5-2016

An adaptable fuzzy-based model for predicting link quality in robot networks.

Christopher J. Lowrance
University of Louisville

AN ADAPTABLE FUZZY-BASED MODEL FOR PREDICTING LINK QUALITY IN ROBOT NETWORKS

By

Christopher J. Lowrance
B.S., Virginia Military Institute, 2000
M.S., George Washington University, 2008

A Dissertation
Submitted to the Faculty of the
J.B. Speed School of Engineering of the University of Louisville
in Partial Fulfillment of the Requirements
for the Degree of

Doctor of Philosophy
in Computer Science and Engineering

Department of Computer Engineering and Computer Science
University of Louisville
Louisville, Kentucky

May 2016

AN ADAPTABLE FUZZY-BASED MODEL FOR PREDICTING LINK QUALITY IN ROBOT NETWORKS

By

Christopher J. Lowrance
B.S., Virginia Military Institute, 2000
M.S., George Washington University, 2008

A Dissertation Approved on

April 8, 2016

by the following Dissertation Committee:

_____
Adrian P. Lauf, Ph.D., Dissertation Director

_____
Mehmed Kantardzic, Ph.D.

_____
Roman V. Yampolskiy, Ph.D.

_____
Hongxiang Li, Ph.D.

_____
Karla Welch, Ph.D.

DEDICATION

This dissertation is dedicated to my older and late brother

Mr. Samuel Mark Lowrance

who always supported me, and also, encouraged me to pursue my Ph.D.

ACKNOWLEDGMENTS

First and foremost, I would like to thank my wife, Bahar, and my son, Keon, for their support over the years while I progressed through graduate school and toward building this dissertation. Their love has provided me the necessary stamina to see this dissertation to fruition. They have sacrificed much over the years for my career and education, and for that, I will be eternally grateful.

I wish to recognize my advisor, Adrian P. Lauf, for his unabated mentorship. He provided me ample guidance and encouragement as I navigated my way toward this dissertation topic. Under his direction, I was given the creative freedom to grow and build the skills necessary to succeed in the future as a scholar and researcher. I am immensely grateful for all his efforts.

I would also like to express my appreciation for the assistance provided by my committee members: Mehmed Kantardzic, Roman Yampolskiy, Hongxiang Li, and Karla Welch. They all offered various guidance and suggestions to improve my work and to ensure my success. Their insightfulness and wisdom ultimately helped to refine my work into its present form.

Finally, I would like to thank my parents, Ronald and Alice Lowrance, for their unwavering love and support.

ABSTRACT

AN ADAPTABLE FUZZY-BASED MODEL FOR PREDICTING LINK QUALITY IN ROBOT NETWORKS

Christopher J. Lowrance

April 8, 2016

It is often essential for robots to maintain wireless connectivity with other systems so that commands, sensor data, and other situational information can be exchanged. Unfortunately, maintaining sufficient connection quality between these systems can be problematic. Robot mobility, combined with the attenuation and rapid dynamics associated with radio wave propagation, can cause frequent link quality (LQ) issues such as degraded throughput, temporary disconnects, or even link failure. In order to proactively mitigate such problems, robots must possess the capability, at the application layer, to gauge the quality of their wireless connections. However, many of the existing approaches lack adaptability or the framework necessary to rapidly build and sustain an accurate LQ prediction model. The primary contribution of this dissertation is the introduction of a novel way of blending machine learning with fuzzy logic so that an adaptable, yet intuitive LQ prediction model can be formed. Another significant contribution includes the evaluation of a unique *active* and *incremental* learning framework for quickly constructing and maintaining prediction models in robot networks with minimal sampling overhead.

TABLE OF CONTENTS

LIST OF TABLES

TABLE PAGE

LIST OF FIGURES

FIGURE PAGE

CHAPTER I

INTRODUCTION

Motivation

Moving and communicating are two fundamental tasks for most robotic systems, but despite the influence of mobility on communications, these tasks are often considered independently. In theory, the network is expected to automatically adapt to the connectivity changes induced by robot movement [1], but in reality, unfettered movement can severely degrade wireless communications due to signal attenuation and multipath fading [2]. For instance, assume that a robot is commanded to traverse to an area that would severely degrade its only wireless connection. Without regularly checking the status of its link, the robot may continue along its planned trajectory until the link eventually fails, which may isolate the robot and prevent it from completing its mission. To avoid such adverse scenarios, it is essential for a robot to regularly assess the quality of its wireless links and incorporate network awareness into its navigation and communication planning. Furthermore, a robot with the ability to regularly predict link quality (LQ) can also use such information to take proactive measures to improve network connections, instead of just avoiding precarious areas. For instance, robots are commonly used as part of a networked collaborative team so that communications can be extended wirelessly from an area of interest back to a data collection point, such as a command center [3-5]. In such a scenario, robots can use their LQ-sensing abilities to control their positioning so that

wireless connectivity is continuously sustained [6]. Or, in another possible scenario, a robot could potentially use LQ awareness to activate communication diversity mechanisms, such as a secondary directional radio, to strengthen its network connectivity once its primary link becomes unreliable [7]. Besides network optimization, estimating the strength of radio signals also plays an important role in assisting robots with other tasks such as localizing the whereabouts of other networked nodes [8, 9]. In conclusion, there are a number of ways where assessing LQ can improve critical robotic functions such as navigation, communication, and localization.

Challenges

Robots are commonly employed in austere, disaster-stricken, or war-torn areas that may not have a reliable infrastructure network, and consequently, robots tend to communicate with team members or command centers in an ad hoc and decentralized fashion [3, 10, 11]. Further complicating communications is the fact that robots are often required to operate at far away from command centers to keep operators a safe distance away from hazardous areas, and the separation often exceeds the range capability of the low-power radios typically used on robots [4, 12, 13]. To overcome this challenge, additional nodes are typically added for the purpose of acting as network relays, and with an appropriate routing protocol, these collective nodes can form decentralized, multi-hop networks known as mobile ad hoc networks (MANETs) [12, 14]. It is common for robots to form MANETs using IEEE 802.11 wireless transceivers because of their low-cost, small packaging, and high-speed capability [3], but the range of these radios is approximately limited to 100 meters (m) [4]. The transceivers operate in the gigahertz range, and consequently, the radio waves emitted by them tend to follow a line-of-sight (LOS)

propagation path [10]. LOS radio waves can easily be obstructed or altered by surrounding objects, thus complicating the decoding process at the receiver. As another disadvantage, command centers and robots tend to be positioned low to the ground, and their low antenna height can restrict Fresnel zone clearance and further limit the range of LOS communications [10]. The aggregation of these factors, as well as the dynamics introduced by mobility, makes MANETs prone to frequent disconnections, network partitions, and latency variations [15]. The capability to assess LQ can potentially alleviate these issues when combined with intelligent control and decision mechanisms.

Unfortunately, gauging LQ in dynamic robot networks is a nontrivial task. There are a number of factors that complicate the ability to decipher the state of the wireless channel. For instance, radio waves undergo a number of wireless phenomena including blocking, absorption, reflection, scattering, and diffraction [16]. The blocking or attenuation of radio signals due to various obstacles and the surrounding terrain is commonly referred to as *shadowing* [16, 17]. The aforementioned effects of radio wave propagation are unpredictable and can change over time, especially when one of the transceivers is mobile. Mobility in robot networks increases the time variability of the propagation parameters due to the spatial changes it induces, and it also sets the conditions for another propagation phenomenon known as *multipath fading* that is characterized by rapid and unpredictable variations in signal strength. The phenomenon arises due to the fact that radio signals can travel various paths to a receiver, and the recombination of these signals may be at times be either constructive or destructive manner [16]. Attempting to model these propagation effects using theoretical equations is usually impractical and inaccurate for such dynamic networks [18]. Consequently, most of the approaches attempt

to statistically or probabilistically quantify link conditions using empirical measurements [16, 18].

There exist well-established methods for empirically modeling channel conditions at the physical layer [16, 19], but this is not the case for upper layers [20]. An overview on the common networking layers can be found in [21]. Unfortunately, the upper layers is where robot subsystems, such as navigation control and ad hoc routing, need details on LQ in order to make optimization decisions. One of the challenges with empirically assessing LQ at the upper layers is that only limited radio information is passed up from the physical layer. Furthermore, the few metrics that are available tend to be convoluted; they are closely coupled with the effects of radio wave propagation, and hence, they are not expressed in lucid terms that an application natively understands [22]. Another complication is that the upper layers tend to observe environmental change more slowly than the rate in which the channel conditions are changing [16], making the translation of these radio metrics more difficult. These issues, as well as others, are explored further in the literature survey provided in Chapter III.

Overview on Existing Approaches and the State of the Art

In general, there are two types of approaches to empirically assess LQ at the higher layers: *estimation* or *prediction* [20]. Both will be discussed further in Chapter III, but in the meantime, the two methods will be briefly distinguished. An *estimate* of LQ is a rough approximation that generally reflects the large-scale attenuation factors of path loss and shadowing; however, the small-scale effects of multipath are usually averaged. Thus, an estimate will tend to contain some margin of error due to multipath being abstracted. On the other hand, a *prediction* of LQ attempts to minimize the error by trying to exploit the

short-term stationary behavior of the wireless channel, and thus predictions generally do not abstract or average out the effects of multipath. However, LQ predictions are short-lived based on the rapidly changing wireless channel. By far, the majority of the works in the literature are LQ *estimators*, likely because of the short validity of LQ predictions.

There are a few common techniques to empirically-based LQ estimation, and each has its own sets of advantages and challenges. One estimation method involves periodically probing wireless links using packet transmissions, and then, forming statistics based on the number of probes successfully received over a finite window of attempts. Despite its attractive simplicity, there are a number of drawbacks to the probing technique that makes it suboptimal in many networks including MANETs. Probing has been shown to be inaccurate in tracking actual link loss [23], as well as slow in detecting sudden changes in LQ [24, 25]. Furthermore, the added network congestion and the energy cost of probes can be concerning in dense networks with energy-constrained nodes such as robots. Another approach to LQ estimation that counters some of these disadvantages is to passively monitor the protocol signaling, such as acknowledgements (ACKs), occurring at the data link layer, but the approach requires complex driver and system modifications that makes it impractical and difficult to scale across different hardware [26]. On the other hand, another approach that *does not* involve driver or kernel modifications is to use the physical layer metrics provided to the operating system by the radio hardware. The radio metrics are updated with every received signal and offer the potential for up-to-date LQ estimates, but the metrics tend to be noisy in the sense that they are more reflective of the rapidly changing disturbances introduced by wireless propagation. Hence, the variance and fluctuation associated with the physical layer metrics makes mapping them directly to

LQ (i.e., throughput or reliability) challenging. Complicating matters further is the fact that the physical layer metrics are hardware dependent and can vary in terms of scaling and precision based on the specific wireless adapter and its properties (e.g., antenna type, calibration, internal noise in receiver, etc.) [20, 27]. Some approaches attempt to overcome these issues by developing customized mapping functions that are built offline using empirical measurements from a particular geographic setting. However, these pre-configured mapping functions tend to remain static and only relevant for the specific hardware and environmental conditions for which the data was collected. Such an approach is inaccurate for mobile systems, such as robots, whose spatial locations and propagation environments change throughout a given mission or from mission-to-mission. Even if such an approach is adopted, the time and effort involved in conducting the onsite experiments necessary to build the mapping function is impractical and not scalable.

There are primarily two state-of-the-art methods for mitigating the drawbacks associated with the common approaches to LQ estimation that were previously mentioned. One method is to form a *hybrid* LQ estimate using a combination of multiple LQ metrics from different protocol layers in a cross-layer fashion. However, cross-layer designs tend to inherit some form of disadvantage from each of the layers, despite attempting to mitigate them. For instance, mixing a physical layer metric with a probing statistic may help offset the responsiveness and accuracy issues of probing, but the overhead of probing still remains. Another drawback to the existing hybrid schemes is that many of them lack any means of adaptability. Most of the techniques combine the metrics in a fixed manner based on observations made previously offline. Thus, without any form of adaptation or learning,

the concept of strengthening an LQ estimate through multiple inputs is marginalized due to the reality that LQ is link-specific and dynamic.

A more recent state-of-the-art development is to use machine learning for quantifying LQ based on several inputs. The statistical learning method automates the process of generalizing a best-fit functional description between the input metrics and the desired output estimate or prediction of LQ. Furthermore, when accompanied with an incremental learning framework, the generalized relationship can evolve over time. This evolutionary process of adapting the functional relationship is essential in LQ estimation because of the streaming and dynamic nature of the application.

There are a few works in the literature that have explored the use of supervised learning in LQ estimation. However, as will be discussed further in Chapter III, there are several opportunities to advance the state of the art. Arguably, the most significant gap in the literature is a lack of a comprehensive sampling and incremental learning framework. Existing works tend to either train only once, or they simply resort to an online learning algorithm that tunes the model on an individual sample basis. Thus far, no works in LQ prediction have made any effort to investigate an incremental framework where the model is updated in batch-to-batch fashion. There are some inherent advantages to such an approach as will be discussed later. Furthermore, there is little research associated with reducing the overhead associated with labeling samples in the domain of LQ estimation. For instance, some authors suggest that robots should randomly move around for a period of time in order to collect diverse samples [28, 29], but such artificial movement consumes time and energy. To mitigate this issue, other authors have suggested that nodes should exchange labeled samples across multiple links [24, 30], but the problem is that the

statistical relationship between the inputs and target are link-specific and should not be aggregated. In summary, there are several opportunities to advance the state of the art in LQ estimation.

Fundamentals of the Proposed Approach

Due to the nature of radio wave propagation in mobile environments, the statistical relationship between the input metrics to the output description of LQ can change over time. However, the existing fuzzy-based LQ estimators in the literature lack the adaptability to adequately deal with the evolving relationship. To overcome this issue, a novel way of adding adaptability to fuzzy systems through supervised learning is introduced. Specifically, a set of classifiers is used to assign the input metrics into fuzzy sets, in contrast to the classical approach of using expert-designed fuzzy sets. By using classification, the discovery of the statistical mapping function between the inputs to the output is automated and optimized according to the learning algorithm. However, learning is a continuous process in the context of robot networks due to the various triggers that may cause the statistical relationship within the LQ data stream to drift. Thus, robots must regularly collect and label new samples to retrain the classifiers, and furthermore, the older samples that no longer represent the evolved relationship must be forgotten in order to avoid a reduction in accuracy. Unfortunately, obtaining labels is relatively expensive given that the sender must query the receiver across the wireless channel for each target value. Thus, an active learning framework is developed where attempts are made to minimize these labeling expenses by selectively requesting labels only for the samples necessary to build an accurate prediction model.

There are several mechanisms within the proposed selective labeling framework that determine whether to request a label within the data stream. One of the labeling mechanisms is designed to uniformly label the feature space so that the sampling process is unbiased and sufficient. A series of first-in first-out (FIFO) queues are used to guide the uniform labeling process. The queues have a maximum size to prevent oversampling of the data stream and to ensure the size of the training set remains manageable for online retraining. Other labeling mechanisms are focused on sample replacement in order to mitigate concept drift. These mechanisms use *change detection* to trigger sample replacement on an as-needed basis, and in either case, the oldest samples stored in the queuing structures are '*popped*' out and forgotten.

In addition to selective labeling, an incremental learning framework is also designed for updating the LQ prediction model over time as new samples are incorporated into the training set. The framework calls for retraining the model incrementally in a batch-to-batch format. The batch-style learning process is ideally suited for the selective sampling scheme because it allows for a reduction in label requests. More specifically, the conservation of labeling resources is based on the proposed scheme maintaining a portion of the samples from the previous batch in each new training batch. This partial memory concept reduces the labeling burden by not requiring the system to continually collect similar types of samples, which would be wasting resources. Furthermore, maintaining some samples from batch-to-batch allows the system to maintain a more comprehensive model, which over time may account for large sections of the sample space. In this case, it may be possible for the robot to conserve its labeling resources, assuming it leaves and reenters the same region of the feature space and drift has not yet occurred.

A fundamental assumption of the batch-style framework is that there is usually no need to throw away all the training examples from the previous batch. Given the robot application, it can be assumed that the large-scale statistical relationship between the LQ inputs to the output target will likely change somewhat slowly, especially in the case of slower-moving ground robotic platforms [18, 31]. As will be shown later in this dissertation, major statistical shifts in the underlying relationship tend to only occur when the attenuation or noise components of the wireless channel change significantly change on a persistent basis. Given these results, there is usually no need to continually label and retrain based on every sample within the data stream. Therefore, it is more logical for the system to conserve its labeling budget and maintain partial memory from batch to batch. Effectively, the rate of learning or retraining of the proposed model is driven by the system's labeling mechanisms and whether the system detects the need to increase its labeling. These critical concepts of detailing a holistic framework for selective sampling, incremental learning, and rapid model startup have yet to be fully developed in the context of LQ prediction or robot networks. Hence, the focus of this dissertation and the following contributions.

Contributions

Several contributions are offered as part of a composite body of work focused on improving LQ assessment in robot networks. Much of this work is extensible to other types of wireless networks and higher-layer applications besides those dealing with robots. The specific contributions within this dissertation can be summarized as follows:

(i) A comprehensive survey of LQ estimation and prediction in IEEE 802.11-based networks. The evolution of the field is presented over the

years, yet focuses on the latest developments. The survey explains how the latest trends in fuzzy logic and machine learning are deficient, and it provides suggestions to the research community on how to improve upon these works.

(ii) **Introduction of a unique application for LQ estimation in robot networks and a cost-effective means to extend the transmission range of robotic systems.** The concept of radio-switching is introduced to robotic systems. Specifically, the idea of adding a passive antenna reflector to a robot as a secondary radio option is presented and evaluated. Fuzzy logic and LQ estimation are used to switch the secondary, directional radio between sleep-mode and active-mode, as needed, in order to conserve energy.

(iii) **Introduction of a novel approach to achieve adaptability in fuzzy-based systems.** Machine learning is incorporated in a unique novel fashion into the fuzzification process of fuzzy-based systems. The approach offers a new way of adding adaptability to fuzzy systems, assuming an incremental learning algorithm or framework is employed.

(iv) **Introduction of a comprehensive framework for selective labeling and incremental learning in robot networks.** **S**everal sampling and model building concepts are introduced that have yet to be explored in the context of LQ prediction or robot networks. The proposed sampling techniques are shown to reduce labeling costs, and the unique incorporation of synthetic samples into the learning framework is shown to improve early prediction

accuracy. Currently in the literature, there exist only single-iteration batch learners that are trained only once offline, or online learning algorithms that tune the model sample-by-sample.

Dissertation Outline

The remaining portion of the dissertation is organized as follows. In the next chapter, an overview is presented on the fundamental concepts associated with fuzzy logic and supervised learning. Afterwards, the existing literature associated with LQ estimation and prediction is surveyed in Chapter III. Subsequently, in Chapter IV, a rudimentary fuzzy controller is presented for the purpose of switching between diverse radios based on LQ estimates. In Chapter V, the deficiencies associated with the previous fuzzy-based design is highlighted and used as motivation for the introduction of a new method for performing fuzzification in an adaptable manner. Then, in Chapter VI, a holistic sampling and learning framework is introduced for the purpose of lowering the costs associated with maintaining prediction accuracy in a streaming-based network. Finally, concluding remarks and suggested future work are provided in Chapter VII.

CHAPTER II

BACKGROUND

Introduction

This chapter is intended to serve as a primer on some fundamental concepts within the domains of fuzzy logic and machine learning. These selected topics serve as the foundation to the research presented in the subsequent chapters. Further information into these subject areas can be found in the sources cited in this chapter, as several sources are comprehensive textbooks covering these domains.

Fuzzy Logic and the Fuzzy Set

Fuzzy or vague boundaries exist in many natural phenomena, and additionally, the human brain tends to perceive and quantify things in a non-crisp or fuzzy way [32]. These are some of the primary reasons Zadeh first introduced the fuzzy set in [33]. The fuzzy set provides a way of quantifying fuzziness or vagueness in measurements and inputs. Due to fuzziness being a frequent and naturally occurring phenomenon, fuzzy logic is often intuitive and fitting in many scenarios.

The fuzzy set serves as the foundation of fuzzy logic [34]. A fuzzy set defines a range of real numbers that either have full *or partial* membership to the set. Before its introduction, elements in classical set theory either fully belonged to a set or not at all. However, with the fuzzy logic, an element's degree of membership to a particular set can range from [0, 1]. More specifically, an element not belonging to a fuzzy set is assigned a value of 0, full membership with a value of 1, and partial membership with any real number between 0 and 1.

There are several attractive features to the fuzzy set. One of them relates to the fact that elements can take partial membership in *multiple* fuzzy sets. This is advantageous when an element has an ambiguous value near the boundaries of other distinct sets. Boundaries between classical sets are defined using crisp cutoffs, such as $x > 7$, but often times, it is difficult to identify such a boundary. Or, such an abrupt transition between sets may be inaccurate at times. In contrast, an element in fuzzy logic can take partial membership in all of the adjacent sets with varying degrees of certainty. In fact, sets in fuzzy logic tend to overlap, to a certain extent, in order to account for these ambiguous regions and to allow for more graceful transitions between adjacent sets.

Another common motivator for using fuzzy sets deals with the uncertainty that sometimes arises when working with empirical measurements. Occasionally, measurements contain some level of imprecision or noise due to the natural phenomenon being measured or due to the instrumentation being used. In these cases, the ability to overlap sets and to classify measurements with varying degrees of confidence can express, and even mitigate, the amount of noise or undependability associated with the measurement.

Fuzzy sets are defined by *membership functions*. More formally, a membership function expresses the degree to which every possible value of the variable, x, belongs to the fuzzy set, and in equation form, a fuzzy set A can be defined as

$$A = [(x, \mu^A(x)): x \in X]$$

where $\mu^A(x)$ is the membership function of A which maps each value of x to a membership degree between 0 and 1 [32, 34]. The variable x can be formally defined as a subset of real

numbers (i.e., $X \subseteq \mathbb{R}$), and this subset of real numbers, X, is referred to as the *universe of discourse* in fuzzy parlance [34].

In most cases, fuzzy sets are labeled with linguistic names, formally referred to as *linguistic values* [32, 34]. These linguistic values are usually one-word or two-word classifications such as '*high*' or '*moderate high*'. The assigned labels naturally describe the group of elements within the set, and the name usually relates to the placement of the fuzzy set with respect to the universe of discourse. For instance, a fuzzy set with the linguistic value of '*high*' would likely occupy the upper region of the universe, while another fuzzy set with the name '*low*' would account for a range of values in the lower region of the universe. The somewhat vague connotation of these linguistic terms accurately reflects the concept of fuzzy sets and the vagueness associated with their boundaries. The naming of the fuzzy sets using natural language plays an important role in fuzzy logic. In fact, the fuzzy reasoning process was intentionally modeled after human reasoning, which generally prefers to perform decision making using linguistics versus crisp numbers [34].

Membership functions are often described using triangular-shaped or Gaussian-shaped functions [34]. These functions offer a relatively straightforward way to express the waning degree of membership that x assumes as it digresses from the center peak of the function. However, a fuzzy set can be described using any shape that best describes the set and its elements.

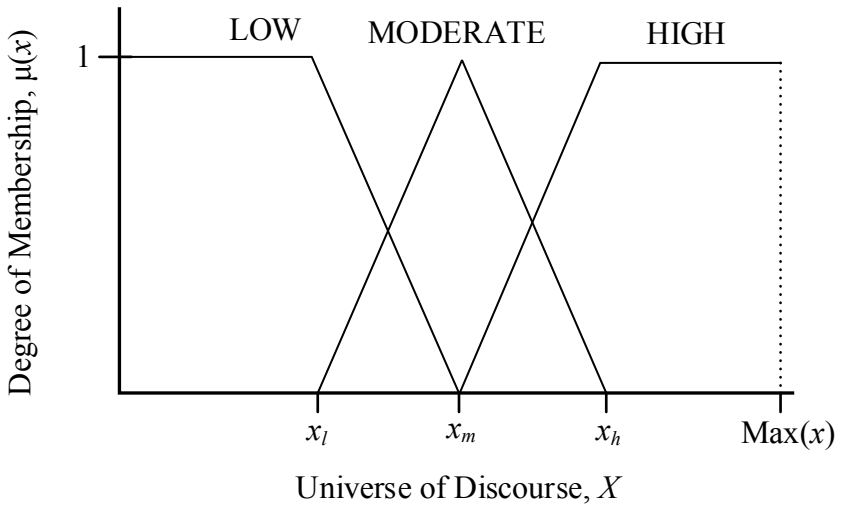

Figure 1. An example of fuzzy sets

An example to illustrate some of the aforementioned concepts about fuzzy sets is provided in Figure 1. A total of three fuzzy sets are shown on the universe of discourse, *X,* which ranges from 0 to max(*X*). Each fuzzy set has a unique linguistic value of either '*Low*', '*Moderate*', or '*High*'. The center of the middle triangle, as well as the saturation points of the corner fuzzy sets, are placed in locations where *x* can be confidently described as belonging with full membership to the labeled fuzzy set. The slope of the triangle and corner edges describe how quickly membership to the fuzzy set attenuates away from the center (or transition) point. The fuzzy sets are shown to overlap in some regions where elements of *X* assume partial membership to multiple fuzzy sets. In general, the amount of overlap depends upon the uncertainty associated with classifying a value of *x* into one of the fuzzy sets. The number of fuzzy sets chosen for a given input variable is design-specific, and it usually involves a tradeoff between accuracy and complexity. Furthermore, the actual placement of the fuzzy sets along the universe of discourse is either determined

using expert knowledge [34] or through some learning process as will be discuss later in Chapter V.

Fuzzy Systems

In general, a multiple-input, single-output (MISO) fuzzy system accepts a vector of *n* crisp inputs (i.e., $\underline{x}(t) \in \mathbb{R}^n$), and then operates on them using fuzzy logic in order to generate a crisp output, y(t), for every instance of *t* [34]. The internal operation of the system involves a fuzzy reasoning process, which is modeled after human reasoning [34]. The process of inferring an output given a set of inputs is derived from a knowledge base that is usually in the form of *if-then* conditions that are easy for humans to comprehend. The knowledge embedded within a fuzzy system can either be imparted via an expert or learned over time through historical input-output pairs [34, 35]. Fuzzy logic systematically handles the partial memberships of the inputs (i.e., the fuzzy sets) and uses them to generate weighted outputs that are crisp (i.e., continuous) in nature. The internal functions of most fuzzy systems can be broken down into a series major subcomponents as illustrated in Figure 2.

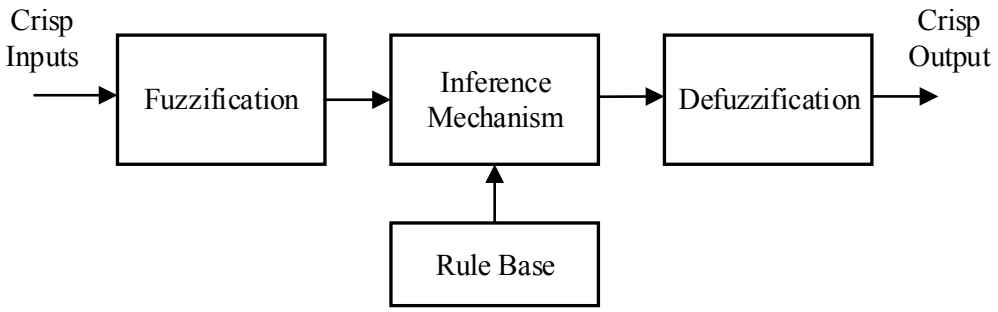

Figure 2. The structure of fuzzy systems

Fuzzification

The first process of fuzzy systems is known as *fuzzification.* This step involves converting the vector of real inputs into fuzzy sets. More specifically, during fuzzification, each input is assigned a level of membership to every fuzzy set defined along its respective universe of discourse using the established membership functions. The membership functions that perform this process can either be configured by an expert or adapted over time through learning [34, 35].

Inference Mechanism

After fuzzification, the membership assignments of the inputs are processed through a set of logic commonly referred to as the *inference mechanism*. Fuzzy systems make inferences based on an inherent set of *if-then* rules, collectively referred to as the *rule base.* One of the strengths of fuzzy systems is that expert knowledge can be imparted into the rule base [34]. However, the rule base can become cumbersome for an expert to manage as the number of inputs and fuzzy sets grow; thus, the rule-base is occasionally auto-generated in some systems [35]. Most fuzzy systems use the *modus ponens* form of if-then logic, and thus, the rules generally take the form

If x is A, then y is B

where the part to the left of the comma constitutes the *premise* of the rule, and the part to the right of the comma is the *consequent* that is inferred (i.e., fired) when the premise is true. The *modus ponens* form of the above rule states that if the premise is true, then the consequent is also true.

In contrast to the above rule, the premise of most fuzzy rules usually consists of a conjunction of n conditions. In other words, a rule R_i in a fuzzy system could be formally described as

$$R_i\text{: If } x_1 \text{is } A_1^k \text{ and } x_2 \text{is } A_2^l \text{ and…and } x_n \text{is } \mathrm{A}_n^m, \text{ then } y \text{ is } Q^i$$

where A_1^k is the linguistic value associated with fuzzy set k on the universe of input x_1, A_2^l is the linguistic value associated with fuzzy set l on the universe of input x_2, and so on. The consequent of rule R_i is described by an output fuzzy set, Q^i, on the universe of Y.

The first function of the inference mechanism is to determine the extent to which each rule in the rule base is fired. Because each input may only partially belong to a fuzzy set, a premise may only be partially true. In other words, the degree of firing a particular rule in the rule base is based on the level of certainty that a given premise is true. The above statement can be generalized for an input vector $\boldsymbol{x}$ by quantifying that R_i is fired to the extent

$$\mu_i(\boldsymbol{x})=\mu_1^k(x_1)*\mu_2^l(x_2)*\ldots*\mu_1^m(x_n)$$

where $\mu_1^k(x_1)$ is the degree of membership that input x_1 assumed in fuzzy set k on universe X_1, and $\mu_2^l(x_1)$ is the degree of membership that input x_2 assumed in fuzzy set l on universe X_2, and so on. The * notation represents the AND operation between the fuzzy sets, and the operation is carried out using some form of a triangular norm or *T-norm* (i.e., either *min*, *product*, or other) [34].

The next step of the inference mechanism uses the degree to which each rule is '*on*' in order to determine all of the implied fuzzy sets on *Y*. For example, the membership function of the implied fuzzy set $\widehat{Q}^i$ for a particular rule R_i can be characterized by

$$\mu^{\widehat{Q}^i}(y)=\mu_i(\boldsymbol{x})*\mu^{Q^i}$$

where the implied fuzzy set $\hat{Q}^i$ takes the form of the consequent membership function μ^{Q^i}, but to the extent that rule R_i is '*on*' defined by $\mu_i(\boldsymbol{x})$.

Defuzzification

The final stage of fuzzy systems is defuzzification. This process involves converting the collection of recommendations generated by all of the fired rules into a crisp output. There are multiple different methods for performing defuzzification, but the most common methods are center of gravity (COG) and center average (CA) [34].

The method of CA defuzzification is demonstrated for more insight into the process of generating a *weighted* average of all the rule consequents. Assuming that each rule consequent is normal, which is usually the case, then the CA defuzzification process can be described by

$$y^{\text{crisp}}=\frac{\sum_{i=1}^{R} q_i\mu_i(\boldsymbol{x})}{\sum_{i=1}^{R} \mu_i(\boldsymbol{x})}$$

where q_i is the consequent of rule R_i defined by its center on *Y* [34].

It may be convenient to express the consequents of rules using singleton membership functions, similar to those displayed in Figure 3. Singleton fuzzy sets eliminate the need to for more computationally-intense COG defuzzification and the need to calculate the area under implied fuzzy sets [34]. In general, singletons can replace

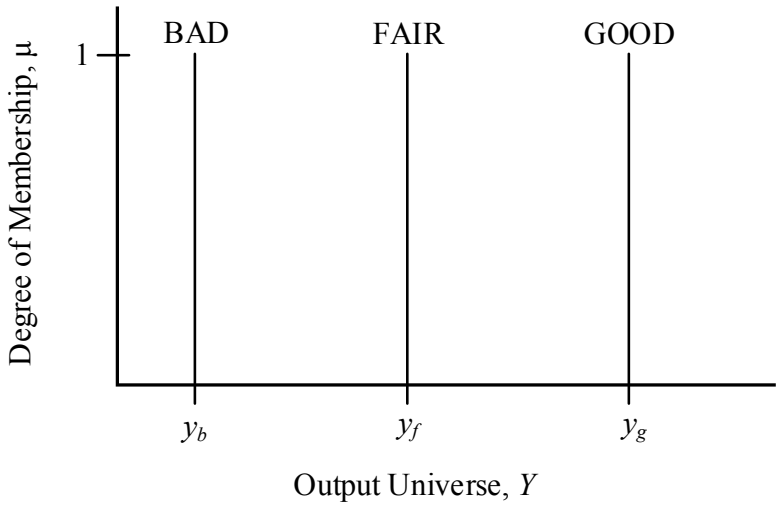

Figure 3. Example of output singletons

ordinary fuzzy sets on the output universe whenever the product T-norm is used and whenever the output fuzzy sets are symmetrical and normal [34].

Mamdani and Takagi-Sugeno Fuzzy Systems

There are two primary types of fuzzy systems: Mamdani systems and Takagi-Sugeno (T-S) systems. They both perform fuzzification and use rule bases to infer consequents; however, the form of the consequents differ between the architectures. In Mamdani systems, the consequents are expressed as fuzzy sets on the output universe. However, in T-S systems, the consequents are mathematical expressions that assume any linear function of variables. In other words, a T-S rule R_i would take the general form

$$R_i\text{: If } x_1 \text{is } A_1^k \text{ and } x_2 \text{is } A_2^l \text{ and…and x}_n \text{is } A_n^m, \text{ then } y \text{ is } Q^i = f(x_1, x_2, \ldots, x_n)$$

where the consequent is usually some polynomial function of the inputs. In contrast with Mamdani systems, the structure of T-S systems makes the identification and adaptive control of dynamic and nonlinear systems possible using fuzzy logic [34].

