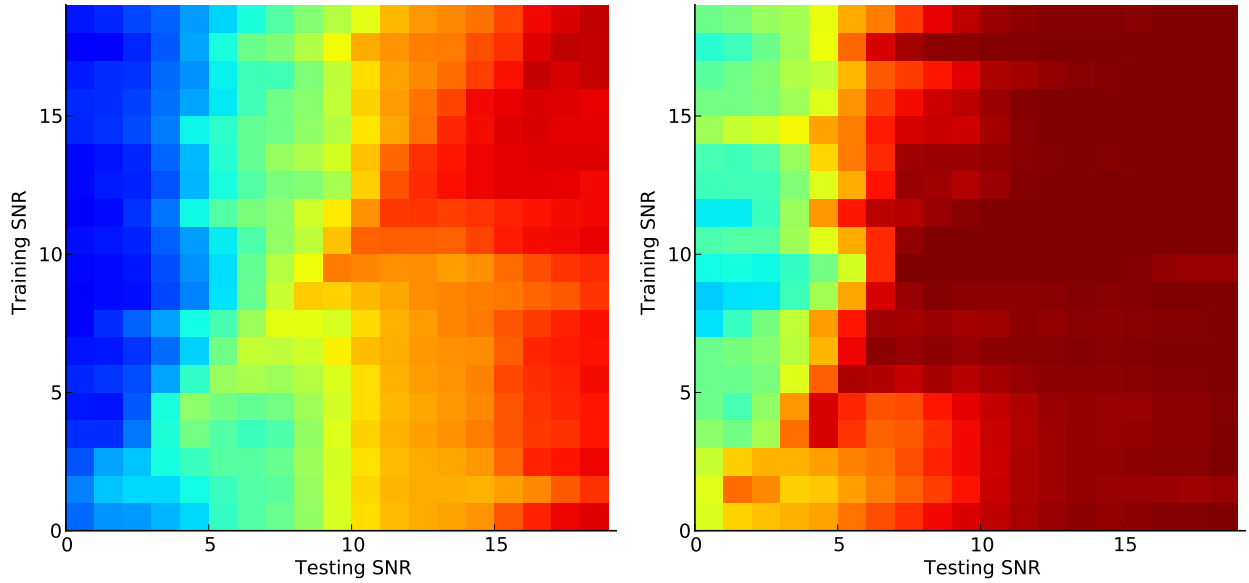
performance is calculated using the testing set. Note that this process trains and tests instances only in $\mathcal{D}_r$. We are also interest in the classifier performance when trained on instances in $\mathcal{D}_r$, but tested on instances in $\mathcal{D}_{r'}$, where $r' \neq r$, and so we add the set $\mathcal{D} \setminus \mathcal{D}_r$ to our testing set.

4.5.1.1 Evaluation Results

We evaluated four implementations of the weight vector classifier using the genetic algorithm process. Classifiers M_1 and M_2 use individual weight vectors for all modulation types after completing 1000, and 5000 generations in the genetic algorithm search, respectively. Classifiers M_3 and M_4 use a single common weight vector for all modulation types after completing 1000, and 5000 generations in the genetic algorithm search, respectively.

Our experiments produce a large set of confusion matrices, from which we create a set of condensed confusion matrices based on the modulation family. Recall that each modulation type in $\mathcal{T}$ consists of a modulation family and order, where the order expresses the number of bits per symbol. For a confusion matrix C, we create a condensed confusion matrix by grouping together the statistics in C by modulation family. The resulting condensed confusion matrix is a 3 by 3 matrix, since we consider PAM, QAM and PSK modulations families. The condensed confusion matrix allows us to analyze how well our classifier performs at the simpler task of determining only the modulation family. Table 4.4 shows a sample confusion matrix and the resulting condensed confusion matrix.

We seek to evaluate the performance of a trained classifier across a wide range of SNR values to determine the degree to which the classifier's performance is robust. For this reason, we train



(a) Classifier M_1 Modulation type

(b) Classifier M_1 Modulation family

Figure 4.8: Heat map performance of classifying the modulation type, and modulation family, for classifier M_1 with the stride equal to 1 for test scenario S_1.

a classifier at a particular SNR value, and then we evaluate its performance over the complete set of SNR values, Σ. For an individual pair of training and testing SNR values, the classifier performance is captured by a confusion matrix, where our evaluation procedure considers 400 SNR value pairs. We use a heat map to better visualize the classifier performance by the correct classification accuracy, where the horizontal axis is the testing SNR value, the vertical axis is the training SNR value, and the color indicates the accuracy. The accuracy is represented using colors that span the color spectrum from red, for 100% accuracy, to blue, for 0% accuracy.

Figure 4.8 illustrates the heat map performance for classifier M_1 using a stride of 1 in test scenario S_1, where the left subfigure shows the accuracy of recognizing the modulation type, and the right subfigure shows the accuracy of recognizing the modulation family only. We see that the modulation-family heat map qualitatively has more intensity throughout the plot compared to the modulation-type heat map, which implies that the classifier performs better at the simpler task of determining only the modulation family. We observed this same effect for all of our classifiers, for each stride value, and every test scenario.

A heat map helps us visualize our data and qualitatively compare results, but it does not directly help us quantify exactly how much better one classifier is than another. To measure this value, we calculate a weighted average error (WAE) to help us quantitatively compare classifiers, as shown in Equation 4.36, where i is the testing SNR value, j is the training SNR value, w_{ij} is the error weight

when testing at SNR i and trained at SNR j, and `heat` is a function that returns the accuracy when tested at SNR i and trained at SNR j.

$$WAE = \sum_{i=0}^{19} \sum_{j=0}^{19} w_{ij} \underbrace{(1 - \texttt{heat}[i,j])}_{\text{error}} \tag{4.36}$$

We chose to partition the SNR value range into four equal sized sets, which corresponds to a coarse quantization of very low SNR, low SNR, high SNR, and very high SNR value regimes. Using this partitioning, we define the distance between tested SNR value i and trained SNR value j as the L_1 distance between the partition sets that include values i and j, as shown in Equation 4.37.

$$\texttt{dist}[i,j] = \left| \left\lfloor \frac{i}{5} \right\rfloor - \left\lfloor \frac{j}{5} \right\rfloor \right| \tag{4.37}$$

We wish to apply more weight to the errors that occur when the tested SNR value is near the trained SNR value. For example, we expect the classifier to perform better when trained and tested in the very low SNR regime compared to the performance when trained in the very high SNR regime and tested in the very low SNR regime. With this consideration in mind, the values of the error weight are expressed in Equation 4.38.

$$w_{ij} = \begin{cases} \frac{5}{48} & \text{if} \quad \texttt{dist}[i,j] = 0 \\ \frac{3}{48} & \text{if} \quad \texttt{dist}[i,j] = 1 \\ \frac{2}{48} & \text{if} \quad \texttt{dist}[i,j] = 2 \\ \frac{1}{48} & \text{if} \quad \texttt{dist}[i,j] = 3 \end{cases} \tag{4.38}$$

We can evaluate the effect of the stride value in the training procedure on the performance of a classifier using the WAE. Table 4.5 lists the WAE for both the modulation-type classification and the modulation-family classification of classifiers M_1 through M_4 in test scenario S_1 for each stride value. Recall that the stride value is inversely proportional to the number of classifiers trained to span the range of possible SNR values. We see that, in most cases, the WAE decreases as we increase the stride value, and for all classifiers, the lowest WAE corresponded to a stride value of 20. This result is desirable because it implies that we can train a single classifier using training instances from all possible SNR values. We observed a similar relationship between stride value and the WAE for the other test scenarios, which, again, implies that a single classifier trained from all possible SNR values perform better than creating a set of classifiers trained at individual SNR values.

We can determine which of the four classifiers performs the best by ranking their WAE for each of the seven test scenarios. Table 4.6 provides these rankings for both the modulation-type classification and the modulation-family classification using the weighted average error with a stride value of

		Stride 1	Stride 2	Stride 4	Stride 10	Stride 20
Modulation Type Classification	M_1	0.415	0.412	0.399	0.376	0.334
	M_2	0.431	0.428	0.398	0.378	0.320
	M_3	0.395	0.395	0.393	0.366	0.311
	M_4	0.412	0.404	0.402	0.367	0.316
Modulation Only Classification	M_1	0.153	0.162	0.139	0.142	0.085
	M_2	0.171	0.173	0.151	0.147	0.078
	M_3	0.190	0.186	0.173	0.152	0.139
	M_4	0.211	0.197	0.184	0.151	0.137

Table 4.5: Weighted average error for modulation type, and modulation family classification by classifiers M_1, M_2, M_3, and M_4, in test scenario S_1.

Test	Classifier			
Scenario	M_1	M_2	M_3	M_4
S_1	4	3	1	2
S_2	4	3	2	1
S_3	4	2	3	1
S_4	4	1	3	2
S_5	4	3	1	2
S_6	4	3	2	1
S_7	4	3	2	1

(a) Classifier Modulation-Type Ranking

Test	Classifier			
Scenario	M_1	M_2	M_3	M_4
S_1	2	1	4	3
S_2	2	1	3	4
S_3	3	1	4	2
S_4	2	1	4	3
S_5	2	1	3	4
S_6	2	1	3	4
S_7	2	1	3	4

(b) Classifier Modulation-Only Ranking

Table 4.6: Classifier ranking by lowest weighted average error for modulation type, and modulation family classification for all 7 test scenarios.

20 for each test scenario. In Table 4.6a, for the modulation-type classification, the classifier M_4 received the best ranking in 4 of the 7 test scenarios. In Table 4.6b, for the modulation-family classification, the classifier M_2 is clearly the best choice, since it had the best ranking in each test scenario. The rankings in Table 4.6 suggests that the classifier M_2 is the best classifier to use, which matches our expectations, because it is the best at modulation-family classification, but also does reasonable well at the modulation-type classification. Recall that classifier M_2 uses an individual weight vector for each modulation template, and the genetic algorithm searched for 5000 generation cycles. Intuitively, we expect that using an individual weight vector for each modulation template would perform better than using a common weight vector among all templates. Also, we expect that allowing the genetic algorithm to iterate for more cycles would produce better results than using fewer cycles. The results match our expectations that classifier M_2 performs the best out of our four classifiers.

Finally, we evaluate our classifiers' performance across different test scenarios. Recall that the

seven test scenarios exercise our classifier in the presence of varying channel conditions and receiver inaccuracies, such as timing and frequency offsets. After analyzing the WAE for classifier M_2 using the stride value of 20, we found that our classifier performed fairly consistently over all seven test scenarios. The WAE for each test scenario, with one exception, is near 0.33 for the modulation-type classification, and near 0.08 for the modulation-family classification. For test scenario S_4, the weighted average error is 0.52 and 0.14 for the modulation-type classification and modulation-family classification, respectively. We expected to see a more variation in the results across the test scenarios, so this result is encouraging, and we believe that our classifier approach is robust to varying channel conditions in the real-world.

In summary, we have four conclusions about our classifier approach. First, we found that our classifier approach does a better job at classifying the modulation-family compared to classifying the modulation-type. Second, we found that the weighted average error of our classifier decreases in value as we increase the stride value. Third, we found that using individual weight vectors for each modulation template worked that best, and also using more generation cycles used in the genetic algorithm yields better weight vectors. Fourth, we found that our classifier performed equally well in non-ideal test scenarios.

4.5.2 Second Evaluation – Examination of our Algorithm with Respect to the Literature

In this second evaluation, we wish to directly compare the correct classification accuracy of our classification algorithm to the performances reported in the literature. This comparison is difficult to achieve for several reasons. First, there is not a standard set of test and training instances to make comparisons, which is a concern mentioned in [128]. Second, the accuracy of the classifier is highly dependent on the modulation types included in the test set.

An additional challenge is that the feature set that we developed for our classifier operates on digital modulations that can be described by a constellation. In order to fairly compare our classifier, we must find in the literature classifier results for classifying modulation types that are digital *and* can be represented by a constellation. For example, we cannot compare with results that classify analog modulations, such as AM or FM [118, 119, 127, 147], nor non-constellation-based digital modulations, such FSK [121, 146]. However, for this comparison, we found six classifiers in the literature that report the correct classification accuracy on a modulation class set that we could apply our feature extraction.

For this evaluation, we ran two experiments. In a first experiment, we performed a direct comparison of our CBDM classification algorithm to six classifiers from the literature and found that our algorithm consistently outperformed the results reported. For example, at 0 dB SNR, the correct modulation classification accuracy of our algorithm averaged 23.8 percentage points better than

the results in the literature, where the most dramatic increase was from 37.5% to 98.3%. A second experiment showed that our algorithm had a slight degradation in accuracy when using a larger class label set, but yet still maintained comparable performance to the first experiment. This outcome is made impressive by the fact that the class label set contained a significantly larger number of modulations. This scalability of a modulation classification algorithm to a larger class label set has not been previously discussed in the literature.

4.5.2.1 Classification Learning Approach for this Evaluation

For this evaluation, we are creating six new classifiers in order to directly compare to the performance of the six reference classifiers reported in the literature. Recall, in our CBDM classification algorithm, we repeat the following process for each of the modulations in our class label set: we cluster our received signal to the modulation constellation set, and then we extract features that are used in conjunction with a weight vector to calculate a score value. The decision rule classifies the signal to the modulation class label that produces the lowest score value. Thus, the overall approach to learn these classifiers is essentially the same of the approach discussed in previous sections of this chapter. However, we vary the class label set in order to match the set used in reference classifier from the literature. Figure 4.9 illustrates an example change in implementation of our algorithm to accommodate a 3-class label classification between QPSK, 8-PSK, and 16-QAM.

For each class label set comparison, we let W_k represent the trained weight vector associated with $\omega_k \in \Omega$. We let $W = \{W_1, ..., W_{|\Omega|}\}$ be the set of the weight vectors for each of the class labels in Ω. In training, we use the penalty function that is the addition of two error terms summed over each training instance in the training set. The first term is a misclassification term that is 1 when the classifier decision label using W does not match the training instance label, otherwise the term is 0. The second term is an error magnitude term that is the difference between the score value for the training instance label and the score value of the classifier decision label. Note that this error is positive and nonzero when a misclassification occurs. We express the penalty function, $\mathcal{P}$, in Equation 4.39, where $\delta[\cdot]$ is an indicator function, ω_d^* is the class label predicted by the classifier on Y_d using W, and `score` returns the score value generated by the argument class label in the classifier on Y_d using W.

$$\mathcal{P} = \sum_{(Y_d, \ell_d) \in T} \underbrace{\delta\left[\omega_d^* \neq \ell_d\right]}_{\text{misclassification}} + \underbrace{\texttt{score}(\ell_d, Y_d, W) - \texttt{score}(\omega_d^*, Y_d, W)}_{\text{error magnitude}} \tag{4.39}$$

Notice in Equation 4.39, we attempt to minimize not only the number of misclassifications, but also the magnitude of the error produced by the classifier when a misclassification does occur. This

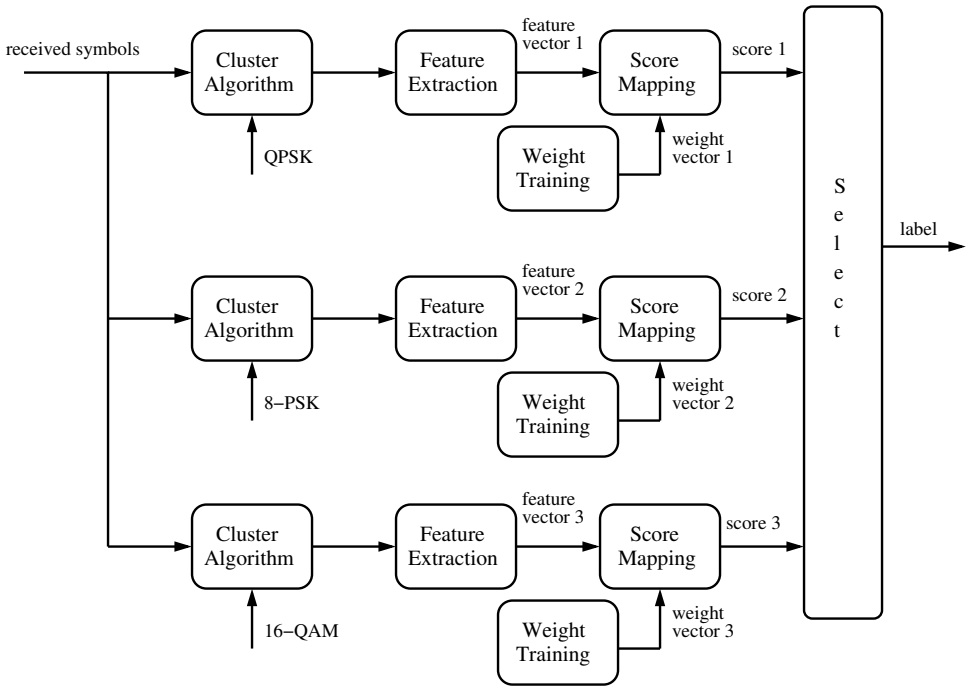


Figure 4.9: Example implementation of the classification algorithm on the class label set {QPSK, 8-PSK, 16-QAM}

corresponds to setting the penalty weight values $\lambda_1 = \lambda_2 = 1$ and $\lambda_3 = \lambda_4 = 0$.

4.5.2.2 Experiments for this Evaluation

In the literature there are six examples of automatic modulation classifiers where the class label set consists only of modulations that can be represented by a constellation set [128, 129, 132]. In [128], fuzzy clustering was used to generate a probability density function of the complex samples, and the resulting density function was used to classify the signal. In [129], the Wavelet transform is used to de-noise the samples, followed by subtractive clustering to determine the number of peaks in their constellation. In [132], a genetic algorithm was used to cluster the received samples in the complex plane, and then a hierarchical clustering algorithm further merged the clusters to feed into a Hamming neural network. These classifiers give us a set of performance results that we can compare to our CBDM classification algorithm. Table 4.7 lists the modulation class label set of each of these six classifiers. Next, we describe our signal generation procedure, the comparison measure, and the two experiments performed to evaluate our algorithm.

We created a collection of noisy, digitally modulated signals, where we vary the SNR between 0 dB and 20 dB. At each SNR value, and for each modulation in Table 4.7, we created 1000 received signal instances, where each instance contains 1000 symbols that are randomly selected from the

		Class Label Set	Reference
Ω_1	=	{QPSK, 8-PSK, 16-QAM}	[128]
Ω_2	=	{V29, 16-QAM}	[128]
Ω_3	=	{V29 fallback, 8-PSK}	[128]
Ω_4	=	{4-QAM, 16-QAM, 32-QAM, 64-QAM}	[129]
Ω_5	=	{4-QAM, 16-QAM, 64-QAM}	[132]
Ω_6	=	{QPSK, 8-PSK, 16-PSK}	[132]

Table 4.7: Class label sets from the literature that can be used for direct comparison with our CBDM classification algorithm.

constellation set of the modulation with equal probability and corrupted by additive Gaussian noise. We perform 5-fold cross-validation on our classification algorithm for each SNR value, which produces a confusion matrix at each SNR value, and we measure the classification accuracy. The classifier performance is defined as the correct classification accuracy, which is the trace of the confusion matrix divided by the total number of test instances, averaged over the five folds.

The results in Table 4.8 evaluate how well our classification algorithm works in comparison with reported results in the literature. For each of the six reference classifiers, we trained and tested our algorithm on the same set of class labels at the SNR values specified in the appropriate paper. Each row in Table 4.8 gives the accuracy of a classifier for each SNR value, where accuracy is expressed as a percentage and the SNR is in terms of dB. The classifier name has the form of $\Omega_i - algorithm$, where Ω_i indicates the class label set and *algorithm* indicates whether the classifier is one of the reference algorithms from the literature or our CBDM algorithm.

We directly compare the performance between the reference and CBDM classifiers that share the class label set Ω_i. We see that our CBDM classification algorithm outperforms each of the classifiers in the literature across all SNR values. In some cases, we see that our classifier yields a significant improvement over the reference classifier. For example, on class label set Ω_6 at 0 dB SNR, our approach had a correct classification accuracy of 98.9% versus 37.5% from the literature.

The results in Table 4.9 show the amount of degradation in our CBDM classification algorithm as we increase the size of the class label set. We created a single classifier using a class label set that is the union of the six class label sets from Table 4.7, i.e. $\Omega_7 = \cup_{i=1}^{6} \Omega_i$. This new class label set has 8 modulation class labels, which makes the classification problem more challenging than the problem faced by each Ω_i-CBDM classifier. Note that we consider QPSK and 4-QAM to be equivalent modulations. We again perform a 5-fold cross-validation on this classifier to produce a confusion matrix on the larger class label set.

Each row in Table 4.9 again gives the accuracy of a classifier for each SNR value. The table includes

Classifier	SNR (dB) 0	5	6	8	10	12	15	20
Ω_1-Reference	85.7	91.7			96.7		99.3	100
Ω_1-CBDM	98.7	100			100		100	100
Ω_2-Reference	83	86			90		96	99
Ω_2-CBDM	99.6	100			100		100	99.9
Ω_3-Reference	93	96			98		100	100
Ω_3-CBDM	99.9	100			100		100	100
Ω_4-Reference	72	100			100			
Ω_4-CBDM	80.7	100			100			
Ω_5-Reference	40	51.7		86.7	96.7	100	100	100
Ω_5-CBDM	76.7	100		100	100	100	100	100
Ω_6-Reference	37.5	58.3	65.8	76.7	87.3	97	100	
Ω_6-CBDM	98.9	100	100	99.9	100	100	100	

Table 4.8: The performance of our CBDM classification algorithm in direct comparison with the reference classifiers from the literature.

Classifier	SNR (dB) 0	5	6	8	10	12	15	20
Ω_1-CBDM	98.7	100			100		100	100
Ω_2-CBDM	99.6	100			100		100	99.9
Ω_3-CBDM	99.9	100			100		100	100
Ω_4-CBDM	80.7	100			100			
Ω_5-CBDM	76.7	100		100	100	100	100	100
Ω_6-CBDM	98.9	100	100	99.9	100	100	100	
Average	92.41	100	100	99.97	100	100	100	99.98
Ω_7-CBDM	73.86	99.78	99.93	100	99.95	100	99.95	100

Table 4.9: Performance of our CBDM classification algorithm on a larger class label set.

the performance of the Ω_i-CBDM classifiers, for $i = 1, 2, ..., 6$, from the first experiment, the average performance of these six classifiers, and the performance of the Ω_7-CBDM classifier which operates on a larger class label set.

We see in Table 4.9 that there is not a significant degradation our CBDM classification algorithm when predicting from a larger class label set. For all SNR values except for 0 dB, the Ω_7-CBDM classifier accuracy was no more than 0.22 percentage points below the accuracy of any classifier from the first experiment or the average performance value. The largest degradation occurs at 0 dB SNR where the average performance in the first experiment was 92.4%, whereas in the Ω_7-CBDM classifier performance was 73.86%.

The result of this second experiment indicates that our CBDM classification algorithm can be trained on a large class label set and maintain an excellent classification accuracy. Recall that the reference classifiers from the literature operate on a smaller class label set and we do not know how these reference classifiers would behave with a larger class label set, such as Ω_7. Based on the results of the first experiment, we would expect our CBDM classification algorithm to remain the better approach.

4.5.3 Third Evaluation – Examination of our Feature Set with Respect to the Literature

The literature contains many attempts to use pattern recognition approaches to classify modulation. Two feature sets have become a popular choice to use in classification algorithms. A decision tree using second moment temporal statistics to differentiate between analog and digital modulations was one of the first attempts to use pattern recognition to identify modulation type [118]. The features used in that decision tree became a popular choice of standard features for many later papers [32, 119, 154, 155]. Higher-order moment and cumulant values are another example of commonly used features [122–127]. The literature contains a wealth of individual reports on feature-based modulation classification accuracy. However, one deficiency is the lack of a direct comparison between the performance of modulation classifiers using different feature sets. To the best of our knowledge, this section describes the first direct comparison of feature sets in modulation classification on the *same* data set.

In this third evaluation, we directly compare the performance of an ensemble SVM classifier to correctly classify the digital modulation of a received signal by using the temporal, cumulant, and our constellation-based feature sets. We trained and tested a classifier using these three feature sets by varying the SNR over a large range and found that the classifier using the constellation-based features outperformed the classifier using the temporal features by 5.31 percentage points, and also outperformed the classifier using the cumulant features by 13.35 percentage points.

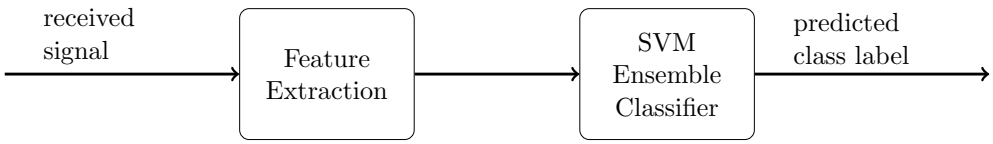


Figure 4.10: Canonical structure of the support vector machine ensemble classifier

4.5.3.1 Classification Learning Approach for this Evaluation

In this section, we utilize an architecture to perform our modulation classification that is different than the structure we previously described and used up to this point. Specifically, for our classification algorithm, we use an SVM, which is a supervised learning algorithm for binary classification problems that creates a hyperplane decision function from a training set of instances [90, 91]. An SVM makes a binary decision, however we have a multi-class problem. The typical approach is to create an ensemble classifier by decomposing the multi-class problem into a set of binary classification problems [97].