Machine Learning

The primary function of machine learning is to automate the process of learning from data [36]. The fundamental approach typically used in machine learning is the inductive form of learning [32]. *Induction* refers to the classical type of inference where a generalization is obtained from a set of samples. The generalized model describes the dependencies or underlying approximation function between the inputs and output. The usual intent of inducing a general model is so that the future values can be predicted using the approximation function, and this process follows the classical inference mechanism of *deduction* [32]. Prediction is beneficial in many scenarios. Some examples include when it is expensive to measure system output relative to the input, or when the intent is to make proactive adjustments and control the output of the system using the inputs. In these scenarios, prediction offers valuable and inexpensive foresight into the system's behavior.

The power of machine learning lies in its ability to extract general tendencies or patterns within data [37]. Statistical or probabilistic analysis, along with other modeling assumptions, are often used to formalize these dependencies. This capability is attractive because, on many occasions, system environments are complex and cannot be analytically described [32, 34]. In this case, machine learning offers an alternative way of describing system behavior through an approximation function that is based on empirical observation.

An example of a complex environment is the wireless channel. It is a convoluted system involving several dynamics, especially when either antenna is mobile. Thus, an analytical description of the wireless environment is usually not feasible [18, 29]. However, with machine learning, it is possible to gather a set of empirical samples consisting of the input and output variables, and afterwards, supply them to a learning

algorithm. Using the training examples, the learner can then systematically find a suitable model or approximation to the underlying dependency between the variables in a statistical or probabilistic sense. It should be noted that with dynamic systems, such as the wireless channel, that learning is a continuous process and the approximation function should continuously evolve with the system. Furthermore, although machine learning automates much of the process, human intervention in terms of incorporating domain knowledge, making modeling assumptions, or tuning modeling parameters is often required for best results [32, 36].

Supervised Learning

Supervised learning is the most common form of inductive learning [38]. It is a type of learning method that uses *labeled* training examples to find an approximation to the unknown function relating the inputs to the output of a system. The training examples in supervised learning are assumed to be labeled with the true values of the target variable.

The process of supervised learning is illustrated in Figure 4. The figure shows the main elements involved in generalizing an approximation function, $g(\mathbf{x})$, to the unknown target function, $f(\mathbf{x})$, which defines the environment or system at hand. Based on the environmental setup, empirical input samples are provided to the system according to some unknown input distribution, $P(\mathbf{x})$. An important assumption is that the inputs are sampled in an independent and identically distributed (IID) fashion from the distribution P on the input space X [38]. Each input vector, $\mathbf{x}_i$, in supervised learning is labeled with the target variable, y_i. After collecting n samples, a training set, consisting of n pairs of input-output examples, is fed into the learning algorithm. Then, the learning algorithm uses the training examples, along with a set of assumptions formalized within the hypothesis set and the

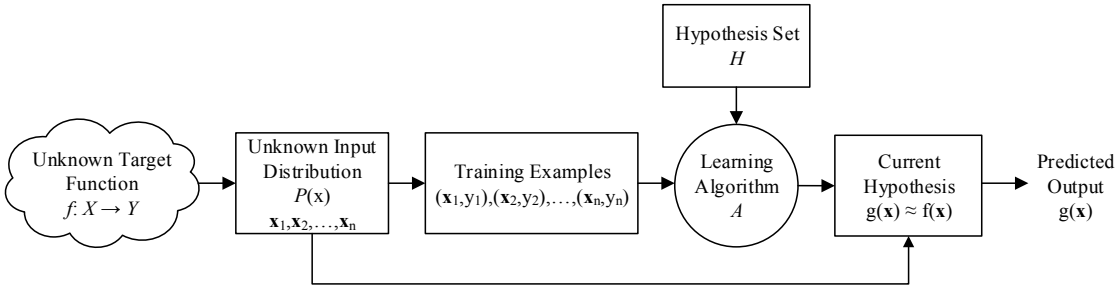

Figure 4. Supervised learning block diagram

algorithm itself, to search for the best approximation function it can find to represent the underlying process exhibited within the examples.

The inductive learning process implemented by the learning algorithm can be explained in more detail. The primary function of the learning algorithm is to search through the constrained space, *H*, of possible hypotheses in order to find a hypothesis for the approximation function that yields the least amount of error. Search heuristics from the domain of artificial intelligence (AI) are used in this process, including various combinatorial and continuous optimization strategies [36]. Meanwhile, general models from the fields of statistics and probability often guide the set of possible hypotheses through which the search proceeds. Hence, machine learning is often considered a combination of AI and statistics along with other fields [32, 39]. The search is constrained based on a set of assumptions incorporated within the algorithm, and these assumptions can vary depending upon the algorithm. These assumptions serve multiple purposes. First, they limit the search space and the complexity of the model so that an approximation can be found in a reasonable amount of time. Secondly, the assumptions assist the algorithm in finding a *generalized* model from the training examples. Generalization is important because tailoring or overfitting the approximation function specifically to the training

examples may not accurately reflect the dependency exhibited among the future data samples. A way of avoiding this potential problem is to restrict the characteristics of the hypothesis. For instance, the hypothesis space could be limited to only linear models. In order to make such assumptions, *a priori* knowledge about the makeup of the data is often required [32]; this ensures the best performance by applying the most appropriate models given the general behavior of the data. It is well-known that every learner must leverage knowledge or assumptions in order to effectively generalize dependencies within the data [36], and this principle is formalized within the so-called "*no free lunch*" theorem that was presented by Wolpert [40].

Linear models tend to generalize well across a wide variety of datasets [32, 41]. Linearity refers to the parameters used in the approximation function to restrict the hypothesis space. As an example, a polynomial regression model could be formally described as

$$g(\mathbf{x}, \mathbf{w}) = w_1 x^n + w_2 x^{n-1} + \ldots + w_0$$

where w_i represents the i^{th} linear coefficient within the polynomial function. In some cases, these weighting coefficients are described using nonlinear functions such as e^{-wx}.

Search heuristics are used to vary these coefficients and to evaluate various forms of the constrained model type. The quality of each attempted approximation function within the search is measured by a so-called *loss function*, $L(y, g[\mathbf{x}, \mathbf{w}])$. For the task of *classification*, the loss function is measured discretely based on the number of errors the model makes in classifying the examples. For instance, in the case of binary classification, the loss function may take the form [32]:

$$L(y, g[\mathbf{x}, \mathbf{w}]) = \begin{cases} 0 \text{ if } y = g(\mathbf{x}, \mathbf{w}) \\ 1 \text{ if } y \neq g(\mathbf{x}, \mathbf{w}) \end{cases}$$

In this case, the loss function increases with the number of incorrect classifications, and thus, the ideal model would be the hypothesis that produces the smallest loss or error. If the target is continuous in nature, then a common loss function for *regression* algorithms is the squared error as defined by [32]:

$$L(y, g[\mathbf{x}, \mathbf{w}]) = (y - g[\mathbf{x}, \mathbf{w}])^2$$

In the process of searching for the hypothesis with the best loss function, it is possible that the search yields a hypothesis that is too customized toward the training examples, and as a result, the approximation function likely will not generalize well with future samples. This problem is known as *overfitting* [32, 41]. There are some common heuristics used during the search to minimize the risk of overfitting. An example heuristic is referred to as *cross-validation* (CV) [36]. With CV, a portion of training examples is withheld during the preliminary search, and then later used to evaluate the model's performance on untrained examples. Another common option is known as *regularization* [36]; in this case, a regularization term is added to the loss function in order to penalize or bias the hypothesis selection. More specifically, regularization allows for models of a higher-degree polynomial to be penalized, thereby favoring less complex models, which tend to be more general and less prone to overfitting.

It is also possible that a search of the hypothesis space may yield multiple consistent hypotheses that have equivalent loss functions. In this case, it is best to prefer the simplest hypothesis. As previously discussed, the less specific model will likely tend to generalize better and avoid the issue of overfitting. Furthermore, the guiding principle of Ockham's razor also supports the decision for preferring the least complex option [42].

Active and Online Learning

There are different variants of supervised learning depending upon the environment and the manner in which examples are presented to the learning algorithm. Typically, datasets are presented to the learning algorithm in their entirety and at the onset of the learning process [38]. However, in some cases such as the intended LQ prediction environment, a pool of training examples are not available at system startup. Instead, samples are presented to the system one after another in an *online* and *streaming* fashion. There exist online learning algorithms that tune the model one example at a time by examining the amount of error between the predicted output and the true target. However, data streams tend to generate an enormous amount of data over time [43, 44], and it may be somewhat expensive to collect labeled samples in this fashion. Therefore, samples within the data stream must be *selectively* labeled. This form of selective sampling is referred to as *active learning* [38, 45]. In active learning, the system queries the supervisor for labels on an as-needed basis. The goal of active learning is to reduce the overhead associated with sampling by only requesting labels which are essential for building an accurate model. There exist numerous active learning heuristics for determining when it is best make a label query [43, 45, 46]. Because the robot LQ application is online and streaming-based, these topics are discussed further in Chapter VI.

Select Classification Algorithms

The fundamentals of some common classification algorithms are reviewed because the classifiers are referenced later as part of the proposed design discussed in Chapter V. Classification arises in supervised learning when the output variable is one of a finite set of values or *classes* such as '*strong*' or '*weak*'. As with any supervised learning problem,

classification uses training examples in the form of ($\mathbf{x}_i$, y_i), where $\mathbf{x}_i = <x_1, x_2, \ldots, x_n>$ is a vector of n features that may be discrete or continuous, while y_i is the discrete output class. If the output contains classes that are ordinarily described using names, such as '*strong'* or '*weak'*, then these terms must be encoded into discrete values such 0 and 1 before entering the training set. The objective of a classifier is to approximate the unknown function $f: X \rightarrow Y$, or equivalently, estimate $P(y|\mathbf{x})$.

The Naïve Bayes classifier is based on the well-known Bayes theorem that is formalized as

$$P(y|\mathbf{x}) = \frac{P(\mathbf{x}|y) * P(y)}{P(\mathbf{x})}$$

where the posterior probability $P(y|\mathbf{x})$ is inferred using the probabilities $P(\mathbf{x})$, $P(y)$, and $P(\mathbf{x}|y)$ [32]. The problem with implementing Bayes theorem directly is that computation of $P(\mathbf{x}|y)$ is complex, especially for larger datasets [32, 39]. Hence, the Naïve Bayes classifier makes the naïve assumption that the features within X are conditionally independent from one another. This assumption reduces the number of parameters needed to estimate $P(\mathbf{x}|y)$ using the training data. However, in reality, there are times when the features are dependent in some fashion. Consequently, the error rate of the NB classifier may suffer more than other algorithms when there is a strong dependency between the input features [32].

In contrast to Naïve Bayesian classification, logistic regression directly estimates $P(y|\mathbf{x})$ and is intended for binary classification problems. Logistic regression can be viewed as a special case of a generalized linear model (GLM) [32]. The challenge with using ordinary linear regression for binary classification is that it would produce probabilities less than zero or greater than one. To overcome this issue, logistic regression describes the

$P(y|\mathbf{x})$ using a special type of sigmoid curve referred to as the logit function that bounds the output between zero and one. The logit function is defined as the log of the odds function $p/(1-p)$, which can be formally described as

$$\log\left(\frac{p}{1-p}\right) = \alpha+\beta_1 x_1+\beta_2 x_2+\ldots+\beta_n x_n$$

where $\alpha+\beta_i x_i$ is some linear combination of the feature space and p is the probability that the output class is one (i.e., $y = 1$). By solving for p in the above equation, the most probable output class can be determined using a decision boundary of 0.5. If p is greater than 0.5, then the prediction would be $y = 1$; otherwise, if p is less than 0.5, then the output prediction would be $y = 0$. This is based on binary nature of the probabilistic problem and that the probabilities of both success and failure must sum together to be a total of one. In general, logistic regression is a simple but powerful classifier [32] that is widely used in a variety of real-world problems [42].

Support vector machines (SVMs) provide another means of making classification predictions using a set of features; the concept of the SVM can also be extended to regression-style problems, but this section will focus on their use in the context of classification. In contrast to Naïve Bayes and logistic regression, the SVM is a non-probabilistic classifier that is based on the principle of structural risk minimization (SRM), where an effort is made to balance the tradeoff between error performance and model complexity [32]. The concept of the SVM is identify a decision boundary that provides the maximum separation distance between the classes. Therefore, the loss function must be modified to include a distance measure that quantifies the margin of separation provided by each hypothesis. The separation boundary is extensible to an n-dimensional space by finding the hyperplane that provides the largest amount of separation between the classes.

An issue arises when the classes are not linearly separable via a hyperplane, but this can be handled by adding a cost factor that penalizes any hypothesis to the degree that each example violates the separation hyperplane. However, the problem with this approach is that optimal separation often requires complex nonlinear models [32]. Fortunately, the SVM mitigates this issue by mapping the data into a higher dimensional space. A sufficiently high-ordered feature space enables the training examples to be linearly separable, but there are some computational costs associated with the mapping and learning processes [32]. To reduce these expenses, it is possible to find a higher dimensional hyperplane and to classify features without explicitly representing the entire feature space; this is accomplished through a *kernel function* [32]. Two common kernel functions include the polynomial and the Gaussian. The Gaussian version is part of a larger subgroup of kernels referred to as radial basis functions (RBFs), which only depend on the geometric distance between the features and the classes [32].

In most real-world scenarios, the RBF model is the preferred choice over the linear or polynomial kernel [32]. In contrast with the linear model, the RBF kernel can handle the separation of highly nonlinear classes. Furthermore, the RBF model is often less complex than the polynomial due to it having fewer parameters. Choosing the best kernel for a given dataset often involves experimental testing, but regardless, the SVM has proven effective in numerous applications and has often been found to outperform other classification methods [32].

Generalized Linear Regression

An inductive learning strategy referred to as *regression* is used when the system output is continuous in nature, as opposed to categorical. Regression is a well-known statistical technique that attempts to reduce the residual error of the fitted approximation using the method of least squares [47]. In fact, regression is the most prevalent form of any prediction model [32]. Generalized linear regression (GLR) models follow the linear expression previously discussed in the *Supervised Learning* section, where the w_i terms represent the regression coefficients, which are solved for using the method of least squares. With regression, it is assumed that the training examples are influenced by some amount of noise, which is likely the result of hidden independent variables that cannot be feasibly measured. Consequently, the regression estimate will not fit the training examples perfectly. In this case, the quality of the fit can be quantified by finding the residuals defined as

$$R_i = y_i - g(\mathbf{x}_i)$$

where y_i is the true system output and $g(\mathbf{x}_i)$ is the regression approximation for the output based on the input vector $\mathbf{x}_i$. Further analysis into a given regression estimate is often accomplished through a statistical procedure referred to as the analysis of the variance (ANOVA) [32]. A useful product of ANOVA is that it can provide machine learning algorithms a means to perform feature reduction. More specifically, it is possible to identify the weakest inputs using an iterative process of comparing the variance of the model's residuals for every combination of predictor inputs. From this information and some simple *F*-statistics, the least informative terms become apparent, and thus, the

associated coefficients of these inputs can effectively be set to zero to reduce model complexity.

Conclusion

In this chapter, an overview into some key concepts within fuzzy logic and machine learning were presented. The research presented in the subsequent chapters is built upon these fundamental domains. Thus, the overview material contained within this chapter served to prepare the reader for these future concepts. In subsequent chapters, it is assumed that the reader possesses this background information.

CHAPTER III

LITERATURE SURVEY

Introduction

Several higher-layer applications, such as those listed in Table 1, require periodic assessments on link quality (LQ) in order to make optimization decisions. As the table conveys, the capability to gauge and optimize LQ can lead to enhanced energy efficiency, network capacity, fault tolerance, and location awareness.

Table 1

Examples of Applications that Rely upon Link Quality Assessments

Application	**Example**
Robot - Formation Control	[6]
Robot - Communication Area Sensing	[2]
Robot - Radio Source Localization	[8]
Robot - Automated Relay Deployment	[12]
Robot - Multi-Radio Control	[7]
Localization - Distance Estimation	[48]
Localization - Direction Finding	[49]
Network - Routing	[50]
Network - Transmission Rate Adaptation	[51]
Network - Transmission Power Control	[52, 53]

The term *link quality* (LQ) generally refers to some target variable that is a derivative of throughput or reliability. There is no standard definition for LQ, nor standard unit of measure for the quantifier [54]. In fact, it can either be a qualitative or quantitative description about a link, depending on the context in which it is used. In most cases, LQ is generally expressed in a probabilistic sense of how reliable the link is in terms of *past* or *expected* packet delivery, and naturally, the measure of such a probability would range from 0 to 1. Additionally, link conditions are sometimes categorically described using linguistic terms such as '*good*', '*intermediate*', or '*bad*', and such language may even be used by LQ systems as part of their estimation or prediction process (e.g., fuzzy systems). Generally, the measure of LQ depends upon the application and its intended use.

Unfortunately, evaluating LQ at layers above the physical is challenging due to the underlying dynamics of wireless propagation and the mismatched temporal perspectives between the layers. Due to its relevance and difficulty, a significant research effort has been devoted to the field that is commonly referred to as '*Link Quality Estimation'* or '*Link Quality Prediction'*, depending upon whether estimates or predictions are being formed. In general, there are two empirical-based approaches to assessing LQ at a higher layer: *prediction* or *estimation*. An LQ prediction is a time-dependent forecast on the future of LQ, but its relevance is limited in time due to the rapidly changing channel conditions. On the other hand, an estimate is a rough approximation of the current state of LQ based on recent observations. Consequently, an LQ estimate is less time sensitive than a prediction and can remain relevant over a spatial area of several wavelengths.

The empirically-based methods for estimating or predicting LQ at the upper layers is different than the existing techniques used at the physical layer. For instance, in some

cases, the physical layer has the capability to exploit so-called *instantaneous* channel state information (CSI). The concept with instantaneous CSI is to exploit the short timeframe where the response of the wireless channel is mostly flat or invariant. More specifically, the physical-layer scheme calls for a training sequence to be transmitted to a receiver so that it can estimate the channel gain matrix in the forward direction; immediately afterwards, this information is sent back to the sender for transmission adaptation prior to the channel conditions changing. However, even at the physical layer, the round-trip delay associated with this action is non-negligible [19], making instantaneous CSI unrealistic for the upper layers. Another approach taken at the physical layer is to model the channel statistically, referred to as *statistical* CSI [19]. However, this information is not in terms of LQ metrics that are directly beneficial to the upper layers. For instance, statistical CSI is generally related to the fading distribution, average channel gain, spatial correlation, and others [19]. On the other hand, applications outside the physical layer likely prefer more straightforward statistics that are directly related to the probability of packet reception or impending throughput potential. Another problem is that CSI is not supplied to the upper layers. The higher layers only have access to select physical layer metrics, such as received signal strength indicator (RSSI) [27]. These select metrics are defined by the radio protocol and supplied to the data link layer for special purposes such as medium access control [55]. Fortunately, higher layers can gain access to these types of metrics in an efficient manner through the operating system and radio driver software [26].

The goal of this survey is to provide a comprehensive overview of the various methodologies used by the upper layers for LQ estimation and prediction in wireless networks. The survey covers works from a variety of network types including ad hoc,

mesh, and wireless sensor networks (WSNs), in contrast to a survey from 2012 that focused exclusively on WSNs [20]. Another distinction between surveys is that this newer one covers the latest developments in the field that relate to the critical elements of learning and adaptability, which is deficient in [20].

Preliminaries and Challenges

Link Asymmetry

Wireless communication links are bi-directional in nature as illustrated in Figure 5. The issue of *link asymmetry* refers to when LQ in one direction of the link differs from the other. The common cause of link asymmetry is usually related to a mismatch in antenna types (i.e. gains) or transmit powers between the two ends of the link [20, 56]. However, as the figure indicates, link asymmetric may also arise when the levels of noise and interference are significantly different in each respective geographic location.

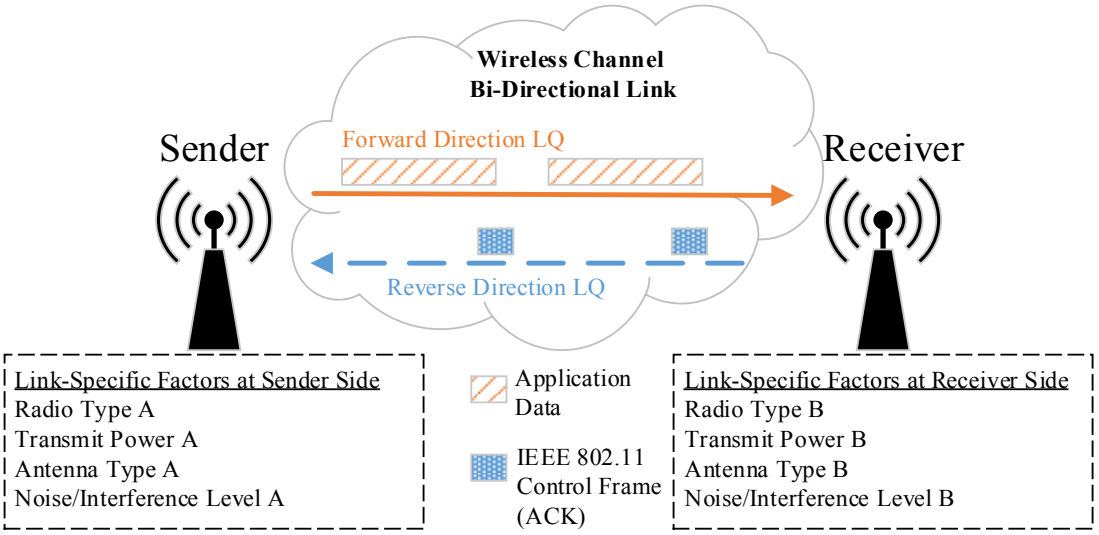

Figure 5. Bi-directional link factors that may precipitate link asymmetry

A holistic description of LQ would consider the conditions of both directions of the link, but this can be difficult and expensive for an upper layer. The problem stems from the fact that a sender is natively unaware of the receiver's LQ conditions in the forward direction of the link. Thus, a sender must be explicitly provided information about its forward direction LQ, but this is costly in terms of energy and channel capacity overhead. Furthermore, the LQ conditions at the receiver are changing rapidly due to the underlying dynamics of wireless communication. Consequently, any feedback mechanism from the receiver to the sender must be processed expeditiously, but in reality, the short timespan that the wireless channel remains stationary may prove too restrictive for many upper layers.

To mitigate these costs and constraints for the higher-layers, it would be better if a sender could infer the forward direction LQ with fewer explicit queries, and ideally, through more readily available and less expensive metrics. For example, the forward and reverse directions of a link are usually highly correlated. Thus, it should be feasible to use LQ metrics related to the reverse direction to infer LQ in the forward direction, assuming the relationship is updated regularly to mitigate any statistical drift. In order to establish a statistical dependency between the reverse and forward directions, it would still require some occasional feedback for the purpose of obtaining labeled training examples. But fortunately, the volume of feedback would be reduced once the general statistical relationship has been establish.

Coherence Time

The hidden and underlying function between the LQ metrics of the forward and reverse directions is dependent upon time. In mobile networks, the fading characteristics of the wireless channel are time-varying as a result of the transmitter or receiver mobility [16]. The coherence time, T_c, of the channel describes the time period in which the channel response or fading is essentially invariant [17]. Therefore, the metrics from both directions should be sampled and paired together within this stationary period of time of less than T_c; otherwise, the samples will have undergone independent fading, which would convolute any attempt to establish dependency between the metrics.

To better understand T_c, it is helpful to look at the effect of mobility on signal propagation in the frequency domain. Mobility introduces a phenomenon known as *Doppler shift* into received signals [17]. Due to multipath, reflected signals may travel along different paths and arrive at different angles [17]. Each reflected copy likely has a different Doppler shift, and the aggregation of the reflected signals results in a Doppler spreading of the transmitted signal [17]. Assuming movement at a constant velocity, v, the magnitude of the maximum Doppler shift component can be described as [17]

$$f_d = \frac{v}{\lambda}$$

where λ is the wavelength of the transmitted signal. The time domain equivalent of Doppler spread describes the coherence time of the channel. The relationship between the frequency-domain and time-domain equivalents can be approximately (within a multiplicative constant) related by [17]

$$T_c \approx \frac{1}{f_d}$$

A more precise relationship can be defined by specifying that if the channel response of two time-delayed signals is correlated to a level of 0.5 or greater, then the approximation becomes [17]

$$T_c \approx \frac{9}{16 * \pi * f_d}$$

However, it is common to use the geometric means of the two previous equations to estimate the coherence time as [17]

$$T_c \approx \frac{0.423}{f_d} \approx \frac{0.5}{f_d}$$

If equation v / λ is substituted into the above equation for f_d, then the coherence time can be described as

$$T_c \approx \frac{0.5 * \lambda}{v}$$

which shows that the coherence time window narrows as antenna velocity increases.

Any LQ assessment system should be generally aware of T_c for several reasons. First, T_c states the time dependency between the inputs and output of any LQ system. Therefore, input-output pairs must be sampled together within this threshold in order to generalize an accurate statistical dependency between them. Furthermore, any LQ prediction based on a set of input samples taken at time instance, t, will only remain relevant or most accurate during the period of $t + T_c$. After that time, the radio wave signals are subject to independent fading in terms of small-scale variation due to the effects of multipath, and thus, any LQ prediction losses its accuracy once T_c has expired. Finally, T_c can also serve as a guide to how frequently samples should be collected. More specifically, samples collected at intervals greater than T_c are statistically independent. Thus, to avoid oversampling and to conserve resources, samples should be collected at a periodic rate near

1 sample every T_c seconds. This would also ensure that samples are collected in an independent and identically distributed fashion, which is a requirement for extrapolating an unbiased statistical dependency [38, 57].

Temporal Mismatch

It can be challenging for an upper layer application to exploit the narrow timeframe that a prediction remains valid. The primary constraint is that an application has several actions to perform within a short time span. Specifically, after rapidly sampling the LQ features and making a prediction, an application must complete some proactive decision, as well as its transmission, before Tc expires. In reality, this constraint limits the applicability of LQ prediction at the upper layers [58]. Only select and rapid processing applications at the network or data link layers (e.g., routing protocols, rate adaptation, and power control) may be able to meet the time constraint.

In other cases, applications may perform processing decisions more slowly, and thus, predictions may be of little value given the temporal perspective of the application. An example of this time scale difference between layers could be a robot navigation system; significant attenuation change, or chances of link failure, likely occur on the order of seconds, in contrast to the underlying small-scale fluctuations in LQ that are pertinent to prediction.

In summary, some applications may not have a need for predictions, while others could benefit from the short-lived forecasts on LQ. Therefore, it is important for LQ systems to make the connection between prediction and estimation, and be able to generate both for maximum versatility. It is possible to easily form LQ estimates from the output of a LQ prediction system by smoothing or averaging a recent series of past predictions.

Accuracy Limitations

LQ is generally quantified at the higher layers statistically based on past observations. In other words, sets of empirical measurements are collected, and then statistical dependencies are formed between the input and output variables. Similar to many empirically observed phenomena [32], any pool of LQ samples collected at the higher layers will inevitably contain some level of noise. This noise or unexplained variance is an artifact of the limited number of cost-effective inputs available at the higher layers that can describe the underlying effects of multipath and other wireless phenomena. In addition, there are factors within the network protocol stack, besides those at the physical layer, that may impact LQ in some fashion; however, several of these factors are likely unavailable for measurement and input into a LQ gauging system. For example, a data link or transport layer protocol may throttle the rate of transmissions at times for flow or congestion control purposes [21], but the flag indicators associated with these mechanism are natively hidden from the application layer. In summary, the upper layers only have access to select inputs that are correlated to LQ. Consequently, the empirical samples will contain some level of noise or unexplained randomness. The purpose of statistical inference is to generalize a relationship between the input-output samples that minimizes the noise within the samples, but inevitably, most estimates or predictions will contain some level of error.

The Fundamentals of Modeling Link Quality

General Methods

There exist some general methods for modeling LQ using empirical measurements as shown in Figure 6. One approach involves analytical modeling, where complex theoretical models are used to approximate the behavior of the wireless channel and its random fading models probabilistically. The use of probabilistic models is commonly necessary because exact mathematical equations are difficult to obtain due to the time-varying and unpredictable nature of radio propagation [18]. In the analytical approach, the models are typically simplified to capture to the underlying dynamics of path loss, shadowing, and multipath fading [18]. The various parameters of these propagation effects are then approximated using measurements taken from the channel. Generally speaking, analytical models do not directly provide a lucid indicator of LQ for upper layer applications; instead, some translation from the models is required to obtain an indicator that is easily distinguishable and that relates to LQ in the sense of throughput or reliability. Hence, analytical modeling is a more indirect approach to modeling LQ.

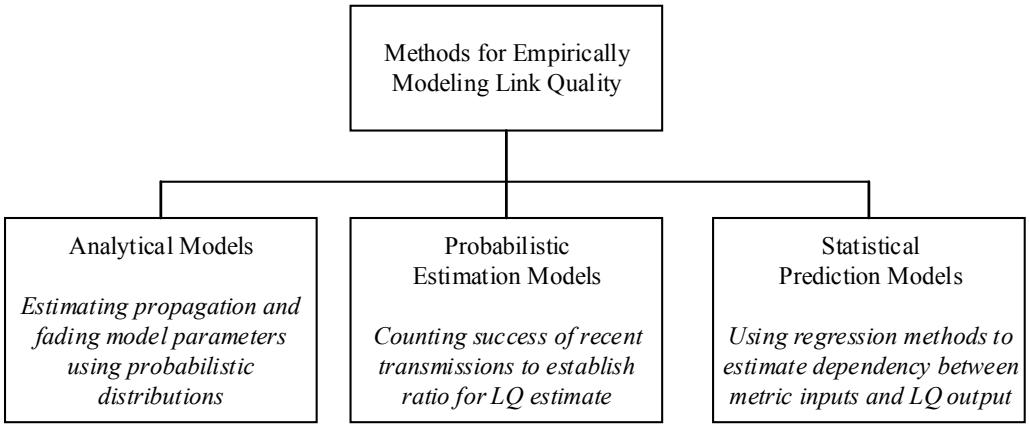

Figure 6. Empirical-based approaches to modeling link quality

A series of works have taken the analytical approach to LQ estimation [18, 31, 59], and the authors propose a probabilistic framework for predicting the spatial variations of the wireless channel using minimal measurements. However, the models are complex and based on several assumptions, making the approach challenging to implement in reality. Furthermore, the application of these sorts of models is typically limited to static or quasi-static scenarios, and the robustness of these models under more dynamic and dense networks may prove challenging [29]. Finally, the information returned by these models is not in a form that can be easily interpreted by higher layer applications, such as routing engines, which need a simple measure to quickly distinguish between link options.

The more common and straightforward approach to assessing LQ at the higher layers is to use one of the two remaining modeling options displayed in Figure 6. The ratio-based approach of counting the success of recent transmissions is the simplest form of LQ estimation. It involves periodically testing (i.e. probing) the channel directly, and based on the outcome of a series of trials, a simple ratio is formed that probabilistically describes the expected chances for packet delivery in the near future based on recent observations. On the other hand, the statistically-based approach uses link features that are empirically measured and are statistically correlated to LQ in some fashion. The relationship is exploited by mapping the empirical measurements to LQ using an approximate mapping function discovered through regression analysis. By far, the majority of works in the literature use one of these two methods for LQ estimation, and therefore, this survey will primarily focus on distinguishing these works, as opposed to analytical models.

Common Target Metrics

As previously alluded, the objective in most environments or applications is for a sender to have a cost-effective means to estimate or predict its LQ in the forward direction. LQ is usually defined as some ratio or statistic that varies from 0 to 1 and relates to the link's reliability in a probabilistic sense. These statistics are commonly referred to as *logical* metrics because they intuitively or directly describe LQ from the perspective of a higher layer [22]. Hence, logical metrics are often treated as the target or output variable.

Table 2

Common Target Metrics

Metric
Frame Delivery Ratio (FDR)
Frame Error Ratio (FER)
Packet Delivery Ratio (PDR)
Packet Reception Ratio (PRR)
Packet Error Ratio (PER)

Table 2 lists some of the common logical metrics referenced in the literature. As indicated by their names, the metrics differ based on where the statistics originate. For instance, the *frame* ratios are built using statistics gathered from the medium access control (MAC) layer, while the *packet* statistics are based on observations made from the network layer. The seemingly subtle differences in nomenclature between the metrics and their measurement locations can actually result in significant differences in values between them [60]. For instance, the reliability and control mechanisms built into the medium access control (MAC) of IEEE 802.11 ensures that unsuccessful frames are retransmitted in accordance with the protocol, and these retransmissions are hidden from the network layer. Therefore, the packet delivery ratio (PDR) metric will tend to be higher (i.e., report higher

reliability) than the frame delivery ratio (FDR) metric due to the perspective differences between counting *frame* and *packet* transmissions [60]. The other significant difference between the metric ratios provided in Table 2 is whether the ratio measures *delivery* or *error* rate. Because the ratios are success or failure rates in a probabilistic sense, FDR and PDR is related to frame error rate (FER) and packet error rate (PER), respectively, by the following:

$$FER = 1 - FDR$$

$$PER = 1 - PDR$$

It should be noted that PDR and packet reception ratio (PRR) measure the same statistic, but the difference in semantics arises due to the bi-directional nature of wireless links and the location where the statistic is being measured (i.e., sender ~ *delivery* and receiver ~ *reception*).

Empirically-based Methods

Active Packet Counting

Some LQ estimation methods call for the use of *active probes* and *packet counting* to estimate LQ using a delivery or reception ratio. The term *'active'* refers to the fact that these schemes rely upon dedicated probes, not application data, for establishing an LQ estimate. The basic concept is for the transmitter and/or receiver to periodically transmit probes to the other end of the link, and then build a ratio that describes the percentage of success over some pre-defined window.

The first work to advocate this approach was presented by De Couto et al. in [61, 62]. The target statistic defined in these works is referred to as "*Expected Transmission*

Count (ETX)". ETX is a combination of the delivery ratios from both directions of the link, and more formally, it is defined as

$$\text{ETX} = \frac{1}{d_f * d_r}$$

where d_f and d_r are the delivery ratios in the forward and reverse directions, respectively. In general, packet (or frame) counting ratios, such as d_f and d_r can be defined as the number of delivered packets (or frames) divided by the total number of attempts made over a pre-defined window, w. Or mathematically, a generic packet counting ratio, R_{pc}, can be described as

$$R_{pc} = \frac{p_{i-1} + p_{i-1} + \ldots + p_{i-w}}{w}$$

where p is equal to 1 if the i^{th} packet was delivered successfully and zero otherwise. Typically, the probing frequency is set to every second and the window size is set to 10 seconds, which was also used in the evaluation of ETX.

After the introduction of ETX, several studies highlighted various deficiencies associated with the metric. Draves et al. found that the ETX statistic is somewhat inaccurate because it does not reflect the actual packet size of data transmissions, nor the bandwidth of the channel [63]. Hence, Drave et al. suggested a modified version of ETX called "*Expected Transmission Time* (ETT)" that can be described as

$$\text{ETT} = \text{ETX} * \frac{L}{B}$$

where L is the length of the packet and B is the bandwidth of the channel. However, Kim et al. noted that if broadcast probes are used to measure ETT as suggested in [63] and if the sizes of the probes remain fixed, then ETT adds no new information to ETX because broadcast transmissions are made at the physical standard's lowest rate [64]. Additionally, Biaz et al. discovered that ETT fails to consider forwarding delays, as well as differences in link loss rates, along multiple hops of a route [23]. Thus, they suggested a modified and more complex version of the metric called '*improved ETT'* or iETT. However, the metric still inherits the challenge of obtaining the bandwidth along each link, in addition to the other parameters called for in iETT.

All of the aforementioned routing metrics [23, 61, 63] use broadcast transmissions for forming their statistics. According to IEEE 802.11, broadcast probes are transmitted at the lowest data rate of the physical layer specification, unlike unicast transmissions that may be transmitted at higher data rates [65]. Transmissions at lower data rates are more likely to be received successfully because they use a more robust modulation scheme. Therefore, these metrics (i.e., ETX, ETT, and iETT) are somewhat inaccurate and may overestimate the quality of the link by calculating their packet delivery ratios based on the delivery success of broadcast messages that are inherently different than unicast data traffic [65]. To address this issue, Qi et al. proposed using unicast packets to send probes at the actual transmission rates of data traffic [65]. However, sending unicast probes is not scalable and would incur significant overhead in terms of bandwidth and energy consumption as the number of nodes in the network increases [66]. For instance, the traditional ETX metric requires a total of n recurring broadcasts within a network of n nodes. On the other hand, a unicast version of ETX would require nearly double

(i.e., $n*(n-1)$) the number of recurring transmissions. As another drawback to the standard ETX metric, Qi et al. revealed that the broadcast rate of every second with a window size of 10 seconds is insufficient for accurately tracking dynamic packet loss rates across a link. Simulation results from [65] show that probes should be sent much more frequently and at a minimum rate of every 25 milliseconds (ms) in order to be more responsive and better track the actual link loss. But, such a drastic modification to ETX would be expensive in terms of bandwidth and energy expenditure.

Another issue with ETX performance was discovered by Tran and Kim [67]. Specifically, the study shows that the accuracy of ETX degrades as the traffic load (i.e., network density) increases. In dense networks, broadcasts are more susceptible to collisions from hidden nodes due to them not using the request-to-send (RTS) / clear-to-send (CTS) mechanism of IEEE 802.11. In addition, the flooding of route requests (RREQ) packets and the questionable fairness of the IEEE 802.11 MAC under heavy load conditions appear to also affect the performance of ETX.

In summary, most of the LQ estimators intended for routing applications utilize some form of probe-based LQ estimation. The primary advantage of probe-based methods is that they are simple and require little prior knowledge [25], other than the sender and receiver sharing a mutual understanding of the metric and its exchange protocol. However, as previously reviewed, there are several deficiencies with the fundamental approach.

There are a couple of other issues with sending probes that the literature appears to have overlooked: energy consumption and network utilization. In terms of energy consumption, the topic is barely mentioned in the literature, but Zhang et al. did highlight that probing links can be inefficient at times in sensor networks because they tend to

undergo extended periods of inactivity, making probes unnecessary during these times [68]. The challenge with precisely quantifying the energy cost and impact of sending probes is that it depends upon a number of variables, including the size of the probe, its transmission frequency, and the type of application (e.g., battery-powered sensor). As for the impact that probing has on network capacity, it appears the issue has never been fully investigated. Therefore, a network simulation was created using NS-3 [69] to take a deeper look into the effects of probing on single-channel throughput.

The simulation results of Figure 7 show that probing has an adverse effect on network throughput, especially in denser networks where more nodes are transmitting probes. The level of impact on channel throughput appears to be somewhat random at times as indicated by the occasional sharp reductions in throughput. These anomalous deviations in throughput are likely a combination of byproducts stemming from the MAC used in IEEE 802.11, as well as the protection mechanisms built into Transmission Control Protocol (TCP). Both of these protocols were used in the simulation, and both contain random back-off components when contention or congestion is detected [55, 70]. Overall, the generally increasing trend in throughput reduction is the result of nodes having to increasingly share the frequency channel as more nodes begin sending probes.

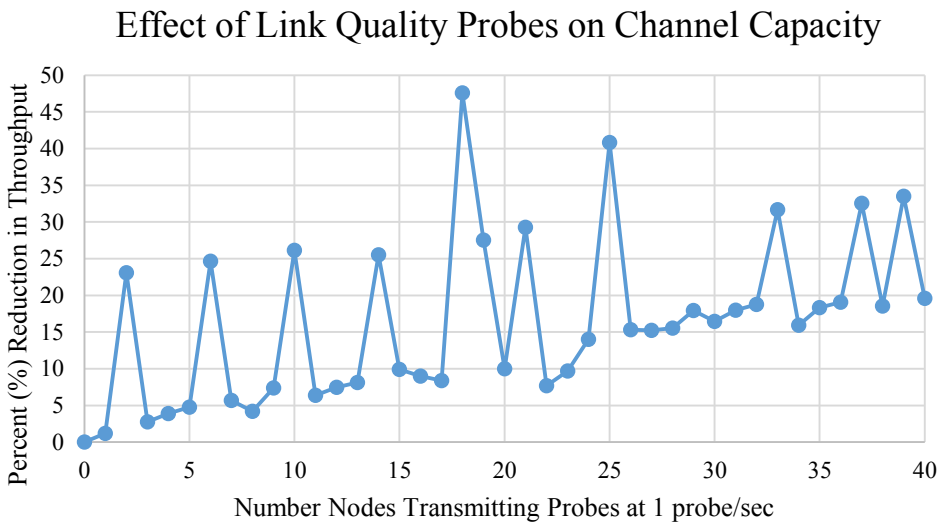

Figure 7. The impact of link quality probes on channel capacity

Other details about the simulation are as follows. The baseline throughput was based on the amount of time it took a sender to transmit one megabyte (1 MB) of data via TCP to a single receiver spaced 100 meters away, and meanwhile, no probes were transmitted during this baseline trial. The transfer size of 1 MB was selected in order model the conditions of a dense or busy network. After the baseline test, all subsequent trials repeated the same process of transmitting 1MB of data, except that additional nodes were introduced for the sole purpose of transmitting periodic probes. Each node that was added transmitted probes every second via user datagram protocol (UDP), and the size of each probe was set to 1200 bytes; both parameters were chosen based on the evaluation of ETX in [62]. The probe-sending nodes were added in a grid-like fashion around the baseline pair so that they were within the energy detection range of each other. Other pertinent simulation details are outlined in Table 3.

Table 3

NS-3 Simulation Parameters

Parameter	Setting
WiFi Standard	IEEE 802.11b
Physical Mode	DSSS Rate 11 Mbps
WiFi MAC	Ad hoc
Energy Detection Threshold	-81 dB_m
Tx Power	20 dB_m
Propagation Delay	Constant Speed
Propagation Loss Models	Friis model combined with Nakagami model
Mobility	Constant Position Mobility Model
Max. Detection Range	~650 meters

Passive Packet Counting

The accuracy, responsiveness, and overhead concerns of sending active (i.e., forced) probes for LQ estimation motivated several researchers to establish packet counting ratios using passive methods [68, 71-75]. Instead of forcing traffic at the network layer, the concept is to passively observe and count data transactions occurring at the MAC layer. In essence, these schemes calculate probabilistic statistics similar to ETX, but the ratios are established by monitoring transmit (TX) and acknowledgement (ACK) indicators through the MAC layer. When compared to active probing, the passive approach conserves bandwidth and energy, while also speeding up by the responsiveness of the statistics by moving down the protocol stack and observing more frequent data exchanges.

Forming ETX-like statistics without forced probes appears promising, but there are several factors that affect its accuracy and ease of implementation. First, passive packet-counting methods depend upon consistent and steady-flows of application data across the links in order to form statistics. Without data traffic, the statistics become stagnant, and

consequently, become more untimely and inaccurate as the gaps between link transmissions grow. Therefore, there is no guarantee that the passive metrics will be up to date when needed due to their dependency on consistent application traffic. To mitigate this issue, Zhang et al. proposed a hybrid approach where active probes are transmitted during idle periods, but the scheme still suffers from the drawback associated with accessing the MAC layer parameters necessary to passively count packets [66]. The problem is that the transmission and acknowledgement flags embedded within the MAC layer are not readily available without system and radio driver modifications. Some schemes call for changes to the kernel in order to access this type of fine-grained link information [68]; however, such approaches are considered inefficient and not scalable [26]. A significant challenge is that customized modifications to device drivers and system configurations are not portable across a wide range of heterogeneous systems and radios that makeup most networks. Furthermore, the processing and latency impacts of these system and radio alterations merit further investigation. As a specific example, research by Kolar et al (2011) revealed that logging overhead and buffer overflows under heavy traffic loads likely caused sporadic packet loss during testing [26], which is similar to the observation from [68]. In conclusion, there are several concerns about the practicality of passive link monitoring.

Mapping Radio Metrics to Link Quality

In addition to packet counting schemes, another common method of gauging LQ involves the use of *hardware* metrics. Hardware metrics are LQ indicators that are measured at the physical layer and supplied by the radio driver to the upper layer protocol functions, as well as the operating system. The radio metrics can be easily monitored by applications via the */proc* file system in Linux [26], but the availability of the metrics is vendor-specific [27, 76]. Additionally, vendors may also perform proprietary filtering and scaling of the hardware metrics prior to supplying them to the operating system [27].

There are several advantages associated with using hardware metrics. First, they are relatively low-cost when compared to the probing overhead associated with packet counting schemes [22]. Hardware metrics are more efficient because they are updated passively from management feedback that is inherent with protocols such as IEEE 802.11 [55]. An additional efficiency is that they are readily-available and do not require any complex driver modifications to access [26]. Secondly, hardware metrics facilitate faster LQ assessment than packet counting schemes. For example, hardware metrics have been shown to track changes in LQ faster than packet counting [25, 77].

On the other hand, there are some challenges to using hardware metrics effectively. One drawback relates to the behavior and scaling of the metrics being specific to the vendor and radio [22, 27]. Additionally, some hardware metrics have been shown to exhibit some inaccuracies at times as a result of anomalous conditions and the way the metrics are measured at the physical layer [22, 27]. Finally, the hardware metrics tend to be somewhat noisy because they are closely coupled with the rapidly fluctuating dynamics of wave propagation.

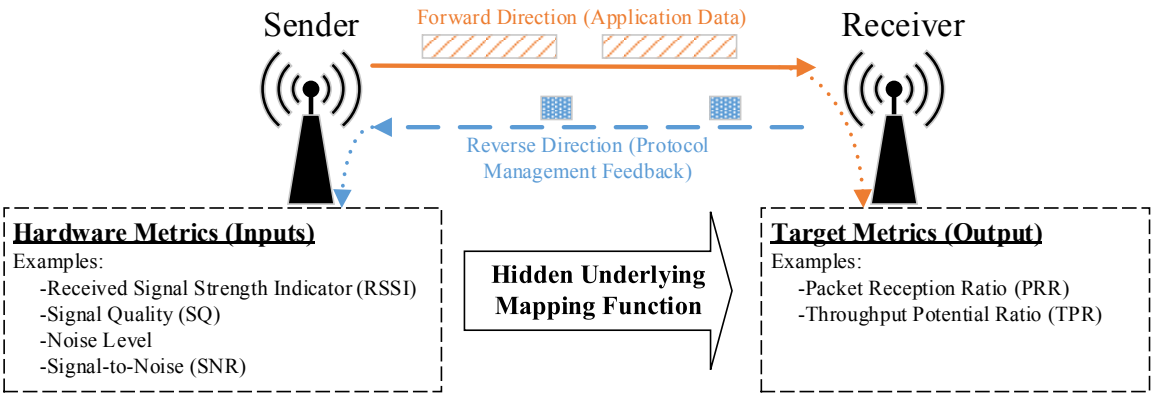

Figure 8. Illustration showing link environment where forward direction LQ statistics are inferred using reverse direction radio measurements.

Therefore, in order to leverage the physical-layer metrics, additional processing is required. The processing includes possible filtering, as well as a mapping function that statistically relates these fluctuating or noisy metrics over to a desired target variable such as PRR. The concept of mapping the hardware metrics over a more intuitive measure of LQ is depicted in Figure 8. The figure shows the scenario where a sender is attempting to estimate or predict LQ in the forward direction using the hardware metrics available from its radio. Some common hardware metrics referenced in IEEE 802.11 [55] and the literature are listed underneath the sender. As the figure implies, the radio metrics are updated passively from protocol traffic coming in the reserve direction. The challenge is that the relationship relating the hardware metrics to the target variable is not known *a priori* and must be somehow discovered.

Several authors suggest discovering these mapping functions through *offline* experimentation and some form of statistical regression fitting [25, 60, 64, 77-79].

However, there are a couple of significant issues with the general approach taken by these works. First, offline experimentation is time-consuming and only reflects the specific environmental conditions observed during experimentation. Secondly, these particular schemes do no attempt to revalidate or relearn the mapping relationship while the system is online, and instead, the mapping relationship is assumed to maintain its accuracy indefinitely. However, in reality, the link environment (e.g., surrounding noise, interference, obstacles, etc.) may change over time. Additionally, mapping functions are link-specific and dependent upon the radio hardware in terms of chipsets, transmit powers, and antenna gains [20]. Other factors are also known to cause radio metric variance including scaling differences, vendor-specific smoothing (i.e., filtering), and calibration imbalances [15, 27]. Therefore, adaptable approaches that learn and update the relationship online are needed.

Some authors have attempted to mitigate the aforementioned lack of adaptability, but issues still remain. Zhang et al. proposed an online calibration technique that initially probes the channel and then builds a piecewise linear approximation of the correlation function using *a priori* knowledge about the sigmoid shape of the mapping curve [51]. To mitigate the impact of interference and drift, the approximation function incorporates 10% safety margins near the high and low thresholds (i.e., knee points or transitions) of the curve. Although it eliminates the need for offline measurements, the margins may over- or under-estimate the expected interference level. Therefore, it would be better to have an online or more correlated means to detect interference, not a fixed 10% margin. A different approach by Judd et al. (2008) proposed a tunable scheme for selecting a transceiver's modulation rate based on past signal-to-noise ratio (SNR) threshold observations [80].

Specifically, the protocol tracks whether packets succeed or fail at a particular selected rate, and then records the SNR difference between the threshold and the actual measurement in a histogram. Based on the histogram frequencies, the protocol adjusts the SNR thresholds every few seconds. For the protocol to work, it assumes that the success or failure of transmission is known by a sender. However, this information may be hidden on a per-packet basis without MAC layer monitoring, which is not transparently available as previously discussed.

Hybrid Techniques

All of radio mapping schemes from the previous section rely upon a single hardware metric as an input. The problem with this approach is that research has shown that a single metric is insufficient in accurately quantifying LQ [20, 27]. To mitigate this issue, several authors have proposed hybrid schemes that effectively combine multiple metrics in order to strengthen the accuracy of the link projection and to offset deficiencies that one metric may have. For instance, Zhou et al. propose supplementing passive packet counting with RSSI whenever the link becomes idle [81]. Another approach by Boano et al. forms a hybrid metric, known as the triangle metric, using two hardware metrics from IEEE 802.15.4 radios. More specifically, the values of the hardware metrics serve as the perpendicular legs of a right triangle, and the magnitude of the resulting hypotenuse serve as the hybrid output indicator [82].

In contrast to custom combinations of metrics, several works related to LQ estimation and routing in wireless sensor networks (WSNs) utilize fuzzy logic for weighting the input of multiple metrics. For instance, the fuzzy-based estimator proposed by Ko and Chang uses the metrics of expected number of transmissions (ETX) and symbol

error rate (SER) variance, and combines them with RSSI to determine LQ [83]. Similarly, Baccour et al. use fuzzy logic to combine four inputs that are related to packet delivery, link asymmetry, stability, and channel quality [54]; all of the inputs into the estimator are PRR-based statistics, except for the channel quality input, which is a moving average of SNR. Lastly, Guo et al. designed a fuzzy-based LQ estimator that uses three logical metrics, which includes PRR, coefficient of PRR variance, and distribution correlation [84].

All of the hybrid estimators that were previously mentioned use some form of packet counting as an input. Therefore, these estimators tend to inherit all or some of the previously-mentioned disadvantages associated with packet counting, despite taking a hybrid approach. Furthermore, all of these schemes lack adaptability, and instead, use hardcoded parameters or settings that are based on the results of offline experiments. The fuzzy-based methods may be somewhat resilient to minor concept drift, but more precise countermeasures are needed to maintain higher accuracy against more significant and inevitable drift.

Learning Methods

Learning algorithms automate the process of learning from data [36], and thus, they possess the potential to discover statistical dependencies between input and output metrics without the need for offline experimentation. Furthermore, machine learning can also perform incremental and online learning, meaning that the mapping function can be tuned over time using observations made while the system is online. This capability significantly reduces the risk of concept drift, which is possible, for instance, if the noise or attenuation

levels persistently change in the wireless channel. The following subsections review various forms of learning or statistical prediction of LQ.

Time Series Analysis

Some works use different forms of time series analysis for predicting LQ. In time series analysis, the ordering of previous LQ observations (e.g., PRR, ETX, etc.) matters, and a finite window of ordered past observations is used to generate future predictions. For instance, Liu et al. use a weighted sum of ordered past observations of PRR to forecast its future value [85]. However, the past observations of PRR are based on estimates formed from two hardware metrics, and the statistical mapping functions of these metrics to PRR are not made online.

In another study, Farkas et al. employs pattern matching to predict LQ [86, 87]. More specifically, current SNR trends are compared with historical time series recordings of SNR in order to find the best match. Predictions are made by performing the cross-correlation of a recent SNR sequence with the previously stored patterns, and the pattern with the highest correlation is predicted as the future LQ state. The scheme is based on the assumption that link behavior follows patterns. However, this assumption does not hold in all networks. Nodes may not continue to operate within the same spatially confined area, thus causing some patterns not to repeat. In reality, pattern prediction would be difficult based on the wide variability in pattern possibilities. Furthermore, the continuous processing of cross-correlation with numerous patterns may be too resource consuming and lagging to be pragmatic.

Another work by Millan et al. uses a software framework that supports taking a machine learning approach to time-dependent data [88]. In essence, the framework

enabled the time dependency of the data to be encoded via additional input fields, referred to as *lagged* variables, and by doing so, the input data (i.e., ETX) can be processed by a standard learning algorithm. In this work, Millan et al. experimented with various machine learning algorithms and different-sized lag windows in order to identify the combination with the best prediction performance. The results indicate that the regression tree (RT) algorithm performed slightly better than three other evaluated algorithms. On the other hand, the best lag window size of previous ETX instances was statistically uncertain. The study also investigated the impact of the training set size and found that more training samples tended to improve prediction accuracy. Finally, the authors concluded that it is important to retrain the model periodically based on the offline model losing accuracy as time progresses. Therefore, an online learning algorithm that progressively updates its model may mitigate this issue. In general, the study was insightful, but it would be more interesting and challenging to evaluate the algorithms in a mobile environment that induces more variability.

Machine Learning

Some researchers have employed the principles of neural networks to predict delivery ratios such as ETX [89, 90]. Specifically, Caleffi and Paura (2009) targeted the replacement of the SMA and EWMA filters, which are commonly used in forming ETX, with an unsupervised neuron estimator [90]. Based on the last n packet reception events, the predictor determines the weights of each event and the biasing coefficient in order to estimate the delivery ratio at next time instance. The neural-based estimator showed promising results in simulation. However, in a follow-up study with Cacciapuoti et al., the performance of the neuron estimator was inconclusive [89]. The evaluation, which used

datasets captured from an actual network, failed to identify a superior filtering approach. The ambiguous outcome was attributed to the number of possible parameters that influence the different techniques. For instance, the ranking of the estimators changed as different parameters were varied, thus making the results uncertain. Another possible concern with the neural-based approach would be its computational requirements compared to other machine learning techniques [29].

The use of regression techniques in the form of supervised learning have also been explored in a series of works by the same research center [28-30, 91]. In [28, 29], a distributed protocol was designed to exploit the mobility of nodes for gathering diverse training samples, and afterwards, use the training samples in an *offline* supervised learning algorithm. The primary difference between the two works is that the study by Flushing et al. (2012) was simulation-based [28], while the work by Kudelski et al. (2014) was a validation study using actual mobile robots [29]. Overall, the framework used in these works can be classified into four basic phases: *collect*, *learn*, *deploy*, and *use*. During the collection phase, nodes vary their positions randomly so that a large number of different network configurations are sampled via the transmission of periodic probes. Each training sample consists of a labeled LQ value (i.e. PRR) and an attribute vector of eight features. The features are related to distance, traffic load, RSSI, transmission rate, and neighborhood state. Using simulation data, attribute selection revealed that RSSI and distance were the most critical to prediction accuracy, but all eight features were retained during the follow-on evaluation. As for the learning phase, both works (i.e., [28, 29]) perform the learning step offline at a distributed node that later disseminates the predication model after training

is complete. The learning algorithm used in both papers was based on Support Vector Regression.

Some concerns about the approaches taken in [28, 29] include the amount of training time required to generate a prediction model, as well as the size of the feature space. For example, both studies appear to spend significant time and resources in gathering training examples before the model is actually trained and ready to use. Specifically, the simulation in [28] collected 10,000 training examples, while in [29], it took 30 minutes for the robots to randomly collect samples despite being spatially confined to testing areas less than 8 meters (m) by 6 m in size. The other concern is the size of the feature set (i.e., eight features), which adds complexity to the design and requires more training samples. Model prediction becomes exponentially harder as the dimensionality (i.e., feature set) grows due to a fixed-size training set covering less of the input space [36]. Therefore, it is important to perform feature selection and eliminate extraneous link attributes.

To address the adaptability issues in [28, 29], Di Caro et al. developed an online learning framework that incrementally retrains its regression model [30]. According to Di Caro et al., the previous works [28, 29] were offline, non-incremental, centralized, and non-cooperative, but the new design removed these constraints. Incremental learning was added through the use of Locally Weighted Projection Regression (LWPR). Di Caro et al. attempted to speed up the slow learning process in [28, 29] by having nodes across disparate links exchange training examples. However, this procedure undoubtedly adds noise to the training set because different links exhibit unique behavior due to hardware and environmental specifics [15, 20, 26]. Another drawback to the study by Di Caro is that

testing was performed in a sensor mote lab, but the statical nature of the nodes likely fails to capture the dynamics that could be expected in a mobile ad hoc network (MANET).

Machine Learning in Wireless Sensor Networks

An early application of supervised learning in LQ estimation can be traced to research by Wang et al. [92]. In [92], both offline and online versions of supervised learning were applied to datasets gathered from a 30-node sensor testbed. Two classification algorithms were evaluated including decision tree learners and rule-based learners. Furthermore, binary and multiclass (i.e., trinary) versions of these classifiers were evaluated. Each algorithm classified LQ as either 'good' or 'bad' (binary classifier) or 'good', 'medium', or 'bad' (trinary classifier). The algorithms were trained using a mixture of seven features that included RSSI, buffer sizes, delivery ratios, and topology information. After performing attribute selection, the feature set was reduced to five attributes due to the prediction accuracy remaining the same. The attributes with the most information gain proved to be RSSI, along with the forward delivery probability. The evaluation revealed that the decision tree learner achieved higher accuracy, while the binary classifier was about 3% more accurate than the multi-class classifier. Finally, the Very Fast Decision Tree (VFDT) algorithm was used in an online fashion, and it showed comparable performance to the offline approach that took multiple hours to train. Despite the simplicity of the classification approach, there are likely times when more granularity in LQ is needed to clearly distinguish between link options. Additionally, the model only considers a single physical metric as part of the feature set, and based on the information gain of RSSI, adding more physical metrics may prove more fruitful than many of the other proposed features.

Liu and Cerpa published two similar works that use machine learning for predicting the chances of successful packet delivery over a short-term window [24, 93]. The works primarily differ in two regards. First, the learning in [93] takes places offline, while in [24] the learning transpires online. Secondly, the prediction windows, or temporal relevance, of the estimators are slightly different: [93] is designed to output whether or not the *next* packet will be successful based on the binary output of a classifier, while the binary classifier output in [24] is intended to be valid for a slightly longer period by predicting whether the probability of packet delivery will be above some predefined threshold (e.g., 90%) during the next short-term window. Both algorithms are designed for short-term routing protocols that attempt to boost delivery efficiency by exploiting the correlation of packet delivery over short timeframes.

Both of the algorithms designed by Liu and Cerpa were tested using a feature set consisting of four attributes: PRR, RSSI, SNR, and LQI. Feature engineering revealed that the attributes could be reduced to PRR combined with any one of the physical metrics with negligible differences. In [93], the authors evaluate three different classifiers (i.e., Naïve Bayes, Logistic Regression, and Neural Network) and found the logistic regression model to be the best performer. Evaluation results indicate that the model could be accurately trained using a minimum of several disparate links (i.e., roughly 5-7) with about 1000 labeled samples per link in the composite training set. The authors note that PRR and physical parameter correlation may be different at times due to hardware-specific variations, yet they use an aggregated set of training data that was collected over different links to train a single classifier. Better accuracy would likely be obtained by maintaining individual classifiers for each link and training each one using individualized training sets

that were collected over its specific link. Overall, the major disadvantage of [93] is that the offline design requires on-site data collection and training to be performed prior to implementation.

To circumvent this issue, the other algorithm in [24] uses online learning. Several online learning frameworks were tested including weight majority, winnow, and stochastic gradient descent (SGD); the authors found that SGD performed the best. The prediction algorithm was designed using a binary classifier that informed the routing protocol whether the future reception ratio will be greater than a predefined threshold over the next short time interval (e.g., 1 second). The model was originally intended to accept a window of historical feature vectors (i.e., PRR and physical metrics), but after testing, the authors concluded that only the last input vector was relevant for prediction based on the short prediction window into the future. The dependence of the learned model on packet inter-arrival time was tested, and the results showed that prediction accuracy decreases as the time between inter-arriving packets grows. For instance, the prediction accuracy of the binary classifier was shown to be only slightly better than 50% when the packet interspacing was set to one second. As a result, the authors admit that the biggest disadvantage of the design is that it depends upon a high volume of incoming traffic in order to make accurate predictions.

There are a few issues with the approach taken by Liu and Cerpa. First, no quantitative analysis into the coherence time was provided, despite the approach attempting to make a prediction that only remains valid for periods less than or equal to the coherence time of the channel. It would be beneficial to know the estimated period of validity for each prediction in a WSN. Additionally, they used thresholds to limit the output from the

regression algorithms to a binary 1 or 0 for the purpose of indicating whether to transmit the next packet. In IEEE 802.11-based networks, most upper-layer applications do not manage transmissions on a per-packet basis; instead, applications delegate lower layer processes to handle the segmentation and transmission of packetized data. Therefore, the utility of their binary output is questionable in other networks outside of WSNs. Furthermore, the binary output of their prediction algorithm would likely fluctuate rapidly between 0 and 1, which would likely be little value to applications such as a robot navigation system and could not be filtered into a more stable and long-term estimate of LQ. In general, most higher-layer LQ estimators strive for output stability [20].

Future Directions for Learning Link Quality

Several potential areas for future work have been briefly implied while reviewing the existing work related to learning LQ. Below is an expanded summary of these suggestions, in addition to some others. The aim of this discussion is to guide future efforts in the field and to improve the next generation of LQ prediction systems.

(i) **More prediction granularity -** Some existing models only predict next packet delivery [24, 93], while others only classify LQ into categories [92]. A binary output or simple classification does not provide much insight into LQ and is probably insufficient for many decision making processes. Thus, regression methods offer more distinction into a precise level of LQ.

(ii) **Pay attention to channel coherence time and offer the capability to convert short-term predictions into long-term estimates.** Some prediction systems are focused solely on short-term predictions, but do not provide an in-depth look into channel coherence time [24, 93]. Meanwhile,

others ignore channel coherence time and do not spatially average radio metrics [28, 29]. Coherence time is important because it outlines the input-output sampling constraints, as well as the time period for which a prediction remains valid. Because channel coherence is usually short (e.g., much less than 1 second), LQ predictions may have limited applicability to some higher-layer applications that process decisions or actions more slowly. In this case, an LQ estimate, which is reflective of average attenuation, is the best alternative. A future contribution may wish to demonstrate the versatility of a short-term prediction system and how it could also serve as an estimation generator by smoothing a window of recent predictions.

(iii) **Faster model startup -** Several learning schemes call for extensive data collection before initially training a prediction model [28, 29]. However, many real-time and streaming applications such as robots, require almost immediate predictions on LQ shortly after link initiation. An option that may facilitate this objective is incorporating synthetic samples into the early training sets, and then eventually replacing them as real samples are eventually collected.

(iv) **Reduce sampling and labeling expenses -** Some schemes ignore the costs associated with collecting labeled training examples as evident in the large training sets used in [28, 29] and the continuous online labeling and tuning used in [24, 30]. However, there are energy and network costs associated with obtaining labels due to them residing at the opposite end of the link.

Another example of this oversight is evident in [28, 29]; in these works, robots perform artificial movement for the sole purpose of collecting diverse training examples. Such an approach is expensive in terms of time and energy. A suggested alternative to reduce labeling costs is to leverage some of the techniques used in active learning or semi-supervised learning.

(v) **Improved accuracy -** Some authors advocate for the mixing of training sets gathered across disparate links to expedite model startup [30, 93]. However, LQ is specific to an individual link hardware and environment [20, 22, 27]. Thus, to avoid inaccuracies, training examples should be kept individualized for specific links.

(vi) Improve Adaptability - Many works are non-incremental learners where learning only takes place once in an offline manner [28, 29, 93], and thus, they are not adaptable to concept drift. LQ prediction is a streaming and dynamic process where the underlying statistical dependency between the input and output variables may drift over time. Therefore, online learning algorithms are essential for accurate LQ prediction.

(vii) **Explore other forms of online learning -** Thus far, only online learning algorithms, where model parameters are tuned one example at a time, have been employed in LQ prediction. However, samples in a wireless environment may contain occasional outliers due to deep fades, and the impact of noisy samples on these types of learning algorithms merits further investigation. Furthermore, maintaining these types of learners may be more expensive in terms of labeling than other batch-style incremental

learners. Overall, the field lacks a holistic sampling and incremental batch-style learning framework. This style of learning may offer more potential to reduce labeling expenses by only tuning the model when there are high chances that concept drift has occurred.

(viii) **Reduce complexity -** Many works advocate for large feature sets with attributes that are loosely correlated to wireless channel quality [28-30, 92]. Model complexity, as well as labeling expenses, can be reduced by eliminating extraneous features. It is well-known that reducing dimensionality also lowers the number of training examples required for a prediction model [36].

(ix) **Introduce new target variable options -** All of the existing models target PRR or some other packet counting statistic as the predicted output variable [24, 28-30, 85, 88, 90, 91, 93]. However, PRR may be inaccurate or difficult for a receiver to passively measure. The problem is that many upper layer applications do not manage per-packet transmissions. Usually, this is performed at the network layer and below, but accessing this information from the application layer is nontrivial. Even when monitoring transmission from the transport layer, there are hidden reliability mechanism at the lower layers that may obscure a true indication of PRR. Therefore, it may be prudent to use some other type of LQ statistic that can be more easily measured by a receiver and more accurately reflect true LQ from the application's perspective. An example could be a *throughput potential ratio* that is a time-based reflection of how quickly recurring

blocks of application data can be transmitted across the wireless channel; the statistic could be normalized into a ratio using the maximum rate observed since link inception.

Conclusion

Assessing the quality of wireless links is a critical function in many higher layer applications. Consequently, the field of LQ estimation has sparked a rich body of work over the past decade and more. Many of the early works in LQ estimation relied upon transmission of periodic probes for assessing the state of the link [23, 61-63, 67]. Later, many works attempted to reduce the overhead and responsiveness issues of periodic probes by developing passive packet counting techniques through MAC layer monitoring [68, 71-75]. However, access to the MAC layer primitives for packet counting requires driver and software modifications that makes it difficult to scale across heterogeneous systems [26]. As an alternative, some works investigated the use of physical layer metrics because they support rapid and passive link information without the need for complex software modifications. Unfortunately, the physical layer metrics are fairly complex to interpret and require the discovery of link-specific mapping functions to estimate LQ [25]. Some authors have proposed the use of experimentation and offline statistical analysis for discovering these functions [25, 60, 64, 77-79]. However, this approach time-consuming, and ultimately, lacks adaptability. A more practical and adaptable approach would be to use an online learning algorithm that automates the functional discovery process, as well as evolves with the link dynamics. Any such adaptable algorithm must consider multiple input metrics because research indicates that relying on a single input is insufficient assessing LQ [20, 27]. Machine learning, as some authors have suggested [24, 28-30, 91-

93], can leverage the information provided by multiple inputs, but learning algorithms usually require some form of *a priori* knowledge and manual tuning in order to obtain the best results [32].