One popular approach to creating an ensemble classifier is called the one-versus-one ensemble classifier, which creates a binary classifier for every possible pair of class labels [90]. For each pair of class labels, one class is trained to a positive value and the other class is trained to a negative value. The remaining class labels are not used in training. The ensemble decision is created by taking a majority vote on the predictions made by the binary classifiers in the ensemble. In Figure 4.10, we show the canonical structure of an SVM ensemble classifier that we implement and test, where the received signal is used to create a feature vector, which is then used by the classifier to predict the class label. The three methods of feature extraction that is used in this evaluation is described next.

The first feature space is a set of cumulant and high-order statistic features. Similar features have been used extensively in prior works [122–127]. In total, this feature set contains 19 values. The first 7 features were originally proposed in [125] and measure the cumulants of the received signal and its conjugate. The remaining 12 features were originally proposed in [126] and measure the cross-cumulants between the real-valued portion and the imaginary-valued portion of the received signal. This feature space should produce good classification results because the cumulants, theoretically, can easily discriminate between the different modulations. We let CMLT-SVM refer to the classifier using feature vectors of this type.

The second feature space is a set of temporal-based features, which were originally proposed in [118]. Many papers have used these features with different types of classification algorithms with varying success [32, 119, 154, 155]. In total, this feature set contains 8 values based on temporal statistics of the received signal, such as the mean and variance of the signal amplitude, phase and frequency. This feature space should produce good classification results because information

transfers by manipulating the signal amplitude, phase and frequency based on the modulation. We let TMPL-SVM refer to the classifier using feature vectors of this type.

The third feature space is the constellation-based digital modulation features that we developed and discussed in Section 4.4.1. For each modulation in our class label set, the following process is repeated: we cluster the received signal to the modulation constellation set and then we extract feature values based on the clustering. In total, this feature set contains 7 values based on statistics after clustering the signal to each class label constellation. For a single signal, this clustering and feature extraction is repeated for each modulation in the class label set. Recall that our problem has 12 digital modulations in our class label set, which give us a total of 84 features for the signal. This feature space should produce good classification results because it is taking advantage of the known constellation structure of each modulation. We let CBDM-SVM refer to the classifier using feature vectors of this type.

4.5.3.2 Experiments for this Evaluation

Our experimentation intends to determine which of the three previously discussed feature sets provides the best classification accuracy when used in conjunction with an SVM classifier. To use an SVM classifier, we must select a kernel type and the values of the parameters specific to the kernel. We do not know beforehand which kernel parameterization will work the best with our data set, so we must try many different parameterizations.

The kernels we chose to evaluate are the linear kernel, the polynomial kernel with degree $d \in \{3, 4, 5\}$, and the radial basis function (RBF) kernel with $\gamma \in \left\{1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}\right\}$. These three SVM kernels are common selections [91]. Additionally, we must set the value of C, which is our penalty for misclassification errors in the classifier training process. We try three values of C: $C = 1$ for unit cost penalty, $C = 0.1$ which lightly penalizes misclassification errors in training, and $C = 10$ which heavily penalizes misclassification errors in training.

An SVM classifier can perform better when the feature values are limited in range [156, 157]. We utilize four normalization methods, which are applied individually to each dimension of the training set feature space. The first normalization method scales the features to fall within the range 0 to 1. The second normalization method scales the features to fall within the range -1 to 1. The third normalization method fits the features to a Gaussian distribution with a zero mean and unit variance. The fourth normalization method does not actually perform a normalization, which allows us to the test the performance of the SVM classifier on unnormalized features.

Let us define an SVM configuration to be the specification of an SVM kernel, the value of C, and the normalization method used on the feature vectors. Thus, there exist 120 possible SVM configurations. For a particular feature set, we train and test using 5-fold cross-validation on each of these SVM configurations to find the best SVM classifier for that feature set. For each fold, the

Classifier Name	SVM Kernel	Penalty C	Normalization Method	Average Accuracy
CMLT-SVM	Linear	10	[0,1]	0.78983
	Linear	10	[-1,1]	0.82725
	Linear	**10**	$\mathcal{N}(0,1)$	**0.83542**
	Linear	10	none	0.77608
TMPL-SVM	RBF ($\gamma = 1$)	10	[0,1]	0.87625
	RBF ($\gamma = 1$)	10	[-1,1]	0.90833
	Linear	**10**	$\mathcal{N}(0,1)$	**0.91583**
	RBF ($\gamma = 1$)	10	none	0.80492
CBDM-SVM	Linear	10	[0,1]	0.96683
	Linear	**10**	**[-1,1]**	**0.96892**
	Linear	0.1	$\mathcal{N}(0,1)$	0.96258
	-	-	none	-

Table 4.10: SVM configurations that produce the top perfomance values for each normalization method per classifier.

classifier is trained on the training set and the correct classification accuracy percentage on the test set is recorded. The SVM configuration performance value is defined to be the correct classification accuracy percentage averaged over the five folds.

To facilitate this experiment, we created a collection of noisy, digitally modulated signals, where we vary the SNR between 0 dB and 19 dB with 1 dB increments. We let the class label set be { BPSK, QPSK, 8-PSK, 16-PSK, 8-QAM, 16-QAM, 32-QAM, 64-QAM, 4-PAM, 8-PAM, 16-PAM, 32-PAM }. At each SNR value and for each modulation in our class label set, we created 50 received signal instances, where each instance contains 1000 symbols that are randomly selected from the constellation set of the modulation with equal probability and corrupted by additive Gaussian noise.

For each feature set, we find the SVM configuration that produces the best performance value on the created data set averaged over all SNR values. For each classifier, we report in Table 4.10 the top-performing SVM configuration for each normalization method with respect to the average classification accuracy over the SNR values, and we boldface the SVM configuration the performed the best. Note that when using the unnormalized CBDM feature set, the SVM training process could not complete the training optimization for any of the kernels. Thus, performance results do not exist for the unnormalized CBDM feature set.

We see that the best average accuracy is 83.5%, 91.5%, and 96.8% for the CMLT-SVM, TMPL-SVM, and CBDM-SVM classifiers, respectively. For each classifier the best SVM configuration used a linear kernel and a cost value $C = 10$. The CMLT-SVM and TMPL-SVM classifiers performed better using the zero mean Gaussian normalization, whereas the CBDM-SVM classifier did better with the [-1,1] normalization. We show the classification accuracy performance of these three

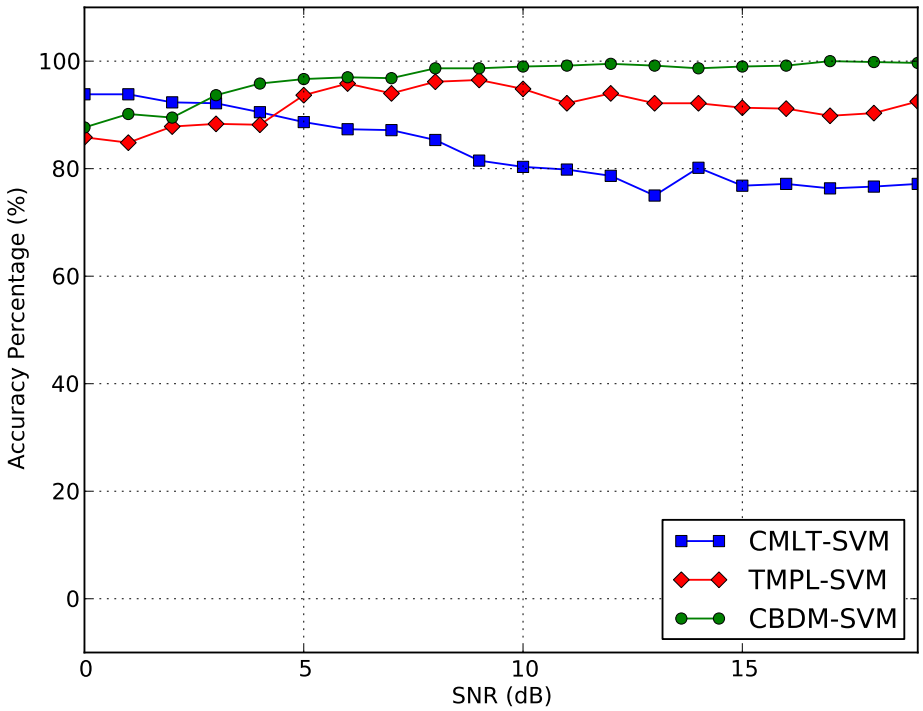


Figure 4.11: The average accuracy percentage for best SVM configuration for the CMLT-SVM, TMPL-SVM, and CBDM-SVM classifiers versus the SNR.

classifiers versus the SNR value in Figure 4.11, where the line marked by squares, diamonds, and circles correspond to the CMLT-SVM, TMPL-SVM, and CBDM-SVM classifiers, respectively.

In the figure, we see that the CBDM-SVM classifier outperforms the other two classifiers for all SNR values except at 0, 1, and 2 dB. At these low SNR values the CMLT-SVM classifier performed slightly better than the other two classifiers. However, the average difference in accuracy percentage points between the CBDM-SVM and CMLT-SVM classifiers over all SNR values is 13.35 percentage points (the average difference in accuracy percentage points between the CBDM-SVM and TMPL-SVM classifiers is 5.31). This fact along with the higher average classification accuracy of CBDM-SVM leads us to believe that the CBDM-SVM classifier, which uses the feature set that we developed, is the best at the task of classifying the digital modulations under consideration among the three classifiers compared.

4.6 Discussion and Future Improvements

We have presented in this section a novel classifier for identifying the modulation type of a received signal based on the EM algorithm, assuming ideal conditions for the receiver. In our first evaluation we saw that the classifier performs significantly better on the simpler task of identifying only

the modulation family of the received signal. Our results suggest that training a classifier with individual weight vectors for each modulation type performs as well as training a classifier with a single common weight vector.

In our second evaluation we saw that our CBDM classification algorithm performs exceptionally well at correctly classifying digitally modulated signals corrupted by noise by using features that intelligently characterize the behavior of the constellation of the true modulation. This novel feature set allows our algorithm to scale well to larger class set sizes, which is an invaluable characteristic since there exists many digital modulations. We compared our CBDM classification algorithm to six classifiers described in the literature, where we found that our algorithm increased the classification accuracy by an average of 9.8 percentage points. Additionally, we showed that our algorithm maintains its correct classification accuracy with only a slight degradation as we increase the size of the class label set. On average the performance was reduced by only 2.4 percentage points even though we had a class label set that contained a significantly larger number of modulations. In contrast, the scalability of the reference modulation classifiers to larger class label sets has not been demonstrated.

In our third evaluation, we studied and directly compared the performance of modulation classification using three different feature sets to create an ensemble SVM classifier on the same data set. To the best of our knowledge, this type of direct comparison on modulation classification performance with respect to feature set has not been previously done. Two of feature sets employed are derived from cumulant and temporal statistics, which are a common choice of features to use in modulation classification. The third feature set, which we created, is based on the structure provided by the modulation constellation set. We found that the classifier using our constellation-based feature performed the best over the SNR range between 0 dB and 19 dB compared to the other feature set classifiers: increased the average accuracy by 13.35 percentage points over the cumulant statistic classifier and by 5.31 percentage points over the temporal statistic classifier.

There are a number of directions we can take in future work to improve our modulation classification algorithm. In this chapter, we tried many modulation types for our class label set. However, there remain many other constellation-based digital modulations that we could try to implement and test. Additionally, there are a number of digital modulations that do not have a constellation representation, such as frequency-shifted keying, as well as the analog modulations. In order for our classification algorithm to be truly well-suited for an SDR or cognitive radio, we should investigate methods to apply our feature set to these modulation types.

Chapter 5

Spectrum Sensing Architecture

Spectrum sensing is the process used by a cognitive radio to identify available channels of both licensed and unlicensed spectrum in order to wirelessly communicate. The frequency spectrum is a dynamic system that changes with time and location. The availability of a channel in the licensed spectrum depends on the activity of the primary user which has priority to the licensed spectrum. A primary user has the highest priority access to the spectrum, and our architecture must not interfere with this access. However, our cognitive radio has equal priority access to the spectrum with other secondary users. For example, the IEEE 802.22 standard requires a secondary user to vacate the channel or reduce power within two seconds of a primary user becoming active [15], Thus, a cognitive radio must quickly detect this appearance to prevent interfering with the primary user [16], which implies that the cognitive radio must devote computational resources to perform spectrum sensing.

As we previously discussed in Chapter 1, there are five common algorithms that enable spectrum sensing: energy detection, cyclostationary detection, waveform-based sensing, radio identification, and match filtering [19–31]. We improve upon these spectrum sensing approaches, in particular the radio identification type of sensing, by applying the waveform knowledge about the signal of a licensed primary user, such as the spectrum frequency location, bandwidth, and modulation type. We developed our architecture to automatically and appropriately execute our narrowband signal detect algorithm and modulation classification algorithm, which were discussed in Chapter 3 and Chapter 4, respectively.

In this chapter, we develop a spectrum sensing architecture to find spectrum frequency locations available for use by a cognitive radio. We treat the spectrum as a wideband signal that contains $M \geq 0$ narrowband signals which are primary or secondary. Our architecture estimates a set of parameters for each narrowband signal in order to determine which frequencies in the spectrum are available to our cognitive radio: the signal carrier frequency (or center frequency), the signal bandwidth, and the signal modulation type. We refer to this parameter set as the signal identification.

We assume that we know the signal identification for primary users in the spectrum.

The novelty of our approach is that we incorporate the knowledge of the modulation type used by the primary user to improve our reporting of available spectrum. A cognitive radio that utilizes this extra information can outperform other cognitive radios that use less information. As we shall discuss, we found that the probability of correctly detecting a spectrum opportunity was 99.4% for the cognitive radio that uses all of the information provided by our spectrum sensing architecture compared to 87.7% and 93.8% for the two other types of cognitive radio that do not make use of this extra information. The probability of correctly detecting a "grey-space" spectrum opportunity was significantly greater, 96.1%, for the cognitive radio that uses all of the information provided by our architecture, compared to 49.1%, for the cognitive radio that do not make use of this extra information. The probability of a false alarm of a primary user using our architecture was 13.3% compared to 46.9% and 62.7% for the competing techniques.

The remainder of this chapter is organized as follows. In Section 5.1, we describe our spectrum sensing architecture. In Section 5.2, we present our methods to evaluate our architecture along with our experimental results. Finally, in Section 5.3, we discuss future work and improvements to our architecture.

5.1 Architecture Description

In this section, we describe our spectrum sensing architecture, which consists of four main components. The first component is the knowledge-base representation of the channels in the spectrum, and the parameterization and configuration of the primary users. The second component receives the wideband spectrum and detects channels that are in use by narrowband signals. The third component is the modulation classification to further identify characteristics of any active channel that might be a primary user. Finally, the last component is the spectrum activity report. We illustrate our spectrum sensing architecture in Figure 5.1 where the four main components are easily seen. Next, we state the spectrum sensing problem more explicitly along with the assumptions we use, and then we describe each of these four components in more detail.

5.1.1 Spectrum Sensing Problem Statement and Assumptions

Spectrum sensing is the cognitive radio task of detecting the activity in the spectrum in order to identify the available spectrum bandwidth for the cognitive radio's utilize for its own communication usage, while not impeding nor harmfully interfering with the licensed primary users of the spectrum.

We assume our received wideband signal is a linear combination of $M \geq 0$ digitally modulated transmitted narrowband signals with differing channel usage (i.e. center frequency and bandwidth)

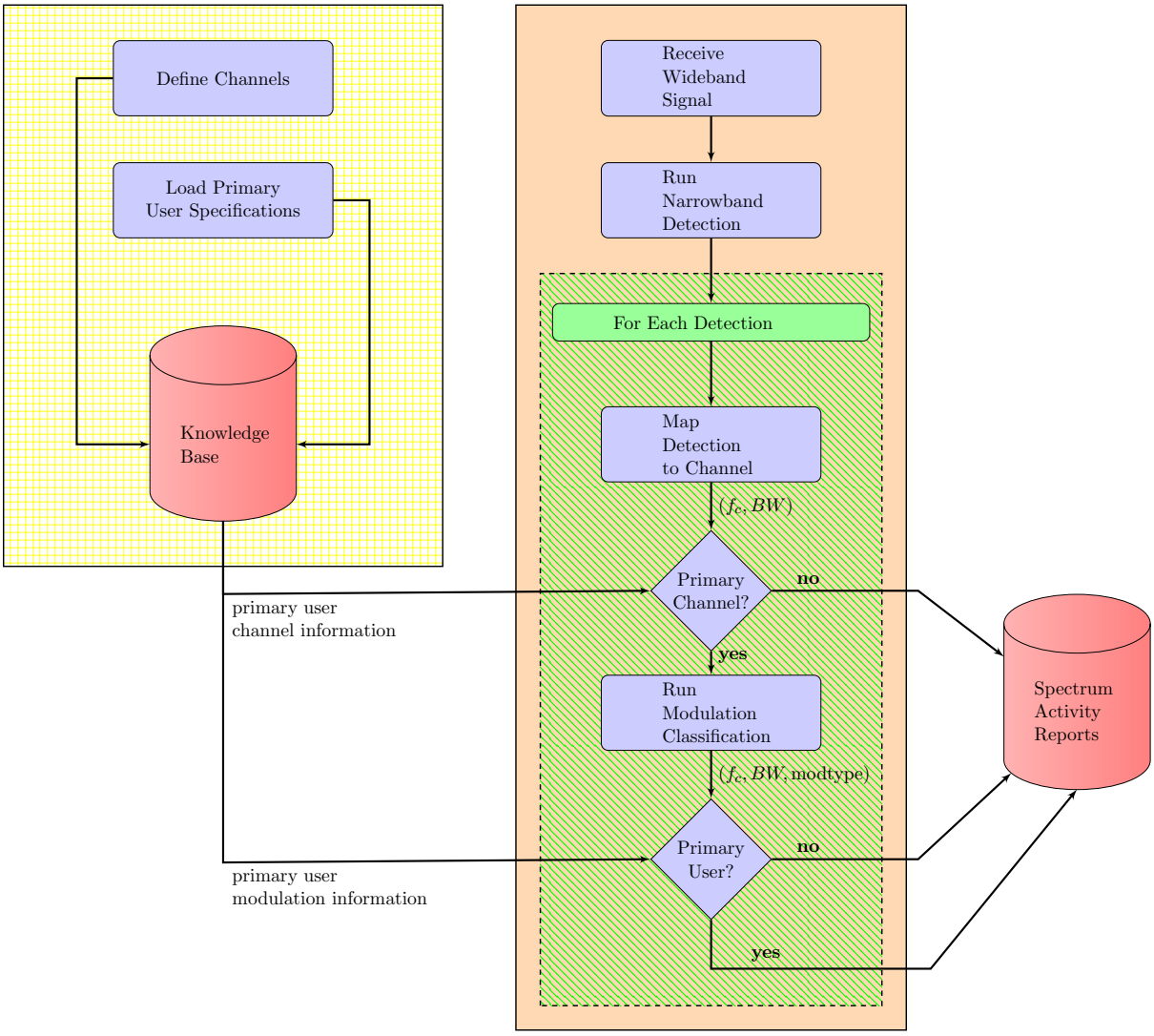


Figure 5.1: Block diagram of our proposed spectrum sensing architecture that incorporates the knowledge about the modulation type used by a primary user.

in addition to noise in a non-fading environment where the characterization of M, modulation type, center frequency and bandwidth are all unknown. We assume that the noise is a circular normal, additive white complex Gaussian noise process. We also assume that we do not have co-channel inference, i.e. in a single channel there can exist at most a single narrowband signal. We assume that we know the signal identification of each primary user that possibly could use the spectrum. We assume that any transmitted signal is digitally modulated using a modulation type that our classifier can identify and also that an ideal rectangular pulse shaping filter is used. These assumptions simplify our testing in order to show the utility of our architecture, which is a common approach towards this goal [158–164]. The results of this testing establishes a baseline performance, so that in the future, we can extend our testing to incorporate to other modulation types, pulse shaping filters, and channel types, and compare against this baseline.

5.1.2 Primary User Knowledge Base

In Figure 5.1, we represent our primary user knowledge base as the yellow block. There are three items that creates this knowledge base: the number of spectrum cells, channel definitions, and the primary user specifications.

The number of spectrum cells, N_c, represents the bandwidth resolution for our spectrum sensing architecture where each cell contains an equal amount of bandwidth. For example, if our wideband spectrum contains 128 MHz of bandwidth and the number of spectrum cells is 16, then each cell is allocated 8 MHz where the i^{th} cell spans from $8 \cdot i$ MHz to $8 \cdot (i+1)$ MHz. In general, if we operate in the spectrum from W_L MHz to W_H MHz, then the total bandwidth is W MHz where $W = W_H - W_L$ and correspondingly, the i^{th} spectrum cell contains the frequencies from $W_L + \frac{W}{N_c}(i)$ MHz to $W_L + \frac{W}{N_c}(i+1)$ MHz for $i = 0, 1, ..., N_c - 1$.

A primary user signal identification specifies the range of spectrum licensed to the primary user and also provides a list of possible modulation types used by the primary user. We define the primary user's spectrum usage in terms of spectrum cells. In order words, we define the licensed spectrum of a primary user by a lower spectrum cell index and an upper spectrum cell index, which denotes the lower and upper boundary of the total licensed bandwidth. For example, if our wideband spectrum goes from 128 MHz to 256 MHz with $N_c = 16$ and a primary user is licensed from 192 MHz to 208 MHz, then this bandwidth corresponds to spectrum cells 8, 9 and 10 where 8 would be the lower spectrum cell index and 10 would be the upper spectrum cell index.

We also use the number of spectrum cells to create our set of channels. We recursively build this set starting with channels equal to the spectrum cells themselves. Next we double the bandwidth and create every possible channel from two contiguous spectrum cells. Next we double again the bandwidth and create every possible channel from four continuous spectrum cells. This process continues until we create channels with a bandwidth size equal to one-quarter of the total wideband

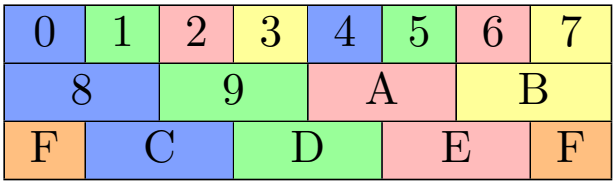


Figure 5.2: Example set of channels created when the number of spectrum cells is set to eight, i.e $N_c = 8$.

spectrum bandwidth.

Suppose we let $N_c = 8$, then Figure 5.2 shows a brickwall pattern of the channels we create from the 8 spectrum cells. We see that there is a total of 16 channels which we enumerate with hexadecimal notation. Channels 0 through 7 are the 8 spectrum cells. Channels 8 through B are the concatenation of consecutive pairs of spectrum cells starting with even numbered cells. Channels C through F are the concatenation of consecutive pairs of spectrum cells starting with odd numbered cells. Notice that channel F wraps around the spectrum as the union of channels 0 and 7, which is due to the periodicity that occurs as a result from sampling the spectrum.

In general, if the number of spectrum cells is a power of 2, i.e. $N_c = 2^k$ for $k > 1$, then the total number of channels in our set is $(k-1) \cdot N_c$. Our spectrum sensing does not require N_c to be a power of 2, however the channel set creation is simplified if that is the case.

We create our channel set in this fashion because it offers an easy method to associate channels with spectrum cells and also report the spectrum activity. If we detect a signal on channel E in Figure 5.2, then we quickly know that we must report some activity on spectrum cells 5 and 6. We discuss the specifics of our spectrum activity reporting in Section 5.1.5.

5.1.3 Narrowband Signal Detection

Narrowband signal detection is the first operation performed by the spectrum sensing architecture, which is depicted as the top two subblocks of the red block in Figure 5.1. This algorithm receives the spectrum as a wideband signal and tries to determine the number of narrowband signals present, and for each signal detected the center frequency and bandwidth. The result of this operation is a list of narrowband signal detections, which is parameterized by the center frequency and bandwidth.

Energy detection is a common approach for this type of signal detection and is the implementation of the maximum likelihood decision rule for detecting purely complex exponential signals in noise [109, 110]. However, we are interested in signals that contain digitally modulated information, which are not purely complex exponential signals. In Chapter 3, we described in detail a novel approach to this signal detection problem that uses the sinc function as an approximation for an unknown signal, which that can be used to estimate the center frequency and bandwidth.

5.1.4 Channelization and Modulation Classification for Primary User Identification

For every signal detected by the narrowband signal detection algorithm, the following steps are used to further identify the signal as a primary or secondary user. This process is depicted as the green subblock of the red block in Figure 5.1.

The first step is to channelize the narrowband detection, i.e. map the detection to a channel. Recall that a detection consists of a center frequency and bandwidth. Due to the estimation process of the narrowband detection algorithm, we do not expect these parameters to exactly match the center frequency and bandwidth of each possible channel. Therefore, we map the detection to the nearest channel in which we use the L_1 metric to measure the distance as shown in Equation 5.1, where `chan` is the returns the nearest channel, (f_d, BW_d) is the estimated center frequency and bandwidth from the narrowband detection, and $\mathcal{C}$ is the set of center frequency and bandwidth tuples for each channel.

$$\texttt{chan}(f_d, BW_d) = \arg \min_{(f,BW)\in\mathcal{C}} |f - f_d| + |BW - BW_d|. \tag{5.1}$$

Once the detection is channelized, we look up the set of spectrum cells associated with that channel to determine if there is an overlap with the known set of spectrum cells licensed to a primary user. If an overlap does not exists, then we mark the detection as a secondary user and proceed to process the next detection. However, if an overlap does exist, then we mark the detection as a potential primary user and perform further processing in order to determine the type of user.