An alternative method for weighting several input metrics is fuzzy logic, as proposed in [54, 83, 84]. However, these particular fuzzy-based methods rely on offline experimentation and lack adaptability. An interesting approach, which has yet to be applied, would be to employ some form of fuzzy learner to LQ estimation or prediction. Fuzzy systems are an attractive design option because they intuitively allow for the injection of domain knowledge. The fuzzy-based approach referred to as *adaptive fuzzy control* can evolve over time with system changes [34], but the method is likely not appropriate for assessing LQ. The *control* paradigm is based on altering the inputs in order to *control* or obtain the desired output. However, most LQ systems have little or no control over the inputs, and instead of making adjustments to the inputs, these systems make assessments based on their given values. There exist other fuzzy-based learners as discussed in [35], but the complexity of genetic algorithms and neural networks likely makes them impractical for rapidly adapting to link dynamics. A more streamlined means of incorporating adaptability into fuzzy systems would be beneficial to goal of having a tunable and adaptable method that is efficient in making LQ assessments.

CHAPTER IV

FUZZY LOGIC FOR RADIO-SWITCHING IN ROBOT NETWORKS

Introduction

As mentioned in the literature survey, there are several higher-layer applications that can benefit from link quality (LQ) estimation. In this chapter, the focus is transitioned to a specific application of LQ estimation that has yet to be explored in the context of robot networks. The concept is to use a LQ estimator for switching between diverse radios that could be options onboard a robotic platform. In this particular application setting, the number of radio choices is limited to two, but the concept is extensible to more. The two radio options chosen for this study were motivated by the desire to extend the transmission range of robotic systems in an efficient manner. The extension capability is enabled by a passive directional antenna, such as the one shown on the robot in Figure 9. The efficiency aspect comes from the concept of putting the directional radio in sleep-mode to conserve power when link conditions are favorable enough for the primary omnidirectional radio. However, when link conditions begin to deteriorate for the primary radio, the objective of the LQ estimator is to awaken the directional radio prior to link failure so that any link disruption is avoided.

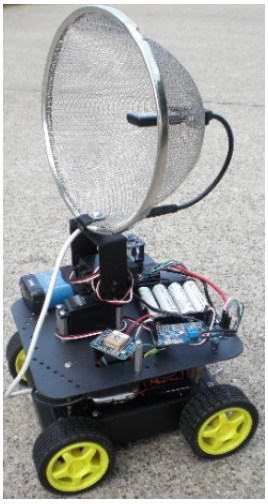

Figure 9. Image of the robot used during the evaluation of the radio-switching controller.

The decision-making process of selecting a particular radio for an impending transmission is enabled through fuzzy logic, which was selected due to its attractive features mentioned in Chapter II. By applying fuzzy logic in a practical application, an appreciation of its capability to incorporate expert knowledge, systematically weight multiple LQ metrics, and mitigate large-scale wireless dynamics could be gained. The study also served as a means to evaluate the potential of fuzzy logic to serve in a more generalized capacity for assessing LQ, beyond its tailored use in this chapter for radio switching decisions.

The radio switching application studied in this chapter does not require high-precision LQ *predictions*, which can vary significantly over short distances due to multipath fading. Instead, the application demands more stable LQ *estimates* that are roughly the same over several wavelengths of movement and tend to reflect the large-scale attenuation factors. The stability aspect of the LQ output is important in order to avoid unnecessary and rapid switching between the two radios. With LQ estimation being the

design objective, the focus in this chapter is more on generating estimates in the most efficient manner possible, versus minimizing prediction error. However, accurate short-term predictions become the predominant concern in the subsequent chapter.

The fuzzy design used in this chapter follows the classical Mamdani architecture discussed in Chapter II. However, Mamdani systems are not natively adaptable to dynamic processes such as wireless LQ [34]. Thus, this chapter also serves to highlight this issue and provide motivation for the modifications in the next chapter. One of the primary issues relates to the fuzzy sets used for the radio controller. Specifically, the fuzzy sets were designed using expert knowledge, and consequently, the controller is not inherently self-configuring or adaptable. Other fuzzy-based approaches in the domain of LQ estimation, which also use expert knowledge, contend that the customized fuzzy sets are naturally robust to noise [54, 83, 84]. This justification for mitigating the dynamics of wireless LQ may suffice in the case of *estimation*; however, when more accurate LQ *predictions* are needed, the fuzzy sets should be more precisely configured and tuned *online* based on the changing dynamics of the environment. These concerns are fully addressed in the next chapter, as the focus transitions to improved adaptability and accuracy. In the meantime, this chapter serves to introduce a novel application and lay the foundation for fuzzy-based LQ estimation.

Motivation for the Radio Controller

The Demand for Range Extension in Robot Networks

Several existing and emerging robotic applications such as ordnance disposal, disaster assessment, and search-and-rescue require high-speed communication links to span large distances [12, 13]. Wireless communication between these systems is usually ideal, as tethered connections complicate mobility and can become tangled around objects [12, 13]. However, there are several challenges to communicating wirelessly over longer distances, especially in mobile ad hoc networks (MANETs). Often times, the radios of robotic systems operate in the gigahertz range, and thus, the emitted radio waves tend to follow a line-of-sight (LOS) propagation path and undergo several propagation dynamics [10]. The LOS propagation path can become especially problematic for unmanned ground vehicles due to their low antenna height and the uneven terrain they must traverse [10]. The proposed approach mitigates these combined effects due to the gain provided by the directional radio, which can be activated on-demand under unfavorable LQ conditions.

The Challenge with Smart Antennas on Robots

The proposed approach calls for the use of passive antenna reflectors, similar to the one shown in Figure 9, in order to provide *directional* gain. However, so-called *smart antennas,* such as the phased array, can also provide directivity through a combination of spatially-diverse antennas and signal processing [94]. Unfortunately, these types of smart antennas require overhead in the form of signal processing, space for multiple antennas, and power for each independent RF chain (i.e. digital-to-analog converter, filter, mixer, power amplifier, etc.). These extensive requirements make the use of smart antennas in small, lightweight, and low-power devices a challenging problem (see Section 10.8 in

[16]). Thus, mounting smart antennas on smaller, battery-powered robots could prove infeasible.

Multiple-Input Multiple-Output (MIMO) beamforming is another multi-antenna and signal processing technique that offers *diversity gain, multiplexing gain,* or a combination of both [16]. MIMO techniques thrive in multipath environments (e.g., indoors) where independent fading paths can be captured by the antennas and be coherently recombined. On the other hand, MIMO performance is degraded in the presence of a strong light-of-sight (LOS) component [95]. Consequently, MIMO gains may be lower than those of traditional reflectors when used in outdoor scenarios that are frequently encountered in applications such as disaster assessment and ordnance disposal.

Why the Radio Switching Approach?

The motivation behind the radio-switching concept is to conserve power. Operating multiple radios simultaneously would be expensive, and to mitigate this expense, the fuzzy logic controller (FLC) contains the intelligence to perform conditions-based switching between the diverse radios. In previous work [7], it was shown that switching the directional radio from idle-mode to sleep-mode can save roughly 50 milliamperes (mA) of current draw.

Furthermore, it was also demonstrated in [7] that that directional gain with respect to the omnidirectional radio increases as the distance between the transmitter and receiver grows. In other words, when the channel conditions degraded for the omnidirectional radio, the directional radio offered much better throughput potential. Thus, it would be more advantageous in a radio-switching scenario, especially when attempting to conserve

power, to only invoke the directional radio when conditions are poor for the omnidirectional radio.

Why Fuzzy Control of Radio Handoffs?

In Chapter II, several strengths associated with fuzzy logic were highlighted, which consequently led to the decision to implement the radio-switching controller using it. One advantage of fuzzy logic is that it provides an intuitive framework for inferring a weighted response based on the values of multiple inputs and a set of embedded rules. Another strength of fuzzy systems is that a human expert can impart knowledge into the rule base. This capability makes designing a controller a much more streamlined and intuitive process; no complex system equations are required, and the *if-then* rules inside the rule base are similar in fashion to the way humans make decisions. Finally, the inherent characteristics of fuzzy sets make them more robust to the inevitable dynamics associated with LQ estimation. For instance, the ability to classify the inputs with a level of certainty during fuzzification, as well as the ability to overlap fuzzy sets, makes them more resilient to noise or other dynamics.

Improve Efficiency and Lessen Constraints

There are existing LQ estimators in the literature that use fuzzy logic (see [54, 83, 84]). However, all of the estimators use some form of packet reception ratio (PRR) as an input. As discussed in the literature survey, PRR is not readily known to the sender and it can be costly to obtain on a recurring basis. Another drawback to these fuzzy-based designs is that they require three or more inputs. Fuzzy systems require several computational steps to generate each crisp output, and the complexity of fuzzy systems grows exponentially with the number of inputs and fuzzy sets [96]. Thus, an effort should be

taken to limit the number of inputs to only those required to obtain sufficient estimation accuracy.

Configuration of the Fuzzy Logic Controller

Input Selection

LQ estimation involves a series of tradeoffs [97], and arguably the most important factors that influence these tradeoffs are the inputs selected for the LQ estimator. In the case of this application, the primary focus was on optimizing the efficiency aspects of the LQ estimator. More specifically, the goal was to lower the sampling and computational overhead of the system, given the resource constraints of most robotic systems. In order to meet these efficiency objectives, an effort was made to minimize the number inputs into the fuzzy system, as well as choose inputs that are relatively inexpensive to sample. Therefore, the search for possible input metrics was limited to the available radio hardware metrics, given the aforementioned discussion in Chapter III about their low overhead. In terms of the appropriate number of inputs, research studies have shown that only one metric is insufficient for LQ estimation, so some combination of at least two inputs is necessary for improved accuracy [20, 27].

As previously discussed in the literature review, there are only a few radio metrics available for LQ estimation. A common metric used in many works and discussed in the survey was signal-to-noise ratio (SNR). However, SNR is not directly provided by off-the-shelf IEEE 802.11 radios [98]. Instead, SNR must be formed using by combining received signal strength indicator (RSSI) and noise-level, but as other works highlight, the noise-level metric is commonly unavailable or invariant [27, 76]. Thus, relying on SNR for LQ estimation could be problematic in some cases depending upon the hardware.

In contrast with SNR, one common metric that is almost always made available by the hardware is RSSI. RSSI provides a measure of composite energy received at the antenna, and therefore, can be used to provide a sense of signal degradation due to attenuation. However, RSSI has been shown to be insufficient for accurate LQ estimation when used by itself [27, 99]. One of the problems with RSSI is that it fails to indicate the level of interference on a channel that may precipitate link failure [27]. Additionally, in the case of sensor radios (i.e., IEEE 802.15.4), researchers have shown that RSSI has weaker correlation with packet reception once its goes below a particular value, and as a result, it is challenging to accurately gauge the quality of intermediate links using solely RSSI [97, 99].

The other commonly available radio metric in IEEE 802.11 networks is signal quality (SQ). However, studies involving SQ are largely absent in the literature, and instead, much of the research concentrates on an analogous metric known as link quality indicator (LQI), which is specified in IEEE 802.15.4 and commonly used in wireless sensor networks [100]. Both of these metrics are effectively a measure of the chip error rate associated with the spread spectrum signal [55, 101], where the term '*chip*' refers to code's spreading sequence. SQ complements RSSI by indirectly providing information about the level of noise and interference in the channel because pseudo-noise (PN) code correlation degrades under such unfavorable conditions [98].

Due to no existing empirical research involving SQ, an experiment validating the ability of the metric to supplement RSSI was performed; more specifically, the intent was to determine whether SQ could effectively detect noise and interference, which has is known to be problematic for RSSI [27]. To setup the experiment, two separate ad hoc

networks were formed, each consisting of two nodes. Both independent basic service sets (IBSSs) were placed on the same frequency channel. All of the nodes were physically placed in separate rooms of a residential home with the nodes from separate IBSSs placed in adjacent rooms. During each experimental trial, a transmitter from one IBSS sent 512 bytes of payload to its receiver every 5 milliseconds via User Datagram Protocol (UDP) for a total of 5 seconds, and the receiver from the same IBSS logged the SQ from its radio immediately after each received packet. Meanwhile, the transmitter from the other IBSS, which was on the same frequency channel, occasionally served as an interferer by also transmitting UDP packets during the same time. Each interference packet consisted of 24 bytes of payload that was sent every microsecond. Measurements of SQ were also recorded for 5 second periods immediately before the interfering IBSS starting transmitting, so that SQ measurements could be compared with and without intentional interference. The outcome of these experiments are shown in Figure 10. The markers in the plot indicate the mean of SQ during each 5-second trial, while the whiskers reflect the standard deviation across all of the SQ measurements made during that same period. As expected, SQ is shown to generally degrade when there is additional transmission noise or interference on the channel, and thus, SQ can potentially complement RSSI in LQ estimation because RSSI is known to lack noise and interference detection.

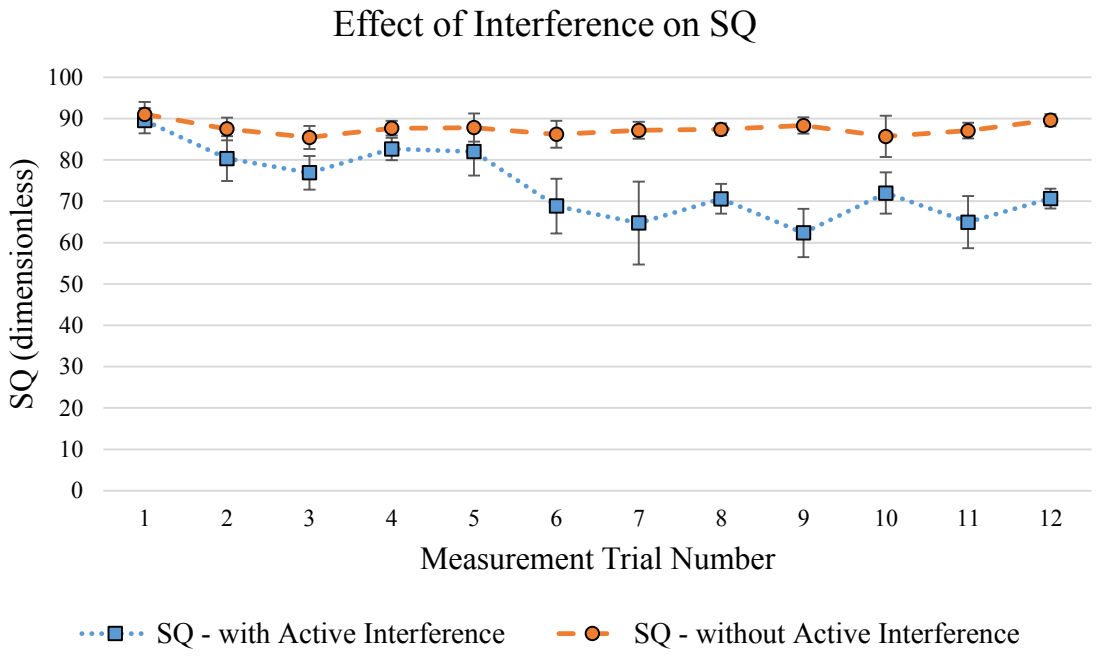

Figure 10. Effect of interference on the hardware metric of signal quality (SQ)

Other studies that were focused on IEEE 802.15.4 also validated the usefulness of chip correlation as a LQ metric [97, 99, 101]. In [97, 99], the chip correlation (i.e., LQI) observed over several packets is shown to be more correlated to link quality (e.g., PRR) than RSSI when the signal strength degrades beyond a certain threshold. In other words, chip correlation was found to be better suited for assessing intermediate quality links than RSSI. Therefore, combining these metrics in some fashion enables a more precise estimate capable of quantifying links ranging from '*good*' to '*bad*'.

Controller Assumptions, Placement, and Feedback

Now that the inputs have been selected, the design of the FLC can be formalized and assumptions can be stated. The FLC is intended to reside at the sender and make the decision as to which radio should be used for each transmission. A block diagram illustrating the placement of the FLC and the entire control process is shown in Figure 11. As indicated in the figure, the inputs into the FLC are the radio metrics, as well as the current state of the system in terms of which radio is currently active. The latter input is used for hysteresis control, which will be discussed in a subsequent section. These inputs are sampled and fed into the system at a periodic rate; therefore, the output of the FLC is a function of time. It is assumed that the input metrics are sampled at a rate that is a function of the robot's velocity. The approximation that independent radio samples are available about every half of a wavelength could be used as a guide in this sampling process [16]. Furthermore, it is assumed that the samples are filtered and smoothed over roughly 10-30 wavelengths in order to abstract the effects of multipath [17]. This allows the estimate to be based on the large-scale attenuation factors of shadowing and path loss, instead of short-term fluctuations due to multipath [17, 18].

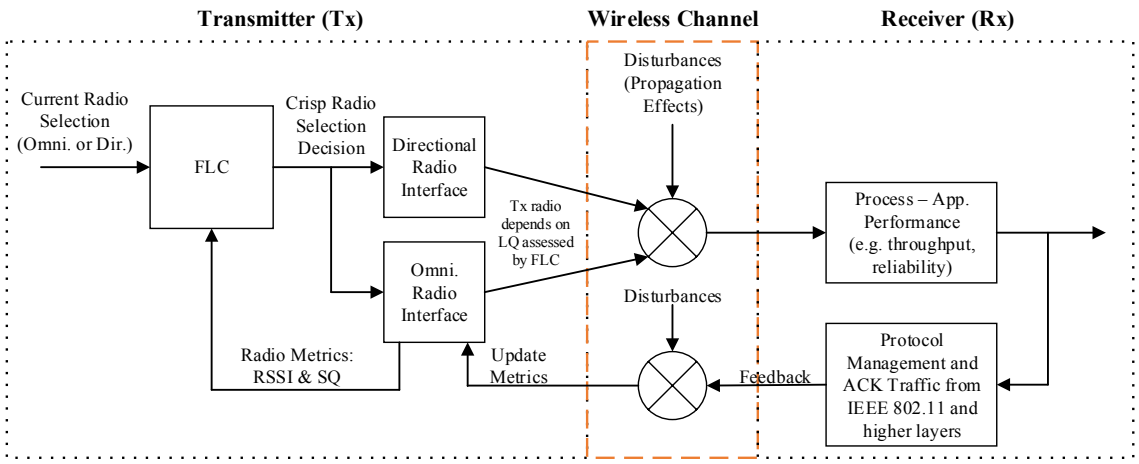

Figure 11. Block diagram of multi-radio control process

Some additional observations and assumptions can be drawn from Figure 11. The figure shows that the disturbances observed in the forward direction may be different than those observed in the reverse direction. It is assumed that the issue of link asymmetry is mostly mitigated due to the system forming estimates, as opposed to short-term predictions. More specifically, the estimation process of averaging the metrics over several wavelengths filters out the small-scale variation differences and reveals the large-scale propagation effects, which should be roughly the same in both directions. The figure also alludes to how LQ is passively sensed without any explicit feedback. In fact, the system relies upon indirect feedback in the form of control packets generated by protocols residing at the data link, network, and transport layers. In order to gain a better appreciation for the volume of feedback generated by these layers, a timeline was constructed based on a Transmission Control Protocol (TCP) packet-capture sequenced obtained from the NS-3 simulator [69]. As the Figure 12 indicates, the built-in protocol management functions of IEEE 802.11 ensures a steady stream of feedback returned to a sender that is actively transmitting application data to a receiver. Even when the link becomes idle, with no

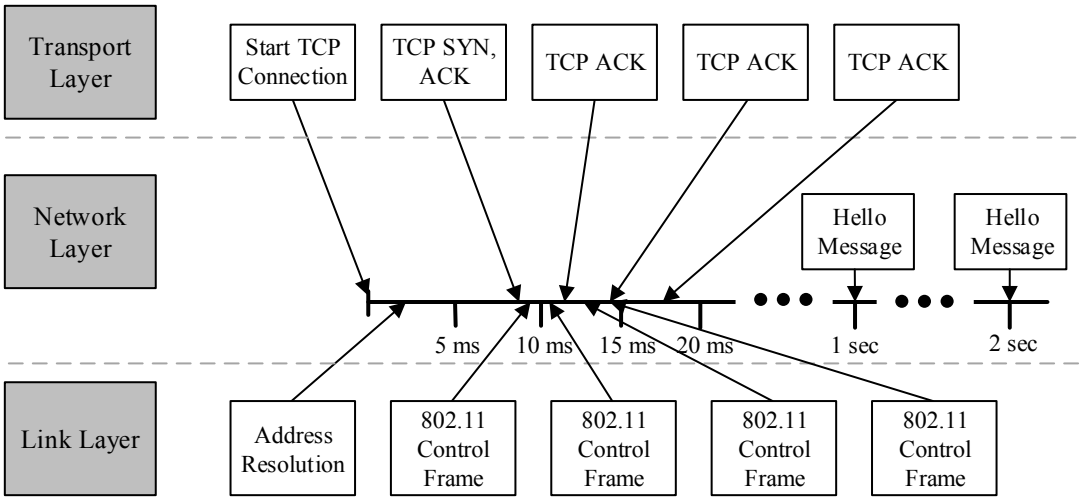

Figure 12. Example illustrating the approximate level of feedback generated by a receiver and sent to an active transmitter on an IEEE 802.11 link.

application data, beacon frames are still periodically transmitted in IEEE 802.11 networks for synchronization and power management purposes [55, 102]; thus, consistent feedback can be assumed regardless of the load of application data. The amount of network and transport layer feedback shown in the figure depends upon the specific types of protocols utilized at these layers.

Gaining Expert Knowledge through Experimentation

An advantage of fuzzy control is that it allows for human reasoning and expert knowledge to be incorporated into the controller [34]. To gain knowledge for this purpose, a series of empirical experiments were conducted using the robot shown in Figure 9. The experiments allowed a *fuzzy* or rough relationship to be established between the input metrics and the output metric of LQ. Throughput was selected as the dependent or target variable due to its importance in time-sensitive robotic applications and due to the drawbacks associated with PRR.

A total of four experiments were conducted in the following locations: a residential neighborhood, both sides of a school track, and a school parking lot. For every experiment, the robot started at the operator control unit (OCU) and then traveled away in a linear fashion until the wireless link failed. Approximately every half meter, the robot transmitted 50 kilobytes (kB) of data via TCP to the OCU. The radio metrics were sampled immediately following each transmission, while the OCU recorded the throughput associated with the 50 kB transmissions; these input-output pairs of measurements were used during the offline relationship analysis.

To generalize a relationship between the input and output indicators, samples from all four experiments were aggregated together. Afterwards, the samples were classified based on throughput. Specifically, the samples were separated according to throughput intervals that were uniformly spaced one megabyte per second apart. For each group of RSSI and SQ samples, distribution fitting was performed to find the mean and standard deviation statistics. The results are plotted in Figure 13, where the markers represent the means and the whiskers show the standard deviations for each group. By analyzing these plots, a better understating of the range of variation of each metric can be obtained. Additionally, the plots can be used to associate fuzzy linguistic values such as '*good*', '*intermediate*', and '*bad*' to actual metric ranges. The plots show that multiple throughput levels overlap in terms of the variation among the metrics. Hence, the overlapping nature of fuzzy sets are well-suited to mitigating the fact that there is no clear cutoff or boundary between LQ quantifiers such as '*good*' and '*intermediate*'.

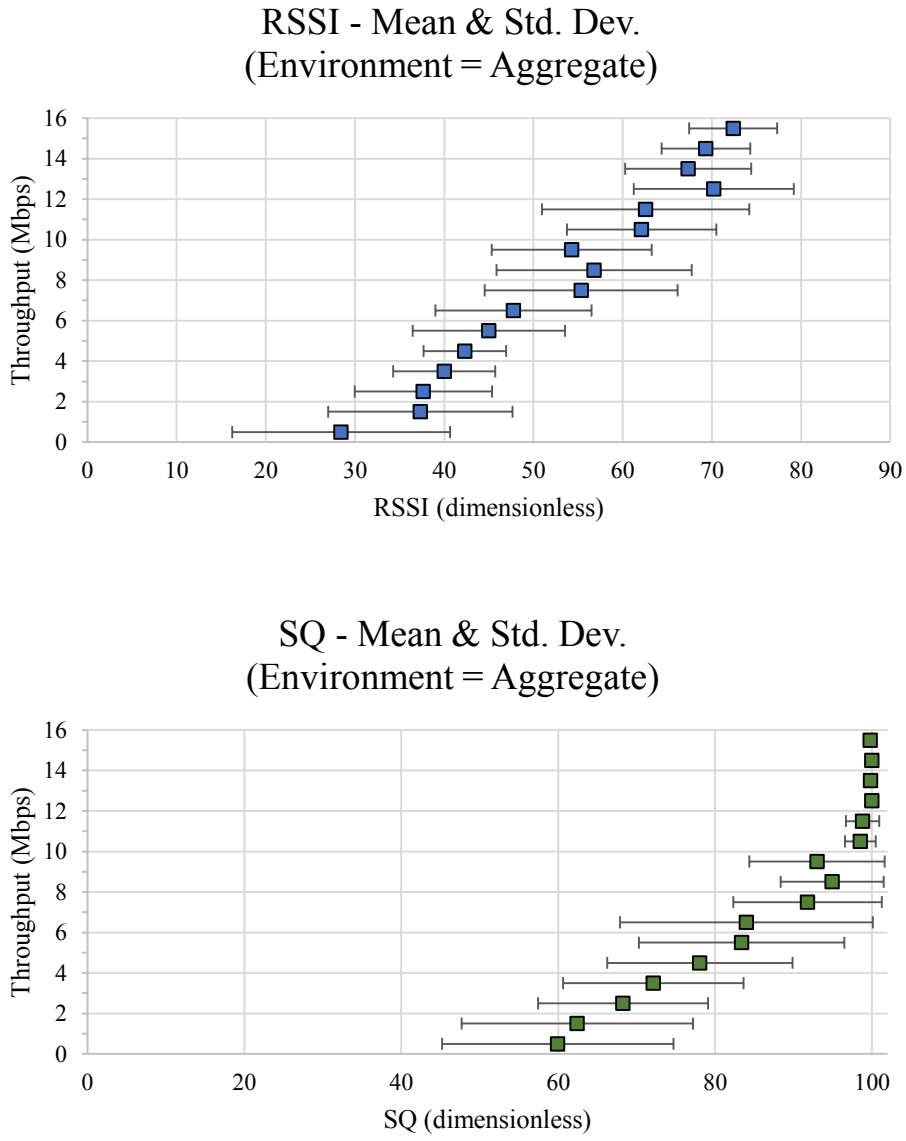

Figure 13. Radio metric statistics used in forming the fuzzy sets.

Configuring the Fuzzy Sets

The fuzzy sets for the radio metric inputs are shown in Figures 14 and 15. The arrows in the figures indicate that there are two fuzzy sets per linguistic value; however, only one is actually used during fuzzification depending upon which radio is currently active. The reason for having pairs of similar, yet shifted fuzzy sets, was to introduce hysteresis into the system. Without hysteresis control, the FLC may at times only slightly favor one radio option over the other. Under these circumstances, the FLC would probably attempt to alternate between the radio states frequently. However, as demonstrated in previous work [7], rapid handoffs are not practical due to the latency involved, and furthermore, the unnecessary switching would waste energy resources.

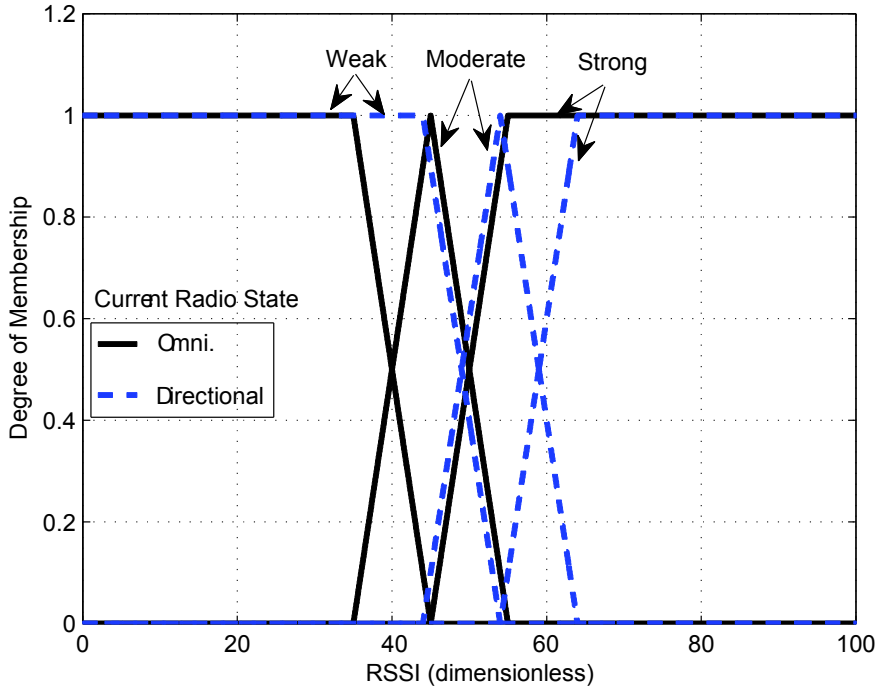

Figure 14. Fuzzy sets for the input of RSSI with duals for hysteresis

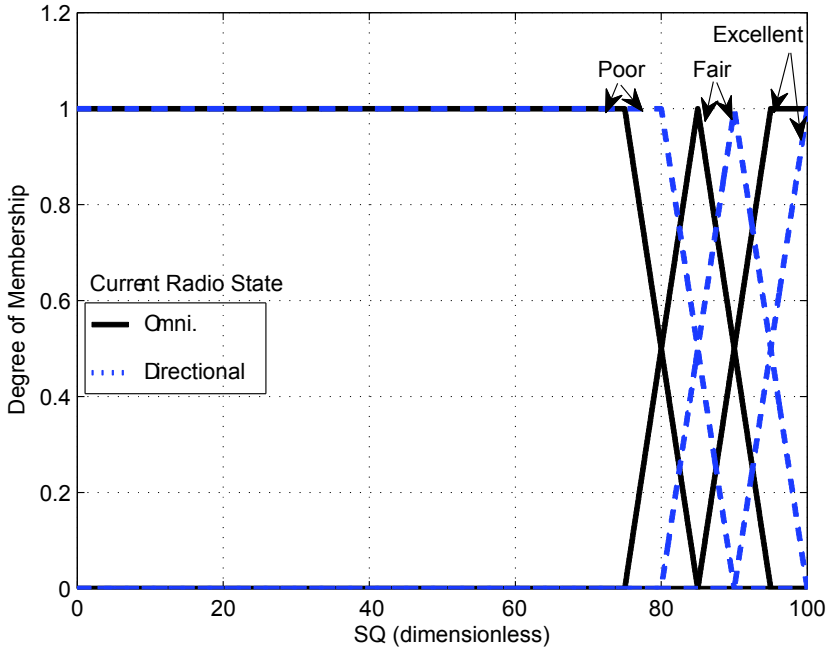

Figure 15. Fuzzy sets for the input of SQ with duals for hysteresis

The current radio state (i.e., the presently preferred radio) determines which series of fuzzy sets are currently in use by the fuzzy controller. More specifically, the fuzzy sets outlined in solid black are employed by the FLC when the omnidirectional radio is preferred by the system, whereas the fuzzy sets outlined in dashed-blue are used by the FLC when the directional radio is active. Both series of fuzzy sets are identical, except that they are offset on each universe: 10% on the RSSI universe and 5% on the SQ universe. The rightward shift of the dashed-blue fuzzy sets ensures that LQ is sufficiently improved before switching transmissions back to the omnidirectional radio and putting the directional radio into sleep-mode.

The positioning of the fuzzy sets on each universe was based on the expert knowledge gained in the previous experiments. In other words, the linguistic values and their associated fuzzy sets logically reflect the throughput performance that can be expected for the respective range of each fuzzy set. The dependency statistics between the inputs and throughput levels displayed in Figure 13 played an important role in the expert placement of the fuzzy sets. Additionally, the average metric values at the time of link failure were instrumental in setting the saturation points associated with the '*weak*' and '*poor*' fuzzy sets; specifically, these points were set slightly higher on the universe so that a handoff would be ideally triggered to the directional radio before link failure.

Tuning the Fuzzy Sets Online

Despite the robust nature of fuzzy sets, it is possible that the placements of the input sets shown in Figures 14 and 15 are suboptimal in some settings. To account for this possibility, a self-correcting algorithm was developed to critique the timeliness of radio handoffs. In the event of an inefficient or untimely switch between the radios, the algorithm tunes the fuzzy sets by shifting them conservatively along their respective universes. Figure 16 shows the flowchart for the performance critic. The highlighted callouts included in the figure explain the possible scenarios that merit corrective action by the algorithm. In essence, the self-tuning capability makes the controller more adaptable to operating in environments that are significantly different than those observed in the offline experiments.