If a detected narrowband signal is marked as a potential primary user, then we need to classify the modulation type of the signal. Our modulation classification algorithm was discussed in Chapter 4. However, that classifier was designed to operate on a complex baseband signal of a narrowband signal. Thus, before we can classify the modulation, we must resample the data. Resampling is a simple signal processing process that changes the effective sample rate of a signal by moving the narrowband signal in the channel to the baseband and then filtering away the signal outside of the channel [88]. We then run our modulation classification algorithm on the resampled signal to predict the modulation type. If the classified modulation type belongs to the set of possible modulations used by a primary user operating in the channel, we mark the detection as a primary user. Otherwise we mark the detection as a secondary user.

5.1.5 Spectrum Activity Reports

The ultimate goal of our spectrum sensing architecture is to produce a report of available frequency bandwidths in the spectrum for use by a cognitive radio. However, it is undesirable for our system to indicate that a portion of spectrum is available when a primary user is transmitting in that spectrum, since our cognitive radio might interfere with that primary user. Similarly, if a portion

of the spectrum is free but our system indicates that the spectrum is not available, the result would be degraded performance of our cognitive radio. However this consequence is less undesirable than interfering with a primary user.

The final component of our spectrum sensing architecture is reporting on the spectrum activity, which combines all of the information generated by the algorithms in our architecture as shown in Figure 5.1. The spectrum activity is reported via the *spectrum cell state vector*, which reports the activity state for each of the spectrum cells in a single quantity. There are four possible activity states for a spectrum cell.

The first state is the *open* state that indicates the absence of any detectable signal and that the cognitive radio can use this channel.

The second state is the *active* state that indicates the presence of a signal, however the spectrum cell is not within the licensed spectrum of a primary user.

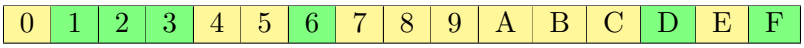
The third state is the *suser* state that indicates presence of a signal and that the spectrum cell is inside the licensed spectrum of a primary user, and our modulation classification algorithm classified the signal to a modulation type that is not a possible primary user modulation type – this state implies that a secondary user is active in this channel.

The fourth state is the *puser* state that indicates the presence of a signal and that the spectrum cell is inside the licensed spectrum of a primary user, and our modulation classification algorithm classified the signal to a modulation type that matches a possible primary user modulation type – this state implies that a primary user is active in this channel.

All states except for the open state indicate that there is a signal is present, but the differences are whether we try to distinguish if the signal is primary or secondary users. These three states that indicate the presence of a signal represent the granularity of information provided to the cognitive radio to help it determine if it wishes to utilize the channel at the risk of an interfering signal being present.

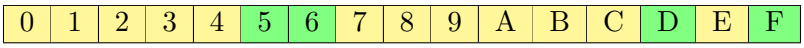
Figure 5.3 shows examples of spectrum cell state vector that might be reported to the cognitive radio where we divide the spectrum into $N_c = 16$ cells, and for reference we label the cells with hexadecimal characters. We use the colors, green, yellow, orange, and red for the open, active, suser, and puser spectrum cell states, respectively.

In our first example, we assume that a primary user may appear in any of the spectrum cells, i.e. the primary user is licensed to operate in the whole spectrum. In Figure 5.3a, we show the spectrum cell state vector after the narrowband detection algorithm completes and we see that a contiguous block of spectrum cells are available from cell 1 to cell 3, and also there are single spectrum cells (6, D, and F) without a signal present. At this point, Figure 5.3a represents a preliminary report that could be made to the cognitive radio on the available spectrum, which is the typical approach when we stop after the narrowband signal detection algorithm. However, our spectrum sensing



(a) Example 1 after narrowband detection algorithm runs

(b) Example 1 after modulation classification algorithm runs



(c) Example 2 after narrowband detection algorithm runs

(d) Example 2 after modulation classification algorithm runs

Figure 5.3: Example spectrum cell state vectors, where the state can be: open, active, secondary user detected, and primary user detected, which are represented by the colors, green, yellow, orange, and red, respectively.

architecture goes further and classifies the modulation on each signal detected within the licensed spectrum. Suppose we determine that the spectrum cells 0, 4, 5, and 7 are secondary users and the remaining detections are primary users. We update the spectrum cell state vector as shown in Figure 5.3b. This vector represents the spectrum activity report that is sent to the cognitive radio.

This extra step to classify the modulation type of a detected signal allows our spectrum sensing architecture to equip the cognitive radio with a better understanding of the spectrum usage. Suppose that our cognitive radio requires the usage of four spectrum cells to transmit its data. In the previous example, if the cognitive radio only possessed the preliminary report, as shown in Figure 5.3a, then the cognitive radio would not be able to transmit and must wait for other channels to open. However, with the final report, as shown in Figure 5.3b, the cognitive radio can decide if it wants to wait for a better spectrum condition or transmit in a channel occupied with by a secondary user with potentially degraded performance.

In our second example, we assume that a primary user may appear in the upper half of the spectrum, i.e. spectrum cells 8 through F. In Figure 5.3c, we show the spectrum cell state vector after the narrowband detection algorithm completes and we see that there are a few cells that do not contain a signal, specifically cells 5, 6, D, and F. In this situation, we perform the modulation classification for each of the signals detected in the upper half of the spectrum. By our assumptions, in this example we can automatically claim that the signals detected in spectrum cells less than 8 are active, but we do not assert if they are primary or secondary users. Suppose after the modulation classification we update the spectrum cell state vector as shown in Figure 5.3d, which indicates that the modulation classified in spectrum cells 8 and 9 did not match a possible modulation that could be used by a primary user, yet the remaining spectrum cells did have a modulation classified that matched a primary user. This vector represents the spectrum activity report sent to the cognitive radio for this example. The extra knowledge about the primary user radio protocol allows our

spectrum sensing architecture to effectively manage the processing resources to run our algorithms. In this example, we did not need to use our modulation classification algorithm on the signals detected in the lower half of the spectrum since we knew that the primary user does not operate in that region of the spectrum.

5.2 Spectrum Sensing Evaluation

In this section, we present the evaluation of our spectrum sensing architecture. We evaluate our architecture by generating synthetic wideband signal sets with varying configurations, which is a desirable approach because we know the exact values of all of the narrowband signal parameters. This ground truth set allows us to measure statistics and judge the performance of our architecture.

The purpose of a spectrum sensing architecture is to produce a report of available frequency bandwidths in the spectrum for use by a cognitive radio. Recall that the spectrum activity report produced by our spectrum sensing architecture is in terms of spectrum cells where each cell is an equally-sized subdivision of the whole spectrum. Our evaluation tries to determine the ability of our architecture to provide a cognitive radio with spectrum opportunities that avoid interfering with the primary user(s).

We assess the architecture's probability of detecting these spectrum opportunities, which are defined as a spectrum cell that has a state value of either open, active, or secondary user active. Additionally, we determine the probability of detecting spectrum "grey spaces", which are defined as portions of the spectrum that are being used by low-powered interferers where these interferers are typically secondary users to the spectrum [165–169]. More specifically to our evaluation, we define a grey opportunity as a spectrum cell that has a state value of either active or secondary user active. We measure the probability of a primary user false alarm, which limits the cognitive radio's ability to access the spectrum. A primary user false alarm occurs when the architecture falsely reports that a spectrum cell contains an active primary user when this is not a true assessment. The success of the architecture's effort to avoid interfering with primary users is quantified by the probability of detecting the primary user. A primary user detection occurs when the architecture correctly reports that a spectrum cell contains an active primary user. We discuss these evaluation measurements in more detail later.

We note that the evaluation of our spectrum sensing architecture is dependent on the type of cognitive radio that uses our spectrum activity reports. Thus, in our analysis of our evaluation, we consider three types of cognitive radios: Typical Cognitive Radio (TYPCR), Primary License(s) Aware Cognitive Radio (PLACR), and Primary Waveform(s) Aware Cognitive Radio (PWACR). We give a brief description of the three types of cognitive radios here.

The TYPCR is the usual, conventional type of cognitive that employs only a narrowband signal

(a) Spectrum cell state vector ground truth

(b) Spectrum cell state vector reported

Figure 5.4: Two example spectrum cell state vectors: the ground truth and the spectrum sensing architecture report where the primary user is licensed to the upper half of the spectrum (cells 8 through F) and where the activity state can be: open, active, secondary user detected, and primary user detected, which are represented by the colors, green, yellow, orange, and red, respectively.

detection algorithm to determine if there is a spectrum opportunity.

The PLACR has knowledge about the spectrum locations in which a primary license resides. This type of cognitive radio can make a better primary/secondary user judgment about an active signal in the spectrum by comparing the narrowband detection locations to the known primary license

The PWACR has knowledge about the primary user waveform(s) that can exist in the spectrum. Our spectrum sensing architecture is intended to work with this type of cognitive radio because our architecture classifies the modulation type used by an active signal in the spectrum which provides the cognitive radio with more information for the decision-making algorithms. Thus, we intend to show that the PWACR is superior to the other two cognitive radio types.

The remainder of this section is organized as follows. In Section 5.2.1, we define the metrics that we use to evaluate our spectrum sensing architecture. In Section 5.2.2, we describe a directed test that explores a configuration space to determine the sensitivity of our spectrum sensing architecture as a function of configuration, and we report and analyze the results of this test. In Section 5.2.3, we describe a random test that randomly samples a configuration space to evaluate our spectrum sensing architecture in a variety of different conditions, and we report and analyze the results of this test.

5.2.1 Evaluation Metrics

The output of our spectrum sensing architecture is the spectrum activity report which is represented using a the spectrum cell state vector. For every test in our evaluation, we compare the reported spectrum cell state vector with the known ground truth vector. We use Figure 5.4 as an example to help describe each of the measures in this section, where the ground truth, and the spectrum cell state vector reported by our spectrum sensing architecture are shown in Figures 5.4a and 5.4b, respectively. The colors green, yellow, orange, and red, correspond to spectrum cell states of open, active, secondary user detected, and primary user detected, respectively.

	O	A	S	P
O	4	1	1	1
A	0	2	0	0
S	1	0	1	1
P	1	0	0	3

Figure 5.5: Example spectrum activity confusion matrix generate from the ground truth and reported spectrum cell state vectors show in Figure 5.4a and Figure 5.4b, respectively, where the rows of the matrix are indexed with respect to the ground truth vector, the columns of the matrix are indexed with respect to the reported vector, and the labels 'O', 'A', 'S', and 'P' correspond to the open, active, secondary user, primary user states, respectively

.

In this example shown in Figure 5.4a, we see that there is a primary user active in cells 8 through B, a secondary user active in cells E and F, an active signal outside of the licensed spectrum in cells 5 and 6, and the remaining spectrum cells are open. In Figure 5.4b, the hypothetical result of our spectrum sensing architecture estimates that there is a primary user active in cells 9 through C, a secondary user active in cells D and E, an active signal in cells 5 through 7, an active signal at cell 0, and the remaining spectrum cells are open. Notice that the primary user and secondary user are detected, but with an offset of a single cell, and the cells 0 and 7 are signal false alarms.

We can compactly represent the similarities and differences between the ground truth and reported spectrum cell state vectors in a confusion matrix. We have four types of cells: open, active, secondary user, and primary user. Thus, the spectrum activity confusion matrix is a 4-by-4 matrix where the rows of the confusion matrix are indexed by the state value for the ground truth vector, and the columns of the confusion matrix are indexed by the state value for the reported vector. We use the labels `O`, `A`, `S`, and `P` to correspond to the open, active, secondary user, primary user states, respectively. We can reference a single element in the confusion matrix by row-column index. For example, `SA` indexes the element the resides in the `S` row and `A` column.

Note that the confusion matrix value at index `PA` always is 0. This index represents the case when the spectrum cell is actually a a primary user, but the spectrum sensing architecture marks the cell active. This situation cannot happen because the architecture marks a cell active only when a signal detection occurs outside of a primary user licensed spectrum, but by definition a cell that contains a primary user signal will be inside of a portion of a primary user licensed spectrum. Thus, the confusion matrix value at index `PA` always is 0.

In Figure 5.5, we show the spectrum activity confusion matrix generated from the ground truth and reported spectrum cell state vectors for the example from Figure 5.4. From Figure 5.4a we see that there are 7 open spectrum cells and if we look at the `O` row in Figure 5.5, we see that index `OO` of the confusion matrix has a value of 4, which means that the spectrum sensing architecture

labeled 4 open spectrum cells correctly as open. The indices `OA`, `OS`, and `OP` each correspond to a value of 1 in the confusion matrix, which implies that the spectrum sensing architecture mistakenly labeled an open spectrum cell to active, secondary user, and primary user.

In addition to compactly representing the spectrum cell activity report, the spectrum activity confusion matrix easily shows the differences in the type of cognitive radios that might use this information. The TYPCR asserts the presence of a primary user whenever a signal is detected in the spectrum, which corresponds to the `A`, `S`, and `P` columns in the confusion matrix. The PLACR asserts the presence of a primary user whenever a signal is detected in a primary user licensed portion of the spectrum, which corresponds to the `S` and `P` columns in the confusion matrix. The PWACR asserts the presence of a primary user whenever a signal is detected in a primary user licensed portion of the spectrum, and when the modulation type of that signal matches the known modulation of a primary user, which corresponds to the `P` columns in the confusion matrix.

Another benefit of the spectrum activity confusion matrix is that we can calculate our evaluation metrics from this matrix. We have four evaluation metrics for our spectrum sensing architecture of which two metrics measure the ability to detect spectrum opportunities and the other two metrics measure the ability to detect and avoid interfering with the primary user(s) in the spectrum.

The first spectrum opportunity metric is the probability of correctly detecting a spectrum opportunity. We define a spectrum opportunity as a spectrum cell that has a state value of either open, active, or secondary user active, i.e. the cell does not have a primary user active. This measure quantifies the ability of our spectrum sensing to identify the spectrum cells available for the cognitive radio. A greater value of this measure implies that the combination of our spectrum sensing architecture and the cognitive radio better utilize the available bandwidth to achieve a potential higher data transmission throughput. We express this probability of the ratio of the number of spectrum opportunities asserted by the cognitive radio divided by the true number of spectrum opportunities, as shown in Equation 5.2.

$$Pr\{\text{ detect spectrum opportunity } | \text{ CR } \} = \frac{\text{CR number of spectrum opportunities}}{\text{true number of spectrum opportunities}} \tag{5.2}$$

In Figure 5.6, we show how to calculate the probability of correctly detecting a spectrum opportunity from the spectrum activity confusion matrix as a function of cognitive radio type. The sum value of the set of grey-shaded indices represents the number of spectrum cells that did not contain an active primary user. This sum becomes the denominator of the ratio. The numerator of the ratio depends on the type of cognitive radio that is using the information reported by our spectrum sensing architecture. The TYPCR type only asserts a spectrum opportunity when a signal is not detected, which in the confusion matrix is represents by the index `OO`, as shown by the red-circled set in Figure 5.6. The PLACR type asserts a spectrum opportunity when a signal is not detected outside of a primary user(s) licensed spectrum, which in the confusion matrix is represents by the

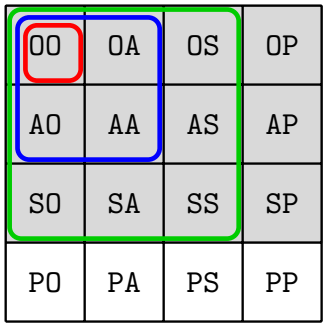


Figure 5.6: Illustration of the probability of correctly detecting a spectrum opportunity from the spectrum activity confusion matrix which is the ratio of two sets of indices from this matrix that depends on the type of cognitive radio in use where the sum value of the set of indices enclosed by the red, blue, and green boxes is the numerator of the ratio for the TYPCR, PLACR, and PWACR types, respectively, and the sum value of the set of grey-shaded indices is the denominator of the ratio.

indices `OO`, `OA`, `AO`, and `AA`, as shown by the blue-circled set in Figure 5.6. The PWACR type asserts a spectrum opportunity when a signal is not detected that matches the primary users' location and modulation type, which in the confusion matrix is represents by the indices `OO`, `OA`, `OS`, `AO`, `AA`, `AS`, `SO`, `SA`, and `SS`, as shown by the green-circled set in Figure 5.6.

Referring back to our example in Figure 5.5, we calculate the probability of spectrum opportunity to be $\frac{4}{12}$, $\frac{6}{12}$, and $\frac{10}{12}$, for TYPCR, PLACR, and PWACR, respectively. Notice that this evaluation metric is a function of both the inactive spectrum cells and the non-primary user active spectrum cells, which means that this metric can be biased by the large underutilization of the spectrum. We balance this potential bias in our evaluation with the next metric.

The second spectrum opportunity metric is the probability of correctly detecting a "grey-space" spectrum opportunity. We define a grey opportunity as a spectrum cell that has a state value of either active or secondary user active. This evaluation metric demonstrates the improvement gained by incorporating the information about the primary user(s) waveform characteristics in the decision-making process of the cognitive radio. We express this probability of the ratio of the number of grey opportunities asserted by the cognitive radio divided by the true number of grey opportunities, as shown in Equation 5.3.

$$Pr\{\text{ detect grey opportunity} \mid \text{CR}\} = \frac{\text{CR number of grey opportunities}}{\text{true number of grey opportunities}} \tag{5.3}$$

In Figure 5.7, we show how to calculate the probability of correctly detecting a "grey-space" spectrum opportunity from the spectrum activity confusion matrix. The sum value of the set of grey-shaded indices represents the number of spectrum cells that contain an active signal outside of a primary user(s) licensed spectrum or an active secondary user. This sum becomes the denominator

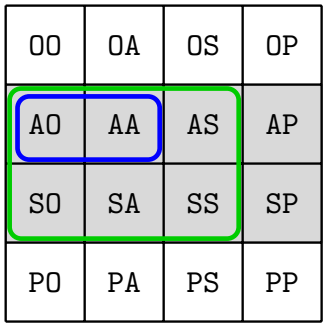


Figure 5.7: Illustration of the probability of correctly detecting a "grey-space" spectrum opportunity from the spectrum activity confusion matrix which is the ratio of two sets of indices from this matrix that depends on the type of cognitive radio in use where the sum value of the set of indices enclosed by the blue and green boxes is the numerator of the ratio for the PLACR and PWACR types, respectively, and the sum value of the set of grey-shaded indices is the denominator of the ratio.

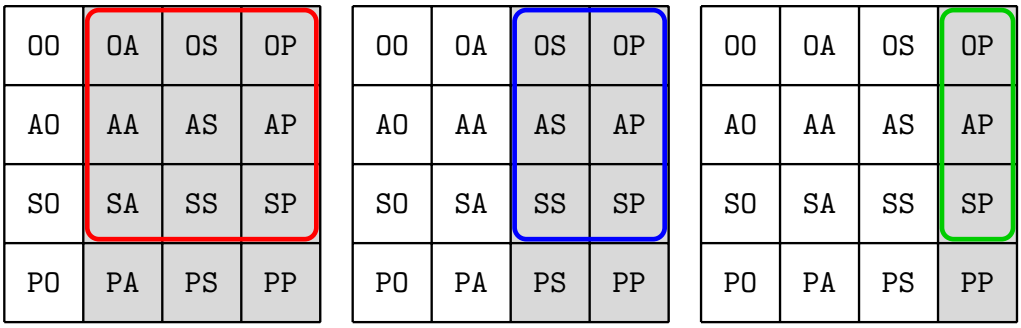
of the ratio. The numerator of the ratio depends on the type of cognitive radio that is using the information reported by our spectrum sensing architecture. The PLACR type asserts a grey opportunity when a signal is detected that is outside of a primary user(s) licensed spectrum, which in the confusion matrix is represents by the indices `AO` and `AA`, as shown by the blue-circled set in Figure 5.7. The PWACR type asserts a spectrum opportunity when a signal is detected that does not match the primary users' location and modulation type, which in the confusion matrix is represents by the indices `AO`, `AA`, `AS`, `SO`, `SA`, and `SS`, as shown by the green-circled set in Figure 5.7.

Referring back to our example in Figure 5.5, we calculate the probability of "grey-space" spectrum opportunity to be $\frac{0}{5}$, $\frac{2}{5}$, and $\frac{4}{5}$, for TYPCR, PLACR, and PWACR, respectively. Notice that the TYPCR type always has a zero-value for this metric because it asserts any signal detection as a primary user. Thus, this type of cognitive radio cannot take advantage of the "grey-space" spectrum opportunity.

The first primary user evaluation metric is the probability of primary user false alarm. A primary user false alarm occurs when the spectrum sensing architecture falsely reports that a spectrum cell contains an active primary user when this is not a true assessment. This metric quantifies the percentage of primary user reports that are falsely reported. A large probability of false alarm implies a degraded ability for the cognitive radio to access spectrum to transmit data. We express this probability of the ratio of the number of CR primary user assertions that are false divided by the total number of CR primary user assertions, as shown in Equation 5.4.

$$Pr\{\text{ primary user false alarm} \mid \text{CR }\} = \frac{\text{CR number of false primary user asserts}}{\text{CR number of primary user asserts}} \tag{5.4}$$

In Figure 5.8, we show how to calculate the probability of primary user false alarm from the



(a) TYPCR (b) PLACR (c) PWACR

Figure 5.8: Illustration of the probability of primary user false alarm from the spectrum activity confusion matrix which is the ratio of two sets of indices from this matrix that depends on the type of cognitive radio in use where the sum value of the set of indices enclosed by the red, blue, and green boxes is the numerator of the ratio for the TYPCR, PLACR, and PWACR types, respectively, and the sum value of the set of grey-shaded indices is the denominator of the ratio, which is also dependent on the type of cognitive radio.

spectrum activity confusion matrix. This probability is a function of the number of primary user assertions made by the cognitive radio. The set of confusion matrix indices that represent a primary user assertion depends on the type of cognitive, which are depicted as subplots for TYPCR, PLACR, and PWACR, as shown in Figure 5.8a, Figure 5.8b, and Figure 5.8c, respectively. In each case, the sum value of the set of grey-shaded indices represents the total number of primary user assertions as a function of the cognitive radio behavior, which becomes the denominator of the ratio. The red-, blue-, and green-circled subset of grey-shaded indices represents the incorrect primary user assertions made by the TYPCR, PLACR, and PWACR, respectively, and the sum value of this set is the numerator of the ratio. Referring back to our example in Figure 5.5, we calculate the probability of primary user false alarm to be $\frac{7}{10}$, $\frac{4}{7}$, and $\frac{2}{5}$, for TYPCR, PLACR, and PWACR, respectively.

The second primary user metric is the probability of primary user detection. This metric quantifies the ability of the spectrum sensing architecture to detect the active primary users in order to avoid interfering with these users. We express this probability of the ratio of the number of CR primary user assertions conditioned on being a primary user, as shown in Equation 5.5.

$$Pr\{\text{ primary user detection} \mid \text{CR }\} = \frac{\text{CR number of correct primary user asserts}}{\text{true number of primary user}} \tag{5.5}$$

In Figure 5.9, we show how to calculate the probability of primary user detection from the spectrum activity confusion matrix. This probability is a ratio of two values where the denominator of this probability ratio is the sum value of the P row of the confusion matrix, as shown by the grey-shaded set of indices in Figure 5.9. The numerator of the ratio depends on the type of cognitive radio that

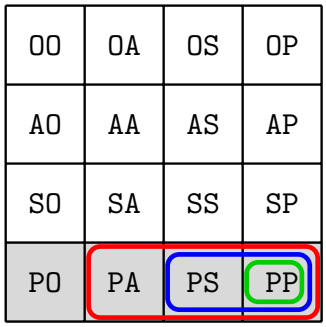


Figure 5.9: Illustration of the probability of primary user detection from the spectrum activity confusion matrix which is the ratio of two sets of indices from this matrix that depends on the type of cognitive radio in use where the sum value of the set of indices enclosed by the red, blue, and green boxes is the numerator of the ratio for the TYPCR, PLACR, and PWACR types, respectively, the sum value of `P` row of the confusion matrix (the set of grey-shaded indices) is the denominator of the ratio.

is using the information reported by our spectrum sensing architecture. The TYPCR type asserts any detect signal as a primary user, which in the confusion matrix is represented by the indices `PA`, `PS`, and `PP`, as shown by the red-circled set in Figure 5.9. The PLACR type asserts a primary user when a signal is detected inside of a primary user(s) licensed spectrum, which in the confusion matrix is represents by the indices `PS` and `PP`, as shown by the blue-circled set in Figure 5.9. The PWACR type asserts a primary user only when a signal is detected in the primary user(s) licensed spectrum and modulation type matches that of the primary user, which in the confusion matrix is represents by the index `PP`, as shown by the green-circled set in Figure 5.9. Referring back to our example in Figure 5.5, we calculate the probability of primary user detection to be $\frac{3}{4}$ for each of TYPCR, PLACR, and PWACR.

The combination of the probability of primary user detection and the probability of primary user false alarm creates a relationship between potentially interfering with a primary user versus limiting the cognitive radio's ability to transmit data. As we increase the probability of primary user detection, then we tend to also increase the probability of primary user false alarm, which reduces the potential interference with a primary user, but also limits the cognitive radio's access to the spectrum. On the other hand, as we decrease the probability of primary user false alarm, then we tend to also decrease the probability of primary user detection, which increases the cognitive radio's access to the spectrum, but also increases the potential interference with a primary user. The cognitive radio's decision-making process must decide how to balance these competing interests to set it's desired probability of primary user detection and false alarm behavior.