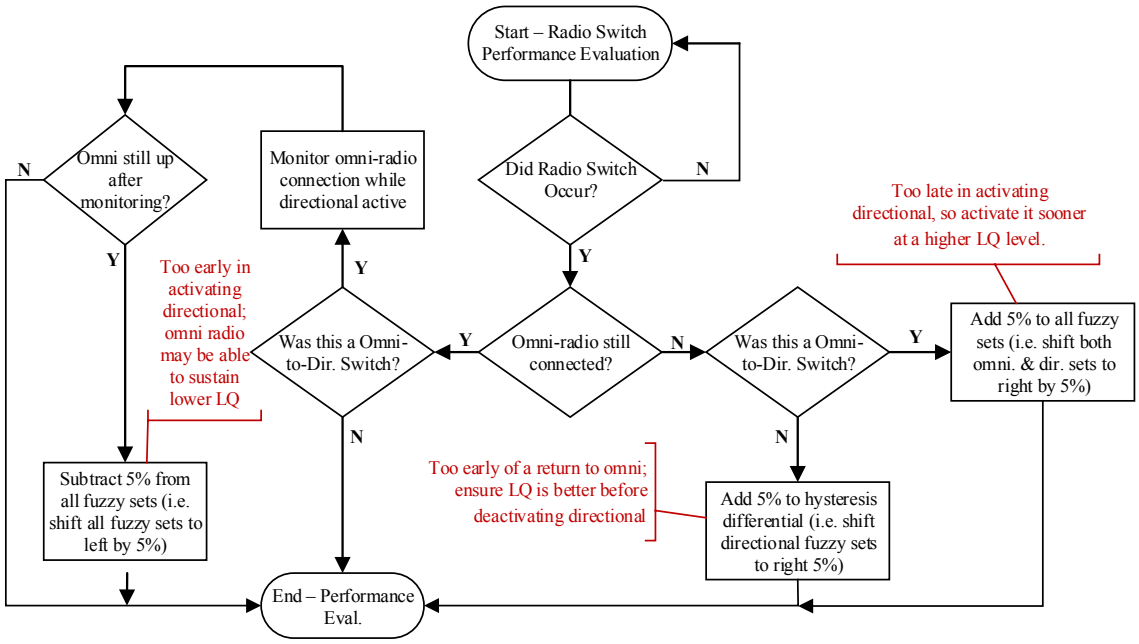

Figure 16. Flowchart for performance critic of radio handoffs

Generating Radio Selection Decisions

Once memberships to the fuzzy sets have been assigned for each input, the inference mechanism determines the extent to which each rule in the rule base is fired. The expert-provided rule base for radio controller is shown in Table 4. Because the proposed system has two inputs with three fuzzy sets each, the rule base contains nine rules (i.e., 3^2). The rules follow the literal form:

Rule1: **If** *RSSI* is *strong* **and** *SQ* is *excellent*, **then** *radio selection* is omni-radio (O)

…

Rule 9: **If** *RSSI* is *weak* **and** *SQ* is *poor*, **then** *radio selection* is directional (D).

Table 4

Rule Base for Radio Controller

Radio Selection		SQ		
		Excellent	**Fair**	**Poor**
RSSI	**Strong**	O	O	D
	Moderate	O	O	D
	Weak	D	D	D

Table 4 concisely lists the premise of each rule, as well as its associated consequent (i.e., radio selection). According to the rule base, the directional radio should be activated whenever one of the radio metrics indicates a poor quality link (i.e., either SQ is '*Poor*' or RSSI is '*Weak*'). Therefore, if RSSI fails to indicate persistent interference, as observed in [27], then SQ can still be used to trigger the directional radio. Otherwise, if the metrics indicate an intermediate or better quality link, then the omnidirectional radio is selected.

The linguistic values of the consequents displayed in Table 4 correspond to fuzzy sets on the output universe as indicated by the singletons shown in Figure 17. The figure shows that there are only two possible output consequents given the number of radio options. However, with fuzzy logic, the final system output is not discrete as suggested by the singletons, and in actuality, it is a weighted average of all the fired rules and the degrees to which they are '*on*'. The defuzzification process performs the weighted averaging to generate the final crisp output. The center-average form of defuzzification, which was previously discussed in Chapter II, is implied given that the output fuzzy sets are defined using singletons. The crisp output can vary continuously from 1 to 2, and thus, an unbiased

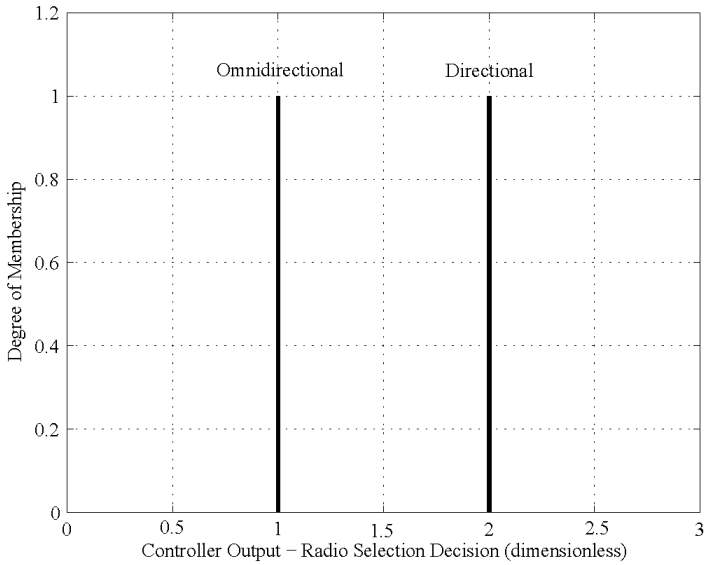

Figure 17. Output singletons for each radio option

decision boundary between radio options would be 1.5. To better visualize the system output, a surface plot is provided in Figure 18 that shows the controller's output for every possible input combination.

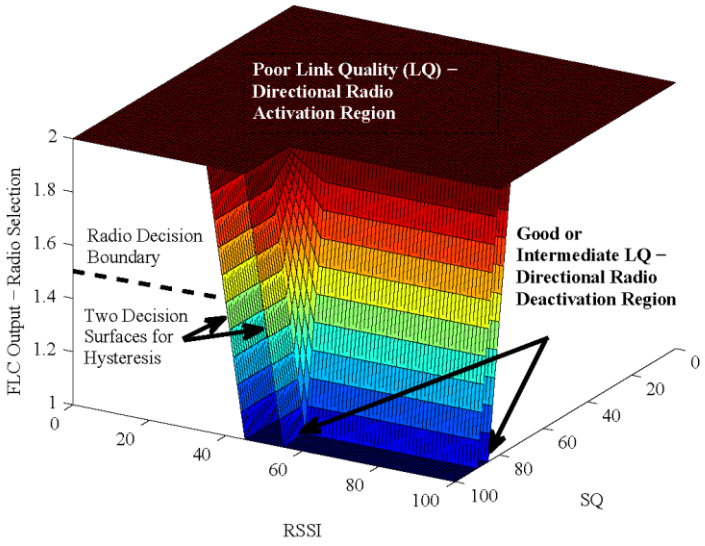

Figure 18. Control surface for the radio controller

An Example Illustrating the Fuzzy Control Process

As an example to illustrate the FLC's radio selection process, assume that the omnidirectional radio is currently preferred and that the moving average (MA) inputs to the system are 37.5 for RSSI and 83.0 for SQ. In that case, the fuzzification process would assign the membership levels to be: $\mu_{\text{rssi}}^{weak}(37.5) = 0.75$, $\mu_{\text{rssi}}^{moderate}(37.5) = 0.25$, $\mu_{\text{sq}}^{fair}(83.0) = 0.8$, $\mu_{\text{sq}}^{poor}(83.0) = 0.2$. Then, the inference mechanism would determine that the following rules would need to be fired: *weak* ∩ *fair*, *weak* ∩ *poor*, *moderate* ∩ *fair*, and *moderate* ∩ *poor*. Using the product t-norm, the degree of firing for each rule would equal 0.6, 0.15, 0.2, and 0.05, respectively. Finally, the weighted average output of the controller using the center-average defuzzification method described in Chapter II would be

$$\frac{[2(0.6)+2(0.15)+1(0.2)+2(0.05)]}{[(0.6)+(0.15)+(0.2)+(0.05)]} = 1.8$$

The crisp output of 1.8 indicates that the directional radio should be selected for transmission based on it being greater than decision boundary of 1.5. The example also demonstrates the steps taken to optimize the FLC's computational process. For instance, the denominator of equation of the above equation results in unity due to the selection of overlapping fuzzy sets that form partitions of unity [34]; thus, the denominator can be omitted during implementation for added efficiency. The defuzzification process was simplified further by using singleton output members for the rule consequents. Singletons eliminate the need for computing the centers of area for each implied output member without any general loss in performance [34].

Performance Evaluation of Radio Controller

Energy Efficiency

The first experiment investigates the efficacy of the dual-radio concept in terms of energy efficiency and whether the gain from the directional radio can lead to energy savings, despite powering two radios simultaneously. An NS-3 simulation was used to carry out the evaluation by comparing the energy consumption of a dual-radio system with diversified antennas to that of a traditional single-radio system with only an omnidirectional antenna. In the simulation, the dual-radio system was configured with two radios: one with an isotropic antenna and another with a gain of 8 decibels relative to isotropic (dB_i). The gain of the directional antenna was selected based on previous theoretical results provided in [7], which shows that approximately 8 dB_i is the expected gain for a small parabolic reflector of average efficiency that is sized near the wavelength of 2.4 GHz (i.e., 12.5 cm). On the other hand, the single-radio system was configured with a standard 0 dB_i antenna. The goal of the simulation was to compare the energy requirements of these systems while conducting a series of one megabyte (MB) transfers to a fixed receiver via TCP at a range of distances. The dual-radio system completed transmissions using its directional radio (i.e. 8 dB_i antenna), while also powering its other omnidirectional radio in an idle-state. The energy consumption of these systems was based on the channel conditions, the gain of the antenna, and the number of radios being powered. The current consumption levels of the radios during transmit and idle periods were configured based on the measurements presented in [7]. The other basic simulation parameters are same as those previously outlined in Table 3 of Chapter III.

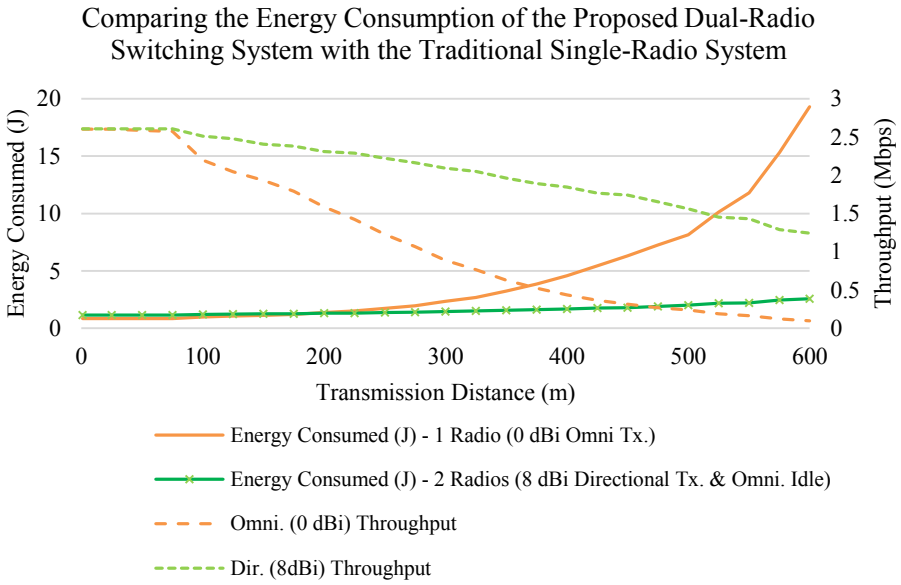

Figure 19. Simulation results evaluating energy consumption of the dual-radio system

The results of the simulation are shown in Figure 19. The chart indicates that the performance of the omnidirectional radio degrades in terms of throughput and energy expenditure as transmission distance increases. The deteriorating channel conditions force the omnidirectional radio to expend more energy, as a result of making prolonged transmissions due to the degraded throughput. On the other hand, the directional radio is able to mitigate these conditions because of its added transmission gain. As a result, the directional radio achieves a higher data rate, and hence, spends less time in the higher current (mA) consumption state of transmit mode. In summary, the results show that, beyond some distance (i.e. ~ 200 meters for this simulation), it is more efficient to use a directional antenna and power two radios concurrently, than trying to conduct larger distance transfers using only an isotropic antenna. Therefore, in addition to extending operational range, the proposed radio-switching concept can potentially lead to energy

savings, or at a minimum, help offset the energy costs associated with the antenna mechanical servo system.

Timeliness of Radio Handoffs

A series of physical experiments were conducted using the robot shown in Figure 9 in order to evaluate the switching timeliness of the FLC. Timeliness was evaluated in terms of LQ estimator's ability to detect imminent link failure and engage the directional radio beforehand, but not too early to avoid needlessly wasting energy. It is worth noting that the switch to the directional radio should be delayed until conditions are significantly degraded for the omnidirectional radio; this ensures a significant gain difference between the radio options as discussed in [7], and it also minimizes the energy consumption associated with the antenna positioning system and the additional radio.

The experiments were conducted at the same four locations used to previously evaluate the radio metrics. The images of these locations are provided in Figure 20. Each image shows the surrounding environment and the significant events that transpired during the testing. The experimental setup and procedure remained the same, except that the radio controller assessed LQ after the completion of each 50 kB data transfer in order to decide the appropriate radio for the next transmission. Once the directional radio was selected, both radios continued to make transmissions until each radio link failed. Ordinarily, only the directional radio would perform transmissions after it has been selected, but for these experiments, both radios continued to make transmissions so that the pairwise measurements could be used for evaluation purposes.

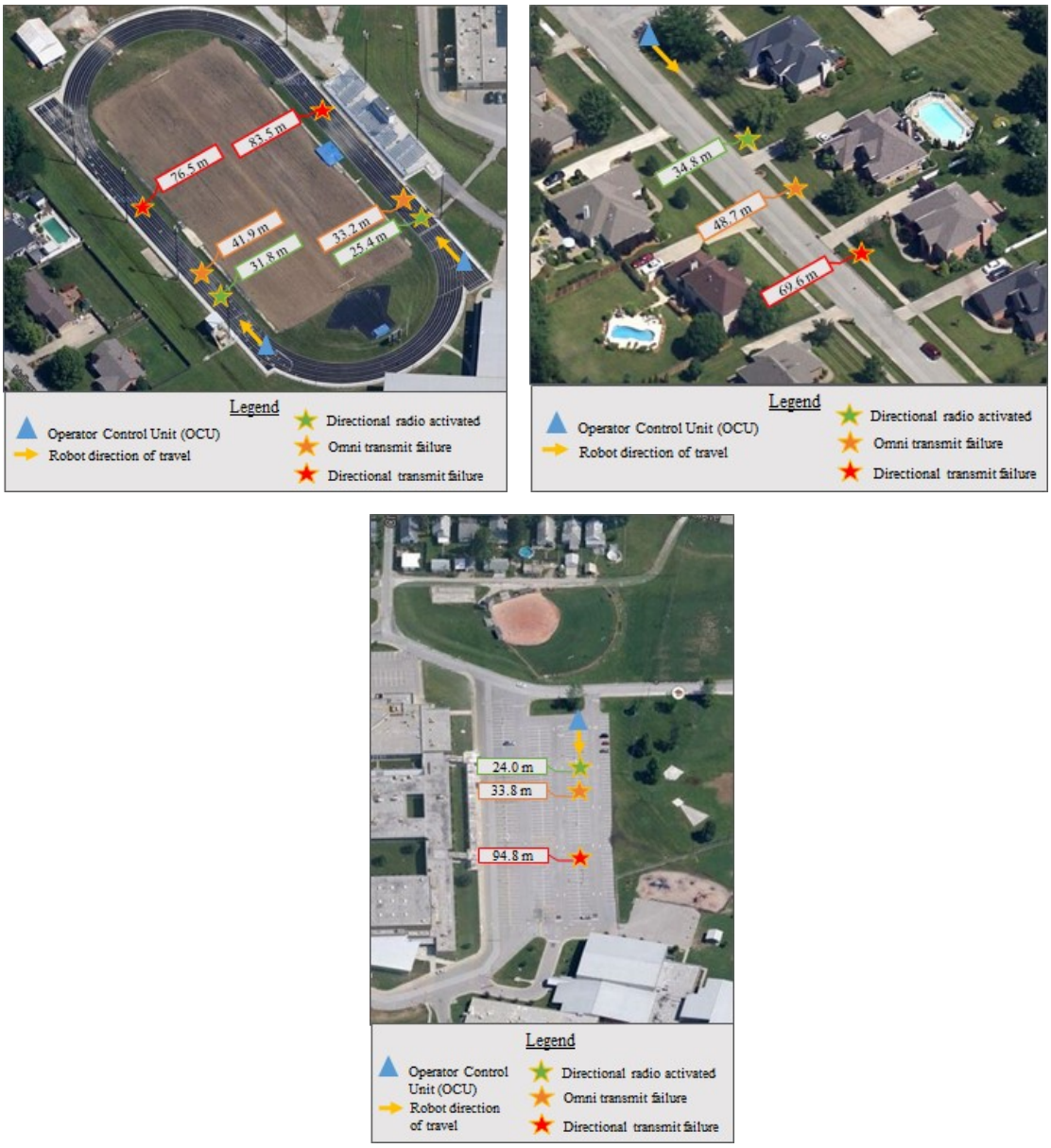

Figure 20. Images [103] showing the geographic environments of the evaluation locations, as well as the significant radio events.

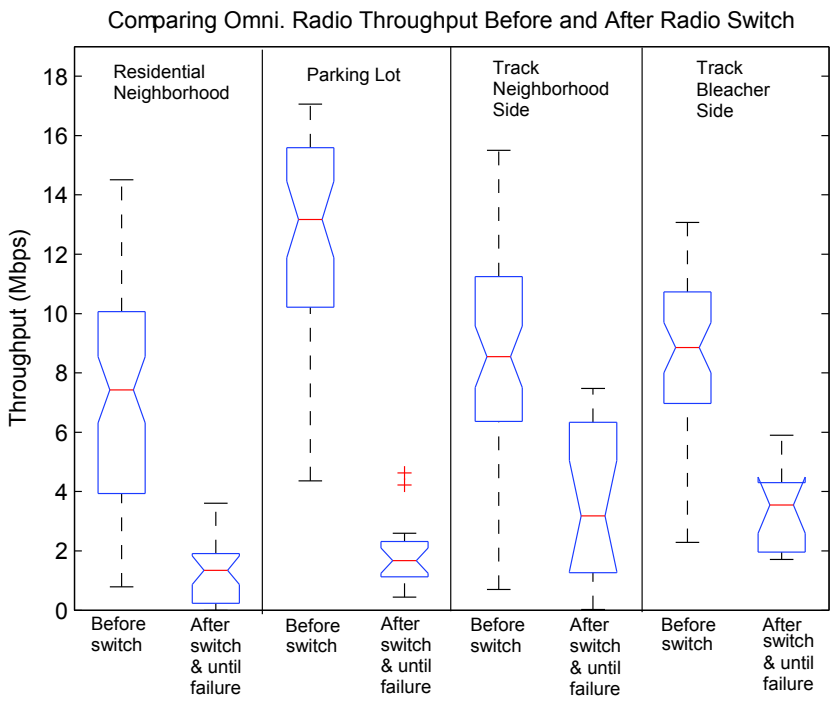

Figure 21. Boxplots used to evaluate the timeliness of the radio handoffs.

The images in Figure 20 show that the FLC successfully triggered the directional radio *prior to* and relatively *near* the omnidirectional radio's failure point, which avoided any disruption to communications and limited energy consumption. The nearness of each radio transition, respective to the failure point of the omnidirectional radio, can be more closely evaluated in Figure 21. The figure compares the throughput achieved by the omni-radio *before* and *after* the radio switch by the FLC. Notched boxplots were used to compare the measurements statistically. The boxplots show that the throughput for the omnidirectional radio was significantly degraded after the FLC decided to make the switch. Therefore, it appears that the FLC made a timely switchover to the directional radio, given the consistently lower throughput experienced by the omnidirectional radio.

Effectiveness of Tuning Agent and Hysteresis

Additional experiments were conducted to analyze the impact of the hysteresis design, as well as the optimization agent. In these experiments, the robot was controlled in the same manner as previously mentioned, except that after every radio switch, the robot would turn 180° and then proceed in the opposite direction. The goal of the robot's back-and-forth movement, either away or toward the OCU, was to create the necessary LQ conditions for a series of subsequent radio handoffs so that the effects of hysteresis and the optimization agent could be observed. Once the FLC triggered each radio handoff, the distance between the robot and the control station was recorded from the robot's GPS sensor so that it could be used to evaluate both effects. The experiments were halted once the adaptation of the fuzzy sets caused a hard handoff, meaning that the radio link failed before a handoff was proactively initiated by the fuzzy controller.

Figure 22 shows the results from the aforementioned testing for a specific trial conducted outdoors in a residential neighborhood. The subplot on the left conveys how the fuzzy sets were progressively lowered by the reinforcement agent after each post-handoff observation period. Only the '*weak*' fuzzy sets on the RSSI universe are shown in the figure in order to simplify its appearance and more easily convey the changes induced by the agent, but in actuality, all the fuzzy sets on both universes (i.e., RSSI and SQ) were shifted after each adjustment. After the optimization agent made the third lowering adjustment, the omnidirectional radio link eventually failed, as indicated by the right-hand subplot; link failure occurred while the robot was moving farther away from the control station and before a handoff was initiated to the directional radio. Consequently, communications were temporarily interrupted by the link failure, and due to the

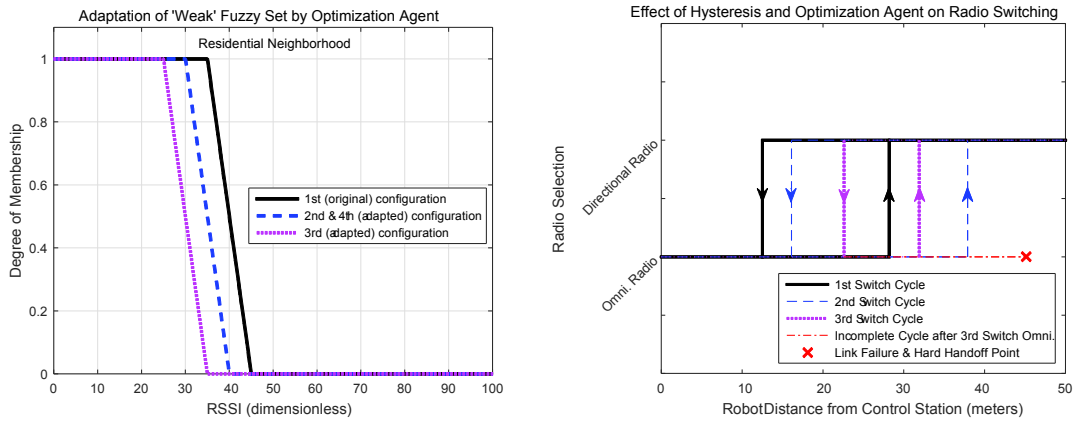

Figure 22. Effects of optimization agent and hysteresis on fuzzy set adaptation and radio switch timing

interruption, the optimization agent inferred that it needed to shift the fuzzy membership functions back (i.e., to the right) 5% to the previous configuration used after the second adaptation. From the results, it is evident that the optimization agent tuned the system to a more optimal operating configuration for the given environment than originally set by the expert.

The right subplot of Figure 22 can also be used to make observations about the effects of hysteresis design. The figure shows the evolution of the hysteresis function due to the fuzzy sets being adjusted by the agent. Generally, the figure shows a rightward shift of the hysteresis function after every fuzzy set adjustment. In terms of distance, each switch cycle is generally shown to have occurred farther away from the control station after each adaptation. However, as indicated between the second and third transitions to the directional radio, the third transition occurred closer to the control station despite it having fuzzy sets placed 5% lower on each universe. The likely explanation for this observation is that other LQ factors besides distance, such as interference and fading, changed somewhat as a result of the channel's time-varying nature and the slightly different

mobility paths taken by the robot. Regardless, the spacing between radio handoffs indicates overall that the hysteresis design proved to be effective at preventing rapid transitions between radio states.

Conclusion

This chapter introduced an efficient alternative means for achieving directivity in mobile robot networks. The concept is based on adding antenna reflectors to robots as part of a secondary radio system. To minimize the energy cost of the added directional radio, it is switched between sleep and idle states based on the assessed link conditions of the primary omnidirectional radio. A FLC was designed to perform the LQ assessment and radio selection decision. It was explained how the fuzzy-based LQ estimator is more efficient than existing approaches in terms of computational, energy, and network overhead. The computational load of the LQ estimator was reduced by minimizing the number of inputs into the FLC and taking steps to simply the defuzzification process. Furthermore, energy consumption was lowered by using the readily-available metrics provided by the radio hardware (i.e., RSSI and SQ) that are transparently updated via protocol feedback that is inherent in IEEE 802.11; this eliminated the need to transmit forced probe messages for the purpose of calculating software-based statistics, thus saving time and energy. Similarly, the frequency bandwidth of the channel is conserved without the need to transmit periodic probe packets for LQ estimation.

After detailing the design of the FLC, it was evaluated using simulation and physical experiments. All of the results corroborate that the dual-radio system enhances throughput and extends transmission distance. Additionally, the physical experiments demonstrated the timely response of the FLC by showing its ability to detect impending

link failure in diverse environments, thus preventing any disruption in communications. The simulation results indicated that the proposed dual-radio system, despite powering two radios, can actually be more energy efficient than the traditional single-radio system at extended distances.

Despite the proven potential for the LQ estimator, there are a few drawbacks to the design that merit further refinement. First, the initial configuration of the controller is based on the knowledge provided by an expert. Unfortunately, the initial hardcoded fuzzy sets may be suboptimal for some locations and hardware configurations. Secondly, the reinforcement agent revealed in this chapter is limited to modifying the fuzzy sets only after observing radio handoffs, and the adjustments can be slow or imprecise due to the nature of its trial-and-error form of learning. It would be better if the fuzzy LQ estimator could perform more optimal self-configuration while operating online, and then, regularly modify its fuzzy sets based on new observations and changes in the environment. These objectives are possible with an incremental learning framework. In the next chapter, machine learning is incorporated into the fuzzy-based design so that it can be seamlessly ported to virtually any environment and adapt its fuzzy membership functions accordingly. However, the radio-switching application is abstracted in the next evolution of the design so that it is more applicable to other types of optimization applications.

CHAPTER V

INCORPORATING MACHINE LEARNING INTO FUZZY-BASED LINK QUALITY PREDICTION

Introduction

One of the primary intentions in the previous chapter was to demonstrate the potential of link quality (LQ) estimation in robotic applications, and how periodic LQ assessments can be used to improve the fault tolerance, efficiency, and transmission range of robot networks. In addition, the application also served as a way to introduce the concepts of fuzzy logic and its suitability in the context of LQ estimation. However, the fuzzy logic controller (FLC) introduced in Chapter IV requires some modifications in order to improve the design's extensibility, as well as its adaptability. Both of these attributes were commonly deficient in many of the LQ estimators discussed in Chapter III, and hence, are the primary motivating factors behind the work in this chapter.

One of the goals is to make the redesigned architecture more extensible to other applications that commonly operate on robotic systems, such as navigation control systems and ad hoc routing protocols. The previous FLC from Chapter IV did not output any estimate of LQ, but instead, it used LQ estimates internally to make radio handoff decisions. In contrast, the objective now is to output a generalized indicator of LQ that could be used in a variety of decision-making applications. A commonly used target variable for quantifying LQ is packet reception ratio (PRR); however, as will be discussed, there are some challenges associated with using PRR as the system's target output. Hence,

a new target variable known as throughput potential ratio (TPR) is introduced to address these concerns. Another deviation from the previous chapter is that the new system is resigned to generate short-term *predictions.* However, the capability to make slower-moving LQ *estimates* is retrained by filtering a recent window of past LQ predictions. By introducing the capability to generate both (i.e., *predictions* and *estimates*), the system becomes more versatile and can be used by applications that demand higher accuracy over a short interval such as a routing protocol, or those that demand more stable and slower-moving estimates such as a navigation system.

The other primary objective of this chapter is to improve the adaptability of the previous fuzzy design. Adaptability was also a primary concern with several of the other LQ estimators previously presented in Chapter III. In terms of the proposed design, the goal is to retain the desirable aspects of fuzzy logic including its intuitiveness, flexibility, and resiliency to noise. However, in contrast to the previous expert-designed system, the new design automates the fuzzification process while the system is online. Traditionally, fuzzification is accomplished using triangular-shaped, Gaussian-shaped, or other types of functions that are often setup using domain knowledge and sometimes remain static after configuration. However, given the dynamic nature of the wireless channel, membership functions need the capability to learn and adapt to changes in link conditions. To perform this function, a novel way of incorporating machine learning into the process of fuzzification is developed. **Specifically, a series of binary classifiers are used to learn the underlying mapping functions relating the LQ inputs to a set of linguistic values defined by fuzzy sets, and degrees of membership to each fuzzy set are assigned using the posterior probabilities from the classifiers**. Through retraining, the classifiers can

update their mapping functions over time using newer training examples, thus achieving the desired attribute of adaptability.

In this chapter, two different ways of incorporating machine learning into fuzzy-based LQ estimation are introduced. The methods are similar in the regards that they both use the posterior probabilities from a collective group of binary classifiers to make fuzzy membership assignments. However, the hybrid classification-to-fuzzy architectures differ in way they use the linguistic values to generate crisp (i.e., continuous) LQ estimates or predictions. In other words, the inference and defuzzification processes of each method differs, and these differences are similar to those that exist between the classical fuzzy architectures of Mamdani and Takagi-Sugeno (T-S) [34].

Motivation

In the previous chapters, several motivating factors for using fuzzy logic in LQ estimation were provided. To briefly summarize, one of the primary benefits of fuzzy logic is its ability to classify inputs with varying degrees of certainty into linguistic values, and afterwards, process them in a systematic and logical way to generate crisp outputs. Fuzzy logic's use of linguistics closely resembles the human reasoning process, and its use of a rule base also facilitates the incorporation of domain knowledge into the LQ estimation process [34]. These features make the setup of a fuzzy system by an expert an attractive option. However, these types of fuzzy systems require offline experimentation and analysis, which tends to be cumbersome and time-consuming. In addition, membership functions designed using expert knowledge tend to remain fixed, and thus for dynamic processes such as the wireless channel, static membership functions will likely become suboptimal over time due to concept drift. Roughly speaking, concept drift occurs

whenever the statistical properties of the data changes over time for some reason [104]; a more precise definition is provided in the next chapter when the focus shifts to online and streaming evaluations of the design. Despite the dynamic nature of LQ, all of the existing fuzzy-based estimators reviewed in Chapter III were configured based on expert knowledge and static membership functions; the justification was that the overlapping structure of the fuzzy sets would be resilient to noise and mitigate its effects. But in reality, fuzzy sets lose accuracy when the membership functions become suboptimal due to concept drift. Therefore, the objective is to add adaptability to fuzzy-based LQ estimation and prediction systems so that membership functions adapt and accuracy can be sustained regardless of drifting.

There exist adaptive fuzzy methods for identifying and controlling dynamic systems, which could potentially be used in LQ estimation. However, to the best of the author's knowledge, the feasibility of applying adaptive fuzzy control in LQ estimation has not been explored. The likely explanation is that adaptive fuzzy control requires frequent sampling of the input-output variables so that the input can be adjusted to control the output. However, in the domain of LQ prediction or estimation, the paradigm is different in that the output cannot be easily controlled, and instead, the objective is to either predict or estimate LQ with minimal sampling overhead. The primary challenge is the fact that the output (i.e., target variable) is measured at the receiver, while the predictor or control system must reside at the sender. Therefore, rapid sampling of the input-output variables would be relatively expensive in the case of adaptive fuzzy control, given the fact that the output measurements must be transmitted across the wireless channel to the sender.

Consequently, the alternative approach taken in this dissertation to add adaptability to fuzzy-based models using machine learning instead.

There are other reasons for the introduction of a new means of achieving adaptability in fuzzy systems. One of them includes the added flexibility of being applicable to either Mamdani or T-S fuzzy systems, in contrast with adaptive fuzzy control which can only be applied to T-S architectures [34]. The drawback with adaptive fuzzy control is that T-S systems tend to be less intuitive than Mamdani systems [34], but with the proposed method, the desirable features of intuitiveness and adaptability are made possible. Additionally, the approach offers a more straightforward and natural way of classifying the inputs (i.e., features) into fuzzy linguistic sets with varying degrees of certainty than existing methods. Finally, the approach lends itself to an easier implementation because of the accessibility of learning algorithms in open-source libraries such as scikit-learn [105] and Weka [106].

The concept of supplementing fuzzy systems with machine learning is not new, but the methodology used in this dissertation to incorporate machine learning into fuzzy logic is unique to the best of the author's knowledge. The closest resemblance to the proposed approach outside the domain of LQ estimation is fuzzy cluster analysis [37]. Both methods share the same fundamental concept of assigning levels of certainty to class memberships using posterior probabilities, but the problem domain, as well as the execution details, are different as will become more evident later. In fuzzy clustering, the principle idea is that an object may not completely belong to just one cluster, but instead, can have levels of membership in multiple clusters. Similarly, the proposed model uses classification to

assign the inputs to linguistic values, such as '*low*' or '*moderate low'*, with varying levels of certainty.

In contrast to the approach taken in this dissertation, most other hybrid methodologies use machine learning to automate the process of fuzzy rule-generation [35]. On the other hand, the proposed approach uses machine learning for adapting the fuzzification process only, as opposed to tuning the inference and defuzzification processes as well. This chosen direction is based on the assumption that the rule base and reasoning process do not need to change, and instead, the functions that classify the inputs into linguistic values only need adjusting over time to handle concept drift. For instance, the fuzzy logic that infers LQ to be '*poor*' when RSSI and SQ are both '*low*' does not need to change, nor does the inference and defuzzification processes that average all of the fired rules together to generate a crisp output. However, what constitutes RSSI to be '*low*' in one environment might be slightly different in another geographic location. Or, said in a different fashion, the functional mapping of a feature to a target variable likely changes over time based on the dynamics of wireless channel. Therefore, the author argues that **the only essential adaptation the system needs to perform is the tuning of its membership functions that assign the inputs into linguistic classes**.

As an alternative to the proposed method of combining machine learning with fuzzy logic, it is possible to use solely machine learning for performing LQ prediction, as previously discussed in Chapter III. However, the author feels that the proposed hybrid method and its overarching fuzzy architecture offers the designer more flexibility and control in tuning the performance (i.e., output) of the system. For instance, an engineer or scientist may have special knowledge about a dataset or process, but it may be cumbersome

to incorporate that domain knowledge into an existing machine learning algorithm such as linear regression. On the other hand, the inference mechanism and knowledge base that come inherent with fuzzy designs make incorporating domain knowledge a more natural process.

Preliminaries

Understanding the Output Variable

Many of the existing designs discussed in Chapter III use packet reception ratio (PRR) as the target variable for prediction or estimation. However, there is a significant challenge with using PRR. To empirically measure PRR, applications must have the ability to track the delivery status of individual packets. In reality, higher-layer applications do not natively have this capability because the job of fragmenting and delivering individual packets is delegated to the lower layers. These internal processes are encapsulated and hidden from the sender's application layer, which is where the LQ predictions are being formed. It is possible to circumvent this issue by modifying the wireless driver or operating system, but these changes can be complex and system specific. A possible exception that could facilitate the use of PRR would be if the application is using a non-guaranteed delivery protocol such as User Datagram Protocol (UDP); however, monitoring the success or failure of packet delivery at the transport layer, where UDP operates, still does not provide a complete picture into the actual reliability of the wireless channel. For instance, IEEE 802.11 has built-in reliability mechanisms at the data link layer, and these mechanisms, which include acknowledgements and retransmissions, are not natively tracked by the upper layers. Therefore, from the standpoint of UDP, the wireless channel

may appear to be better than actuality due to the hidden reliability mechanism residing at the lower layers.

To circumvent the issues associated with PRR, a new target statistic is introduced. The LQ indicator is referred to as *throughput potential ratio* (TPR), and it does not inherit the packet counting issues of PRR. Instead of counting packets, TPR tracks the amount of time that it takes to transmit a finite set of application data. A given TPR can be calculated by a receiver that uses a timer to track the starting and ending time of any reception, and such a task can be accomplished assuming that the application encapsulates individual transmissions with a distinguishable header and trailer sequence. Using this approach, the receiver can calculate the data rate for the i^{th} transmision, r_i, by dividing the amount of information received (in bytes) by the time (in seconds) it took to receive the information. In order to convert a given data rate into TPR, it is normalized by dividing r_i by the maximum data rate observed over the link since its inception. In summary, TPR can be concisely defined as

$$\text{TPR}_i = \frac{r_i}{\max(r)}$$

The normalization of each data rate instance by max(r) transforms the statistic into ratio that varies in range from 0 to 1. Its continuous range can be used by a number of optimization applications, such as those previously mentioned in Chapter III, to clearly distinguish LQ. However, the challenge with TPR, similar to PRR, is that the target variable is measured at the receiver and is not readily known by the sender. But, in order for a sender to proactively make network optimizations, the sender should possess an accurate estimate of its TPR in the forward direction of the link. Ideally, these predictions or estimates of TPR can be formed by the sender on an as-needed basis with minimal

feedback from the receiver so that overhead (i.e., energy and network congestion) is minimized.

Supervised Learning in the Context of Link Quality Prediction

Given the complex nature of radio wave propagation in mobile environments, the wireless channel tends to be classified either probabilistically or statistically [16]. Thus, empirical measurements of the inputs and outputs are essential in order to establish properties about the data. In the case of supervised learning applied to this application, statistical inference can be used to extract dependencies between the inputs and the output variable of TPR. More specifically, a sender can use empirical measurements and a supervised learning algorithm to find an approximation function, $g(\mathbf{x})$, to the unknown target function, $f(\mathbf{x})$, that relates the LQ metrics (i.e., features) to the target variable of TPR. A generalized depiction of supervised learning applied to the domain of LQ estimation is shown Figure 23. The figure shows that supervised learning uses *labeled* training samples to induce a mapping relationship between a set of LQ features from the domain of X and to the target variable, TPR, in Y. As illustrated in the figure, the LQ features are generally available at the sender, but the target variable is measured at the receiver and must be transmitted to the sender in order to form labeled sample pairs. Generalizing a function, $g(\mathbf{x})$, can limit the need for the receiver to continuously send TPR measurements, but due to the dynamics of the wireless channel, the model must be periodically refreshed using new training examples.

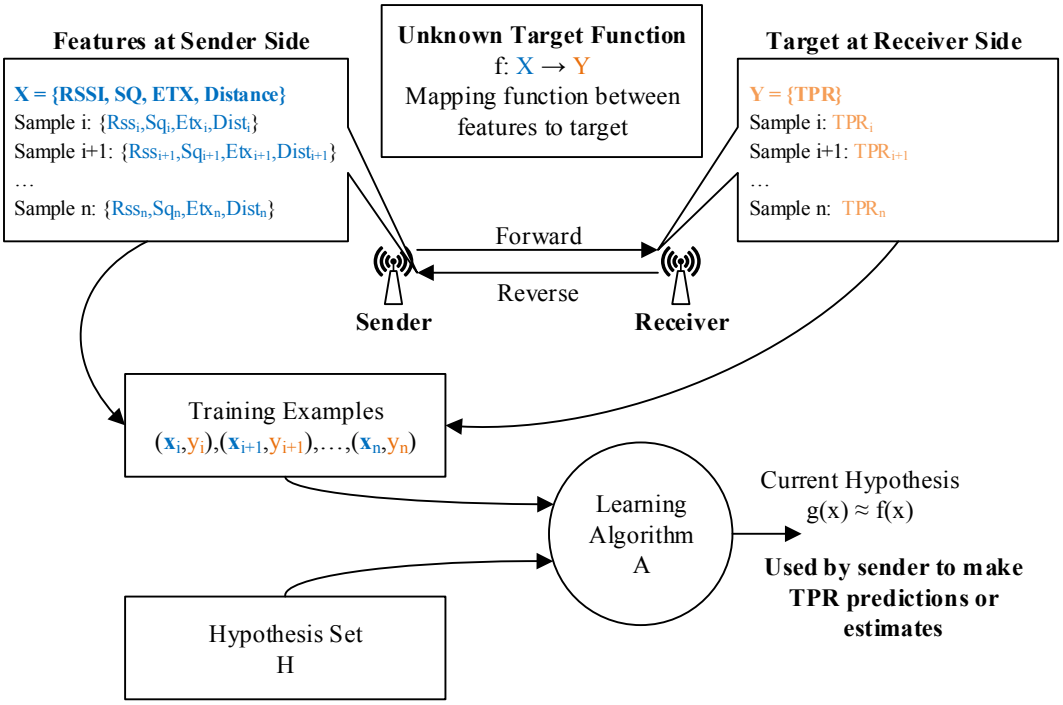

Figure 23. The process of supervised learning applied to LQ prediction.

In the proposed model soon to be revealed in this chapter, supervised learning is not used for the entire process of making LQ predictions as depicted in Figure 23. Instead, more flexibility and intuitiveness into the prediction process is desired by using supervised learning as part of a fuzzy architecture. **Specifically, the proposed model uses supervised learning to perform the initial fuzzification step of assigning the LQ features into fuzzy sets.** To do so requires that the sender convert the TPR measurements from the receiver into class labels that are reflective of linguistic values commonly used in fuzzy logic such as '*high*' or '*low*'. Afterwards, a set of classifiers are used to perform the fuzzification process. The added benefit of using supervised learning for fuzzification is that the process becomes automated and adaptable to changes in the underlying mapping functions, assuming that the classifiers are periodically retrained using new empirical samples.

Adaptable Fuzzification using Binary Classifiers

Mamdani Ensemble-to-Fuzzy Architecture

Mamdani fuzzy systems tend to be more intuitive than Takagi-Sugeno (T-S) systems based on the structure of their rules [34]. In Mamdani systems, natural language is used to describe the premise and consequent of every rule. On the other hand, T-S systems employ linear algebraic functions to define the consequents, making them more difficult to comprehend. The goal of the Mamdani ensemble-to-fuzzy (ETF) architecture is to retain the intuitiveness of classical Mamdani systems, yet transform its fuzzification process so that the system becomes adaptable.

In order to keep most of traditional Mamdani architecture intact, each input (i.e., LQ feature) must be fuzzified independently so that it remains a multiple-input, single-output (MISO) fuzzy system. In other words, the unique process of classifying the predictors into fuzzy sets is repeated individually for each predictor without regard to any others. Once fuzzification is complete, the ordinary Mamdani mechanisms of inference and defuzzification are then applied to generate a crisp LQ estimate.

The novel process of assigning fuzzy memberships relies on a series of binary classifiers that are diversely emplaced with respect to the target variable as depicted in Figure 24. The collection of classifiers is referred to as an *ensemble*. However, the employment of these classifiers does not meet the typical definition of an ensemble. Traditionally, the term *ensemble* refers to a diverse and independent set of predictive models that operate on the same training data, and the results of which are then combined in some fashion to generate a collective prediction that is consistently more accurate than

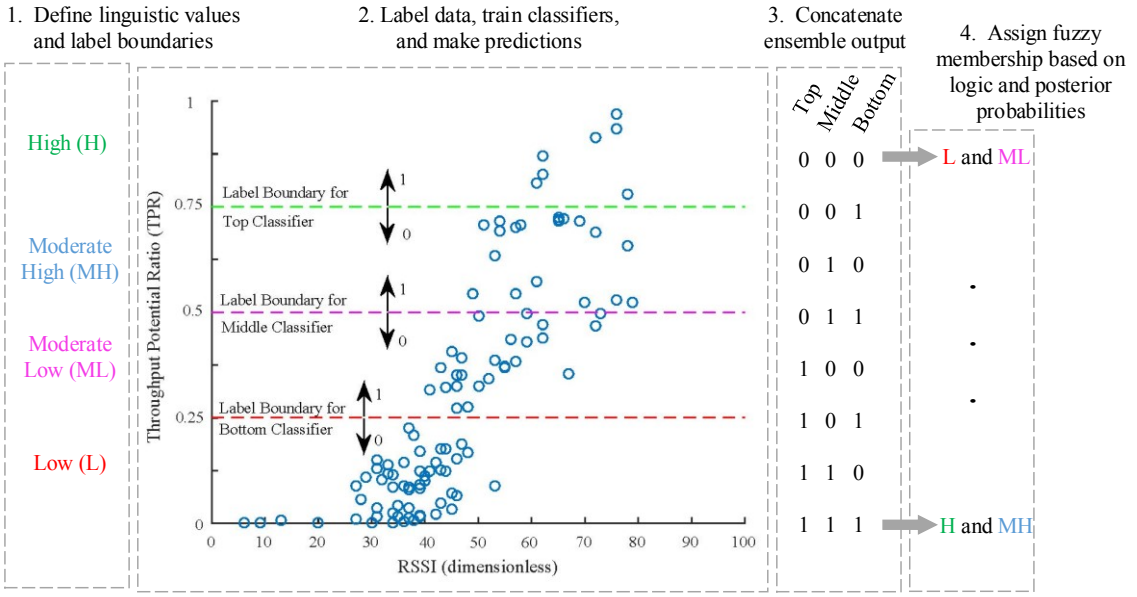

Figure 24. Steps for adaptable fuzzification using binary classifiers

any individual model [32]. On the other hand, the ensemble used in the ETF model consists of classifiers that use the *same* learning algorithm, not different ones that are typically found in ensembles. The diversity of the ensemble comes in the fact that the classifiers operate on different class labels due to their unique and strategic placement on the target universe, as indicated in Figure 24. Due to their diverse positioning, each classifier forms a unique perspective with regard to the classification of a predictor sample. After classification, the outputs from the classifiers are then combined using a series of *if-then-else* logic to determine the two most fitting fuzzy sets that any sample belongs.

The proposed method of fuzzification using an ensemble can be broken down into a series of steps. The first step is to divide the target universe into the desired number of equally-spaced regions or fuzzy sets. Each fuzzy set is then logically assigned a linguistic value. As shown on the left side of Figure 24, the sample space was classified into four different fuzzy sets that correspond to linguistic TPR levels of '*high*' (H), '*moderate high*'

(MH), '*moderate low'* (ML), and '*low*' (L). The number of binary classifiers needed to make the fuzzy distinctions is one less than the number of selected fuzzy sets; in this example case, three binary classifiers are used distinguish between four fuzzy sets.

As new training examples are received, each classifier must appropriately label and store them for training purposes. Labels are assigned differently for each classifier based on their perspective or boundary positioning on the TPR universe. The scheme requires that the classifiers are labeled consistently, as shown in Figure 24. In other words, each classifier within the ensemble would output a binary '1' if the target TPR value resides above its respective boundary; otherwise, a binary '0' would be outputted by the classifier.

Once labeling is complete, the classifiers can be trained and the classification process can begin. At this point, it is worthwhile to note a couple of temporary assumptions with this step. These assumptions will be removed and fully addressed in the next chapter. For the moment, it is assumed that a sufficient sample representation has been collected above and below each classifier boundary so that each classifier can be trained. Additionally, it is assumed that a streaming framework is in place for the systematic labeling and retraining of the classifiers so that effects of concept drift are minimized.

Assuming the classifiers are trained, the rest of the fuzzification process proceeds as follows. When presented with a predictor sample (e.g., an RSSI sample), each classifier makes a classification prediction as to whether the sample most likely belongs above or below its boundary perspective. Afterwards, the outputs from each classifier are concatenated together as shown in Figure 24 and then fed through *if-then-else* logic to identify the *two* most appropriate fuzzy sets to which the sample belongs. This selection is based on the level of agreement among the classifiers. If all of the classifiers point to a

particular fuzzy set, then the classifier nearest to this region, with respect to its boundary position, is used for making membership assignments. For instance, in the case of an ensemble output of '000', all of the classifiers agree that the sample primarily belongs to the fuzzy set of '*low'* (L), but the sample may also partially belong with a degree of less than 0.5 to the neighboring or secondary fuzzy set of *'moderate low'* (ML). Because the bottom classifier divides these two fuzzy sets, its posterior probabilities are used to determine the membership levels of the fuzzy sets above and below its boundary. A similar fuzzification process, as indicated in Table 5, is used for each of the eight possible ensemble outputs shown in Figure 24. The table shows that the posterior probabilities from a particular classifier are used in each case, and thus, the membership levels of the two selected fuzzy sets always sum to one.

The other possible ensemble outputs displayed in Table 5, besides the straightforward cases of '000' and '111', require some additional consideration before selecting the two most appropriate fuzzy sets. For instance, sub-nested *if-then* logic shown in Table 5 is used in the cases of '001' and '011' in order to identify the most-fitting secondary fuzzy set for a given sample. As an example, consider the ensemble output of '011'; in this case, the classifiers agree that the sample primarily belongs to the *'moderate high'* (MH) fuzzy set. However, it is unclear whether the secondary fuzzy set should be *'high'* (H) or *'moderate low'* (ML) because the sample may, in actuality, lie either toward the top or middle classifier boundary. To make this determination, the two nearest classifiers are ranked with regard to their posterior probabilities. Specifically, in this example case, if the middle classifier is more confident about its primary classification than the top classifier, then it logically implies that the sample resides closer to the top boundary

Table 5

Fuzzification Logic used in the Mamdani Form of the Ensemble-to-Fuzzy Method

Ensemble Output			Fuzzy Set Membership Assignments
Top	**Middle**	**Bottom**	
0	0	0	L = $P_{bottom}(0\|\mathbf{x})$ ML = $P_{bottom}(1\|\mathbf{x})$
0	0	1	*if* $P_{bottom}(1\|\mathbf{x}) > P_{middle}(0\|\mathbf{x})$, then ML = $P_{middle}(0\|\mathbf{x})$ and MH = $P_{middle}(1\|\mathbf{x})$ *else*: ML = $P_{bottom}(1\|\mathbf{x})$ and L = $P_{bottom}(0\|\mathbf{x})$
0	1	0	ML = $P_{middle}(0\|\mathbf{x})$ MH = $P_{middle}(1\|\mathbf{x})$
0	1	1	*if* $P_{middle}(1\|\mathbf{x}) > P_{top}(0\|\mathbf{x})$, then MH = $P_{top}(0\|\mathbf{x})$ and H = $P_{top}(1\|\mathbf{x})$ *else*: MH = $P_{middle}(1\|\mathbf{x})$ and ML = $P_{middle}(0\|\mathbf{x})$
1	0	0	ML = $P_{bottom}(1\|\mathbf{x})$ L = $P_{bottom}(0\|\mathbf{x})$
1	0	1	ML = $P_{middle}(0\|\mathbf{x})$ MH = $P_{middle}(1\|\mathbf{x})$
1	1	0	MH = $P_{top}(0\|\mathbf{x})$ H = $P_{top}(1\|\mathbf{x})$
1	1	1	MH = $P_{top}(0\|\mathbf{x})$ H = $P_{top}(1\|\mathbf{x})$

and the *'high'* (H) fuzzy set; otherwise, it implies the sample resides more toward the middle boundary and the *'moderate high'* (MH) fuzzy set.

The remaining ensemble outputs in Table 5 are more ambiguous than those previously discussed due to partial disagreement among the classifiers. For instance, the outputs of '100' and '110' indicate that only two of the three classifiers agree on the classification of the sample. In these cases, majority-rule logic is employed to select the two fuzzy set nearest to the region where most of the classifiers agree that the sample belongs. However, in the cases of '010' and '101', there is no adjacent agreement between the classifiers and majority rule does not make sense. In the event of these rare cases, the

best option is to make the fuzzy assignment decision using the middle classifier and to assign the sample into the two moderate fuzzy sets as shown in Table 5. This is the most conservative approach that may minimize the margin of error when compared to choosing other fuzzy sets closer to a particular extreme.

Once each feature has been fuzzified using the above approach, the traditional fuzzy steps of inference and defuzzification would commence. In other words, the assigned memberships and the rule base would be used to infer a number of consequents. Then, the defuzzification of these consequences (i.e., averaging of the fired rules) would take place so that a crisp (i.e., continuous) LQ prediction could be generated.

Takagi-Sugeno Ensemble-to-Fuzzy Architecture

A drawback to the aforementioned Mamdani architecture is that it faces the so-called *combinatorial explosion problem* as a result of its rule structure [96]. Essentially, the problem is that the complexity of the fuzzy system grows exponentially with the number of inputs and fuzzy sets used [96]. As part of this complexity problem, the rule base becomes increasingly unmanageable without any sort of learning algorithm that can auto-generate the rule-base. For instance, a four-input system with four fuzzy sets per input would require $4^{\wedge 4}$ rules or 256 rules. It would be cumbersome for an expert to impart domain knowledge into a system with such a large rule base. As an added scalability issue, the proposed adaptable fuzzification method for Mamdani systems requires a certain number of classifiers per predictor as previously discussed. For instance, using the same four-input, four fuzzy set system example, a total of 12 binary classifiers would be needed in this case, given that three classifiers are required per ensemble to create four fuzzy sets and one ensemble is required per input. The Mamdani architecture may be intuitive and

easily allow for the interjection of domain knowledge into the fuzzy reasoning process, but a significant drawback is that the design is does not scale well and is only practical for systems requiring a few inputs. Unfortunately, complex systems sometimes demand a large number of features in order to obtain the desired predictive accuracy, and in these scenarios, the Mamdani architecture may prove challenging to implement.

To mitigate the scalability issue, a more streamlined ETF architecture is introduced. In contrast to the previous approach, the input features are not classified independently using a separate ensemble for each individual feature. Instead, only a single ensemble is used to make the fuzzification assignments, and each classifier within the ensemble is trained using the full set of features. In other words, each training example, $<\mathbf{x}_i, y_i>$, contains a feature vector $\mathbf{x}_i$, where all the selected features are combined with a target label, y_i. Without the need for multiple ensembles, the computational burden of fuzzification is significantly reduced.

Once the ensemble has been trained, the system is ready to perform its intended fuzzification process. Similar to the previous Mamdani architecture, the binary predictions from each classifier are concatenated and then processed through rule-base logic (i.e., *if-then-else* logic) to determine the appropriate consequents. However, for this T-S style architecture, the consequents take the form of algebraic equations, not fuzzy sets like the Mamdani architecture.

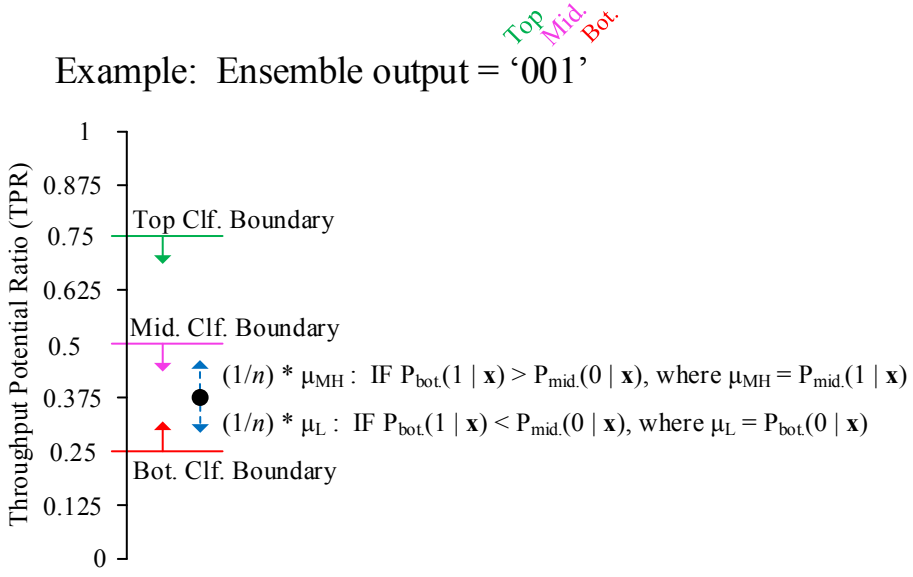

Figure 25. An example to illustrate the inference mechanism used in Takagi-Sugeno ensemble-to-fuzzy architecture.

An illustration was created to better explain the fuzzification process and the makeup of the consequent equations. The example shown in Figure 25 assumes that the concatenated binary output from the ensemble is '001', where the top and middle classifier outputs are '0' and the bottom classifier output is '1'. Given this example sequence, it is evident that the all of the classifiers tend to agree, as indicated by the colored arrows in the figure, that the sample primarily belongs somewhere in the '*moderate low*' fuzzy set. However, to be more precise with the classification, fuzzy logic can be used to try and narrow down whether the sample lies closer to the boundary of the middle classifier or whether it lies closer to the boundary near the bottom classifier. As with traditional fuzzy logic, a sample usually has some level of secondary membership to another fuzzy set, and the aim of the proposed approach is to determine the most appropriate secondary fuzzy set. In order to make this determination, the confidence of the classifications between the two

surrounding classifiers can be compared using their posterior probabilities. For instance, if the bottom classifier in Figure 25 is more confident about its classification than the middle classifier, then it can be reasonably inferred that the sample lies more toward the middle boundary and the 'moderate high' fuzzy set; otherwise, it implies an opposite shift toward the secondary fuzzy set of '*low*'. In either case, the level of membership that the sample assumes in the secondary fuzzy set will always be less than 0.5, and the precise membership level is determined by the nearest classifier and its posterior probability that points to the secondary fuzzy set.

Next, the proposed T-S ETF system will be generalized in terms of its rule-base and consequent logic. The rule base logic consists of two tiers of *if-then-else* logic. The outer level of *if-then* logic processes the concatenated ensemble output and identifies a primary fuzzy set, *j*, where the sample most likely belongs. Afterwards, an inner set of *if-then* logic identifies the secondary fuzzy set, *k*, using the posterior probabilities of the surrounding classifiers. Based on the outcome of the inner set of *if-then* logic, a consequent equation would then be applied to determine the crisp output (i.e., TPR prediction) from the ETF system, and the generalized form of such a consequent would be

$$y^{\text{crisp}} = m_j \pm \frac{1}{n}\mu_k$$

where m_j is the TPR midpoint associated with the primary fuzzy set, *j*, and μ_k is the level of membership to secondary fuzzy set, *k*. The constant *n* is the number of fuzzy sets used to divide the TPR universe, and it is responsible for bounding the maximum amount of shift the sample point can move away from the midpoint, m_j, given μ_k. The assignment of plus (+) or minus (-) sign in the above equation corresponds to whether the shift in TPR

from the fuzzy set midpoint is higher or lower, respectively, and the sign is determined by the rules established within the inner if-then logic.

Evaluation

The evaluations of this section focus exclusively on the T-S version of the ETF architecture because of the model's scalability advantages previously discussed. The T-S style architecture calls for an ensemble of binary classifiers, and as a result, standard evaluation procedures from machine learning can be used to evaluate the ensemble portion of the design. One of the primary objectives of this section is to identify the best performing supervised learning classification algorithm for the problem set. It is well-known from the so-called "No Free Lunch Theorem" that there is no single learning algorithm that is best-suited for all applications [32]; hence, the requirement to compare the performance of several leading supervised learning classification algorithms in the context of the given problem.

Another objective of the evaluation is to perform feature selection. The investigative process establishes a ranking of the features in terms of their predictive power or usefulness towards making accurate predictions. As a result, the rankings can be used to perform feature reduction if needed.

The final evaluation measures the accuracy of the entire hybrid system. It is important to note that the complete system acts like a learning regression algorithm due to its crisp (i.e., continuous) output. Therefore, the accuracy of the proposed model is compared with that of a generalized linear regression (GLR) model. The GLR model was selected to baseline the performance of the proposed system given that linear regression modeling is the most frequently applied statistical technique [32]. All of the learning

algorithms, as well as the associated evaluation algorithms in this section, were implemented using scikit-learn [105], an open-source library of machine learning algorithms written in Python.

Dataset Collection Procedure

In order to evaluate the proposed model, a series of real-world datasets were collected in a variety of environments. The datasets were gathered by the robot shown in Figure 26, and a description of each collection environment is included in Table 6. At each location, the robot was controlled by an operator control unit (OCU). The OCU consisted of a laptop with a graphic user interface (GUI), as shown in Figure 27. The GUI provided a means for controlling the robot, as well as for displaying images and prediction statistics received from the robot.

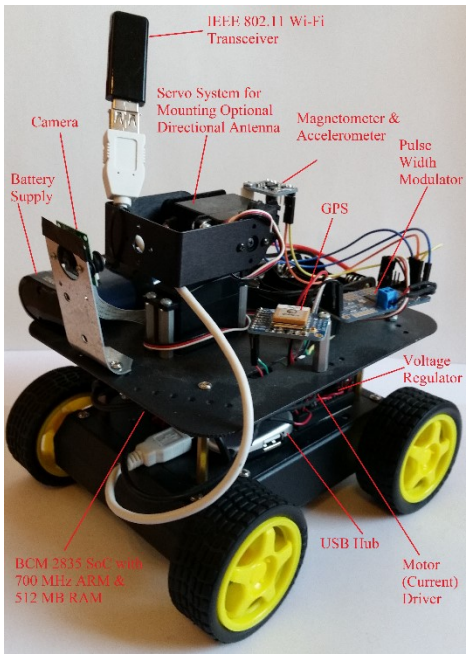

Figure 26. Picture of the robot used during the evaluation of the ensemble-to-fuzzy architecture.

Table 6

Description of Sample Collection Sites

Site Location	Dataset Name	Sample Size	Description
Residential Neighborhood	Residential (Outdoor)	400	Robot moved along residential sidewalk surrounded by homes.
Inside Residential Home	Residential (Indoor)	429	Robot moved through several rooms in basement while OCU was positioned on second floor.
Park	Park (LOS)	415	Robot moved through a large parking lot that was free of any obstacles.
Park	Park (NLOS)	377	An automobile was placed between the networked radios for a portion of the dataset.
Track	Track (LOS)	417	Robot moved along length of an outdoor track which was lined with residential homes on one side.
Track	Track (NLOS)	161	A box was placed over the robot for a portion of the dataset.

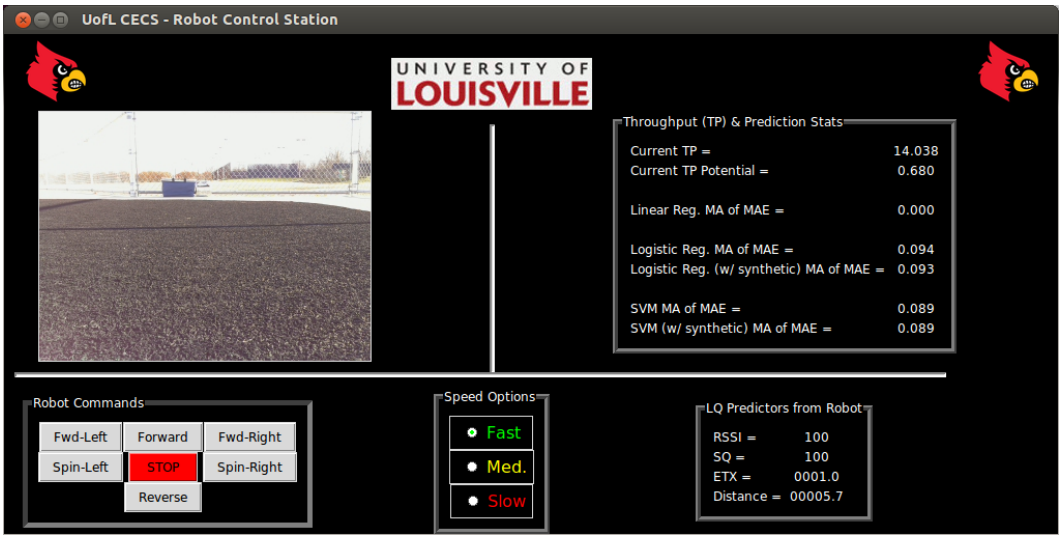

Figure 27. Snapshot of the operator control unit's graphical user interface

With the exception of the indoor experiment, the sampling of the empirical data generally proceeded as follows. The robot started next to the OCU and then was commanded to travel away from the OCU in a near linear fashion. Once the wireless IEEE 802.11 ad hoc connection was close to failing, the robot then reversed its direction and returned back to its originating position near the OCU. The route of travel was selected in order to force the robot to collect measurements from much of the LQ sample space that ranged from the very good (i.e., TPR near one) to the very poor (i.e, TPR near zero).

The LQ features selected for evaluation included received signal strength indicator (RSSI), signal quality (SQ), expected number of transmissions (ETX), and distance. With the exception of ETX, all of the features were collected with relatively little overhead. For instance, the radio metrics of RSSI and SQ were passively sampled from the robot's IEEE 802.11 radio, while distance was calculated using coordinates obtained from the robot's global positioning system (GPS) sensor. On the other hand, the sampling of ETX incurred the overhead associated with sending and receiving 1200 byte probes to and from the OCU every second.

The target variable of TPR was measured at the OCU and associated with the robot's LQ predictors using a numbering system. The OCU measured each TPR instance using the amount of time it took to receive an application packet from the robot via the transmission control protocol (TCP). Each application packet transmitted from the robot to the OCU consisted of a picture image and some other sensor data. A header and trailer encapsulated the application data so that the OCU could distinguish between the streaming packets, as well as time the reception of each. The typical payload of each packet ranged in size from about 60 kilobytes (kB) up to 1 megabyte (MB), depending upon the amount

of information needed to quantify each picture image. The robot transmitted these packets in a stream-like fashion approximately every 1.5 seconds based on the speed of the camera hardware and other factors. Prior to each discrete transmission, the robot would sample its predictors, and these predictors would eventually be paired with the TPR of that transmission in order to form *labeled* samples for offline testing. At this time, the robot did not actually generate any online predictions. However, the robot sampled the features shortly before each transmission, knowing that the intention was to eventually incorporate online *predictions* for each transmission.

As indicated in Table 6, the locations and conditions of the sampling processes were varied between experiments. The intent was to force changes in the underlying mapping function between the features and the target variable. By doing so, it would test the resiliency or robustness of the proposed model to generalize mapping functions under varying conditions. During one of the experiments on the track, a box wrapped in aluminum foil was placed over the robot for a portion of its route. The intent was to degrade the signals of the radio and GPS. In addition to those effects, it also caused the robot to skip a portion of the predictor space while sampling. Other anomalies were introduced into one of the datasets collected at the park. Specifically, sudden changes were introduced in the attenuation conditions of the wireless link by inserting and removing an automobile between the communicating radios for a portion of the robot's sampling route. For these two deviated datasets, the use of the generic label of 'non-line-of-sight' (NLOS) is used to distinguish them as a result of the line-of-sight (LOS) component being temporarily broken for a portion of these datasets. Collectively, the six datasets cover a variety of different operating conditions that could be expected in realistic robot scenarios.

Therefore, the datasets should facilitate a thorough feature and classifier selection process, as well as a complete model evaluation.

Classifier Selection

The first step of the classifier selection process was to prepare the datasets for classification. The samples within the datasets were originally stored with the continuous form of the target variable, TPR. However, for binary classification, the target variable needs to be labeled using one of two possible values. Therefore, the original target variables were re-labeled to either a '1' if they were above the classifier's respective positioning on the TPR universe, or a '0' otherwise. The process of re-labeling was repeated for each classifier based on its unique positioning (i.e. ¼ for the bottom classifier, ½ for the middle classifier, and ¾ for the top classifier), as denoted in Figure 24. Another preprocessing step included the normalization of the features so that the classification algorithms would not be biased by any disparity among the feature ranges, and for this purpose, the method of standard deviation normalization was used [32].

After preparing the datasets, different classifier options were compared using well-established evaluation methods. Three common classifier algorithms were chosen as part of the comparison: support vector machines (SVM), logistic regression (Log. Reg.), and Naïve Bayes (NB). The method of *k*-fold cross-validation (CV) was used to compare the prediction accuracy of each classifier given the prepared datasets. The number of folds, *k*, performed on each of the aforementioned datasets was set to 10, which is commonly used in k-fold CV [41].