Parameter Set	Values
$\mathcal{N}_c$	8, 16, 32, 64
$\mathcal{L}$	$[0, N_c)$, $[\frac{1}{2}N_c, N_c)$, $[\frac{1}{4}N_c, \frac{3}{4}N_c)$
$\mathcal{M}_T$	BPSK, QPSK, 8-PSK, 16-PSK, 8-QAM, 16-QAM, 32-QAM, 64-QAM, 4-PAM, 8-PAM, 16-PAM, 32-PAM
$\mathcal{M}_F$	PSK, QAM, PAM
$\mathcal{M}_P$	$\mathcal{M}_T \cup \mathcal{M}_F$
$\mathcal{S}$	0 dB, 5 dB, 10 dB, 15 dB, 20 dB

Table 5.1: Directed test configuration parameter sets and values: $\mathcal{N}_c$ for the number of spectrum cells, $\mathcal{L}$ for the primary user licensed spectrum cells, $\mathcal{M}_P$ for the primary user modulation set, which is the union of $\mathcal{M}_T$ for the primary user modulation type set and $\mathcal{M}_F$ for the primary user modulation family set, and $\mathcal{S}$ for the SNR value.

5.2.2 Directed Test Evaluation

The first evaluation is a series of directed tests that exhaustively exercises our spectrum sensing architecture over the configuration parameter space to understand the performance behavior of our architecture when a single primary user license is defined and a single signal is active in the spectrum. In each test, we must define the primary user license along with the active signal's characteristics. This evaluation intends to show that our spectrum sensing architecture is robust against varying these configuration parameters, where the configuration parameter space, which is described later, is shown in Table 5.1.

We define the primary user license, as described in Section 5.1.2, by the number of spectrum cells, N_c, the range of primary user licensed spectrum cells, L, and the primary user modulation list, $\mathcal{M}$. We let $\mathcal{N}_c$ be the possible set of number of spectrum cells for our spectrum sensing architecture. We vary $N_c \in \mathcal{N}_c$ to study the effect of the spectrum resolution on our architecture's metric evaluations. We let $\mathcal{L}$ be the possible set of licensed spectrum cell ranges, and we select $L \in \mathcal{L}$ to understand our architecture's sensitivity to the location of the primary license in the spectrum. The set $\mathcal{L}$ contains 3 ranges that are capable of testing this sensitivity. The range $[0, N_c)$ contains the whole spectrum. The range $[\frac{1}{2}N_c, N_c)$ contains the upper-half of the spectrum, which exercises the system when there is a single boundary between the licensed and unlicensed spectrum. The range $[\frac{1}{4}N_c, \frac{3}{4}N_c)$ contains the middle-half of the spectrum, which exercises the system when there are an upper and lower boundaries between the licensed and unlicensed spectrum. Finally, we select the primary user modulation list $\mathcal{M} \in \mathcal{M}_P$. The set $\mathcal{M}_P$ contains 15 elements where 12 elements are the modulation types in $\mathcal{M}_T$ that our modulation classifier can recognize and the final 3 elements are the modulation families in $\mathcal{M}_F$: PSK, QAM, and PAM. If $\mathcal{M}$ is a single modulation type, then this implies that a primary user only uses that type of modulation. However, when $\mathcal{M}$ is a modulation family, then this implies that a primary user can select any modulation in that family. For example,

if $\mathcal{M}$ = PSK, then the primary user can modulate the data with BPSK, QPSK, 8-PSK, or 16-PSK.

To characterize the active signal in the test, we must define whether the signal is a primary or secondary user, the channel location, the modulation type, and the signal strength via the SNR value. In our directed test, we test both cases in which the active signal is either a primary or secondary user, and we select a SNR value from $\mathcal{S}$. The selection of the channel location and modulation type depends on the user type of the signal. If the signal is a primary user, then the channel location must be contained within the licensed range defined by the primary user license, and the modulation type must be from $\mathcal{M}$ defined by the primary user license. If $\mathcal{M}$ contains a single modulation type, then the choice is simply that modulation, otherwise we randomly choose from the modulation family. If the signal is a secondary user, then the channel location can be any defined channel in the spectrum, and the modulation type must be from $\mathcal{M}^C$, where $\mathcal{M}^C$ is the modulation type set-complement of $\mathcal{M}$, as defined by the primary user license, such that $\mathcal{M}^C = \{m : m \in \mathcal{M}_T, m \notin \mathcal{M}\}$.

We exhaustively test all configurations of the parameter space $\mathcal{N}_c \times \mathcal{L} \times \mathcal{M}_P \times \mathcal{S}$, which contains 900 parameter configurations. We test each parameter configuration 500 times with a primary user signal active and 500 times with a secondary user signal active. The choice to repeat each configuration test 500 times is to improve our statistical confidence in our evaluation metrics. In each test, we run our spectrum sensing architecture on a wideband signal that is randomly generated based on the parameter configuration. The result of each test is a spectrum cell activity report. From this report and the ground truth, we can generate a spectrum activity confusion matrix for the test. From all of the tests in this evaluation, we create a set of spectrum activity confusion matrices that can be conditioned by L, N_c, and SNR and, we analyze the performance of our spectrum sensing architecture with our evaluation metrics.

The first measure that we analyze is the probability of correctly detecting a spectrum opportunity. We are interested in the behavior of this measure versus three variables of interest: the primary user license location, L, the number of spectrum cells (N_c), and SNR. We tabulate these results for the TYPCR, PLACR, and PWACR in Tables 5.2, 5.3, and 5.4, respectively. In each table, we have a subtable for the three ranges for L, and each subtable lists the probability of correctly detecting a spectrum opportunity as a percentage that can be indexed by SNR and N_c value.

From Table 5.2 we first notice that this probability of detection for the TYPCR for a fixed N_c is fairly constant across the SNR values. When we consider the whole spectrum as the primary user license location in Table 5.2a, the average value and standard deviation is calculated to be $87.9\% \pm 2.7\%$, $91.2\% \pm 1.4\%$, $93.2\% \pm 0.7\%$, and $94.2\% \pm 0.5\%$ for N_c = 8, 16, 32, and 64, respectively. In fact, we see each subtable in Table 5.2 are very similar, and the TYPCR has a consistent performance regardless of the value of the primary user license location ranges tested.

From Table 5.3, we see that the PLACR's probability of correctly detecting spectrum opportunities

	N_c			
SNR	8	16	32	64
0 dB	82.5	88.3	91.8	93.3
5 dB	89.3	91.9	93.7	94.5
10 dB	89.4	91.9	93.5	94.6
15 dB	89.2	92.1	93.7	94.5
20 dB	89.0	91.5	93.5	94.0

(a) $L = [0, N_c)$

	N_c			
SNR	8	16	32	64
0 dB	82.4	88.7	92.1	93.1
5 dB	89.4	91.9	93.8	94.4
10 dB	89.5	91.9	93.6	94.7
15 dB	89.5	92.0	93.8	94.6
20 dB	88.9	91.7	93.5	93.8

(b) $L = [\frac{1}{2}N_c, N_c)$

	N_c			
SNR	8	16	32	64
0 dB	82.4	88.4	91.9	93.3
5 dB	89.2	91.8	93.7	94.6
10 dB	89.4	92.0	93.9	94.5
15 dB	89.3	92.0	93.6	94.3
20 dB	89.0	91.6	93.5	93.9

(c) $L = [\frac{1}{2}N_c, \frac{3}{4}N_c)$

Table 5.2: Probability of correctly detecting spectrum opportunities for TYPCR operating in the three different primary user license locations in this test where the rows of each table correspond to a SNR value and the columns correpsond to the N_c value.

	N_c			
SNR	8	16	32	64
0 dB	82.5	88.3	91.8	93.3
5 dB	89.3	91.9	93.7	94.5
10 dB	89.4	91.9	93.5	94.6
15 dB	89.2	92.1	93.7	94.5
20 dB	89.0	91.5	93.5	94.0

(a) $L = [0, N_c)$

	N_c			
SNR	8	16	32	64
0 dB	91.8	94.4	96.0	96.2
5 dB	95.0	95.8	96.8	96.9
10 dB	95.6	95.9	96.4	97.2
15 dB	95.2	95.8	96.7	97.2
20 dB	94.8	95.9	96.6	96.6

(b) $L = [\frac{1}{2}N_c, N_c)$

	N_c			
SNR	8	16	32	64
0 dB	88.3	91.7	93.6	94.5
5 dB	91.4	93.1	94.6	95.4
10 dB	91.6	93.3	94.9	95.2
15 dB	91.8	93.4	94.6	95.1
20 dB	91.6	93.0	94.5	94.7

(c) $L = [\frac{1}{2}N_c, \frac{3}{4}N_c)$

Table 5.3: Probability of correctly detecting spectrum opportunities for PLACR operating in the three different primary user license locations in this test where the rows of each table correspond to a SNR value and the columns correpsond to the N_c value.

	N_c			
SNR	8	16	32	64
0 dB	99.3	99.4	99.6	99.6
5 dB	99.3	99.5	99.6	99.5
10 dB	99.3	99.4	99.6	99.6
15 dB	99.1	99.5	99.6	99.6
20 dB	99.0	99.3	99.6	99.4

(a) $L = [0, N_c)$

	N_c			
SNR	8	16	32	64
0 dB	99.7	99.6	99.9	99.7
5 dB	99.7	99.8	99.8	99.8
10 dB	99.7	99.7	99.8	99.8
15 dB	99.6	99.8	99.9	99.7
20 dB	99.6	99.7	99.7	99.5

(b) $L = [\frac{1}{2}N_c, N_c)$

	N_c			
SNR	8	16	32	64
0 dB	99.5	99.6	99.7	99.6
5 dB	99.5	99.5	99.7	99.6
10 dB	99.4	99.6	99.7	99.7
15 dB	99.4	99.5	99.7	99.6
20 dB	99.4	99.4	99.6	99.5

(c) $L = [\frac{1}{2}N_c, \frac{3}{4}N_c)$

Table 5.4: Probability of correctly detecting spectrum opportunities for PWACR operating in the three different primary user license locations in this test where the rows of each table correspond to a SNR value and the columns correpsond to the N_c value.

is also consistent for a fixed N_c, and does not depend on the SNR values nor the L range. For the PLACR, we calculate the average value and standard deviation to be $91.2\% \pm 1.8\%$, $93.2\% \pm 0.9\%$, $94.7\% \pm 0.4\%$, and $95.3\% \pm 0.4\%$ for $N_c = 8$, 16, 32, and 64, respectively.

For the PWACR in Table 5.4, we see a stronger consistency where the PWACR's probability of correctly detecting spectrum opportunities is $99.4\% \pm 0.3\%$ regardless of the N_c value, SNR value, and L range.

The first remark we can make about our spectrum sensing architecture is that the probability of correctly detecting a spectrum opportunity is not sensitive to the spectrum location of the primary user(s) license, regardless of the type of cognitive radio that utilizes the information produced by our architecture. In fact, we found, as we analyzed our test data with the remaining evaluation measures, that all four of our measures maintain a consistent performance that is independent of the L range. This performance independence from the license location is a good quality for our spectrum sensing since the sensing environment is dynamic.

The reported values of the probability of correctly detecting a spectrum opportunity as a function of cognitive radio type matches our intuition. As a cognitive radio uses more of the information provided by our spectrum sensing architecture, then we see this probability increase: 94.2% for TYPCR ($N_c = 64$), 95.3% for PLACR ($N_c = 64$), and 99.4% for PWACR.

Finally, the increase in probability of correctly detecting a spectrum opportunity as we increase the number of spectrum cells also matches our intuition. A larger N_c value implies a better frequency resolution, which allows us to better estimate the bandwidth of a signal and make a better spectrum cell activity report.

The second measure that we analyze is the probability of correctly detecting a "grey-space" spectrum opportunity. Recall that this evaluation metric is defined as the ratio of spectrum cells available to the cognitive radio from the spectrum cells in which a non-primary user is active. By this definition, the TYPCR type of cognitive radio never detects a "grey-space" spectrum opportunity, and we analyze this measure with respect to the PLACR and PWACR types.

In Table 5.5, we see that the probability of correctly detecting a "grey-space" spectrum opportunity is consistent for both the PLACR and PWACR, and independent of the L, N_c, and SNR values. The calculated average and standard deviation for this metric is $21.8\% \pm 2.5\%$ for the PLACR and $93.7\% \pm 1.3\%$ for the PWACR. We immediately see a dramatic increase in the detection of "grey-space" spectrum opportunities with PWACR type of cognitive radio. This result is further evidence that a cognitive radio should utilize all the information provided by our spectrum sensing architecture, as is the case with PWACR, in order to maximize the radio's spectrum access for data transmission with a high throughput.

The third measure that we analyze is the probability of primary user false alarm. We found that this measure is independent of the L and N_c values. We plot this measure for each of the cognitive

	N_c			
SNR	8	16	32	64
0 dB	25.2	21.9	20.0	19.2
5 dB	24.4	21.5	20.5	20.2
10 dB	26.4	21.6	20.1	19.7
15 dB	25.7	21.9	20.3	20.5
20 dB	25.4	22.6	20.8	20.3

(a) PLACR

	N_c			
SNR	8	16	32	64
0 dB	95.5	94.6	95.8	94.6
5 dB	95.1	95.1	95.1	95.3
10 dB	94.6	94.6	95.4	95.2
15 dB	94.2	94.9	95.4	95.3
20 dB	93.4	93.7	94.4	95.3

(b) PWACR

Table 5.5: Probability of correctly detecting "grey-space" spectrum opportunities where the rows of each table correspond to a SNR value and the columns correpsond to the N_c value.

radio types versus the SNR value, as shown in Figure 5.10 where the vertical axis is the probability of primary user false alarm, the horizontal axis is the SNR value, and the red, blue, and green line corresponds with the evaluation of TYPCR, PLACR, PWACR, respectively.

From Figure 5.10, we see that the PWACR has a significantly lower probability of primary user false alarm than compared to both the TYPCR and the PLACR. The PWACR has a false alarm rate under 10% for SNR values 10 dB and greater, whereas the TYPCR is near 50% and the PLACR is near 43% over the same range of SNR values. We can intuitively explain and compare the three cognitive radio behaviors in the figure.

The difference between TYPCR and PLACR (red versus blue) is due to way the cognitive radio handles a signal detection that occurs outside of a primary user licensed spectrum. Recall that the TYPCR asserts any signal detection as a primary user, but the PLACR asserts any signal detection inside the licensed spectrum as a primary user. Thus, when a detection occurs outside this licensed spectrum, the TYPCR incorrectly make a primary user assertion whereas the PLACR does not. This difference leads to the higher probability of primary false alarm for the TYPCR compared to the PLACR.

The difference between PLACR and PWACR (blue versus green) is due to way the cognitive radio handles a signal detection that occurs inside a primary user licensed spectrum. Again, in this region of spectrum, the PLACR asserts any signal detection as a primary user, but the PWACR checks the modulation type with the known possible modulations used by a primary user. If there is a match in modulation type, then the PWACR asserts the signal detection as a primary user. Thus, any secondary user operating inside the licensed spectrum of a primary user causes a higher probability of primary user false alarm for the PLACR compared to the PWACR.

Again, this result is further evidence that a cognitive radio should utilize all the information provided by our spectrum sensing architecture, as is the case with PWACR, in order to maximize the radio's spectrum access for data transmission with a high throughput.

The fourth measure that we analyze is the probability of primary user detection. We desire this

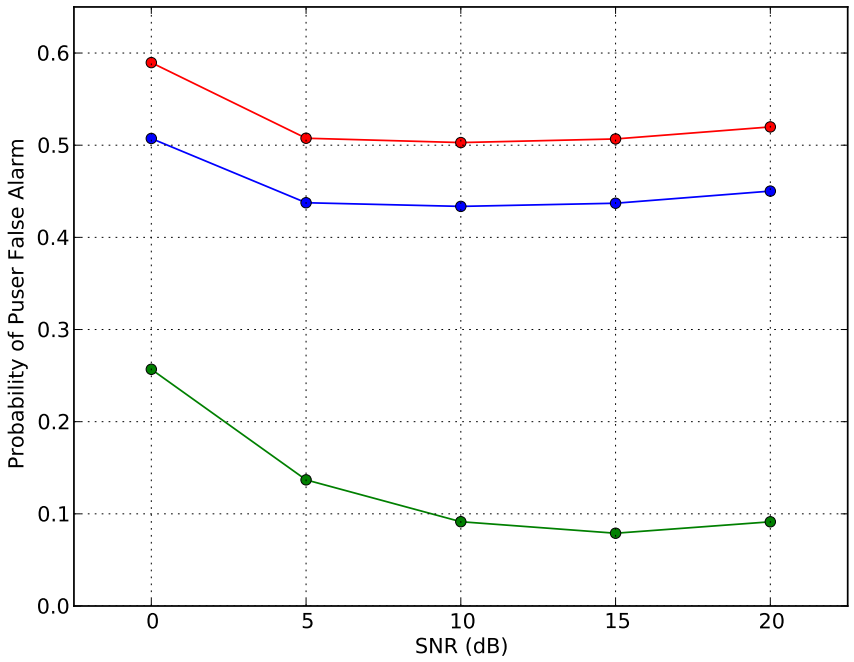


Figure 5.10: Probability of primary user false alarm versus SNR value where the red, blue, and green lines correspond with TYPCR, PLACR, and PWACR, respectively.

measure to have a value near 100% because a high probability of detection prevents a cognitive radio from interfering with the primary user(s). Similar to the probability of primary user false alarm, we found that this measure is independent of the L and N_c values. We plot this measure for each of the cognitive radio types versus the SNR value, as shown in Figure 5.11 where the vertical axis is the probability of primary user detection, the horizontal axis is the SNR value, the constant blue line corresponds to both TYPCR and PLACR, and the green bars represent the probability of detection for the PWACR.

We notice that TYPCR and PLACR have the same probability of primary user detection that is constant at $99.3\% \pm 1.0\%$ across our range of SNR values. Recall from Figure 5.9 that this measure is calculated from the `P` row of the spectrum activity confusion matrix. For the TYPCR this measure is proportional to the sum of confusion matrix values in indices `PA`, `PS`, and `PP`, and for the PLACR this measure is proportional to the sum of confusion matrix values in indices `PS` and `PP`. However, recall from Section 5.2.1 that the confusion matrix value at index `PA` is always 0. Thus, this measure is the same for both TYPCR and PLACR.

More importantly, we see that the probability of primary user detection for the PWACR is significantly lower that 99.3%. This reduction in performance can result in higher interference with the primary user by the cognitive radio because the radio believes that the spectrum is available when a primary user is actively transmitting. To understand this degradation in performance, let us understand further the differences between the PWACR and PLACR with respect to this evaluation

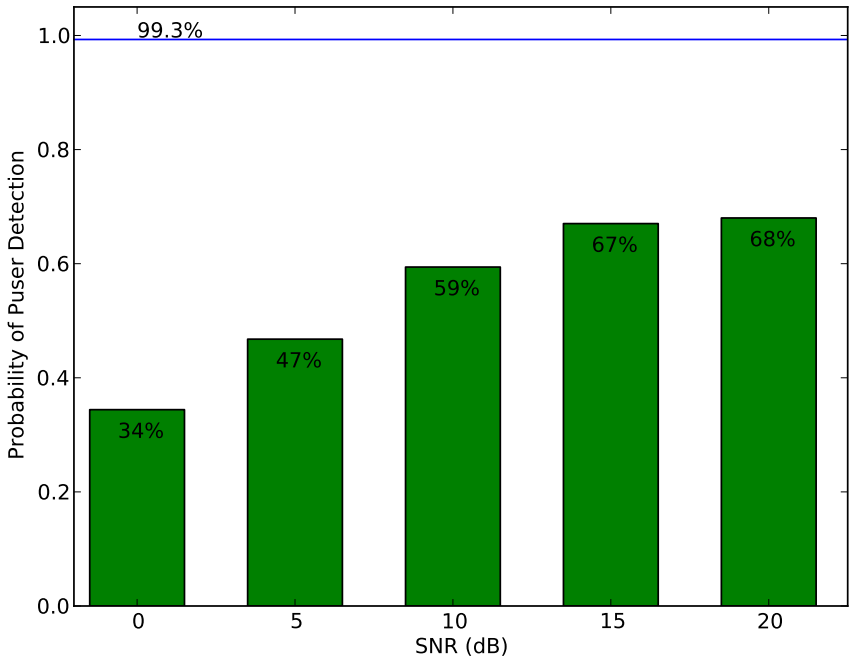


Figure 5.11: Probability of primary user detection versus SNR value where the constant blue line corresponds to the probability of detection for both TYPCR and PLACR and the green bars indicate the probability of detection for PWACR.

measure.

For our spectrum sensing architecture to label a spectrum cell as a primary user, three events must occur. First, this spectrum cell must be inside the range of spectrum cells licensed to a primary user. Second, the architecture must detect a signal present that that contains this spectrum cell. Third, the architecture must classify the signal's modulation to type that is a known possible type employed by a primary user. This third event is responsible for the difference in probability of primary detection between PWACR and PLACR.

Let E be the (third) event where the spectrum sensing architecture classifies the modulation of a signal detected inside the range of licensed spectrum cells to match a possible modulation type employed by a primary user. We mathematically define E, as shown in Equation 5.6, where $\mathcal{M}$ is the set of modulations that our architecture can classify, x is the detected signal, $\beta(x, m)$ is the modulation classification algorithm's calculated probability that the signal x is modulated by the m modulation type, and $\mathcal{M}_P$ is the set of possible modulation types that a primary user can use.

$$E = \begin{cases} 1, & \arg\max_{m \in \mathcal{M}} \beta(x, m) \in \mathcal{M}_P \\ 0, & \text{otherwise} \end{cases} \tag{5.6}$$

Recall that for the PWACR this measure is proportional to the confusion matrix value at index

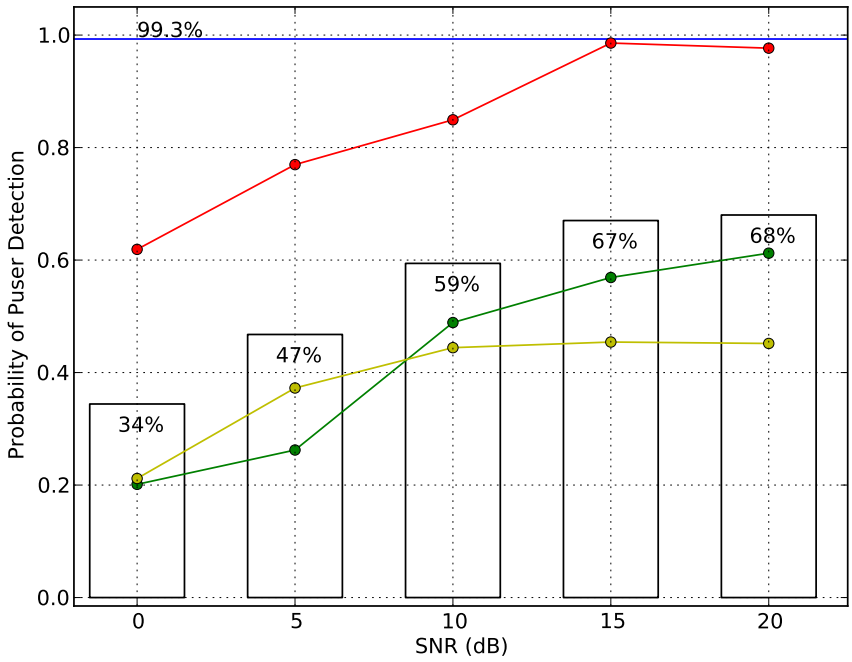


Figure 5.12: Probability of primary user detection versus SNR value where the red, green, and yellow lines correspond to conditioning the primary user modulation type to the PSK, PAM, and QAM modulation family, respectively, and where the constant blue line corresponds to the probability of detection for PLACR and the black-outline bars indicate the probability of detection for PWACR, which is reshown from Figure 5.11.

PP and, as we previously mentioned, for the PLACR this measure is proportional to the sum of confusion matrix values in indices PS and PP. The first and second events are prerequisites for us to increment the count when tabulating the spectrum activity confusion matrix in either index PS or PP. We increment the confusion matrix at index PP if E equals 1. Otherwise we increment the confusion matrix at index PS, which occurs when the modulation classification algorithm incorrectly labels the modulation type of the detected primary user signal.

As a result, the probability of primary user detection for PWACR is related to the probability that E equals 1, which is further related to the correct classification accuracy of the modulation classification algorithm used by the spectrum sensing architecture. Therefore, the degradation in this evaluation measure for PWACR compared to PLACR is due to the modulation classification inaccuracy.

This conclusion gives us another avenue to further investigate. In Figure 5.12, we again show the results from Figure 5.11 along with presenting the conditional probability of primary user detection for PWACR where the red, green, and yellow line corresponds to conditioning the primary user modulation type to the PSK, PAM, and QAM modulation family, respectively. We clearly can see that the detection of the primary user improves when the primary user modulates its data with

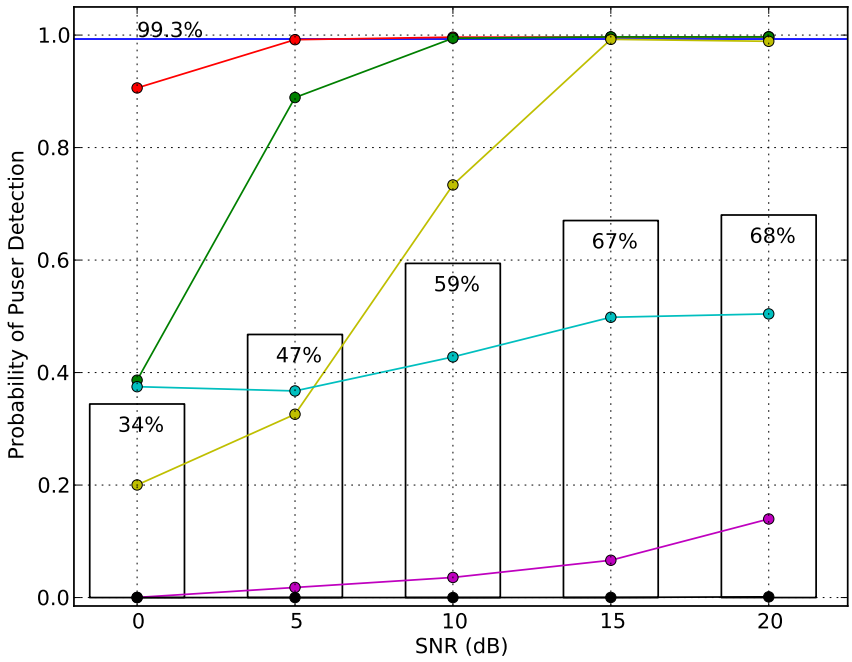


Figure 5.13: Probability of primary user detection versus SNR value where the red, green, yellow, cyan, magenta, and black lines correspond to conditioning the primary user modulation type to ones that have 1, 2, ..., 6 bits-per-symbol, respectively, and where the constant blue line corresponds to the probability of detection for PLACR and the black-outline bars indicate the probability of detection for PWACR, which is reshown from Figure 5.11.

a modulation in the PSK family. However, this conditioning still underperforms the PLACR's performance value of 99.3%.

Rather than conditioning on the modulation family, we consider the performance of this evaluation measure when we condition on the number of bits-per-symbol of the modulation type. Understanding the performance behavior with this type of conditioning is justified because a modulation type with more bits-per-symbol requires a higher SNR value to achieve the same data transmission bit error rate than compared to a modulation type with fewer bits-per-symbol [84, 86, 87, 170–172].

In Figure 5.13, we again show the results from Figure 5.11 along with presenting the conditional probability of primary user detection for PWACR where the red, green, yellow, cyan, magenta, and black line corresponds to conditioning the primary user modulation type to ones that have 1, 2, ..., 6 bits-per-symbol, respectively. In this figure, we see a more favorable results for the PWACR.

For the red-line, which is 1 bit-per-symbol and corresponds to the BPSK modulation, the probability values are 90.6%, 99.2%, 99.6%, 99.6%, and 99.7% for the SNR values of 0 dB, 5 dB, 10 dB, 15 dB, and 20 dB, respectively. If the primary user transmits with a BPSK modulation, then the PWACR performs reasonable well in comparison with PLACR.

For the green-line, which is 2 bits-per-symbol and corresponds to the QPSK and 4-PAM modula-

tions, the probability values are 38.6%, 88.9%, 99.4%, 99.7%, and 99.7% for the SNR values of 0 dB, 5 dB, 10 dB, 15 dB, and 20 dB, respectively. If the primary user transmits with a QPSK or 4-PAM modulation, then the PWACR performs comparably with PLACR for SNR values of 10 dB or greater.

For the yellow-line, which is 3 bits-per-symbol and corresponds to the 8-PSK, 8-PAM, and 8-QAM modulations, the probability values are 20.0%, 32.6%, 73.4%, 99.2%, and 98.9% for the SNR values of 0 dB, 5 dB, 10 dB, 15 dB, and 20 dB, respectively. If the primary user transmits with one of these 3 bits-per-symbol modulation, then the PWACR performs comparably with PLACR for SNR values of 15 dB or 20 dB.

For any of the modulations that have 4 or greater bits-per-symbol, the probability of primary user detection for PWACR remains significantly lower than the PLACR over the range of SNR values that we tested. In reality, a communication system that expects to use these types of modulations requires a channel with a greater SNR value to reliably communicate, so we might not be discouraged by the performance of the PWACR. However, we cannot completely ignore this issue due to the fact that fading and shadowing environments also occur in reality. These environments can create a situation where the primary user does have a good channel to facilitate the use of a modulation type with a large number of bits-per-symbol, but our spectrum sensing architecture is located such that it experiences a faded or shadowed signal and yet the cognitive radio is still expected to avoid this hidden primary user.

Although the performance in detecting the primary user is degraded due to the modulation classification inaccuracy, when we consider all four evaluation measures on this directed test, we observe that the extra information provided by our spectrum sensing architecture enhances the ability of a cognitive radio to detect when to access the spectrum.

When we analyzed our test data with our evaluation measures, we noticed that all four of our measures maintain a consistent performance that is independent of the range of spectrum cells contained in the primary user's license. This performance independence from the license location is a good quality for our spectrum sensing since the sensing environment is dynamic.

The probability of correctly detecting a spectrum opportunity was 99.4% for PWACR. This value outperforms the other two types of cognitive, which we saw was 94.2% for TYPCR and 95.3% for PLACR. The probability of correctly detecting a "grey-space" spectrum opportunity was significantly greater for PWACR compared to PLACR, which was calculated to be 93.7% for the PWACR versus 21.8% for the PLACR. These two results provides evidence that a cognitive radio should utilize all the information provided by our spectrum sensing architecture, as is the case with PWACR, in order to maximize the radio's spectrum access for high data transmission throughput.

The probability of primary user false alarm was significantly lower for PWACR than compared to both the TYPCR and the PLACR. The PWACR has a false alarm rate under 10% for SNR values

10 dB and greater, whereas the TYPCR is near 50% and the PLACR is near 43% over the same range of SNR values. Unfortunately, the probability of primary user detection for the PWACR is significantly lower than 99.3%, which is the performance value of both the TYPCR and the PLACR.

We saw that if we conditioned the probability to when the primary user modulates its data with a modulation in the PSK family, then the performance improves, but still underperforms the PLACR's performance value of 99.3%. However, if we conditioned the probability on the number of bits-per-symbol of the modulation used by the primary user, in particular when there are less than 4 bits-per-symbol, then the probability of primary user detection for the PWACR is comparable to that of the PLACR.

In the next evaluation, we test our spectrum sensing architecture with many random configurations to generate a variety of input wideband signals. This test exercises our architecture with multiple signals and allows us to compare the results with this evaluation that only had one signal in each test.

5.2.3 Random Test Evaluation

The second evaluation is a series of tests that randomly selects configurations from the configuration parameter space. These tests intend to show that our spectrum sensing architecture performs well when operating in a variety of different signal environments that contain a mixture of primary users and secondary users. The configuration parameter space is the same as the one described in Section 5.2.2.

To specify a random test configuration, we must select a number of items: the number of spectrum cells, N_c, the number of primary user licenses, N_L, the total number of signals, M, and the number of primary user signals, M_p. The number of secondary user signals, M_s, is the difference $M - M_p$. We randomly choose N_c from the set $\mathcal{N}_c$ with an uniform distribution where $\mathcal{N}_c$ was previously defined in Table 5.1. We randomly choose N_L from the set $\{1, 2, 3\}$ where we have a $Pr\{\ N_L = i\ \}$ equal to 90%, 9%, and 1% for $N_L = 1$, 2, and 3, respectively. We randomly choose M from the set $\{1, 2, 3\}$ with an uniform distribution. Finally, we randomly choose M_p from the set $\{0, 1, ..., \xi\}$ with an uniform distribution where ξ is the minimum value of M and N_L, which guarantees that at most a single primary user is active in each of the primary user licenses.

For the i^{th} primary license, for $i = 1, ..., N_L$, we must define the primary user licensed spectrum cells, L_i, and the primary user modulation list, $\mathcal{M}_i$. We randomly select a contiguous block of spectrum cells between 0 to $N_c - 1$ to assign to each L_i for $i = 1, .., .N_L$ where we ensure that each set of licensed spectrum cells do not overlap, i.e. $L_i \cap L_j = \emptyset$ for all pairs i, j where $i = 1, ..., N_L$, $j = 1, ..., N_L$, $i \neq j$. We randomly choose $\mathcal{M}_i$ from the set $\mathcal{M}_P$ with an uniform distribution where $\mathcal{M}_P$ was previously defined in Table 5.1.

After we define the N_L primary licenses, we characterize the M_p primary users that are active. We randomly create an one-to-one mapping between the M_p primary users, and the N_L primary user licenses. The mapping is randomly generated because $M_p \leq N_L$ and we do not want to only use the first M_p licenses. Let `PUL` be a function that returns the primary user license index for the input primary user index. For example, suppose $M_p = 2$ and $N_L = 3$, then a possible mapping is $\{0 \rightarrow 2, 1 \rightarrow 0\}$, which implies that the 0^{th} primary user occupies the 2^{nd} primary license and 1^{st} primary user occupies the 0^{th} primary license. Further, with this example, the function $\texttt{PUL}(0)$ returns 2 and the function $\texttt{PUL}(1)$ returns 0. For the m^{th} primary user, for $m = 1, ..., M_p$, let $i_m = \texttt{PUL}(m)$. We randomly choose a channel that is contained within the primary license spectrum cells set L_{i_m}. The modulation type is defined by the set $\mathcal{M}_{i_m}$ in the primary license. If $\mathcal{M}_{i_m}$ contains a single modulation type, then the choice is simply that modulation, otherwise we randomly choose from the modulation family with an uniform distribution. Finally, we randomly choose a SNR value from the set $\mathcal{S}$ with an uniform distribution where $\mathcal{S}$ was previously defined in Table 5.1.

After we define the M_P primary users, we characterize the M_s secondary users that are active. For the s^{th} secondary user, for $s = 1, ..., M_s$, we randomly choose a channel from the complete range of spectrum cells as long as the channel selected does not interfere with a primary user. The channel selection determines the set of possible modulation types we allow the secondary user to employ. If any spectrum cells in the selected channel overlaps with the i^{th} primary license's spectrum cells for any $i = 1, ..., N_L$, then the modulation type is randomly chosen from $\mathcal{M}_i^C$ with an uniform distribution. We define $\mathcal{M}_i^C$ as the modulation type set-complement of $\mathcal{M}_i$ such that $\mathcal{M}_i^C = \{m : m \in \mathcal{M}_T, m \notin \mathcal{M}_i\}$ where $\mathcal{M}_i$ is defined by the primary license and $\mathcal{M}_T$ is previously defined in Table 5.1. Otherwise, if all of the spectrum cells in the secondary user's selected channel do not overlap with any of the primary licenses' spectrum cells, then the modulation type is randomly chosen from $\mathcal{M}_T$ with an uniform distribution. Finally, we randomly choose a SNR value from the set $\mathcal{S}$ with an uniform distribution.

We complete a Monte Carlo sampling of this configuration parameter space by performing 1,000,000 test runs. Similar to the first evaluation, the result of each random test is a spectrum cell activity report. From this report and the ground truth, we can generate a spectrum activity confusion matrix for the test. Note that in the random test evaluation we cannot simply report our finding versus the SNR value since the input wideband signal can contain multiple narrowband signals, which each of these narrowband signals may differ in SNR value. Instead of enumerating all of the SNR combinations that could occur, we sum all of the test results together conditioned on the number of spectrum cells, as shown for reference in Table 5.6. We now perform an analysis similar to the first evaluation.

The first measure that we analyze is the probability of correctly detecting a spectrum opportunity. We found this measure to have the value $88.5\% \pm 3.8\%$ for TYPCR, $95.1\% \pm 1.0\%$ for PLACR, and

	O	A	S	P
O	1392923	36927	23016	4126
A	39260	182819	0	0
S	17592	112	83159	7011
P	33266	0	85695	97574

(a) $N_c = 8$

	O	A	S	P
O	3135188	58808	37696	6504
A	50671	262666	6	6
S	23255	232	131130	10608
P	42849	0	103773	135872

(b) $N_c = 16$

	O	A	S	P
O	6578384	107584	71266	11696
A	76149	443688	89	5
S	40595	393	242693	18554
P	55919	0	151799	205282

(c) $N_c = 32$

	O	A	S	P
O	13321305	232636	178609	31727
A	114178	821899	487	71
S	70984	875	477743	37789
P	76908	0	260701	341000

(d) $N_c = 64$

Table 5.6: Sum total spectrum activity confusion matrix for the random test evaluation of our spectrum sensing architecture conditioned on the N_c value from which we calculate our evaluation measures.

	N_c					
CR	8	16	32	64	Mean	Std. Dev.
PLACR	67.3%	65.5%	63.2%	61.4%	64.4%	2.2%
PWACR	97.9%	97.8%	97.7%	97.5%	97.7%	0.1%

Table 5.7: Probability of correctly detecting "grey-space" spectrum opportunities where the rows of each table correspond to a cognitive radio type, the columns correspond to the N_c value, and the final two columns are the mean and standard deviation average over the N_c values.

$99.5\% \pm 0.1\%$ for PWACR. Theses results are consistent with the results from the first evaluation, with the exception of a decrease probability for TYPCR. This decrease is most likely due to allowing more than one signals to be present in the received wideband signal.

The second measure that we analyze is the probability of correctly detecting a "grey-space" spectrum opportunity. In Table 5.7, we show this probability as a percentage for both PLACR and PWACR where we have a column for each of the four possible N_c values, and we provide the mean and standard deviation values for both cognitive radio types that are averaged over the N_c values. We see that the probability of correctly detecting a "grey-space" spectrum opportunity is consistent for both the PLACR and PWACR. The calculated average and standard deviation for this metric is $64.4\% \pm 2.2\%$ for the PLACR and $97.7\% \pm 0.1\%$ for the PWACR. We see the increase in the detection of "grey-space" spectrum opportunities with PWACR type of cognitive radio, which is consistent with the results from the first evaluation, however the difference is not as dramatic in this evaluation.

The third measure that we analyze is the probability of primary user false alarm, which we measured

to be $69.6\% \pm 5.0\%$ for TYPCR, $50.3\% \pm 7.5\%$ for PLACR, and $16.0\% \pm 4.2\%$ for PWACR. These results show that the PWACR has a significantly lower probability of false alarm than both the PLACR and TYPCR. The pattern is also consistent with the results in the first evaluation.

The fourth measure that we analyze is the probability of primary user detection. In this test, we saw that TYPCR and PLACR have a probability of primary user detection that is constant at $95.7\% \pm 5.6\%$, which is slightly lower than the value found in the first evaluation, but this is most likely due to the more challenging random configurations in this evaluation. We found the probability of primary user detection for PWACR to be $53.5\% \pm 3.7\%$, which again in this evaluation is significantly lower than TYPCR and PLACR.

In Figure 5.14, we plot this measure for each of the cognitive radio types along with the performance of PWACR conditioned on the primary user modulation type, where the vertical axis is the probability of primary user detection, the solid blue line corresponds probability of detection for both TYPCR and PLACR, the dashed blue line corresponds probability of detection for PWACR, and the bars correspond to the probability of detection for PWACR conditioned on a particular modulation type where we colored the bars according to modulation family: red for PSK modulation, green for PAM modulation, and yellow for QAM modulation.

Similar to the first evaluation, we see in Figure 5.14 that the detection of the primary user improves when the primary user modulates its data with a modulation in the PSK family: 69.2% on average for the PSK family compared to 35.8% and 37.6% on average for the PAM and QAM families, respectively. However, this conditioning for PWACR still underperforms the PLACR's performance value of 95.7%.

We compute the probability of primary user detection conditioned on the number of bits-per-symbol, 82.4%, 72.5%, 52.5%, 48.1%, 12.6%, and 15.4% for 1, 2, ..., 6 bits-per-symbol, respectively. These results are consistent with the results found in the first evaluation. The PWACR does a better job detecting the primary user when that user modulates its signal with a modulation type that has a lower number of bits-per-symbol.

In the this second evaluation, we saw that the performance in detecting the primary user is degraded due to the modulation classification inaccuracy, but again when we consider all four evaluation measures on this test, we still are encouraged that the extra information provided by our spectrum sensing architecture enhances the ability of a cognitive radio to detect when to access the spectrum.

The probability of correctly detecting a spectrum opportunity was 99.5% for PWACR. This value outperforms the other two types of cognitive, which we saw was 88.5% for TYPCR and 95.1% for PLACR. The probability of correctly detecting a "grey-space" spectrum opportunity was significantly greater for PWACR compared to PLACR, which was calculated to be 97.7% for the PWACR versus 64.4% for the PLACR. These two results provide evidence that a cognitive radio should utilize all the information provided by our spectrum sensing architecture, as is the case with

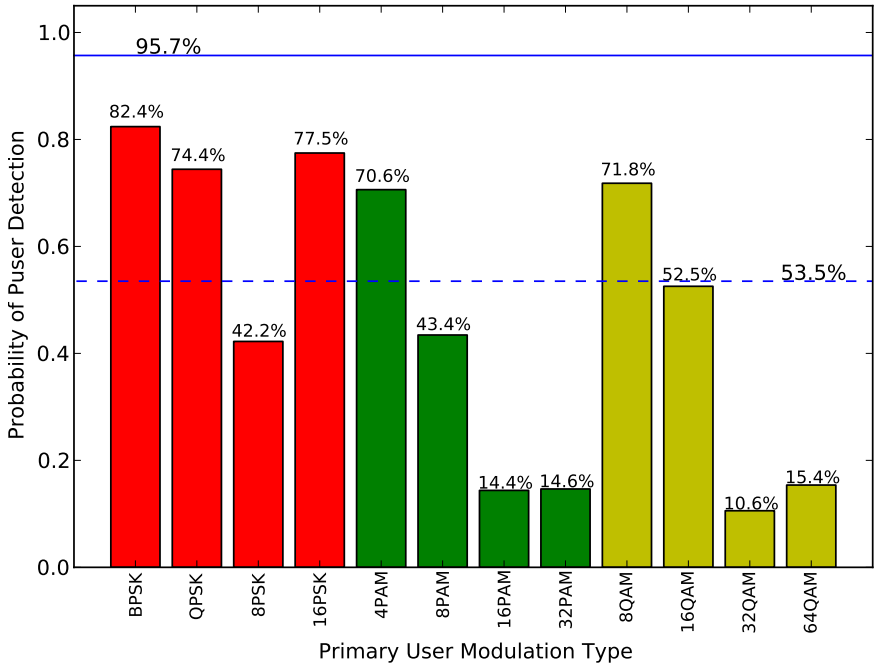


Figure 5.14: Probability of primary user detection in the random test evaluation where the solid blue line corresponds to the probability of detection for both TYPCR and PLACR, the dashed blue line corresponds probability of detection for PWACR, and the bars correspond to the probability of detection for PWACR conditioned on a particular modulation type where we colored the bars according to modulation family: red for PSK modulation, green for PAM modulation, and yellow for QAM modulation.

PWACR, in order to maximize the radio's spectrum access for high data transmission throughput.

The probability of primary user false alarm was significantly lower for PWACR than compared to the rate for both TYPCR and PLACR. The PWACR has an average false alarm rate of 16%, whereas the rate for TYPCR was over 69% and the rate for PLACR was near 50% over the same range of SNR values. Unfortunately, the probability of primary user detection for the PWACR is significantly lower than 95.7%, which is the performance value of both the TYPCR and the PLACR. The performance of the PWACR improves if we condition the probability on subsets of modulation types that the primary user can use.

5.3 Discussion and Future Work

We performed two types of evaluations to test our spectrum sensing architecture to determine if the extra information contained in our spectrum cell activity reports can improve the performance of a cognitive radio. The first evaluation performed a systematic directed test over the configuration parameter space with one signal active. An aspect of this test determined the sensitivity of our architecture to the varying parameters, such as the number of spectrum cells, SNR values, and the range of licensed spectrum cells by the primary user. The second evaluation performed a random sampling of the configuration parameter space with the potential for multiple primary and secondary user signals present in the spectrum. The combination of these two evaluations gave us a wide range of tests to exercise our architecture.

We found that our spectrum sensing architecture is robust to variations in the number of spectrum cells and the location of primary user licenses in the spectrum. This robustness was revealed through the consistency of the evaluation metric values as we changed these variables. We also found that the ability of the architecture increased with SNR, which intuitively makes sense.

We compared three types of cognitive radios that incorporate a different amount of information provided by our spectrum sensing architecture. Recall, the TYPCR is a conventional type of cognitive that employs a narrowband signal detection algorithm to determine if there is a spectrum opportunity. The PLACR has knowledge about the locations that a primary license(s) resides within the spectrum, and PWACR additionally has knowledge about the modulation types that a primary user can use. These latter two types of cognitive radio can make a better judgment about if an active signal is a primary user or secondary user, because they have the additional knowledge of the primary user(s)' waveform.

For three of our evaluation measures, the probability of spectrum opportunity detection, the probability of "grey-space" spectrum opportunity detection, and the probability of primary user false alarm, PWACR significantly outperforms the other two types of cognitive radios. This result is evidence that a cognitive radio should utilize all the information provided by our spectrum sensing

architecture. With this information the cognitive radio can make the best decisions about when to access the spectrum in order to maximize the radio's data transmission throughput.

Overall, when combining the results from both evaluations, the probability of correctly detecting a spectrum opportunity was 99.4% for PWACR. This value outperforms the other two types of cognitive, which we saw was 87.7% for TYPCR and 93.8% for PLACR. The probability of correctly detecting a "grey-space" spectrum opportunity was significantly greater for PWACR compared to PLACR, which was calculated to be 96.1% for PWACR versus 49.1% for PLACR. The probability of primary user false alarm was significantly lower for PWACR than compared to both the TYPCR and PLACR, which we measured to be 62.7% for TYPCR, 46.9% for PLACR, and 13.3% for PWACR.

Unfortunately, the probability of primary user detection for PWACR is 50.4% which is significantly lower than 92.9% for both the TYPCR and PLACR. The performance of PWACR improves if we condition the probability on subsets of modulation types that the primary user can use. However, we feel that this degradation in performance is not enough justification to dismiss all of the information provided by our spectrum sensing architecture. In fact, when we consider the performance of our architecture with respect to all four evaluation measures, we are encouraged that the extra information provided by our architecture enhances the ability of a cognitive radio to detect when to access the spectrum.

The classification accuracy of our modulation classification algorithm dictates which spectrum cells are correctly labeled as a primary user, and also this accuracy directly influences the cognitive radio's probability of detecting a primary user. We have three ideas for future work to improve the performance of our spectrum sensing architecture: 1. Improve the classification accuracy of the modulation classification algorithm, 2. Integrate the performance behavior of the modulation classification algorithm into the spectrum cell labeling, and 3. Incorporate a probabilistic approach to the spectrum cell labeling from the modulation classification. Any improvement in modulation classification accuracy would directly increase the probability of primary user detection for a cognitive radio.

The modulation classification algorithm was trained on very ideal-case instances where there was a single narrowband, modulated signal that is perfectly reconstructed from a signal processing point-of-view. In the future, we could revisit our derivation of our modulation classification algorithm to incorporate the real-world inaccuracies and channel effects in order to better train and classify the modulation types. The modulation classification algorithm provides a belief vector that the signal x is modulated by the modulation type m for all types of modulation, which is proportional to the $Pr\{$ modulation $m \mid$ signal $x\}$. However, in training and testing our algorithm, we can learn an estimate on the joint probability distribution of the the true modulation type label and the estimated modulation type label. In the future, we could integrate this information into our calculation of $Pr\{$ modulation $m \mid$ signal $x\}$ by marginalizing from the joint distribution the esti-

mated modulation type label from the classification algorithm's belief vector. Finally, our spectrum sensing architecture labels a spectrum cell as a primary (or secondary) user based on the argmax of the belief vector from the modulation classification algorithm. In the future, instead of making this hard decision on the modulation type of x, we could investigate if the performance improves by considering any modulation that produces a belief value than a threshold α where the value of α must also be learned.

Chapter 6

Summary

Spectrum sensing is the process of identifying available channels of both licensed and unlicensed spectrum in order to wirelessly communicate. This process is a challenging problem because the frequency spectrum is a dynamic system that changes with time and location. Additionally, the availability of the channel in licensed spectrum depends on the activity of the primary user. The underutilization of the licensed spectrum presents an opportunity for cognitive radios to access the spectrum via the information provided by spectrum sensing architectures.

In this dissertation, we focused on algorithmic development towards a radio identification spectrum sensing architecture to better identify primary users versus secondary users by further characterizing any detected signal in the spectrum. We assumed that we have knowledge of the location of the licensed spectrum belonging to the primary user, as well as the set of modulation types a primary user may use. We compared and matched the identified characteristics of the unknown signal with the known parameters of a primary user. Our spectrum sensing architecture generates more informative reports about the spectrum availability to a cognitive radio, because of this comparison.

Our research contributed to the improvement of three main components of a spectrum sensing architecture. The first contribution of our research is a narrowband signal detection algorithm which tries to locate all of the narrowband signals in the received wideband signal with higher accuracy at estimating the signal's carrier frequency and bandwidth than the energy detection algorithm, in particular at lower SNR values. The second contribution of our research is a constellation-based digital modulation classification algorithm that exploits the constellation structure with higher classification accuracy than a set of literature comparisons' results, where the most dramatic improvement of 61.4 percentage points occurred at 0 dB SNR. Finally, the third contribution of our research is a spectrum sensing architecture that produces an informative spectrum activity report that can be utilized by a cognitive radio with a higher accuracy of detecting spectrum access opportunities and with a significantly lower primary user false alarm rate.