The results of the CV testing are shown in the bar chart of Figure 28. The colored bars reflect the mean prediction accuracy from the ensemble-level, or in other words, the

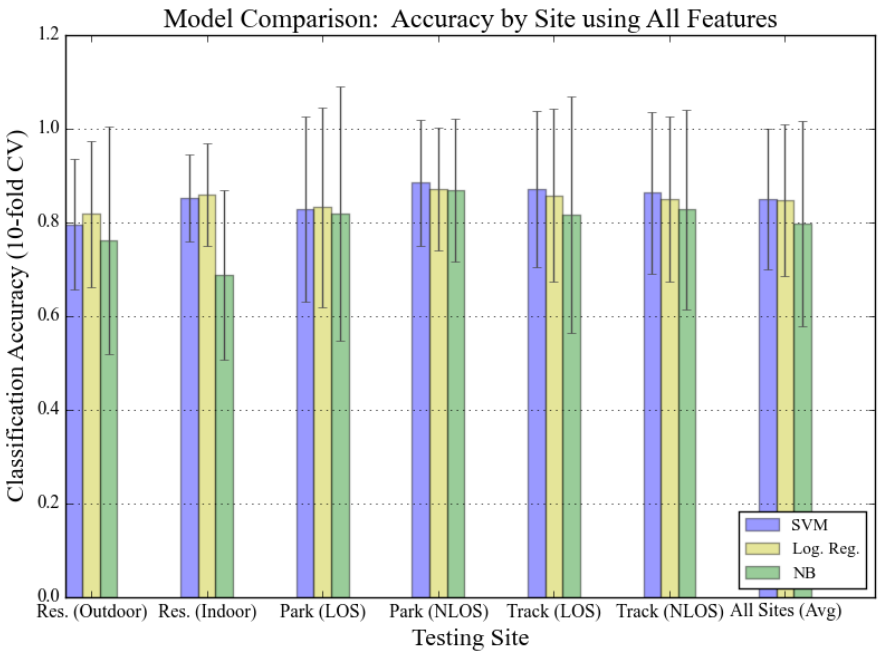

Figure 28. Comparing classifier accuracy using cross-validation

average accuracy achieved by all three classifiers (i.e. bottom, middle, and top). The error bars show the standard deviation in prediction accuracy. It is important to note on the *x*-axis that all of the results are separated and labeled according to a unique dataset, except for the far-right set of bars, which represent an overall average of the six other datasets.

Some conclusions can be drown from Figure 28. First, it is evident that the NB ensemble consistently underperformed the other two algorithms in terms of average accuracy. In addition, the NB algorithm had the undesirable attribute of having a wider standard deviation in accuracy. A likely explanation for these observations is the strong assumption that the NB algorithm makes in regards to the independence of each feature. On the other hand, the other two algorithms (i.e., SVM and Log. Reg.) demonstrated comparable performance. In fact, a close look at the figure shows that they each achieved the highest average accuracy on three occasions.

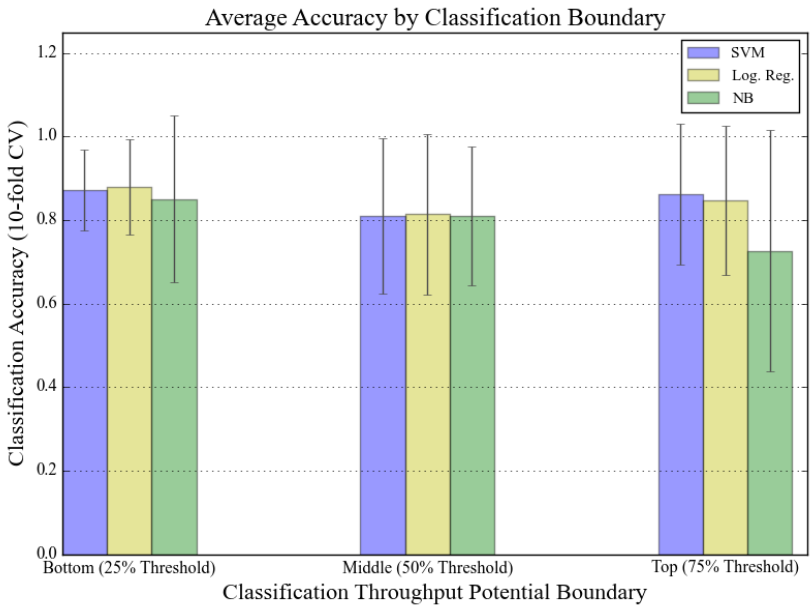

Figure 29. Accuracy of individual classifiers within the ensemble averaged from all datasets

Instead of only looking at classification accuracy from the ensemble-level, a deeper look was taken into the performance of each type of classifier within the ensemble. More specifically, the average CV accuracy achieved by the top, middle, and bottom classifiers were compared. To concisely summarize the findings displayed in Figure 29, the CV accuracy achieved across all six datasets was averaged together. The results again tend to show that the SVM and logistic regression algorithms performed nearly the same. Other interesting results from the figure include that fact that the middle classifier tended to have the most difficulty in distinguishing whether a sample was above or below its TPR threshold. Intuitively, this observation makes sense based on how the positioning of the classifier influences the difficulty of its classification task. The middle classifier has the most ambiguous (i.e. neutral) position along the TPR universe, and thus, its classification

task is likely more difficult than the other two classifiers, which have an off-centered perspective along the TPR universe. Another consideration that likely contributed to the task difficulty of the middle classifier is the fact that immediate quality links (i.e., those around 0.5 TPR) are known to have the most variance in LQ metrics [20]. Regardless, the accuracy of the middle classifier was not significantly lower than the other two classifier positions.

Another consideration, in addition to classifier accuracy, is the model's speed in terms of training and predicting. These factors were investigated knowing that the ETF architecture will eventually be implemented on the robot shown in Figure 26. Given the robot's somewhat resource-constrained hardware, the ensemble should be as computationally efficient as possible because of the streaming nature of the application and the number of other processes (or threads) the robot's central processing unit (CPU) must continuously load balance.

The first computational aspect investigated was the speed at which the robot's hardware could train each classification model when given different-sized training sets. The experiments were conducted on the robot's 700 MHz ARM SoC processor [107], and a total of 100 independent trials were conducted for each of the different-sized training sets shown in Figure 30. The results displayed in the figure clearly show that the SVM incurs the most training time, which increases with the size of the training set. This is likely the result of the extra time required by the SVM algorithm to build out the prediction probability model when the constructor option *'probability'* is set to *'true'* at the time of model fitting [108]. The problem resides in the fact that SVMs do not directly provide probability estimates [109], which the ETF architecture requires. As a result, the SVM

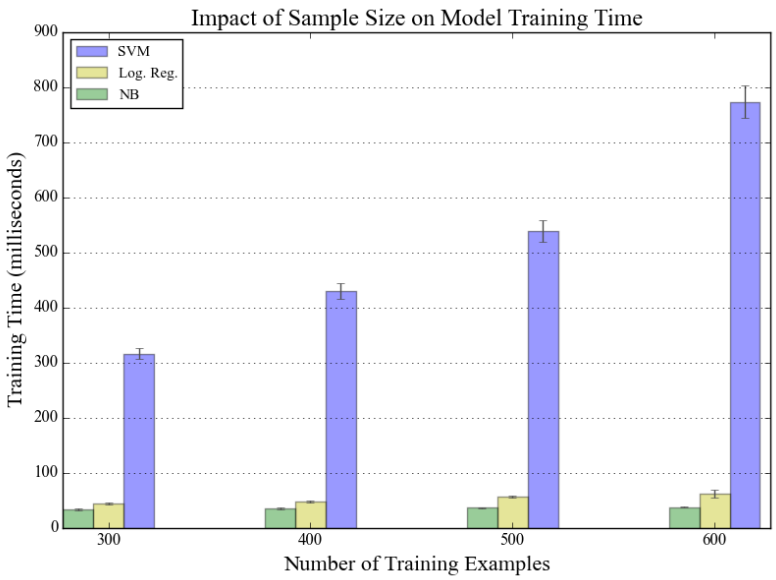

Figure 30. Comparing the time to train each classifier algorithm for various training set sizes

algorithm must calculate the model for these posterior probabilities using an expensive five-fold cross-validation process [108]. In contrast, the other two algorithms were significantly faster at training, and in addition, these algorithms were less sensitive than the SVM to changes in the training set size.

The other computational factor evaluated was the prediction speed of each classifier. Again, the robot's processor hardware was used to perform the experiments. All three classifier models were previously trained prior to the timing of each prediction, and training was conducted using 300 samples with all four features. After training the models, a total of 100 independent predictions were generated and timed. The mean prediction times from these trials are displayed in Figure 31. It shows that the Naïve Bayes (NB) algorithm took the most time, on average, to generate predictions. The observation

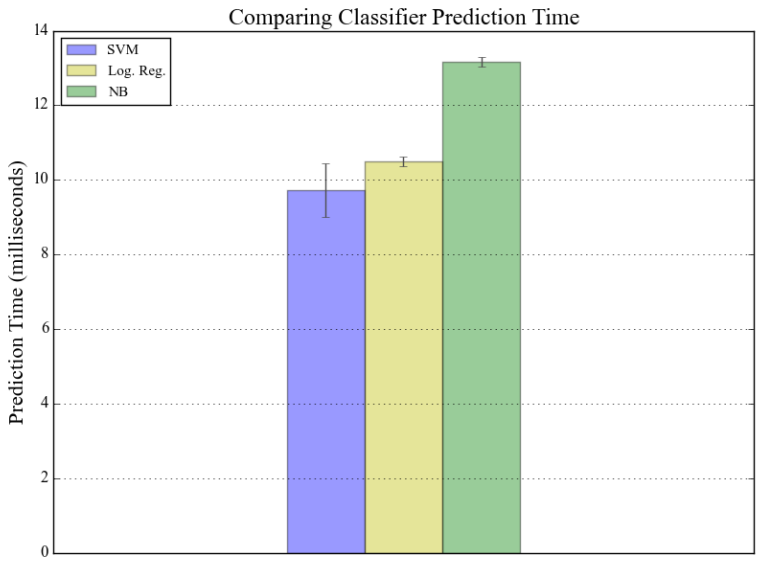

Figure 31. Mean prediction time of each classifier type after 100 trials on target platform

likely results from the algorithm's need to count events and calculate the probabilities associated with Bayes theorem, and the fact that not all of these probabilities can be calculated during model training. On the other hand, the other two algorithms can do much of their model preparation work needed to separate the classes during training, and thus, these models exhibited faster prediction times.

Given the previous experiments, the author felt that logistic regression would serve as the best classifier algorithm for intended application. Its classification accuracy was comparable to that of SVM. As for its computational speed, logistic regression was only slightly slower than the SVM by a millisecond or so. However, the training time of logistic regression proved to be significantly less than the SVM by hundreds of milliseconds. It is important to note that periodic re-training will be essential to combat concept drift, and therefore, retraining speed is a critical aspect given the streaming nature of the application, as well as the limited speed of the robot's hardware.

Feature Selection

The set of LQ features captured during the dataset experiments included the radio hardware metrics of RSSI and SQ, the probing-based statistic known as ETX, as well as the transmission distance between the robot and OCU. Each of these features were sampled shortly before the robot transmitted each picture image to the OCU, while the target variable of TPR was recorded by the OCU once the transmission completed.

Even though the feature set is relatively small, feature selection can still be useful, especially for streaming and resource-constrained applications such as ours. The primary function of feature selection is to identify the most informative features so that the total number of features collected and modeled could possibly be reduced [41]. By eliminating unproductive features, the dimensionality of the sample space is decreased, and the adverse effects of the so-called "curse of dimensionality" problem can be mitigated [32]. With reduced dimensionality, fewer samples are required to generalize an accurate predictive model [36], and this would be desirable when collecting samples is relatively expensive, or when prediction speed is critical. In robot networks, both of these factors are important considerations given the limited resources of most robots (e.g., battery supply, processor speed, etc.), in addition to the requirement of generating predictions quickly in a streaming fashion.

To perform feature selection, the well-known process of recursive feature elimination (RFE) was utilized [41]. The results from stratified 10-fold CV testing were used by the RFE algorithm to rank and recursively eliminate the weakest feature after each iteration. The RFE process was repeated a total of 36 times given that two algorithms (i.e., SVM and Log. Reg.) were tested on all three classifiers (i.e., top, middle, and bottom) using

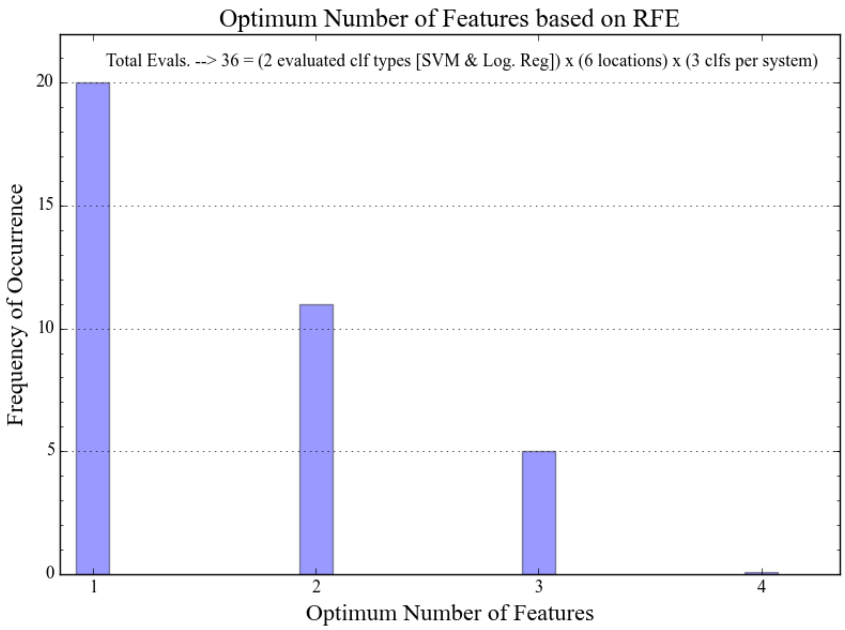

Figure 32. Histogram showing the optimum number of features based on 36 trials of RFE

the six datasets. For each trial, RFE returned the optimum number of features that should be used to achieve the highest possible accuracy. The summarized results from these trials are shown in Figure 32. It is evident from the figure that a single feature proved to be the optimal option for the majority of the time (i.e. on 20 occasions). However, the figure does not distinguish which of the features contributed to these 20 instances. Furthermore, the other trials (i.e., over 44%) demonstrated that at least two or three features should be used for optimal performance, and thus, these results tend to support the use of multiple features in the proposed model.

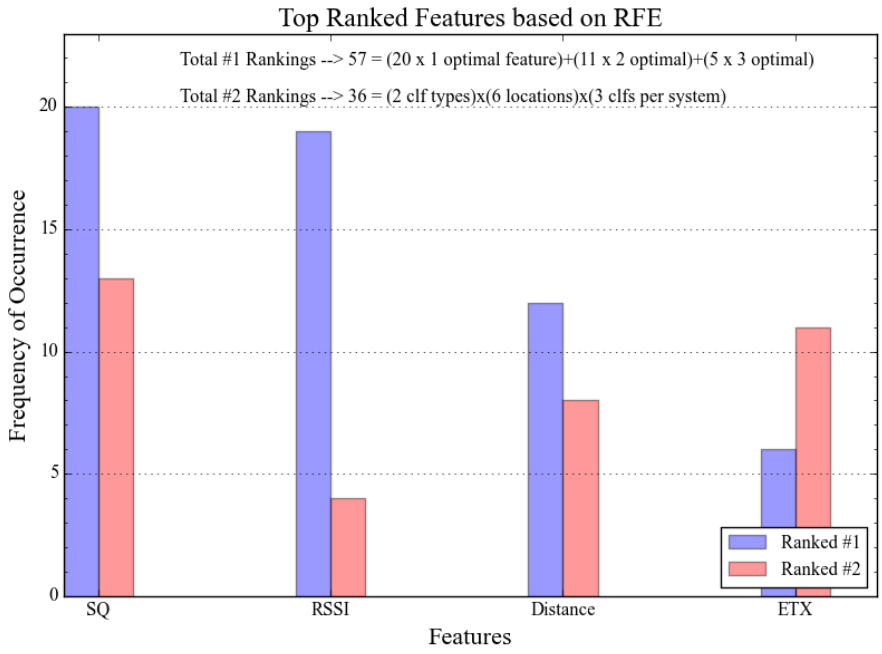

Figure 33: Histogram showing the number of occasions that each feature was ranked either first or second most informative during RFE.

The results from the RFE algorithm can also be presented in another way to better distinguish how informative each feature was to the task of classification. The RFE algorithm ranks the features in order of their predictive power, but it does not differentiate which features are superior when more than one feature proves optimal. Instead, the RFE algorithm lists them with the same top ranking of number one. Given these results, a histogram was plotted in Figure 33 to show the number of times each feature was ranked either first or second most informative. Collectively, the histogram provides a rough ranking order of the features in terms of their predictive power, and in fact, the features are plotted accordingly from left-to-right, starting with the most valuable feature (i.e., SQ) to the least informative (i.e., ETX).

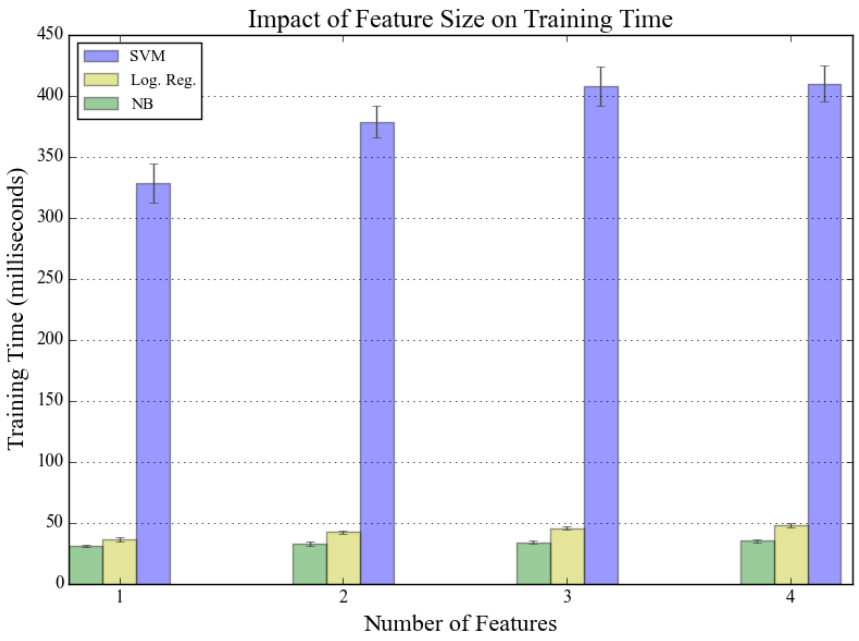

Figure 34. Classifier training time based on the number of features used in the model

Naturally, the next question becomes whether any features should be eliminated. Based on the previous results, the least informative feature is ETX. Another setback for ETX is that it is the most relatively expensive feature to sample because it requires the periodic transmission and reception of probes in order to calculate the statistic. Despite its weaknesses, ETX still managed to rank first or second most informative feature on several occasions as indicated in Figure 33. To further assist with the feature reduction decision, it is possible to investigate whether incorporating four features, instead of three, has a significant impact on model training time. In fact, such an experiment was performed using the robot's hardware, and as Figure 34 indicates, the impact on training time is shown to be marginal, regardless of the algorithm type, when going from three to four features. Given the marginal impact on training time and the occasional utility of all the features, all four were retained in the feature set.

Model Comparison with Generalized Linear Regression

Now that a classifier for the architecture has been selected, as well as the feature set, the *entire* ETF model can be evaluated in terms of its *prediction* accuracy. Each TPR prediction generated by the ETF model is continuous in nature, and therefore, the complete system acts like a regression prediction model. Consequently, the model is evaluated accordingly and its prediction accuracy is compared with the ubiquitous GLR prediction model.

Both models were evaluated using the aforementioned datasets and 10-fold CV. To generate well-balanced training and test sets, the samples were shuffled prior to folding. For each individual test case, both models generated an LQ prediction in a pairwise fashion: one using ETF and another using GLR. After each prediction, mean absolute error (MAE) was used to quantify the accuracy of each prediction [41]. As a result of the CV testing, a list of MAEs was generated for each algorithm and dataset.

The next step of the evaluation was to compare the MAE statistics generated by each algorithm and determine whether there existed any statistical difference between the models. Initially, box plots were used, as shown in Figure 35, to visually inspect some key statistical attributes about the errors generated by each algorithm. However, it is difficult to discern whether any significant difference exists between the models using the box plots. Therefore, paired t-tests were used to continue to look for statistical difference. Because the MAE observations were made in pairs (i.e, one for each algorithm), it was possible to apply the paired t-test in order to determine whether the mean difference (μ_d) between the models was statistically significant. The results of the paired t-tests on each dataset are displayed in Table 7. As the table indicates, the null hypothesis (i.e., $\mu_d = 0$) was rejected

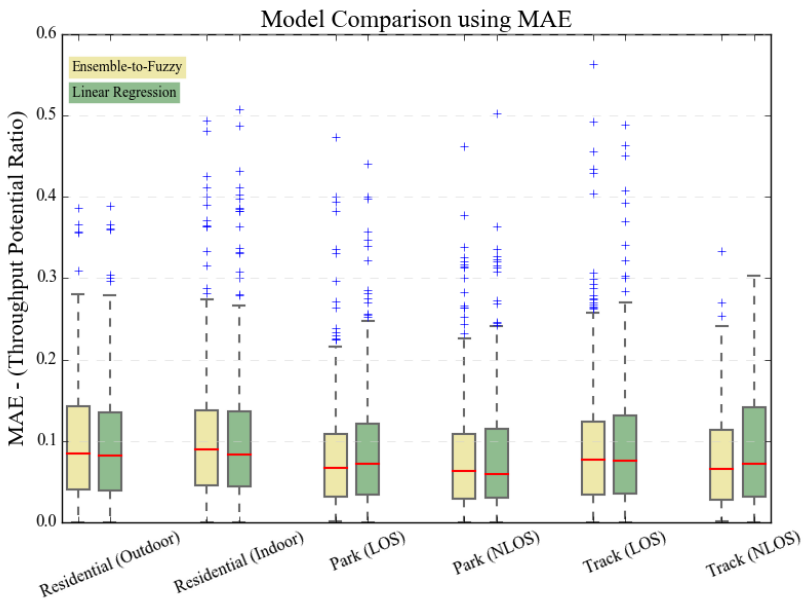

Figure 35. Comparing accuracy of complete ETF system with GLR using box plots

with 5% significance for each dataset. Therefore, the ETF model is shown to perform as equally-well as GLR.

Despite any significance difference in accuracy when compared to GLR, the ETF model offers other potential contributions. For instance, the ETF model arguably offers a more intuitive and flexible design than many other algorithms including GLR. Due to its fuzzy architecture, the model facilitates the interjection of domain knowledge, and the added flexibility increases its potential to exceed existing prediction models with tuning. As a tuning example, the proposed ETF model could be modified to include another fuzzy set, as well as a modified rule base, with the intent of narrowing the mapping relationship between the features and TPR. As another consideration, all learning models tend to perform differently on separate problems [32, 41], and because the proposed ETF model can be extended to other domains, its architecture may be well-suited for other types of problems outside of LQ prediction.

Table 7

Results of Paired t-tests Comparing Model Accuracy Differences

	Residential (Outdoor)	Residential (Indoor)	Park (LOS)	Park (NLOS)	Track (LOS)	Track (NLOS)
Null hypothesis (μ_d = 0 to 5% significance)	Not Rejected	Not Rejected	Not Rejected	Not Rejected	Not Rejected	Not Rejected
p-value	0.5309	0.8412	0.1439	0.9684	0.6303	0.1137

Conclusion

In this chapter, a novel way of making fuzzy systems adaptable to dynamic changes was introduced. It is based on the assumption that often times the rule-bases of fuzzy systems do not need tweaking, but instead, the mapping of the inputs into linguistic categories needs adjusting over time. Given that assumption, the unique method focuses on incorporating adaptability into the fuzzification step of fuzzy systems. Specifically, a classifier ensemble was used to perform the mapping of the inputs into linguistic (i.e., fuzzy) sets, and through the periodic retraining of the classifiers, the mapping process can adapt to drifting.

The proposed ensemble-to-fuzzy (ETF) model was shown how to be incorporated into both Mamdani and T-S fuzzy architectures. Afterwards, the T-S version of the model was evaluated offline using a series of empirical datasets that were collected by a real robot in diverse environments. The offline evaluation included classifier and feature selection. In addition, the entire ETF model was compared to GLR. The CV testing of both prediction methods revealed similar accuracy performance. However, as previously

discussed, a distinguishing feature between the methodologies is that the ETF model offers several other advantages given its overarching fuzzy architecture.

The next evolution after completing an offline evaluation of the ETF model is to test it in an *online* fashion. However, a preliminary step toward that objective is to establish a framework for sampling and retraining. In the next chapter, several novelties for such a framework are introduced, and afterwards, they are evaluated along with the online-version of the ETF model.

CHAPTER VI

AN ACTIVE AND INCREMENTAL LEARNING FRAMEWORK FOR STREAMING LINK QUALITY PREDICTIONS

Introduction

In the previous chapter, the ensemble-to-fuzzy (ETF) model was evaluated in an *offline* manner using nearly complete datasets. The intent of this chapter is to implement the model in an *online* setting, where LQ predictions are made in real-time. However, the paradigm shift to online predictions presents several new challenges. In robot networks, LQ samples are generated in a streaming fashion without any guaranteed uniform data distribution as a result of robot mobility, and the training batches may be incomplete, especially during the early sample collection phase. Even after initially discovering the sample space, the relationship between the features and target variable can suddenly change over time. Hence, robots must occasionally gather new examples and incrementally update their prediction models. Unfortunately, robotic systems typically do not have sufficient resources to label, store, and process every sample in the data stream, and therefore, they must perform some sort of selective sampling procedure. These particular challenges of efficiently handling the streaming collection of samples in robot networks has been largely overlooked by the related work in Chapter III. To bridge this gap, this work introduces an active and incremental learning framework that can guide robots in intelligently labeling samples with minimal overhead, while also incrementally learning to guard against concept drift and to maintain accuracy.

Motivation for Incremental Learning and Selective Sampling

LQ data streams in robot networks are dynamic due to a number of possible changes that can occur in the wireless channel or during robot positioning. In general, there are different types of change that can occur in a data stream, and all of them are commonly referred to as *concept drift* [104, 110, 111]. One type of concept drift relates to when the data distribution from which the features are drawn, $p(\mathbf{x})$, changes over time. In the case of robot networks, this form of drift is a frequent occurrence due to robot mobility. However, changes in $p(\mathbf{x})$ does not necessarily imply that the underlying *concept* between the features and target variable, $p(y|\mathbf{x})$, has drifted [104, 110, 111]. In the case of LQ, this type of concept drift occurs when large-scale characteristics related to the wireless channel, such as the shadowing component, suddenly changes. As a result of concept drift, it essential for an online prediction model to have an *adaptable* learning mechanism that can maintain prediction accuracy over time.

A primary challenge with online and streaming applications is that samples arrive continuously one-after-another, and after some time, they can add up to an enormous amount of data [43, 44, 104]. Therefore, conventional batch learners that attempt to store and process every sample are impractical, and often, infeasible [104]. This is especially true in the case of robots, which tend to have limited resources. Consequently, for the intended robot application, data must be processed either one sample at a time or in manageable batches. This process of handling only a portion of the data stream implies a forgetting mechanism, which is critically important due to the system's inability to remember every sample it encounters.

Another challenge that complicates the process of maintaining an online model for LQ prediction is the expense associated with obtaining labels for samples. Given the nature of the environment, a robot must query its receiver in order to obtain a label, and the resulting transmissions across the wireless channel consume energy and network capacity. To limit these expenses, a sampling strategy known as *active learning* can be used, where only labels essential to building an accurate prediction model are queried [44, 111].

There are primarily two types of *incremental algorithms* for handling examples in an online manner [104]. One type is commonly classified as *online algorithms*, and these kinds of learners only handle a single sample at a time. In other words, learning takes place in an error-driven fashion, where the model is tuned after every sample based on the observed prediction error, and afterwards, the sample is discarded. The learning style assumes that labels are received frequently in order to calculate the error difference between the predicted label and its actual feedback value. As previously stated, labels are relatively expensive in robot networks, and thus, such an approach could be costly given the consistent feedback these adaptation algorithms demand. Another drawback to these types of online learners is that they are known to have slow adaptation to sudden change [104], and the precise adaptation rate is a sensitivity tradeoff that is set by variables within the algorithm. Thus, it is possible for online algorithms to be heavily influenced by noise and outliers, which is a common occurrence in wireless propagation. Consequently, single-instance learners could be problematic in a link quality application setting.

Another online and incremental algorithm option is to update the prediction model using multiple examples at a time in batch-to-batch fashion. In contrast to single-instance learners, it is possible for these multiple-instance algorithms to have memory if a portion

of each new batch consists of examples from the previous batch. This style of learning is sometimes referred to as *incremental learning with partial instance memory* [112]. Due to its memory capabilities, the author feel this style of batch incremental learning is best-suited for the intended application given the objective to conserve labeling costs. Intuitively, the online learning approach that disposes of each sample, after the model is tuned with it, seems wasteful given the resources devoted to obtaining the label. On the other hand, a batch learner can attempt to conserve resources and the knowledge gained by the labeled sample if it is retained along with other relevant samples for a longer period of time. In that case, future labeling expenses would be reduced by leveraging the historical knowledge already obtained. However, retaining old samples can have an adverse effect on predictive accuracy if they no longer represent the actual model due to concept drift [113]. Therefore, it is essential for incremental batch learners to incorporate some form of forgetting mechanism given the dynamic nature of LQ.

Background

The Challenges behind Making Labeling Decisions

Often times, the process of selecting which samples to label in a data stream is difficult because decisions must be made rapidly and without any foresight into the future availability of samples [43]. Both of these factors hold true in robot networks. In terms of the time constraint, the streaming environment dictates that the robot must make a labeling decision shortly after it samples its features; this allows for any label request to be embedded within the next impending transmission to the receiver. In terms of future sample availability, it is assumed that the robot does not perform any artificial movement to collect desired samples, and thus, future available cannot be guaranteed.

Some Related Approaches

There have been several heuristics proposed in the literature that use various decision criteria to determine whether to request a label. Some examples of these heuristics include uncertainty sampling, maximum disagreement, model variance minimization, model/space pruning, and estimated error reduction [43]. Many of these schemes are related to the concept of requesting labels for the most ambiguous cases (i.e. when the posterior probability is low or uncertain) [46]. The problem with this approach is that the labeling effort tends to be concentrated around the decision boundary, and hence, the labeled training data evolves into a distribution that is much different from the original data distribution [43, 44, 111]. In addition to biasing the data distribution, the other problem is that the system may fail to detect concept drift in areas outside the boundary region, and as a result, may stop learning [111].

Some countermeasures have been proposed to mitigate the aforementioned issues associated with sampling solely based on ambiguity. For instance, probabilistic schemes place importance weights to labels in an effort to remove sampling bias [43]. Another approach incorporates randomization in the labeling process by either splitting the data into two streams (i.e., one random and another selective) or by adding a random component into the selection decision [111]. Because active learning schemes tend to selectively label points based on their entropy, preventing bias in the sampling process remains one of the most fundamental challenges posed to active learners [46].

The Active Learning Elements

Queues for Preventing Bias and Oversampling

For the intended robot application, a new selective sampling scheme is introduced that avoids the uncertainties and complexities surrounding the existing methods. The novel framework uses a set of queues, as depicted in Figure 36, to guide the selective sampling process. It is based on placing a series of equally-sized bins or queues across the range of a selected feature. This attribute helps to ensure that samples are labeled in an independent and identically distributed (IID) fashion, which is a common stipulation among supervised learning algorithms [57]. The selectivity aspect of the framework comes from the fact that each queue has a maximum number of samples it can store. The storage threshold prevents oversampling (i.e., over-labeling) in a particular region of the sample space, which could bias the prediction model and waste labeling resources. Another advantage of the queue-based model is that it facilitates a simple and expedited decision on whether a label is needed. If the queue corresponding to a particular sample is not yet full, then a new label is requested in order to gather sufficient samples necessary to generalize an accurate prediction model. On the other hand, if the queue is already full, the decision of whether to request a label requires slightly more consideration. It is possible that the relationship between the features and target variable has drifted since the time the samples were originally collected. In that case, the oldest sample in the queue should be replaced by flushing it out of the queue in a first-in, first-out (FIFO) fashion. This replacement or forgetting process is an important adaptation function, but in order to conserve labeling resources, special care should be taken to ensure that the disposal of samples is truly necessary [113].

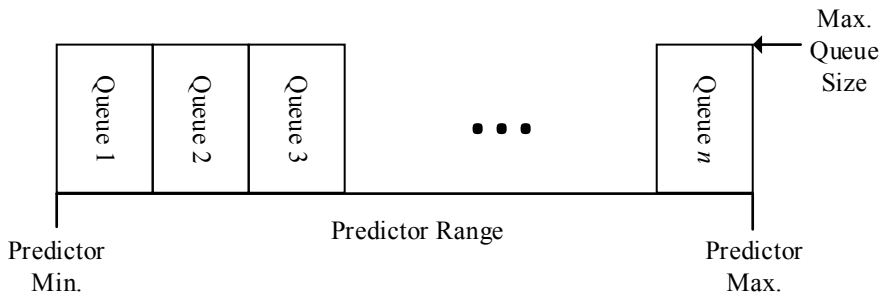

Figure 36. Queue-based structure used to guide the selective sampling scheme

Slow Concept Drift Mitigation

In many cases, data streams have some form of temporal significance and samples should be forgotten as time expires [114]. Otherwise, maintaining inaccurate training examples could negatively impact prediction accuracy [113]. In application of LQ prediction, concept drift is a real possibility, and therefore, older samples may distract from generalizing an accurate mapping function if the underlying function has drifted since the older samples were originally collected. However, precisely detecting concept drift in this LQ application setting is challenging, and there is no accepted means for doing so. As a result, a failsafe is embedded within the sampling framework to ensure the queues are occasionally refreshed with new samples based on time. The failsafe procedure ensures that the system continues to learn, regardless of the success of any additional change mechanisms.