In Chapter 3, we derived, implemented, and tested a narrowband signal detection algorithm that

jointly estimates the center frequency and bandwidth of individual narrowband signals contained within a received wideband signal. Our algorithm is an iterative approach that operates in the frequency domain where we model the wideband signal as a mixture of $M \geq 0$ sinc functions, for which each sinc function corresponds to a single narrowband signal. In each iteration, we estimated the center frequency, bandwidth, and amplitude of a new narrowband signal to add to our mixture model.

Our narrowband signal detection algorithm has a number of tunable parameters that affect its operation, such as the DFT size, the maximum number of detections allowed, and the number of algorithmic iterations. We showed that our algorithm maintains a consistent probability of signal detection and probability of signal false alarm as we varied these parameter values in our tests. We concluded that this results creates a trade-space between parameter resolution and computation-time complexity without sacrificing performance.

We analytically compared the performance of our narrowband signal detection algorithm with the expected performance of an energy detection algorithm. We found that our algorithm outperformed the energy detection algorithm with respect to both the probability of signal detection and the probability of signal false alarm, in particular at lower SNR values.

In Chapter 4, we presented a new constellation-based digital modulation classification algorithm that exploits knowledge of the shape of digital modulation constellations. We used the Expectation-Maximization algorithm to cluster the received data to a discrete set of cluster points in the complex plane by modeling the problem as a Gaussian mixture model problem. After clustering, our algorithm creates a feature vector that uses a novel feature set that incorporate the knowledge about how a noisy signal should behave given the structure of the constellation set used to modulate the transmitted information. A score value is calculated using a weight vector that is learned by a genetic algorithm in the training process. The classification rule chooses the modulation that produce the smallest score.

We performed a series of experiments to determine the quality of our modulation classification algorithm. Specifically, we first evaluated the performance of our classifier with test instances that operate in varying real-world conditions, such as different SNR values, and receiver inaccuracies. The second evaluation examined the performance of our classification algorithm with respect to the performance of other classification algorithms reported in the literature. The final evaluation directly compared the classification accuracy of an ensemble SVM using our feature set with other classifiers using two popular choices in feature sets.

We saw that our classifier performs significantly better on the simpler task of identifying only the modulation family of the received signal and our results show that our classifier is robust to non-ideal, real-world conditions by performing equally well in these test scenarios. We found that our algorithm increased the classification accuracy by an average of 9.8 percentage points

when compared against six classifiers described in the literature. Additionally, we showed that our algorithm maintains its correct classification accuracy with only a slight degradation as we increase the size of the class label set. We found that the classifier using our constellation-based feature set performed the best over the SNR range between 0 dB and 19 dB compared to the other feature set classifiers that are common in the literature: increased the average accuracy by 13.35 percentage points over the cumulant statistic classifier and by 5.31 percentage points over the temporal statistic classifier.

We concluded that the constellation-based digital modulation classifier that we developed provides an automatic modulation classifier that is robust and scalable to a large set of digital modulations that can be represented by a constellation set. This quality makes the algorithm well-suited for SDR applications such as spectrum sensing.

In Chapter 5, we described our spectrum sensing architecture that incorporates knowledge about the primary user's waveform characteristics in order to generate a more informative spectrum activity report to a cognitive radio. This architecture takes advantage of the narrowband signal detection algorithm and modulation classification algorithm by first detecting narrowband signals from a received wideband signal, and then classifying the modulation on the detected signals that might be a primary user. After this processing, our spectrum sensing architecture creates a spectrum activity report by combining all of the information generated. We performed a series of tests to evaluate our spectrum sensing architecture, which tries to determine the ability of our architecture to provide a cognitive radio with spectrum opportunities and primary user avoidance.

We showed that our approach which incorporates the knowledge of the modulation type used by the primary user improves our reporting of available spectrum. A cognitive radio that utilizes this extra information can outperform other cognitive radios that use less information. We found that the probability of correctly detecting a spectrum opportunity was 99.4% for the cognitive radio that uses all of the information provided by our spectrum sensing architecture which outperforms two other types of cognitive which we saw had a measure of 87.7% and 93.8%. The probability of correctly detecting a "grey-space" spectrum opportunity was significantly greater for PWACR compared to PLACR, which was calculated to be 96.1% for PWACR versus 49.1% for PLACR. The probability of primary user false alarm was significantly lower for for the cognitive radio that uses all of the information provided by our architecture which was measured as 13.3% compared to 46.9% and 62.7% for the two other types of cognitive we compared.

Although the performance in detecting the primary user was reduced due to the modulation classification inaccuracy, we feel that this degradation in performance was not enough justification to dismiss all of the information provided by our spectrum sensing architecture. In fact, when we considered the performance of our spectrum sensing architecture with respect to all four evaluation measures, we are encouraged that the extra information provided by our architecture enhances the ability of a cognitive radio to detect when to access the spectrum.

In summary, this dissertation focused on algorithmic development of a radio identification spectrum sensing architecture to better identify primary users versus secondary users by further characterizing any detected signal in the spectrum, which provides a cognitive radio with increased spectrum opportunities to utilize. We developed a novel technique to detect narrowband signals from a received wideband signal and a novel technique to classify constellation-based digital modulations. We created a radio identification spectrum sensing architecture that incorporates these novel techniques to provide more informative spectrum activity reports for use by a cognitive radio. We found that our architecture detected spectrum opportunities at a probability of 99.4% compared to 87.7% and 93.8% for the two comparisons. Our architecture detected "grey-space" opportunities with a probability of 96.1% compared to 49.1%. Also, the false alarm rate was significantly lower for our architecture, 13.3% compared to 46.9% and 62.7% for the two comparisons. Consequently, we conclude that a cognitive radio that is aware of the primary user waveform characteristic can achieve better spectrum utilization by using our spectrum sensing architecture.

Bibliography

[1] J. Mitola, III, "Cognitive radio for flexible mobile multimedia communications," *Mob. Netw. Appl.*, vol. 6, pp. 435–441, September 2001.

[2] M. Dillinger, *Software Defined Radio: Architectures, Systems and Functions*. John Wiley and Sons, 2003.

[3] B. Ramkumar, "Automatic modulation classification for cognitive radios using cyclic feature detection," *Cir. and Sys. Mag.*, vol. 09, pp. 27–45, June 2009.

[4] J. Mitola III, *Cognitive Radio: Model-Based Competence for Software Radios*. PhD thesis, Royal Institute of Technology, 1999.

[5] J. Nuechterlein and P. Weiser, *Digital Crossroads: American Telecommunications Policy in the Internet Age*. MIT Press, 2005.

[6] D. Cabric, M. S. W. Chen, D. A. Sobel, S. Wang, J. Yang, and R. W. Brodersen, "Novel radio architectures for uwb, 60 ghz, and cognitive wireless systems," *EURASIP J. Wirel. Commun. Netw.*, vol. 2006, pp. 22–22, April 2006.

[7] M. McHenry, "Nsf spectrum occupancy measurements projects summary," tech. rep., Shared Spectrum Company, August 2005.

[8] M. McHenry and D. McCloskey, "Multi-band multi-location spectrum occupancy measurements," in *Proc. of International Symposium on Advanced Radio Technologies (ISART)*, (Boulder, CO), March 2006.

[9] M. McHenry, P. Tenhula, and D. McCloskey, "Chicago spectrum occupancy measurements and analysis and a long-term studies proposal," in *Proc. of Workshop on Technology and Policy for Accessing Spectrum (TAPAS)*, (Boston, MA), August 2006.

[10] K. Kim, I. A. Akbar, K. K. Bae, J.-S. Um, C. M. Spooner, and J. H. Reed, "Cyclostationary Approaches to Signal Detection and Classification in Cognitive Radio," in *New Frontiers in Dynamic Spectrum Access Networks, 2007. DySPAN 2007. 2nd IEEE International Symposium on*, pp. 212–215, Apr. 2007.

[11] F. C. Commission, "Spectrum policy task force report," washingtion: *ET Docket* 2-135, 2002.

[12] F. 03-322, *NPRM - Facilitating Opportunities for Flexible, Efficient, and Reliable Spectrum Use Employing Cognitive Radio Technologies.* Federal Communication Commission, Dec. 2003.

[13] A. Sahai, R. Tandra, S. M. Mishra, and N. Hoven, "Fundamental design tradeoffs in cognitive radio systems," in *Proceedings of the first international workshop on Technology and policy for accessing spectrum*, TAPAS '06, (New York, NY, USA), ACM, 2006.

[14] J. Unnikrishnan and V. Veeravalli, "Cooperative spectrum sensing and detection for cognitive radio," in *Global Telecommunications Conference, 2007. GLOBECOM '07. IEEE*, pp. 2972–2976, nov. 2007.

[15] S. Hussain and X. Fernando, "Spectrum sensing in cognitive radio networks: Up-to-date techniques and future challenges," in *Science and Technology for Humanity (TIC-STH), 2009 IEEE Toronto International Conference*, pp. 736 –741, sept. 2009.

[16] H. Li, C. Li, and H. Dai, "Quickest spectrum sensing in cognitive radio," in *Information Sciences and Systems, 2008. CISS 2008. 42nd Annual Conference on*, pp. 203–208, march 2008.

[17] A. Bleicher, "Peaceful coexistence," *Spectrum, IEEE*, vol. 50, no. 4, pp. 42–56, 2013.

[18] G. Ding, Q. Wu, Y.-D. Yao, J. Wang, and Y. Chen, "Kernel-based learning for statistical signal processing in cognitive radio networks: Theoretical foundations, example applications, and future directions," *Signal Processing Magazine, IEEE*, vol. 30, no. 4, pp. 126–136, 2013.

[19] T. Yucek and H. Arslan, "A survey of spectrum sensing algorithms for cognitive radio applications," *IEEE Commun. Surveys & Tutorials*, vol. 11, pp. 116–130, Mar. 2009.

[20] B. Zayen, A. Hayar, and K. Kansanen, "Blind spectrum sensing for cognitive radio based on signal space dimension estimation," in *Communications, 2009. ICC '09. IEEE International Conference on*, pp. 1–5, june 2009.

[21] S. Xu, Z. Zhao, and J. Shang, "Spectrum sensing based on cyclostationarity," in *Power Electronics and Intelligent Transportation System, 2008. PEITS '08. Workshop on*, pp. 171–174, aug. 2008.

[22] V. Turunen, M. Kosunen, A. Huttunen, S. Kallioinen, P. Ikonen, A. Parssinen, and J. Ryynanen, "Implementation of cyclostationary feature detector for cognitive radios," in *Cognitive Radio Oriented Wireless Networks and Communications, 2009. CROWNCOM '09. 4th International Conference on*, pp. 1–4, june 2009.

[23] D. Noguet, L. Biard, and M. Laugeois, "Cyclostationarity detectors for cognitive radio: architectural tradeoffs," *EURASIP J. Wirel. Commun. Netw.*, vol. 2010, pp. 5:1–5:8, January 2010.

[24] Y. Zeng, Y.-C. Liang, A. T. Hoang, and R. Zhang, "A review on spectrum sensing for cognitive radio: challenges and solutions," *EURASIP J. Adv. Signal Process*, vol. 2010, pp. 2:2–2:2, Jan. 2010.

[25] I. F. Akyildiz, B. F. Lo, and R. Balakrishnan, "Cooperative spectrum sensing in cognitive radio networks: A survey," *Phys. Commun.*, vol. 4, pp. 40–62, Mar. 2011.

[26] H. Sun, A. Nallanathan, C.-X. Wang, and Y. Chen, "Wideband spectrum sensing for cognitive radio networks: a survey," *Wireless Communications, IEEE*, vol. 20, no. 2, pp. 74–81, 2013.

[27] P. Paysarvi Hoseini and N. Beaulieu, "An optimal algorithm for wideband spectrum sensing in cognitive radio systems," in *Communications (ICC), 2010 IEEE International Conference on*, pp. 1–6, 2010.

[28] E. Axell, G. Leus, E. Larsson, and H. Poor, "Spectrum sensing for cognitive radio : State-of-the-art and recent advances," *Signal Processing Magazine, IEEE*, vol. 29, no. 3, pp. 101–116, 2012.

[29] Y. Zeng and Y.-C. Liang, "Eigenvalue-based spectrum sensing algorithms for cognitive radio," *Communications, IEEE Transactions on*, vol. 57, no. 6, pp. 1784–1793, 2009.

[30] D. Ariananda, M. K. Lakshmanan, and H. Nikookar, "A survey on spectrum sensing techniques for cognitive radio," in *Cognitive Radio and Advanced Spectrum Management, 2009. CogART 2009. Second International Workshop on*, pp. 74–79, 2009.

[31] A. Taherpour, S. Gazor, and A. Taherpour, "Adaptive spectrum sensing and learning in cognitive radio networks," in *18th European Signal Processing Conference (EUSIPCO-2010)*, vol. 4, pp. 860–864, 2010.

[32] R. Chen and J.-M. Park, "Ensuring Trustworthy Spectrum Sensing in Cognitive Radio Networks," in *Networking Technol. for Softw. Defined Radio Networks*, pp. 110–119, Sept. 2006.

[33] M. Rashidi, K. Haghighi, A. Owrang, and M. Viberg, "A wideband spectrum sensing method for cognitive radio using sub-nyquist sampling," *CoRR*, vol. abs/1010.2157, 2010.

[34] K. Chowdhury and T. Melodia, "Platforms and testbeds for experimental evaluation of cognitive ad hoc networks," *Communications Magazine, IEEE*, vol. 48, no. 9, pp. 96–104, 2010.

[35] Z. Han and H. Jiang, "Replacement of spectrum sensing and avoidance of hidden terminal for cognitive radio," in *WCNC*, pp. 1448–1452, 2008.

[36] B.-L. To, M. Otmani, T. M. T. Nguyen, A. Fladenmuller, and G. Pujolle, "Decentralized cooperative spectrum sensing for cognitive radio ad-hoc network: Hidden terminal awareness approach," in *Information Science, Signal Processing and their Applications (ISSPA), 2012 11th International Conference on*, pp. 146–151, 2012.

[37] R. Balamurthi, H. Joshi, C. Nguyen, A. Sadek, S. Shellhammer, and C. Shen, "A tv white space spectrum sensing prototype," in *New Frontiers in Dynamic Spectrum Access Networks (DySPAN), 2011 IEEE Symposium on*, pp. 297–307, 2011.

[38] H.-S. Chen and W. Gao, "Spectrum sensing for tv white space in north america," *Selected Areas in Communications, IEEE Journal on*, vol. 29, no. 2, pp. 316–326, 2011.

[39] C. Lei, Q. Jing, A. Viessmann, C. Kocks, G. Bruck, P. Jung, and R. Hu, "A spectrum sensing prototype for tv white space in china," in *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, pp. 1–6, 2011.

[40] C. Weber and G. Hildebrandt, "Evaluation of blind sensing algorithms in the 2.4 ghz ism-band on gnu radio and usrp2," in *Wireless Communication Systems (ISWCS), 2012 International Symposium on*, pp. 551–555, 2012.

[41] N. Shankar, C. Cordeiro, and K. Challapali, "Spectrum agile radios: utilization and sensing architectures," in *New Frontiers in Dynamic Spectrum Access Networks, 2005. DySPAN 2005. 2005 First IEEE International Symposium on*, pp. 160–169, 2005.

[42] A. Ghasemi and E. Sousa, "Optimization of spectrum sensing for opportunistic spectrum access in cognitive radio networks," in *Consumer Communications and Networking Conference, 2007. CCNC 2007. 4th IEEE*, pp. 1022–1026, 2007.

[43] M. A. H. Md. Shamim Hossain, Md. Ibrahim Abdullah, "Energy detection performance of spectrum sensing in cognitive radio," pp. 11–17, IJITCS, 11 2012.

[44] M. Hejazi and B. Abolhassani, "Energy detection based spectrum sensing in cognitive radio networks over spatially-correlated channels," in *Industrial Electronics Applications (ISIEA), 2010 IEEE Symposium on*, pp. 738–743, 2010.

[45] L. Y.-H. Oh Dong-Chan, "Energy detection based spectrum sensing for sensing error minimization in cognitive radio networks," IJCNIS, Jan. 2009.

[46] A. Leu, K. Steadman, M. McHenry, and J. Bates, "Ultra sensitive tv detector measurements," in *New Frontiers in Dynamic Spectrum Access Networks, 2005. DySPAN 2005. 2005 First IEEE International Symposium on*, pp. 30–36, 2005.

[47] D. M. M. Plata and ngel Gabriel Andrade Reatiga, "Evaluation of energy detection for spectrum sensing based on the dynamic selection of detection-threshold," *Procedia Engineering*, vol. 35, no. 0, pp. 135 – 143, 2012. International Meeting of Electrical Engineering Research 2012.

[48] K. S. Saqib Saleem, "Performance evaluation of energy detection based spectrum sensing technique for wireless channel," IJMSE, May 2012.

[49] S. Yarkan, W. Halbawi, and K. A. Qaraqe, "An experimental setup for performance evaluation of spectrum sensing via energy detector: indoor environment," in *Proceedings of the 4th International Conference on Cognitive Radio and Advanced Spectrum Management*, CogART '11, (New York, NY, USA), pp. 42:1–42:5, ACM, 2011.

[50] F. Hu and S. Wang, "Energy detection for spectrum sensing in cognitive radio sensor network over fading channels," in *Wireless Communications, Networking and Mobile Computing, 2009. WiCom '09. 5th International Conference on*, pp. 1–4, 2009.

[51] M. Di Felice, K. Chowdhury, and L. Bononi, "Cooperative spectrum management in cognitive vehicular ad hoc networks," in *Vehicular Networking Conference (VNC), 2011 IEEE*, pp. 47–54, 2011.

[52] T. S. Dhope and D. Simunic, "Spectrum sensing algorithm for cognitive radio networks for dynamic spectrum access for ieee 802.11 af standard," *International Journal of Research and Reviews in Wireless Sensor Networks*, March 2012.

[53] H. Urkowitz, "Energy detection of unknown deterministic signals," *Proceedings of the IEEE*, vol. 55, no. 4, pp. 523–531, 1967.

[54] S. Xie, L. Shen, and J. Liu, "Optimal threshold of energy detection for spectrum sensing in cognitive radio," in *Wireless Communications Signal Processing, 2009. WCSP 2009. International Conference on*, pp. 1–5, 2009.

[55] X.-p. Zhai, H.-g. He, and G.-x. Zheng, "Optimization of threshold for local spectrum sensing with energy detector," *Journal of Shanghai University (English Edition)*, vol. 15, no. 2, pp. 132–136, 2011.

[56] G. YU, C. LONG, M. XIANG, and W. XI, "A novel energy detection scheme based on dynamic threshold in cognitive radio systems," *Journal of Computational Information Systems*, vol. 8, no. 6, pp. 2245–2252, 2012.

[57] W. Yue and B. Zheng, "A two-stage spectrum sensing technique in cognitive radio systems based on combining energy detection and one-order cyclostationary feature detection," in *Pro-*

ceedings of the 2nd International Symposium on Web Information Systems and Applications (WISA 2009), pp. 327–330, May 2009.

[58] C. Spooner and W. Gardner, "The cumulant theory of cyclostationary time-series. ii. development and applications," *Signal Process., IEEE Trans. on*, vol. 42, pp. 3409–3429, dec 1994.

[59] W. Gardner and C. Spooner, "The cumulant theory of cyclostationary time-series. i. foundation," *Signal Process., IEEE Trans. on*, vol. 42, pp. 3387–3408, dec 1994.

[60] D. Rawat, G. Yan, and C. Bajracharya, "Signal processing techniques for spectrum sensing in cognitive radio networks," *International Journal of Ultra Wideband Communications and Systems*, no. x/x, pp. 1–10, 2010.

[61] M. Oner and F. Jondral, "Cyclostationarity based air interface recognition for software radio systems," in *Radio and Wireless Conference, 2004 IEEE*, pp. 263–266, 2004.

[62] A. Mossa and V. Jeoti, "The evaluation of cyclostationarity-based spectrum sensing in multipath fading channel," in *Intelligent and Advanced Systems (ICIAS), 2010 International Conference on*, pp. 1–6, 2010.

[63] A. Mossaa and V. Jeoti, "Cognitive radio: Cyclostationarity-based classification approach for analog tv and wireless microphone signals," in *Innovative Technologies in Intelligent Systems and Industrial Applications, 2009. CITISIA 2009*, pp. 107–111, july 2009.

[64] M. Ghozzi, M. Dohler, F. Marx, and J. Palicot, "Cognitive radio: methods for the detection of free bands," *Comptes Rendus Physique*, vol. 7, no. 7, pp. 794–804, 2006. Towards reconfigurable and cognitive communications.

[65] P. Sutton, K. Nolan, and L. Doyle, "Cyclostationary signatures in practical cognitive radio applications," *Selected Areas in Communications, IEEE Journal on*, vol. 26, pp. 13–24, jan. 2008.

[66] J. Zhang, G. Liao, and J. Wang, "Robust direction finding for cyclostationary signals with cycle frequency error," *Signal Process.*, vol. 85, pp. 2386–2393, December 2005.

[67] A. Zahedi-Ghasabeh, A. Tarighat, and B. Daneshrad, "Spectrum sensing of ofdm waveforms using embedded pilots in the presence of impairments," *Vehicular Technology, IEEE Transactions on*, vol. 61, no. 3, pp. 1208–1221, 2012.

[68] N. Han, S. Shon, J.-H. Chung, and J.-M. Kim, "Spectral correlation based signal detection method for spectrum sensing in ieee 802.22 wran systems," in *Advanced Communication Technology, 2006. ICACT 2006. The 8th International Conference*, vol. 3, pp. 6 pp.–1770, 2006.

[69] Y. Zeng and Y.-C. Liang, "Robustness of the cyclostationary detection to cyclic frequency mismatch," in *Personal Indoor and Mobile Radio Communications (PIMRC), 2010 IEEE 21st International Symposium on*, pp. 2704–2709, IEEE, 2010.

[70] A. M. Farrukh Javed, Imran Shafi, "A novel radio mode identification approach for spectrum sensing in cognitive radios," *International Journal of Communication Networks and Infromation Security*, Aug. 2012.

[71] H. Tang, "Some physical layer issues of wide-band cognitive radio systems," in *New Frontiers in Dynamic Spectrum Access Networks, 2005. DySPAN 2005. 2005 First IEEE International Symposium on*, pp. 151–159, 2005.

[72] S. Geirhofer, L. Tong, and B. Sadler, "A measurement-based model for dynamic spectrum access in wlan channels," in *Military Communications Conference, 2006. MILCOM 2006. IEEE*, pp. 1–7, 2006.

[73] S. Geirhofer, L. Tong, and B. M. Sadler, "Dynamic spectrum access in wlan channels: empirical model and its stochastic analysis," in *Proceedings of the first international workshop on Technology and policy for accessing spectrum*, TAPAS '06, (New York, NY, USA), ACM, 2006.

[74] J. Li, Z. Tan, S. Xu, and H. Wang, "Cooperative correlation based spectrum sensing for dmb-t systems," in *Vehicular Technology Conference (VTC 2010-Spring), 2010 IEEE 71st*, pp. 1–5, 2010.

[75] D. Cabric, A. Tkachenko, and R. Brodersen, "Spectrum sensing measurements of pilot, energy, and collaborative detection," in *Military Communications Conference, 2006. MILCOM 2006. IEEE*, pp. 1–7, 2006.

[76] D. Rawat and G. Yan, "Signal processing techniques for spectrum sensing in cognitive radio systems: Challenges and perspectives," in *Internet, 2009. AH-ICI 2009. First Asian Himalayas International Conference on*, pp. 1–5, 2009.

[77] T. Yucek and H. Arslan, "Spectrum characterization for opportunistic cognitive radio systems," in *Military Communications Conference, 2006. MILCOM 2006. IEEE*, pp. 1–6, 2006.

[78] J. Palicot and C. Roland, "A new concept for wireless reconfigurable receivers," *Communications Magazine, IEEE*, vol. 41, no. 7, pp. 124–132, 2003.

[79] M. Gandetto, M. Guainazzo, and C. S. Regazzoni, "Use of time-frequency analysis and neural networks for mode identification in a wireless software-defined radio approach," *EURASIP J. Appl. Signal Process.*, vol. 2004, pp. 1778–1790, Jan. 2004.

[80] K. A. Yasmin Hassan, Mohamed El-Tarhuni, "Learning-based spectrum sesning for cognitive radio systems," *Journal of Computer Networks and Communications*, p. 13 pages, 2012.

[81] R. Tandra and A. Sahai, "Fundamental limits on detection in low snr under noise uncertainty," in *Wireless Networks, Communications and Mobile Computing, 2005 International Conference on*, vol. 1, pp. 464–469 vol.1, 2005.

[82] S. Kapoor, S. V. R. K. Rao, and G. Singh, "Opportunistic spectrum sensing by employing matched filter in cognitive radio network," in *Communication Systems and Network Technologies (CSNT), 2011 International Conference on*, pp. 580–583, 2011.

[83] D. J. Kadhim and S. Q. Jobbar, "A spectrum sensing framework for uwb-cognitive network.," *Wireless Sensor Network*, vol. 2, no. 8, pp. 649–654, 2010.

[84] J. G. Proakis and M. Salehi, *Communication systems engineering*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1994.

[85] P. Paolo and V. Martin, *Signal Processing for Communications*. EPFL Press, 2008.

[86] R. E. Ziemer and W. H. Tranter, *Principles of Communications*. Wiley Publishing, 6th ed., 2008.

[87] T. S. Rappaport, *Wireless Communications Principles and Practice*. Prentice Hall, 2nd ed., 2002.

[88] A. V. Oppenheim, R. W. Schafer, and J. R. Buck, *Discrete-time signal processing (2nd ed.)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1999.

[89] R. Yeung, *Information theory and network coding*. Information technology–transmission, processing, and storage, Springer, 2008.

[90] C. M. Bishop, *Pattern Recognition and Machine Learning*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[91] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2003.

[92] L. I. Kuncheva, *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience, 2004.

[93] Y. LeCun and C. Cortes, "The mnist database of handwritten digits." http://yann.lecun.com/exdb/mnist/.

[94] E. Alpaydin, *Introduction to machine learning*. MIT Press, 2004.

[95] K. Fukunaga, *Introduction to statistical pattern recognition (2nd ed.)*. San Diego, CA, USA: Academic Press Professional, Inc., 1990.

[96] T. M. Mitchell, *Machine Learning*. New York, NY, USA: McGraw-Hill, Inc., 1 ed., 1997.

[97] E. L. Allwein, R. E. Schapire, and Y. Singer, "Reducing multiclass to binary: a unifying approach for margin classifiers," *J. Mach. Learn. Res.*, vol. 1, pp. 113–141, 2001.

[98] V. Guruswami and A. Sahai, "Multiclass learning, boosting, and error-correcting codes," in *Proceedings of the twelfth annual conference on Computational learning theory*, COLT '99, (New York, NY, USA), pp. 145–155, ACM, 1999.

[99] B. Scholkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA, USA: MIT Press, 2001.

[100] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Min. Knowl. Discov.*, vol. 2, pp. 121–167, June 1998.

[101] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. New York, NY, USA: Cambridge University Press, 2004.

[102] R. Herbrich, *Learning Kernel Classifiers: Theory and Algorithms*. Cambridge, MA, USA: MIT Press, 2001.

[103] B. Clarke, E. Fokoue, and H. H. Zhang, *Principles and Theory for Data Mining and Machine Learning*. Springer Publishing Company, Incorporated, 1st ed., 2009.

[104] A. J. Izenman, *Modern Multivariate Statistical Techniques: Regression, Classification, and Manifold Learning*. Springer Publishing Company, Incorporated, 2008.

[105] T. G. Dietterich and G. Bakiri, "Solving multiclass learning problems via error-correcting output codes," *J. Artif. Int. Res.*, vol. 2, no. 1, pp. 263–286, 1994.

[106] T. G. Dietterich, "Machine-learning research – four current directions," *AI MAGAZINE*, vol. 18, pp. 97–136, 1997.

[107] T. J. Sejnowski and C. R. Rosenberg, "Parallel networks that learn to pronounce english text, complez systems," 1987.

[108] K. Dharmawansa, R. Rajatheva, and K. Ahmed, "On the bivariate and trivariate rician distributions," in *Vehicular Technology Conference, 2006. VTC-2006 Fall. 2006 IEEE 64th*, pp. 1–5, sept. 2006.

[109] H. V. Poor, *An introduction to signal detection and estimation (2nd ed.)*. New York, NY, USA: Springer-Verlag New York, Inc., 1994.

[110] L. Palmer, "Coarse frequency estimation using the discrete fourier transform (corresp.)," *Inf. Theory, IEEE Trans. on*, vol. 20, pp. 104–109, jan 1974.

[111] D. Cabric, A. Tkachenko, and R. W. Brodersen, "Experimental study of spectrum sensing based on energy detection and network cooperation," in *Proceedings of the first international workshop on Technology and policy for accessing spectrum*, TAPAS '06, (New York, NY, USA), ACM, 2006.

[112] Z. Quan, S. Cui, H. Poor, and A. Sayed, "Collaborative wideband sensing for cognitive radios," *Signal Processing Magazine, IEEE*, vol. 25, pp. 60–73, November 2008.

[113] Y. Yan and Y. Gong, "Energy detection of narrowband signals in cognitive radio systems," in *Wireless Communications and Signal Processing (WCSP), 2010 International Conference on*, pp. 1–5, Oct. 2010.

[114] S. Atapattu, C. Tellambura, and H. Jiang, "Spectrum sensing via energy detector in low snr," in *Communications (ICC), 2011 IEEE International Conference on*, pp. 1–5, June.

[115] S. Elramly, F. Newagy, H. Yousry, and A. Elezabi, "Novel modified energy detection spectrum sensing technique for fm wireless microphone signals," in *Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on*, pp. 59–63, May 2011.

[116] O. Olabiyi and A. Annamalai, "Extending the capability of energy detector for sensing of heterogeneous wideband spectrum," in *Consumer Communications and Networking Conference (CCNC), 2012 IEEE*, pp. 454–458, Jan. 2012.

[117] O. Dobre, A. Abdi, Y. Bar-Ness, and W. Su, "Survey of automatic modulation classification techniques: classical approaches and new trends," *Communications, IET*, vol. 1, pp. 137–156, april 2007.

[118] E. Azzouz and A. Nandi, "Procedure for automatic recognition of analogue and digital modulations," *Commun., IEEE Proc-*, pp. 259–266, Oct 1996.

[119] A. K. Nandi and E. E. Azzouz, "Modulation recognition using artificial neural networks," *Signal Processing*, vol. 56, no. 2, pp. 165–175, 1997.

[120] C.-S. Park and D. Y. Kim, "A novel robust feature of modulation classification for reconfigurable software radio," *Consumer Electronics, IEEE Trans. on*, vol. 52, pp. 1193–1200, nov. 2006.

[121] B. gang Zhu, S. rui Peng, Y. wei Su, and T. Tang, "Automatic identification method of link4a," in *Networks Security, Wireless Communications and Trusted Computing, 2009. NSWCTC '09. International Conference on*, vol. 2, pp. 239–243, april 2009.

[122] V. Chaithanya and V. Reddy, "Blind modulation classification in the presence of carrier frequency offset," in *Signal Process. and Commun., 2010 Int. Conf. on*, pp. 1–5, July 2010.

[123] L. Wang and Y. Ren, "Recognition of digital modulation signals based on high order cumulants and support vector machines," in *Computing, Commun., Control, and Management, 2009 Int. Colloq. on*, pp. 271–274, Aug. 2009.

[124] A. E. Zadeh, "Automatic recognition of radio signals using a hybrid intelligent technique," *Expert Systems with Applications*, vol. In Press, Corrected Proof, 2010.

[125] A. Swami and B. M. Sadler, "Hierarchical digital modulation classification using cumulants," *IEEE Trans. on Commun.*, vol. 48, no. 3, pp. 416–429, 2000.

[126] M. L. D. Wong and A. K. Nandi, "Automatic digital modulation recognition using artificial neural network and genetic algorithm," *Signal Process.*, vol. 84, pp. 351–365, February 2004.

[127] M. e. a. Petrova, "Multi-class classification of analog and digital signals in cognitive radios using support vector machines," in *Wireless Commun. Sys., 7th Int. Symp. on*, pp. 986–990, Sept. 2010.

[128] B. G. Mobasseri, "Digital modulation classification using constellation shape," *Signal Processing*, vol. 80, no. 2, pp. 251–277, 2000.

[129] C. Yin, B. Li, and Y. Li, "Modulation classification of mqam signals from their constellation using clustering," in *Communication Software and Networks, 2010. ICCSN '10. Second International Conference on*, pp. 303–306, feb. 2010.

[130] N. Liu, B. Liu, S. Guo, and R. Luo, "Investigation on signal modulation recognition in the low snr," *Measuring Technology and Mechatronics Automation, International Conference on*, vol. 2, pp. 528–531, 2010.

[131] N. Ahmadi and R. Berangi, "Modulation classification of qam and psk from their constellation using genetic algorithm and hierarchical clustering," in *Information and Communication Technologies: From Theory to Applications, 2008. ICTTA 2008. 3rd International Conference on*, pp. 1–5, april 2008.

[132] N. Ahmadi, "Using fuzzy clustering and ttsas algorithm for modulation classification based on constellation diagram," *Eng. Appl. Artif. Intell.*, vol. 23, pp. 357–370, April 2010.

[133] M. Aslam, Z. Zhu, and A. Nandi, "Automatic digital modulation classification using genetic programming with k-nearest neighbor," in *MILCOM 2010*, pp. 1731–1736, Nov. 2010.

[134] M. D. Wong and A. K. Nandi, "Semi-blind algorithms for automatic classification of digital modulation schemes," *Digital Signal Processing*, vol. 18, no. 2, pp. 209–227, 2008.

[135] L. Hong and K. Ho, "Bpsk and qpsk modulation classification with unknown signal level," in *MILCOM 2000. 21st Century Military Communications Conference Proceedings*, vol. 2, pp. 976–980 vol.2, 2000.

[136] J. Xu, M. Zhou, and W. Su, "Discrete likelihood ratio test for intelligent signal recognition in software defined radio," in *Wireless and Optical Communications Conference (WOCC), 2010 19th Annual*, pp. 1–6, may 2010.

[137] M. Pedzisz and A. Mansour, "Automatic modulation recognition of mpsk signals using constellation rotation and its 4th order cumulant," *Digit. Signal Process.*, vol. 15, pp. 295–304, May 2005.

[138] F. Wang, R. Xu, and Z. Zhong, "Low complexity kolmogorov-smirnov modulation classification," in *Wireless Communications and Networking Conference (WCNC), 2011 IEEE*, pp. 1607–1611, march 2011.

[139] O. A. Dobre, Y. Bar-Ness, and W. Su, "Higher-order cyclic cumulants for high order modulation classification," in *Proceedings of the 2003 IEEE conference on Military communications - Volume I*, MILCOM'03, (Washington, DC, USA), pp. 112–117, IEEE Computer Society, 2003.

[140] M. Mirarab and M. Sobhani, "Robust modulation classification for psk /qam/ask using higher-order cumulants," in *Inf., Commun. Signal Process., 6th Int. Conf. on*, pp. 1–4, Dec. 2007.

[141] S. Xi and H.-C. Wu, "Robust automatic modulation classification using cumulant features in the presence of fading channels," in *WCNC*, pp. 2094–2099, 2006.

[142] A. Abavisani, M. Soleimani, and V. Tabatabavakili, "A novel algorithm for blind adaptive recognition between 8-psk and $\pi/4$-shifted qpsk modulated signals for software defined radio applications," in *Cognitive Radio Oriented Wireless Networks and Communications, 2009. CROWNCOM '09. 4th International Conference on*, pp. 1–6, june 2009.

[143] H. Hu, J. Song, and Y. Wang, "Signal classification based on spectral correlation analysis and svm in cognitive radio," in *Advanced Information Networking and Applications, 2008. AINA 2008. 22nd International Conference on*, pp. 883–887, march 2008.

[144] H. Roufarshbaf and J. Nelson, "Modulation classification in multipath environments using deterministic particle filtering," in *Digital Signal Processing Workshop and 5th IEEE Signal Processing Education Workshop, 2009. DSP/SPE 2009. IEEE 13th*, pp. 292–297, jan. 2009.

[145] T. Drumright and Z. Ding, "Qam constellation classification based on statistical sampling for linear distortive channels," *Signal Process., IEEE Trans. on*, vol. 54, pp. 1575–1586, may 2006.

[146] Z. Yaqin, R. Guanghui, W. XueXia, W. Zhilu, and G. Xuemai, "Automatic digital modulation recognition using artificial neural networks," in *Neural Networks and Signal Processing, 2003. Proceedings of the 2003 International Conference on*, vol. 1, pp. 257–260, dec. 2003.

[147] Y. Jin, S. Li, Z. Yang, and Y. Wang, "Study of a novel key feature in non-cooperative modulation automatic recognition," in *Wireless Communications, Networking and Mobile Computing, 2007. WiCom 2007. International Conference on*, pp. 1240–1243, sept. 2007.

[148] D. Grimaldi, S. Rapuano, and L. De Vito, "An automatic digital modulation classifier for measurement on telecommunication networks," *Instrum. Meas., IEEE Trans.*, vol. 56, pp. 1711–1720, oct. 2007.

[149] M. Narendar, A. Vinod, A. Madhukumar, and A. Krishna, "Automatic modulation classification for cognitive radios using cumulants based on fractional lower order statistics," in *General Assembly and Scientific Symposium, 2011 XXXth URSI*, pp. 1–4, aug. 2011.

[150] O. Azarmanesh and S. Bilen, "New results on a two-stage novel modulation classification technique for cognitive radio applications," in *MILITARY COMMUNICATIONS CONFERENCE, 2011 - MILCOM 2011*, pp. 266–271, Nov. 2011.

[151] A. Webb, *Statistical pattern recognition.* Wiley, 2002.

[152] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification (2nd Edition).* Wiley-Interscience, 2000.

[153] S. Russell and P. Norvig, *Artificial intelligence: a modern approach.* Prentice Hall series in artificial intelligence, Prentice Hall, 2010.

[154] H. kui Wang, X. qing Yao, J. ping Wu, Y. zhen Han, and L. li Feng, "Modulation recognition scheme using mixed model," in *Pervasive Computing Signal Process. and Applications (PCSPA), 2010 First International Conference on*, pp. 594–597, sept. 2010.

[155] Z. Jianli and W. Tingting, "Identification of cognitive radio modulation," in *Mechatronic Science, Electric Engineering and Computer, 2011 Int. Conf. on*, pp. 1773–1776, Aug. 2011.

[156] S. Ali and K. A. Smith-Miles, "Improved support vector machine generalization using normalized input space.," in *Australian Conf. on Artif. Intell.*, vol. 4304 of *Lecture Notes in Computer Science*, pp. 362–371, Springer, 2006.

[157] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, pp. 27:1–27:27, May 2011.

[158] D. Datla, R. Rajbanshi, A. M. Wyglinski, and G. Minden, "An adaptive spectrum sensing architecture for dynamic spectrum access networks," *Wireless Communications, IEEE Transactions on*, vol. 8, no. 8, pp. 4211–4219, 2009.

[159] L. Bixio, M. Ottonello, M. Raffetto, and C. Regazzoni, "Comparison among cognitive radio architectures for spectrum sensing," *EURASIP Journal on Wireless Communications and Networking*, vol. 2011, no. 1, p. 749891, 2011.

[160] P. Paysarvi-Hoseini and N. Beaulieu, "Optimal wideband spectrum sensing framework for cognitive radio systems," *Signal Processing, IEEE Transactions on*, vol. 59, no. 3, pp. 1170–1182, 2011.

[161] C.-K. Yang, C.-H. Hsieh, and Y.-H. Huang, "An energy-saving spectrum sensing processor based on partial discrete wavelet packet transform," in *VLSI Design, Automation, and Test (VLSI-DAT), 2012 International Symposium on*, pp. 1–4, 2012.

[162] Z. Tian, Y. Tafesse, and B. Sadler, "Cyclic feature detection with sub-nyquist sampling for wideband spectrum sensing," *Selected Topics in Signal Processing, IEEE Journal of*, vol. 6, no. 1, pp. 58–69, 2012.

[163] C.-H. Lee, C.-J. Chang, and S.-J. Chen, "Parallelization of spectrum sensing algorithms using graphic processing units," in *Cross Strait Quad-Regional Radio Science and Wireless Technology Conference (CSQRWC), 2012*, pp. 35–39, 2012.

[164] F. Penna and R. Garello, "Decentralized neyman-pearson test with belief propagation for peer-to-peer collaborative spectrum sensing," *Wireless Communications, IEEE Transactions on*, vol. 11, no. 5, pp. 1881–1891, 2012.

[165] S. Haykin, "Cognitive radio: brain-empowered wireless communications," *Selected Areas in Communications, IEEE Journal on*, vol. 23, no. 2, pp. 201–220, 2005.

[166] M. Li, Y. Xu, and Y. Cai, "A resource scheduling scheme utilizing grey space spectrums for ofdm based cognitive radio systems," in *Wireless Communications, Networking and Mobile Computing, 2009. WiCom '09. 5th International Conference on*, pp. 1–4, 2009.

[167] B. Zayen, A. Hayar, and D. Nussbaum, "Blind spectrum sensing for cognitive radio based on model selection," in *Cognitive Radio Oriented Wireless Networks and Communications, 2008. CrownCom 2008. 3rd International Conference on*, pp. 1–4, 2008.

[168] J. C. Clement, K. V. Krishnan, and A. Bagubali, "Article: Cognitive radio: Spectrum sensing problems in signal processing," *International Journal of Computer Applications*, vol. 40, pp. 37–40, February 2012. Published by Foundation of Computer Science, New York, USA.

[169] D. Grace and H. Zhang, *Cognitive Communications: Distributed Artificial Intelligence (DAI), Regulatory Policy and Economics, Implementation.* Wiley, 2012.

[170] S. Stein, "Unified analysis of certain coherent and noncoherent binary communications systems," *Information Theory, IEEE Transactions on*, vol. 10, no. 1, pp. 43–51, 1964.

[171] M. Fitz, "Further results in the unified analysis of digital communication systems," *Communications, IEEE Transactions on*, vol. 40, no. 3, pp. 521–532, 1992.

[172] P. Malm and T. Maseng, "Optimum number of signal alternatives in mobile cellular systems," in *Vehicular Technology Conference, 1998. VTC 98. 48th IEEE*, vol. 2, pp. 944–948 vol.2, 1998.

[173] O. Pujol, P. Radeva, and J. Vitria, "Discriminant ecoc: A heuristic method for application dependent design of error correcting output codes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, pp. 1007–1012, June 2006.

[174] E. Pimenta, J. Gama, and A. C. P. L. F. de Carvalho, "Pursuing the best ecoc dimension for multiclass problems," in *FLAIRS Conference*, pp. 622–627, 2007.

[175] S. Escalera, D. M. J. Tax, O. Pujol, P. Radeva, and R. P. W. Duin, "Subclass problem-dependent design for error-correcting output codes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, pp. 1041–1054, June 2008.

Appendix A

Formal Proof on Decision Exactness of Error-Correcting Output Code Framework and One-Versus-All and One-Versus-One Ensemble Classifiers

A.1 Introduction

Error-correcting output codes (ECOC) is a framework to create an ensemble classifier for multi-class problems by decomposing the class label set into binary labelings where coding theory is a guide to determine such labelings.

For example, suppose there are three class labels and four classifiers in the ensemble. One possible set of binary strings are $++++$ for class ω_1, $----$ for class ω_2, and $++--$ for class ω_3. In this example, all classifiers would train to produce +1 for class ω_1, and all classifiers would train to produce -1 for class ω_2. For class ω_3, the first and second classifiers would train to produce +1, and the third and fourth classifiers would train to produce -1. Suppose, for a test instance, the binary classifiers produced the binary string $+-++$. This string has a Hamming distance of 1, 3, and 3 from the binary strings associated with class ω_1, ω_2, and ω_3, respectively. In this example, the ensemble would classify the instance to class ω_1.

There are many other works that look at different variations of using ECOC ensembles [173–175]. There is also theoretical evidence that using ECOC ensembles is an improvement over OVA [98]. However, the great insight was provided by Allwein in [97] that the commonly used one-versus-all (OVA), and one-versus-one (OVO) ensemble approaches can be represented in the ECOC framework, i.e. an ECOC coding matrix can be created to build an ensemble equivalent to either

OVA or OVO ensembles. However, the ECOC decision rule is not exactly the same as the decision rules used by OVA and OVO ensembles, and we are unaware of a formal proof to show that the ECOC framework does produce the same result. In this appendix we intend prove that using an ECOC coding matrix that creates an ensemble equivalent to either OVA or OVO ensembles in the ECOC framework produces the same results as using the OVA or OVO ensemble directly.

An ECOC ensemble classifier is defined by a coding matrix, $M \in \{-1, 0, +1\}^{N \times D}$, where N is the number of class labels, and D is the number of classifiers in the ensemble. The n^{th} row in M corresponds to the codeword associated with the class label ω_n, and the d^{th} column in M creates a dichotomy of the class labels for training the d^{th} classifier. We mathematically represent the d^{th} classifier by the function $f : \mathcal{X} \to \mathbb{R}$, where for any $x \in \mathcal{X}$ the sign of $f_d(x)$ is the class label prediction and $|f_d(x)|$ is a measure of confidence on that label.

An ECOC ensemble using the coding matrix M has the decision rule that chooses the class label with the minimum loss summed over all of the classifiers in the ensemble, as shown in Equation A.1, where L is a loss function and we use the notation $M_d(\omega_k)$ to indicate the matrix element in M at row k and column d.

$$\omega_{\text{ECOC}} = \arg\min_{\omega_k \in \Omega} \sum_{d=1}^{D} L(\ f_d(x) \cdot M_d(\omega_k)\ ) \tag{A.1}$$

In [97], the loss function $L : \mathbb{R} \to [0, \infty)$, used in Equation A.1 has the constraint that $\frac{1}{2}\left[L(z) + L(-z)\right] \geq L(0) > 0$, which holds for any convex loss function, but convexity was not required. However, in order to show, in our proof, that ECOC framework produces the same decision results as the OVA and OVO ensemble classifiers, we must further constrain the loss function L. We will specify these constraints later.

Next, we describe the structure of the coding matrices used to produce the OVA and OVO equivalent ECOC ensembles.

The coding matrix M that mimics the OVA ensemble with N class labels is an $N \times D$ matrix, where $D = N$, and the diagonal elements are +1 and the non-diagonal elements are -1. For example, if $N = 4$, the coding matrix M to produce an OVA equivalent ensemble is shown in Equation A.2.

$$M = \begin{bmatrix} + & - & - & - \\ - & + & - & - \\ - & - & + & - \\ - & - & - & + \end{bmatrix} \tag{A.2}$$

For N class labels, an OVA ensemble consists of N classifiers. The n^{th} classifier is trained to return +1 one when presented an instance of class label n and -1 otherwise. We see that the coding matrix in Equation A.2 encapsulates this ensemble structure into the ECOC framework. The decision rule for OVA is to select the class associated with the classifier that produces the largest positive value,

as shown in Equation A.3.

$$\omega_{\text{OVA}} = \arg\max_{\omega_d \in \Omega} f_d(x) \tag{A.3}$$

The coding matrix M that mimics the OVO ensemble with N class labels is an $N \times D$ matrix with $D = \binom{N}{2}$, which creates a classifier for each possible pair of class labels. Thus, each column of M has exactly two non-zero elements and each row of M has exactly $N-1$ non-zero elements. For example, if $N = 4$, the coding matrix M to produce an OVO equivalent ensemble is shown in Equation A.4.

$$M = \begin{bmatrix} + & + & + & 0 & 0 & 0 \\ - & 0 & 0 & + & + & 0 \\ 0 & - & 0 & - & 0 & + \\ 0 & 0 & - & 0 & - & - \end{bmatrix} \tag{A.4}$$

An OVO ensemble creates a classifier for every possible pair of class labels. For a problem with N class labels, there are $\binom{N}{2}$ classifiers in an OVO ensemble. For each pair of class labels, one class is trained to a positive value. The other class is trained to a negative value. The remaining other class labels are not used in training. Again, we see that the coding matrix in Equation A.4 encapsulates this ensemble structure into the ECOC framework. In OVO, the ensemble classifier will select the class label by a majority vote from all the classifiers, as shown in Equation A.5, where $\delta[\cdot]$ is an indicator function that returns 1 when the argument is true, otherwise the function returns 0, and `sign`(z) returns + if $z > 0$ and returns − otherwise.

$$\omega_{\text{OVO}} = \arg\max_{\omega_k \in \Omega} \sum_{d=1}^{D} \delta\left[\texttt{sign}\left(f_d(x)\right) = M_d(\omega_k)\right] \tag{A.5}$$

A.2 ECOC Framework Proofs

We now prove that the decision rules $\omega_{\text{OVA}} \Leftrightarrow \omega_{\text{ECOC}}$ and $\omega_{\text{OVO}} \Leftrightarrow \omega_{\text{ECOC}}$ indeed produce the exact same results under certain loss function constraints. First, we show that an ECOC ensemble can be created to mimic an OVA ensemble. Second, we show that an ECOC ensemble can produce an equivalent output compared to an OVO ensemble.

A.2.1 OVA Proof

We show that an ECOC framework ensemble using a coding matrix of the type seen in Equation A.2 produces the same prediction as an OVA ensemble. In order for this statement to be true, we must require the loss function, L, used by the ECOC decision rule to have the following constraints:

- L must be a monotonically decreasing function

- For any nonequal z_1 and z_2, the equalities $L(z_1) = L(z_2)$ and $L(-z_1) = L(-z_2)$ both cannot be true

The first constraint says that for any $z_1 > z_2$, then $L(z_1) \leq L(z_2)$. The two constraints combined say that for any $z_1 > z_2$, then $L(z_1) + L(-z_2) < L(-z_1) + L(z_2)$. Commonly used loss functions that satisfy our constraints are the exponential function $L(z) = exp\{-z\}$, the logistic regression function, $L(z) = \log(1 + exp\{-2z\})$, and the hinge loss function, $L(z) = (1 - z)_+$ [97]. However, the 0-1 loss function or Hamming loss function do not satisfy the constraints since they have the functional form $L(z)$ that returns a 0 when the $z > 0$ and returns a constant value c for $z < 0$ [105].

First, we show forward direction that given the decision made by an OVA ensemble, then the ECOC ensemble will produce the same decision.

Proof of $\omega_{OVA} \Rightarrow \omega_{ECOC}$. Without loss of generality, let us assume the OVA ensemble decision rule selects class label ω_1. We want to show that $\omega_{\text{OVA}} = \omega_1$ implies $\omega_{\text{ECOC}} = \omega_1$. In order for the OVA decision rule to produce this classification, Inequality A.6 must be true for all $k \neq 1$. We apply the loss function to both sides of the inequality, which is monotonically decreasing, as shown in Inequality A.7. Likewise, if we negative Inequality A.6, and apply the loss function to both sides, then we have Inequality A.8.

$$f_1(x) > f_k(x) \tag{A.6}$$

$$L(\ f_1(x)\ ) \leq L(\ f_k(x)\ ) \tag{A.7}$$

$$L(-f_k(x)\ ) \leq L(-f_1(x)\ ) \tag{A.8}$$

We combine Inequality A.7 and Inequality A.8 to produce Inequality A.9, which is a strict inequality by the fact that $f_1(x)$ and $f_k(x)$ are not equal in conjunction with our constraints on the loss function. We get Inequality A.10 by adding a constant positive term, C, to both sides of Inequality A.9, where $C = \sum_{d=2,d\neq k}^{D} L(-f_d(x)\ )$.

$$L(\ f_1(x)\ ) + L(-f_k(x)\ ) < L(\ f_k(x)\ ) + L(-f_1(x)\ ) \tag{A.9}$$

$$L(\ f_1(x)\ ) + L(-f_k(x)\ ) + C < L(\ f_k(x)\ ) + L(-f_1(x)\ ) + C \tag{A.10}$$

Recall that the value of $M_d(w_n)$ in the OVA coding matrix is +1 when $d = n$, and the value is -1, otherwise. We see that the both sides of Inequality A.10 can rewritten using the ECOC decision rule notation, as shown in Inequality A.11.

$$\sum_{d=1}^{D} L(\ f_d(x) \cdot M_d(\omega_1)\ ) < \sum_{d=1}^{D} L(\ f_d(x) \cdot M_d(\omega_k)\ ) \tag{A.11}$$

Therefore, we have shown that $\omega_{\text{OVA}} = \omega_1 \Rightarrow \omega_{\text{ECOC}} = \omega_1$ to be true. This completes the proof of the forward direction of the claim that the ECOC ensemble decision rule produces the same result as the OVA ensemble decision rule. □

To complete the proof, we show the reverse direction that given the decision made by an ECOC ensemble, then the OVA ensemble will produce the same decision.

Proof of $\omega_{ECOC} \Rightarrow \omega_{OVA}$. Without loss of generality, let us assume the ECOC ensemble decision rule selects class label ω_1. We want to show that $\omega_{\text{ECOC}} = \omega_1$ implies $\omega_{\text{OVA}} = \omega_1$. In order for the ECOC decision rule to produce this classification, Inequality A.12 must be true for all $k \neq 1$.

$$\sum_{d=1}^{D} L\left(f_d(x) \cdot M_d(\omega_1) \right) < \sum_{d=1}^{D} L\left(f_d(x) \cdot M_d(\omega_k) \right) \tag{A.12}$$

Clearly, we can manipulate Inequality A.12 to produce Inequality A.13 by repeating the algebraic steps in the forward direction proof in the reverse order.

$$f_1(x) > f_k(x) \tag{A.13}$$

Therefore, we have shown that $\omega_{\text{ECOC}} = \omega_1 \Rightarrow \omega_{\text{OVA}} = \omega_1$ to be true. This completes the proof of the reverse direction of the claim that the ECOC ensemble decision rule produces the same result as the OVA ensemble decision rule. □

We have shown both the forward and reverse directions of our proof, which shows that the decision rule for an ECOC ensemble using a coding matrix to mimic an OVA ensemble, given our constraints on the loss function, does produce the same classification as an ensemble explicitly created as an OVA ensemble.

A.2.2 OVO Proof

We show that an ECOC framework ensemble using a coding matrix of the type seen in Equation A.4 produces the same prediction as an OVO ensemble. In order for this statement to be true, we must require the loss function, L, used by the ECOC decision rule to have the form shown in Equation A.14 with the constraints that $a < c$ and $b \in \mathbb{R}$

$$L(z) = \begin{cases} a, & \text{if } z > 0 \\ b, & \text{if } z = 0 \\ c, & \text{if } z < 0 \end{cases} \tag{A.14}$$

The required loss function form is a strong constraint, which disallows the use on many common loss functions, such as the exponential function, the logistic regression function, and the hinge loss function. We are essentially restricted to functions similar to the 0-1 loss function, where $a = 0$ and $b = c = 1$, or Hamming loss function, where $a = 0$, $b = \frac{1}{2}$, and $c = 1$. Note that any linear operation on a valid loss function such that $L'(z) = \alpha L(z) + \beta$, where $\alpha > 0$, $\beta \in \mathbb{R}$, produces another valid loss function. We must also assume that all classifiers produce a nonzero value, i.e. $|f_d(x)| > 0$.

First, we show forward direction that given the decision made by an OVO ensemble, then the ECOC ensemble will produce the same decision.

Proof of $\omega_{OVO} \Rightarrow \omega_{ECOC}$. Without loss of generality, let us assume the OVO ensemble decision rule selects class label ω_1. We want to show that $\omega_{\text{OVO}} = \omega_1$ implies $\omega_{\text{ECOC}} = \omega_1$. In order for the majority vote OVO decision rule to produce this classification, Inequality A.15 must be true for all $k \neq 1$. We negate this equality and add a constant value (D) to both sides to produce Inequality A.16.

$$\sum_{d=1}^{D} \delta\left[\texttt{sign}\left(f_d(x)\right) = M_d(\omega_1)\right] \;>\; \sum_{d=1}^{D} \delta\left[\texttt{sign}\left(f_d(x)\right) = M_d(\omega_k)\right] \tag{A.15}$$

$$\sum_{d=1}^{D} 1 - \delta\left[\texttt{sign}\left(f_d(x)\right) = M_d(\omega_1)\right] \;<\; \sum_{d=1}^{D} 1 - \delta\left[\texttt{sign}\left(f_d(x)\right) = M_d(\omega_k)\right] \tag{A.16}$$

Let $g_{d,j}(x) = 1 - \delta\left[\texttt{sign}\left(f_d(x)\right) = M_d(\omega_j)\right]$, which allows us to re-write Inequality A.16 as Inequality A.17

$$\sum_{d=1}^{D} g_{d,1}(x) < \sum_{d=1}^{D} g_{d,k}(x) \tag{A.17}$$

The value of $g_{d,j}(x)$ falls into three categories for class labels ω_1 and ω_k. These three categories allow us to select three values a, b, and c in order to make an equivalent loss function $\tilde{L}$. The value of $g_{d,j}(x)$ can be 0 and corresponds to a classifier making a prediction in favor of the j^{th} class label, and sets $\tilde{a} = 0$. The value of $g_{d,j}(x)$ can be 1 and corresponds to a classifier making a prediction against of the j^{th} class label, and sets $\tilde{c} = 1$. Also, the value of $g_{d,j}(x)$ can be 1 and corresponds to a classifier that was not trained on the j^{th} class label, and sets $\tilde{b} = 1$. Using this loss function along with the three category partitioning on $g_{d,j}(x)$, we have Inequality A.18.

$$\sum_{d=1}^{D} \tilde{L}\left(\; f_d(x) \cdot M_d(\omega_1) \;\right) < \sum_{d=1}^{D} \tilde{L}\left(\; f_d(x) \cdot M_d(\omega_k) \;\right) \tag{A.18}$$

We can translate the loss function $\tilde{L}$ to the loss function L used by the ECOC ensemble by calcu-

lating α and β such that $L = \alpha\tilde{L} + \beta$. By using linear algebra, we see that $\alpha = (c-a)/(\tilde{c}-\tilde{a})$ and $\beta = a - \tilde{a}$, where a and c are associated with the loss function L. Notice that we do not need to worry about the relationship of b with respect to $\tilde{L}$ parameters. The reason is that the value of b contributes to the summation in the decision making process only when the argument to the loss is $z = 0$. We assumed that $|f_d(x)| > 0$, and thus the only time $z = f_d(x) \cdot M_d(\omega_j) = 0$ is when $M_d(\omega_j) = 0$, which corresponds to d^{th} classifier that was not trained on the j^{th} class label. There are exactly $D - N - 1$ classifiers for each class label with this characteristic and produces a constant across all class labels. Therefore, we can replace $\tilde{L}$ with L to produce the ECOC decision rule that select ω_1, as shown in Inequality A.19.

$$\sum_{d=1}^{D} L\left(\ f_d(x) \cdot M_d(\omega_1)\ \right) < \sum_{d=1}^{D} L\left(\ f_d(x) \cdot M_d(\omega_k)\ \right) \tag{A.19}$$

Therefore, we have shown that $\omega_{\text{OVO}} = \omega_1 \Rightarrow \omega_{\text{ECOC}} = \omega_1$ to be true. This completes the proof of the forward direction of the claim that the ECOC ensemble decision rule produces the same result as the OVO ensemble decision rule. □

To complete the proof, we show the reverse direction that given the decision made by an ECOC ensemble, then the OVO ensemble will produce the same decision.

Proof of $\omega_{ECOC} \Rightarrow \omega_{OVO}$. Without loss of generality, let us assume the ECOC ensemble decision rule selects class label ω_1. We want to show that $\omega_{\text{ECOC}} = \omega_1$ implies $\omega_{\text{OVO}} = \omega_1$. In order for the ECOC decision rule to produce this classification, Inequality A.20 must be true for all $k \neq 1$.

$$\sum_{d=1}^{D} L\left(\ f_d(x) \cdot M_d(\omega_1)\ \right) < \sum_{d=1}^{D} L\left(\ f_d(x) \cdot M_d(\omega_k)\ \right) \tag{A.20}$$

Let A_j denote the number of classifiers such that $L(f_d(x) \cdot M_d(\omega_j)) = a$. Let B_j denote the number of classifiers such that $L(f_d(x) \cdot M_d(\omega_j)) = b$. Let C_j denote the number of classifiers such that $L(f_d(x) \cdot M_d(\omega_j)) = c$. Using these new variable we can re-write Inequality A.20 as Inequality A.21.

$$a(A_1) + b(B_1) + c(C_1) < a(A_k) + b(B_k) + c(C_k) \tag{A.21}$$

Next, we translate the inequalities such the A_j is multiplied by 0 and C_j is multiplied by 1. Thus, we subtract $D \cdot a$ from both sides of the inequality and multiple by the scalar fraction $\frac{1}{c-a}$, as shown in Inequality A.22.

$$0(A_1) + \frac{b-a}{c-a}(B_1) + 1(C_1) < 0(A_k) + \frac{b-a}{c-a}(B_k) + 1(C_k) \tag{A.22}$$

Again, recall the scaled value of B_j is constant over all class labels. We can easily replace $\frac{b-a}{c-a}$ with the value of 1 and still have a valid inequality, as shown in Inequality A.23.

$$0(A_1) + 1(B_1) + 1(C_1) < 0(A_k) + 1(B_k) + 1(C_k) \tag{A.23}$$

We see that Inequality A.23 is equivalent to Inequality A.24, since the summations on the right-hand side is equal in both inequalities, and the same is true for the left-hand side of both inequalities. We add D to both sides of the inequality and multiple by -1 to produce Inequality A.25.

$$\sum_{d=1}^{D} 1 - \delta\left[\texttt{sign}\left(f_d(x)\right) = M_d(\omega_1)\right] < \sum_{d=1}^{D} 1 - \delta\left[\texttt{sign}\left(f_d(x)\right) = M_d(\omega_k)\right] \tag{A.24}$$

$$\sum_{d=1}^{D} \delta\left[\texttt{sign}\left(f_d(x)\right) = M_d(\omega_1)\right] > \sum_{d=1}^{D} \delta\left[\texttt{sign}\left(f_d(x)\right) = M_d(\omega_k)\right] \tag{A.25}$$

Therefore, we have shown that $\omega_{\text{ECOC}} = \omega_1 \Rightarrow \omega_{\text{OVO}} = \omega_1$ to be true. This completes the proof of the reverse direction of the claim that the ECOC ensemble decision rule produces the same result as the OVO ensemble decision rule. □

We have shown both the forward and reverse directions of our proof, which shows that the decision rule for an ECOC ensemble using a coding matrix to mimic an OVO ensemble, given our constraints on the loss function, does produce the same classification as an ensemble explicitly created as an OVO ensemble.

As a quick aside, we can a simple example for the need of the stronger constraints on the loss function used with an ECOC ensemble to produce the *exact* same classification as an OVO ensemble. Suppose $N = 5$ and we have $D = 10$ to create our OVO equivalent ECOC ensemble. For simplicity, let us use the linear loss function of the form shown in Equation A.26. Note that this loss function would satisfy the constraints we used in the OVA ensemble.

$$L(z) = \begin{cases} -z, & z < 0 \\ 0, & \text{else} \end{cases} \tag{A.26}$$

Without loss in generality, suppose that for the instance x the OVO ensemble casts 3 votes for ω_1, 2 votes for ω_2, 2 votes for ω_3, 2 votes for ω_4, and 1 vote for ω_5. Thus, the OVO ensemble would classifier x to belong to class label ω_1.

Let $S_j = \sum_{d=1}^{D} L(f_d(x) \cdot M_d(\omega_j))$ be the score that the ECOC ensemble calculates for the j^{th} class

label. In this example, the ECOC ensemble would produce the following score values:

$$\begin{aligned} S_1 &= 3 \cdot 0 + L(z_{11}) + 6 \cdot L(0)) \\ S_2 &= 2 \cdot 0 + L(z_{21}) + L(z_{22}) + 6 \cdot L(0)) \\ S_3 &= 2 \cdot 0 + L(z_{31}) + L(z_{32}) + 6 \cdot L(0)) \\ S_4 &= 2 \cdot 0 + L(z_{41}) + L(z_{42}) + 6 \cdot L(0)) \\ S_5 &= 1 \cdot 0 + L(z_{51}) + L(z_{52}) + L(z_{53}) + 6 \cdot L(0)) \end{aligned}$$

The value $6 \cdot L(0)$ is a constant bias across each class label, the value $n_j \cdot 0$ corresponds to the number of votes casted in favor of the j^{th} class label, and the value $L(z_{jm})$ corresponds to the loss associated with a classifier that votes against the j^{th} class label. Necessarily $z_{jm} < 0$ and thus $L(z_{jm}) = -z_{jm}$.

Comparing the scores of ω_1 and ω_2, we need $S_1 < S_2$ in order to have the ECOC ensemble classify x to belong to class label ω_1. Thus, we need $-z_{11} < -z_{21} - z_{22}$ in all cases. Since all of the classifiers produce numbers on the real lines, clearly, there are many examples where this inequality is not true. Therefore, with this type of loss function, it is possible for the ECOC ensemble to produce a different classification than the OVA ensemble.

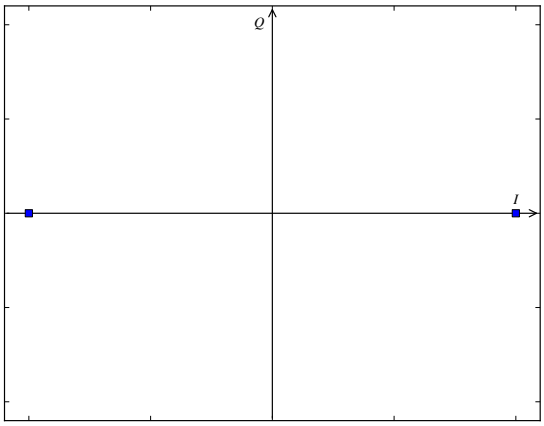
A.3 Conclusions

We have shown that an ensemble created with the ECOC framework using the OVA and OVO coding matrices and specific types of loss functions will produce the same classification as ensembles explicitly using those approaches. This is a new result that has not been previously proven. Intuitively, we expect the coding matrices for the ECOC ensembles that mimic the OVA and OVO structures to produce the correct label, and maybe, for this reason, the result was assumed to be true. However, as we saw in the proof, it is nontrivial to show that the two ensembles do behave the same, given the additional *restrictive* constraints on the loss function L. Further work is required to relax the constraints on our proof.

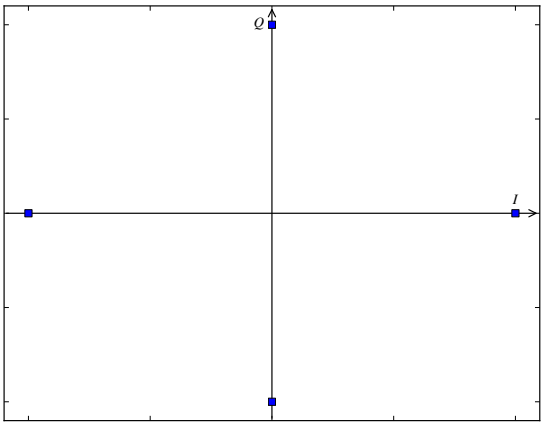
Appendix B

The Modulation Constellation Sets

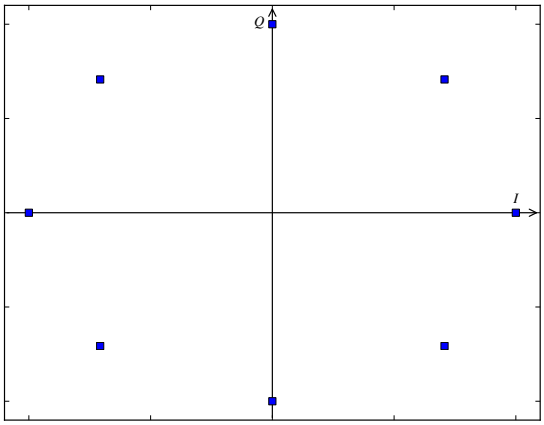
This appendix contains the constellation plots for all of the template modulation constellation sets we considered in our work. In each figure, the horizontal axis is the in-phase (I) component, and the vertical axis is the quadrature phase (Q) component. Figure B.1 shows the four types of M-ary PSK modulations used. Figure B.2 shows the four types of M-ary PAM modulations used. Figure B.3 shows the four types of M-ary QAM modulations used. Each figure has a subplot that indicates the specific modulation that is being visualized.



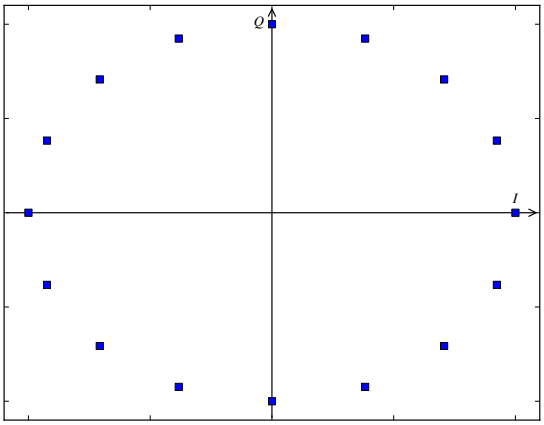
(a) 2-PSK or BPSK



(b) 4-PSK or QPSK

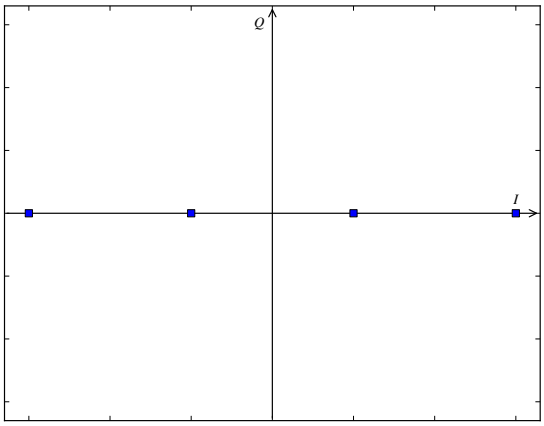


(c) 8-PSK

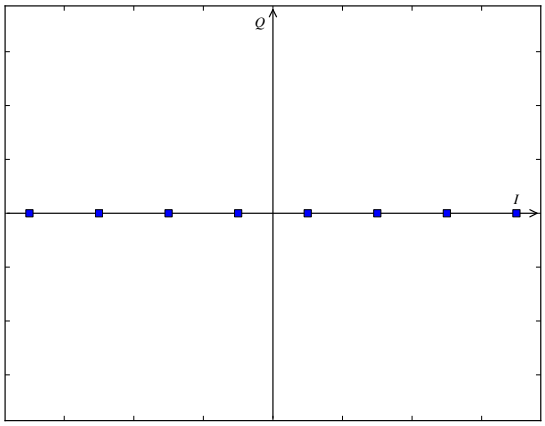


(d) 16-PSK

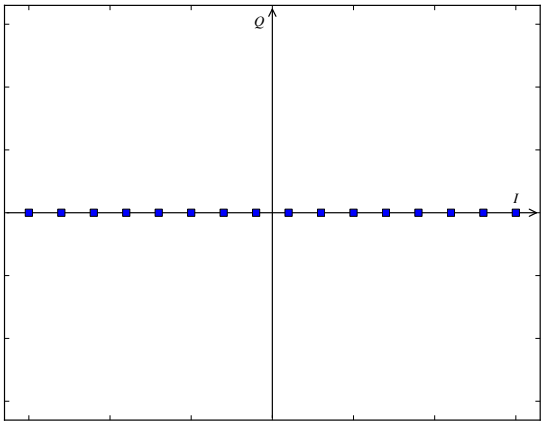
Figure B.1: Constellation Plots for PSK



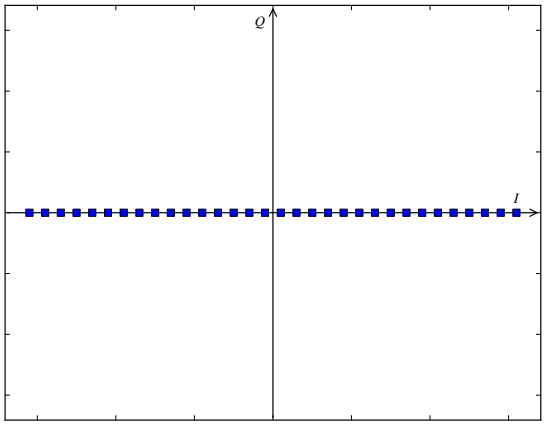
(a) 4-PAM



(b) 8-PAM

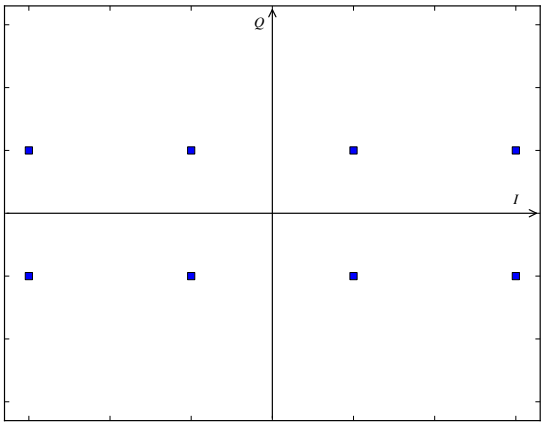


(c) 16-PAM

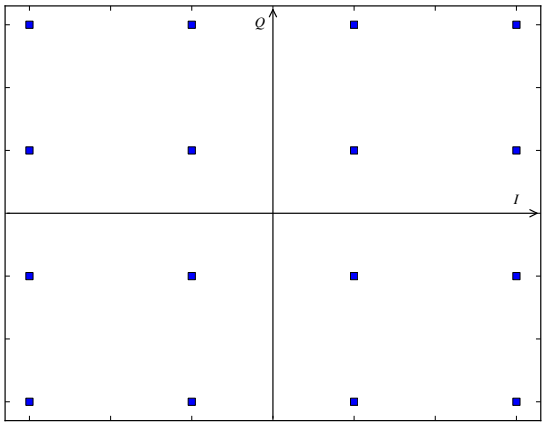


(d) 32-PAM

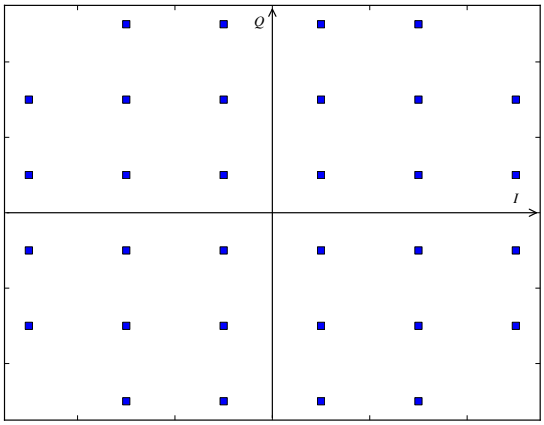
Figure B.2: Constellation Plots for PAM



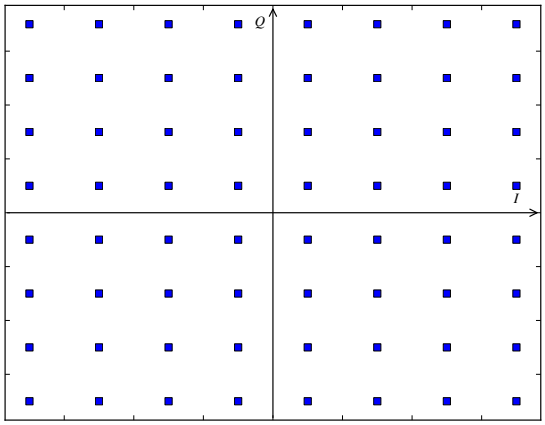
(a) 8-QAM



(b) 16-QAM



(c) 32-QAM



(d) 64-QAM

Figure B.3: Constellation Plots for QAM