The time-based replacement scheme works as follows. Once a new sample enters a queue, the current time is also recorded. Later, if a queue is full and another potential replacement sample becomes available, then the elapsed time of the oldest sample in the queue is calculated. If the transpired time exceeds a predetermined threshold, then a label

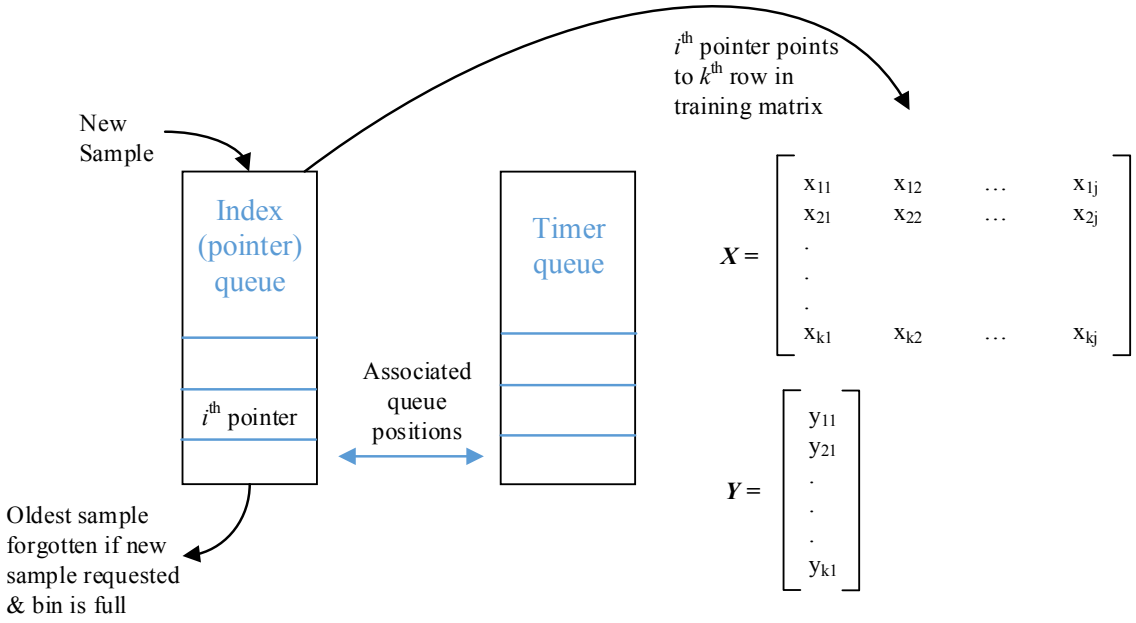

Figure 37. Illustration of data structures used to implement the selective sampling framework.

is requested. Otherwise, no label is requested in order to conserve resources, and thus, the oldest sample remains in the queue.

The specifics behind the various data structures used to implement the sampling model and the time-based replacement mechanism are illustrated in Figure 37. It shows that two interrelated FIFO queues are employed to record the pointers and times associated with each labeled sample in the training matrix. The stored index pointers enable the forgetting mechanism by identifying which row in the training matrix should be deleted once an expired sample is replaced.

Fast Concept Drift Mitigation

The above time-based approach is referred to as *slow-concept drift mitigation* because it may fail to rapidly adapt the model to sudden changes in the relation between the features and the target variable. To ameliorate this potential problem, a change detection mechanism is incorporated within the design so that the model more quickly adjusts to sudden concept drift. The *fast concept drift mitigation* scheme is based on passively monitoring the unlabeled samples (i.e., the predictors) and looking for sudden drift in $p(\mathbf{x})$. Once an abrupt change is detected, the scheme triggers a temporary increase in labeling, and if any of the associated queues are full, the samples in them are replaced regardless of whether they are expired. The concept is based on the assumption that a sudden shift in $p(\mathbf{x})$ *likely* indicates a drift in $p(y|\mathbf{x})$. However, detecting concept drift (i.e., changes in $p(y|\mathbf{x})$) requires true labels [111], and thus, the selected approach to monitor unlabeled samples may inevitably trigger some occasional false positives and unnecessary spurts in labeling.

The Incremental Batch Retraining Elements

Batch Size Selection

As previously alluded, the proposed framework is based on an incremental learning algorithm that retrains its prediction model in batch format. Each subsequent batch consists of *old* and *new* samples with respect to the previous batch. As a result, the system intentionally contains memory from the old samples so that future labeling costs are reduced. The memory in the system also allows the robot to rapidly move through the sample space and still maintain its prediction accuracy, assuming the space has already been sampled and no relation drift (i.e., $p[y|\mathbf{x}]$) has occurred.

The size of each training batch largely depends upon the specifics of the aforementioned queue-based structure. Specifically, the number of queues spread across the predictor space, as well as the queue size, determines the maximum number of samples that will be included in each batch. Therefore, special attention should be given to configuring the queues due to the impact the structure has on the batch size, which in turn impacts the accuracy and costs of the prediction model. The costs impacted by batch size not only includes the labeling expenses, but also the costs of maintaining and building a prediction model. The real-time nature of robot networks demands that the prediction model be relatively lightweight and fast. Therefore, an effort should be taken to minimize the batch size due to its impact on the speed at which a prediction model is generalized, but the batch size should only be reduced to a point that it does not adversely affect accuracy.

Retraining Frequency

The system must be occasionally retrained in batch format to remain adaptable to dynamics related to concept drift. Naturally, the retraining frequency is a function of the number of new labels, x, requested since the last training batch. However, there is a tradeoff associated with identifying exactly how often the model should retrained. For example, a small value for x supports quick adaptation, but at the same time, it places more load on the robot's processor for potentially minor model refinements. On the other hand, a large value for x reduces the computational load, but meanwhile, makes the model slower to adapt.

Expediting Model Completeness using Synthetic Samples

There are some challenges associated with building an accurate prediction model shortly after link initiation. For instance, once a robot establishes a new wireless connection, it does not have the benefit of immediately possessing a pool of samples to fit an accurate predictive model. Instead, labeled samples must be collected over time in a streaming fashion as they become available to the robot during normal operation. Therefore, in these early stages of sample collection, the model may be incomplete or inaccurate due to insufficient samples. In robot networks, this problem would certainly surface whenever the robot enters new regions of the sample space for the first time without any prior knowledge or examples built into its model.

Other authors have attempted to solve this problem in a couple of ways. One approach involved robots generating artificial and random movement so that diverse samples could be collected before making predictions [28, 29]. However, the approach is time-consuming and expensive in terms of energy expenditure. Other approaches had nodes exchange training examples that were collected across disparate links in order to expedite the collection of samples [24, 30], but such an approach introduces inaccuracies into the generalized model due to each wireless link likely having different characteristics (e.g., hardware, shadowing, etc.).

A novel alternative is introduced in this work where synthetic samples are generated so that a predictive model can be utilized shortly after link inception. The concept of using artificial samples has been leveraged in other domains [32], but not in the field of LQ prediction to the best of the author's knowledge. In the proposed design, the artificial samples serve as temporary placeholders in the batch framework until real samples are

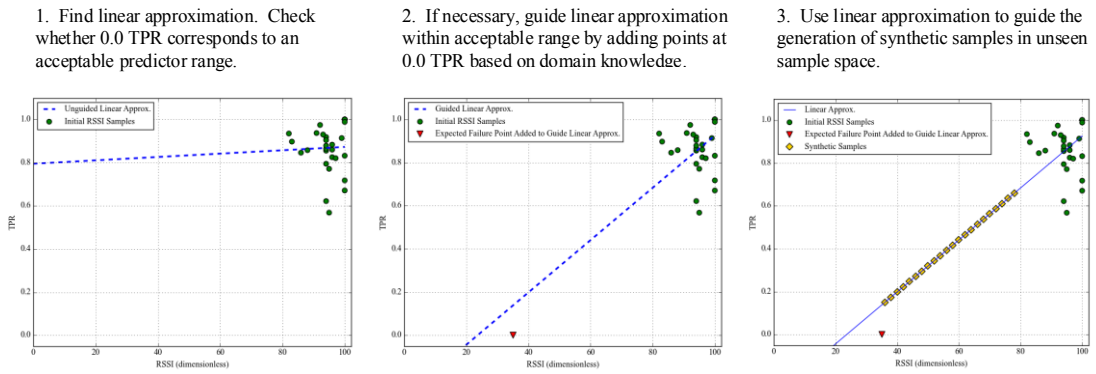

Figure 38. Fundamental steps in generating synthetic samples

collected as replacements. Not only do the synthetics facilitate more accurate predictions as the robot enters new regions of the sample space, but they also facilitate the early training of the diverse classifiers within the ETF architecture. Without synthetics, it would take more time, depending upon link conditions and robot movement, before sufficiently diverse samples (i.e., class labels) would be collected to train all three classifiers (i.e., top, middle, and bottom).

Domain knowledge is leveraged in process of generating the synthetic samples. The purpose of domain knowledge is to ensure the artificial samples are close approximations to the real samples that will eventually take their place, assuming the robot eventually explores that portion of the sample space. The entire process can be summarized as a series of steps as outlined in Figure 38. As indicated in the figure, a particular feature is used to initiate the process. In this work, received signal strength indicator (RSSI) was selected for this purpose. Assuming an initial batch of RSSI samples have been labeled, as depicted in the left-most subfigure, then then linear regression can be used to predict future values of RSSI that have not yet been seen. However, as illustrated in the far-left image, the linear approximation could be significantly inaccurate. To mitigate this issue,

domain knowledge can be interjected; for instance, based on the previous offline experiments discussed in the previous chapter, a RSSI level of roughly 35 (dimensionless units) corresponds to zero TPR given the employed Wi-Fi transceiver. Using this knowledge, the linear approximation can be guided to intercept the x-axis (i.e., the zero TPR level) within an acceptable range of RSSI (e.g., 0 to 50 based on domain knowledge). As shown in the middle image of Figure 38, only a single guiding point (i.e., RSSI = 35, TPR = 0) was sufficient to force the regression line within an acceptable window. Finally, as indicated in the far-right subfigure, synthetic samples can be added to areas of the sample space that currently do not have any examples. Once these synthetic RSSI and TPR pairs are known, the TPR values of the pairs are used to find the other predictor values that makeup the feature vector within each artificial sample. In essence, the process of using linear regression and domain knowledge is repeated for each of the predictors so that complete input-output pairs are formed.

The process of generating artificial examples is repeated immediately before each batch training cycle. During this time, any previous synthetics are deleted and new ones are refitted based on any new real samples collected since the last training event. This ensures that the synthetics are placed as accurately as possible within the model given the existing pattern of behavior among the real examples. The precise number of artificial samples generated depends upon how much of the sample space has been explored by the robot, and only those unexplored areas are filled with a few synthetics. To make this determination, the robot scans the queues placed evenly across a selected predictor's range, such as RSSI, and any queue found empty is designated for a number of artificial samples, depending upon the width of each queue.

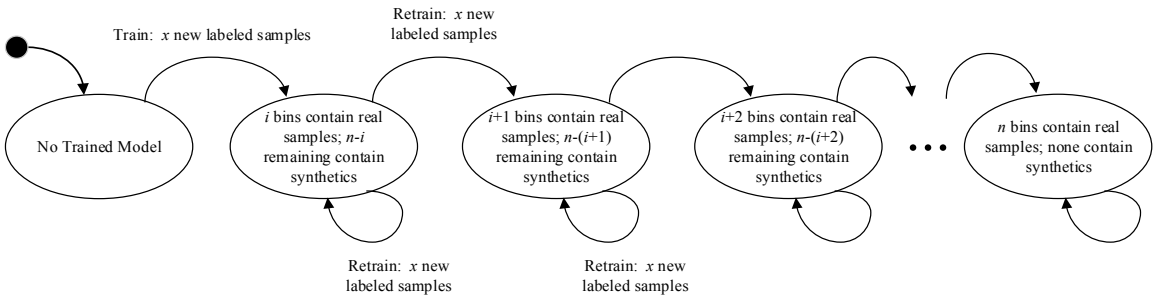

Figure 39. State machine description of incremental learning algorithm

The state of the system in terms of the number of real and artificial samples that exist in the model at any given point is best described as a state machine process as depicted in Figure 39. In other words, for any new batch, the state of the system may change in terms of the number of queues (or bins) that have real samples and those that have synthetics. These state variables, as labeled in Figure 39, depends upon the amount of time the link has been in existence, as well as the amount of movement the robot has performed across the sample space. The state machine also conveys the batch-to-batch nature of how the framework works, and how the state can transform after collecting *x* new samples between batches.

Evaluation

Parameter Selections

Several parameters were previously discussed while detailing the active and incremental learning framework. These variables required assignments in order to implement and evaluate the framework, and the author's selections for this purpose are summarized in Table 8. As the table indicates, RSSI was selected as the feature for guiding the decision of whether to request a label based on the queuing structure previously discussed. By limiting the queues to a single feature, the process was simplified and

Table 8

Parameter Selections used during Evaluation

Parameter Description	Selection
Feature for queue-based selective labeling	RSSI
Number of queues across predictor range	20
Individual queue storage size	20
Timer expiration for slow concept drift mitigation	5 minutes
Predictors used in change detection mechanism	RSSI and SQ
Window size, *w*, used in change detection	10
Number of subsequent labels requested if change detection mechanism triggered	30
Number of new labels, *x*, that triggered retraining	30

eliminated the need to manage multiple queues for every predictor. The approach was based on the assumption that the features are *dependent* in the sense that as one varies, the others also likely vary. Based on this assumption, diverse samples from *all* of the predictors would be gathered as RSSI varies across its range. RSSI was the best-fit feature for this purpose given that fact that it has a fixed range and generally varies across its range. In addition, as discussed in the last chapter, the feature was shown to be the second most informative and only slightly behind SQ; however, SQ is less fit for the queuing model because it only tends to assume values near the top quarter of its range. The range of RSSI, which varies from 0 to 100 on a dimensionless scale, was divided into 20 queues, each 5 points apart. The labeling budget for each queue (i.e., max size) was set to 20 samples. Therefore, each training batch would contain a maximum of 400 examples, assuming each queue is full. That maximum batch size was evaluated in the previous chapter and found to be processed in an acceptable amount of time.

The change detection mechanism used to mitigate fast concept drift involved monitoring the *unlabeled* behavior of RSSI and SQ. Specifically, the mechanism looked

for sudden and significant change by comparing the predictor's current standard deviation, σ_i, over the past *w* samples to its *average* deviation witnessed over the same-sized window, but delayed in time by one sample. The average deviation calculation can be generalized as

$$\sigma_{avg} = \frac{1}{w} \sum_{j=i-w}^{w} \sigma_j$$

where *i* is the index associated with the present feature sample and σ_{avg} is the mean standard deviation calculated over a delayed window of past standard deviations, not including the present σ_i. The change detection mechanisms for RSSI and SQ were configured to trigger a temporary increase in sampling if σ_i was found to be greater than three times (3x) σ_{avg}. If the condition was met for either RSSI or SQ, then labels were requested for the next 30 samples regardless of whether any were expired in the queues. The other selections are self-explanatory given the discussion in the previous sections.

Experimental Overview

A series of experiments were conducted to evaluate the online prediction accuracy of the ETF model, as well as the effectiveness of the active and incremental learning framework. The same robot and operator control unit (OCU) introduced in Chapter V were used during the evaluations. In addition, the locations also remained the same, except for a new testing location inside of an academic building at the University of Louisville. Finally, the experiments were also conducted in a similar fashion. Generally speaking, the robot would start near the OCU and then proceeded to travel farther away to a point near its transmission limit, and in some cases, it returned to the OCU and then repeat a similar travel path. The testing scenario was selected not only for the variety of predictors it would

introduce, but also for its realism. In several applications, robots are often tasked to travel relatively far away from the OCU in order to get a closer visual inspection of some area of interest that would otherwise be dangerous for humans to do so [12, 13]. The other realistic aspect of the testing scenario is the fact that no artificial movements were performed to collect diverse training samples.

In contrast to the *offline* testing performed in the last chapter, the robot made *online* predictions in these experiments. In other words, the robot sampled the features and made its predictions shortly before transmitting each application payload to the OCU. Each transmission was sent via Transmission Control Protocol (TCP) and consisted of a picture image along with some other sensor data. The periodic rate of sampling, predicting, and transmitting was approximately every 1.5 seconds given the imaging speed of the robot's camera.

Online Prediction Comparison

Before each transmission, the robot made three separate predictions using different models. GLR was one of the prediction models used, while the other two were different versions of the ETF model. Similar to the last chapter, GLR was included in order to baseline the accuracy of the ETF model to the most widely used prediction method. On the other hand, the two ETF versions differed in order to evaluate the impact of incorporating synthetics into the training batches; thus, one of the models included synthetics, while the other did not.

Figure 40 shows the accuracy comparison of all three models based on testing performed at four separate locations. Accuracy is plotted in terms of a moving average of mean absolute error (MAE) over the past 150 predictions. Based on the four subplots,

there are some general trends worth noting. First, the plots show that adding synthetics enabled one version of the ETF model to start making predictions right after the first 30 samples were labeled; on the other hand, the ETF version that omitted synthetics had to withhold making predictions until later due to an initial lack of diverse examples. Hence, it is evident that synthetic samples are critical to the rapid startup of the ETF prediction model. Secondly, it is also clear that synthetics play an important role in the accuracy of the ETF model during its early stages. In each subfigure, the ETF model with synthetics is shown to initially have a lower average MAE, but eventually, both ETF versions tend to converge as more samples are stored. Lastly, the ETF model is shown to initially outperform GLR for a period of time, but then the error difference tends to shrink as time progresses. In other words, it appears that the ETF model performs better than GLR when the sample space is only partially explored, but as more samples are added, the difference tends to be less significant. The window of significant difference between ETF and GLR was smallest for the experiment conducted in the academic building. This observation likely resulted from the robot being restricted in movement to a single corridor during testing, and thus, the models converged much more quickly due to the smaller sample space.

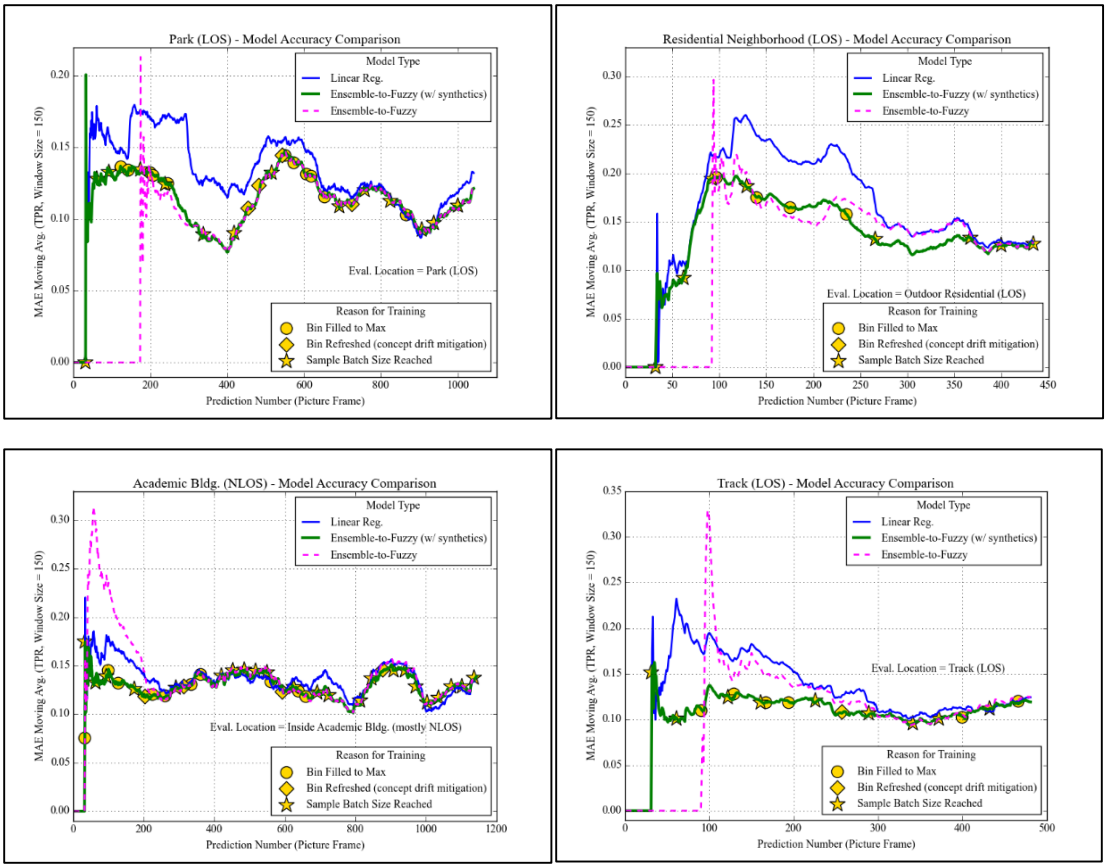

Figure 40. Comparing online prediction accuracy at various locations

The boxplots of Figure 41 were generated in order to determine whether the three models are statistically different. Each box corresponds to a particular model and its associated MAE from every prediction shown in the four experiments of Figure 40. The figure shows that the quartile edges and the maximum bars of the ETF boxes are slightly lower than the GLR box. In addition, the ETF models have fewer extreme outliers than GLR. Finally, the root mean squared error (RMSE) was also calculated based on the aggregated MAEs corresponding to each box, and as the figure indicates, the ETF model has a lower RMSE than GLR.

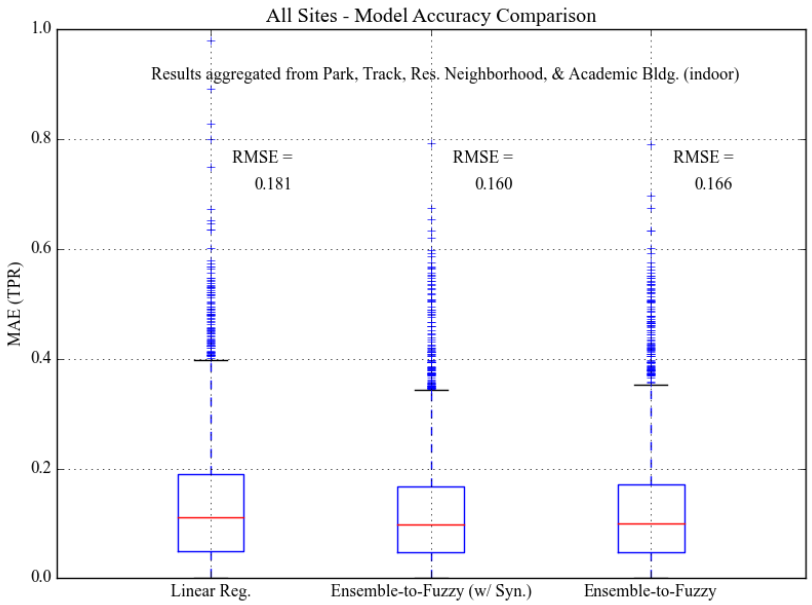

Figure 41. Boxplots comparing predictive models using aggregated MAE results.

In order to solidify whether any significant difference exists, paired t-tests were performed on the predictions made by the ETF model with synthetics and GLR. The results of the paired t-tests are displayed in Table 9. The table concludes that there was a small mean difference between the ETF and GLR models in terms of their prediction error at each location. The amount of difference varied between a fraction of a percent and up to roughly 4%. The improvement margin of ETF over GLR is relative to the difficulty of the task. The GLR algorithm was shown to have a median accuracy of roughly 90% according to the boxplots of Figure 41, and 90% accuracy is fairly common among prediction algorithms in a variety of problems [32]. Therefore, the slight margin of improvement shown in Table 9 is put somewhat into perspective knowing that learning tends to saturate and exceeding 90% accuracy can be difficult at times.

Table 9

Paired t-tests Quantifying Mean Difference between Online Prediction Models

	Location			
	Park	Track	Residential Neighborhood	Academic Building
Null hypothesis ($\mu_d = 0$)	Rejected	Rejected	Rejected	Rejected
p-value	2.26E-13	2.48E-05	0.0014	1.62E-04
5% Confidence Interval	$0.0136 < \mu_d < 0.0233$	$0.0093 < \mu_d < 0.0253$	$0.0095 < \mu_d < 0.0395$	$0.0026 < \mu_d < 0.0083$

Other Prediction Performance Considerations

It is important to compare a model's prediction error to some form of target variability so that a sense of predictive power can be gained [41]. Figure 42 shows the variability of the target by plotting the true value of TPR that was measured during one of the above experiments. The average MAE of the ETF model corresponding to this experiment is also included in order to compare the variance of each. The figure shows that the target varied widely from 0 to 1, yet the prediction error tended to have significantly less variance. The periodic shape of the target is an artifact of the robot's intentional movement during that experiment; specifically, on two repeated occasions, the robot moved away from the OCU for some distance and then later returned near the OCU. The rapid variation between adjacent target samples is likely the result of multipath propagation or other transmission phenomena.

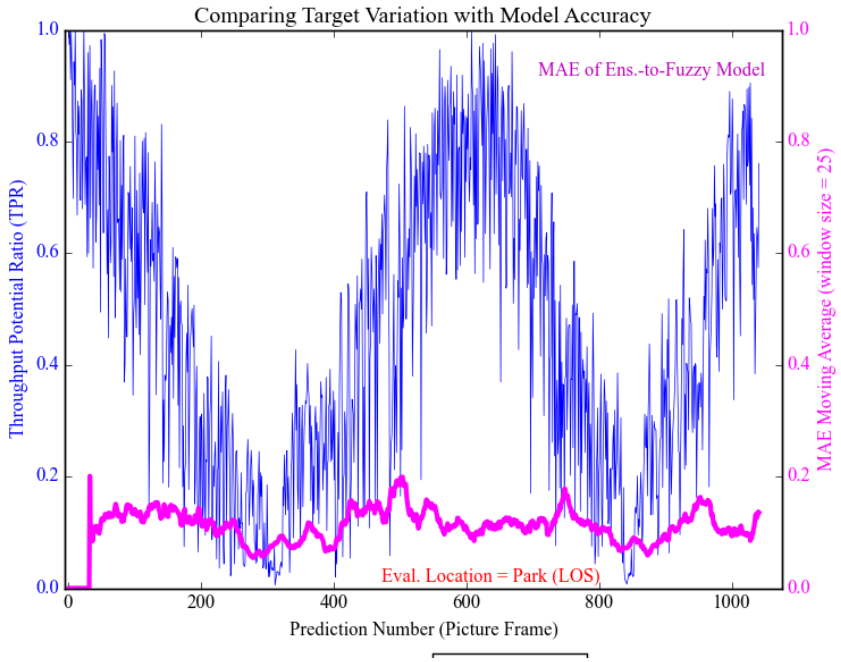

Figure 42. True target compared to average MAE of ETF model

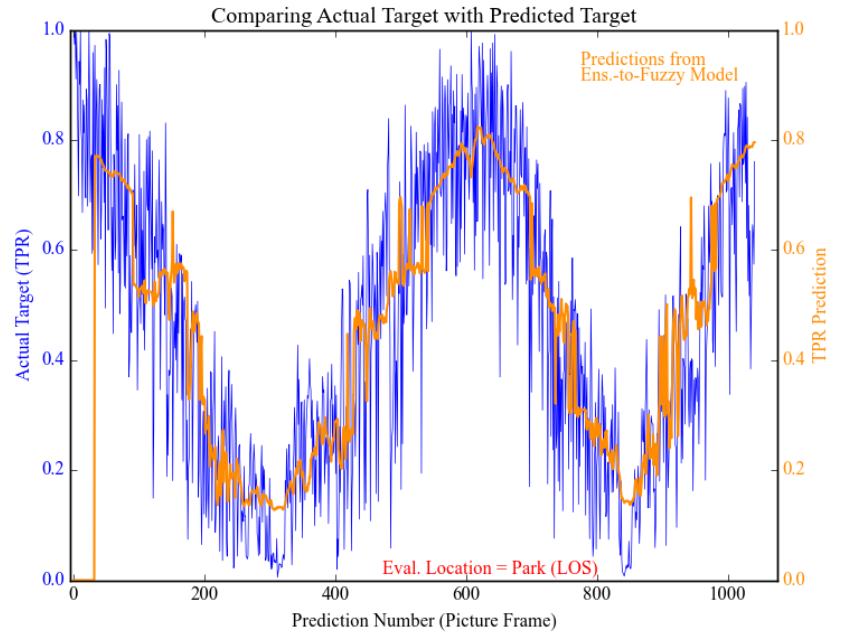

Figure 43. True target compared to the predicted value of the target from the ETF model

Another general sense of model performance can be obtained by plotting the individual predicted values alongside the target values as shown in Figure 43. The figure shows that the predicted values tend to have less variance and generally reside near the mean of the moving target. This result is expected based on the prediction model *generalizing* a best-fit function of the underlying LQ process, given the *available* predictors.

The error in the prediction model is likely the result of several factors. Arguably the most prevalent factor was the fact that the model used samples stored over time, and the age of the samples far exceeded the coherence time of the wireless channel. Thus, the training batches consisted of examples influenced by the small-scale and statistically-random fluctuations induced by the multipath phenomenon. In that case, the best hypothesis function possible was a generalization that abstracted this random fluctuation. This explains the trend of the prediction plot in Figure 43 generally following the mean of the TPR target function.

Other factors could have also contributed to some of the prediction error observed in Figure 43. For instance, important predictors were likely missing in the prediction model that could have better explained the rapid variation between adjacent TPR samples. But unfortunately, the application layer, where the prediction model resides, only has access to a limited number of predictors that offer insight into the conditions of the wireless channel. Another contributing factor to some of the observed error may be related to the fact that many radio vendors tend to smooth the LQ metrics supplied to the operating system using some form of filtering [27]. As a result, the radio metrics may have lost some of their correlation to the target variable.

Effectiveness of Concept Drift Mitigation

The next set of experiments focused on evaluating the effectiveness of the *slow* and *fast* concept drift mitigation strategies previously discussed in this chapter. During all but one of the experiments, some form of disturbance was introduced in order to force a drift in the underlying relation between the predictors to the target variable. The goal was to see whether the selective sampling scheme, as well as the incremental batch learner, could eventually adapt to the change, and if so, how quickly. In order to gauge these attributes, predictions were made by the robot using two models: one *with* the active learning and forgetting mechanisms and another *without* them. In other words, one model would selectively replace samples based on either the embedded *slow* or *fast* detection mechanisms, while the other would simply store samples until the queues filled and would never forget them.

The results from the aforementioned experiments are grouped together in Figure 44. The subfigures are labeled with the location where the testing was performed, as well as an indication to the type of disturbance that may have been introduced (e.g., interference or non-line-of-sight (NLOS) obstruction). Each subfigure contains data that are plotted with respect to two different y-axes: moving average of MAE on the left-side and the predictor range of RSSI and SQ on the right-side. The features of RSSI and SQ were included in the plots in order to note the points when their respective change detection mechanisms triggered an increase in temporary labeling.

One of the subfigures corresponds to testing performed inside a residential home, and the disturbance in this case was sudden interference caused by another pair of transmitting nodes placed on the same frequency channel. The interfering nodes were on

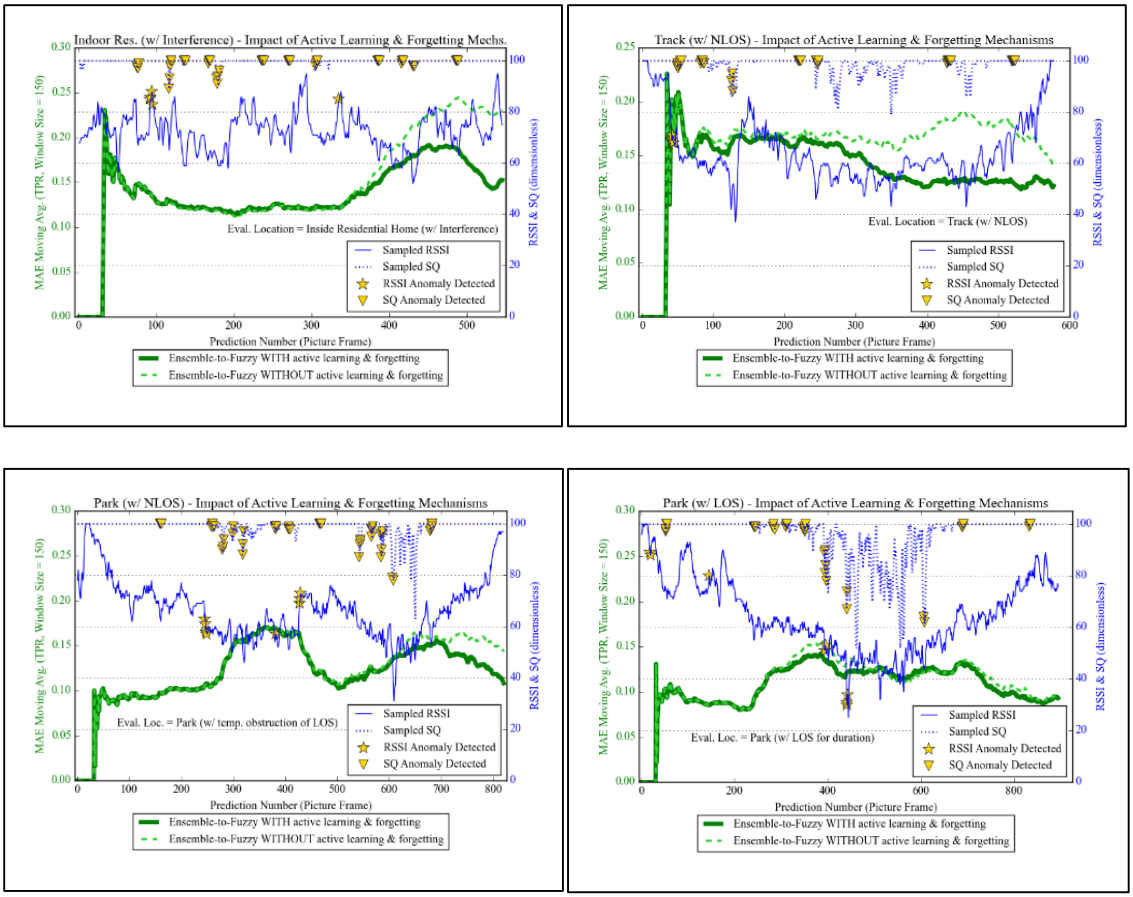

Figure 44. Evaluation of concept drift mitigation strategies under various scenarios

a separate ad hoc network from the robot and OCU, so that they would not directly negotiate sharing the wireless medium. During the disturbance period, one of the interfering nodes transmitted 1200 byte packets approximately every 5 milliseconds (ms) in order to simulate interference. The subfigure shows that, around prediction number 350, the prediction error of the *static* model began to drift more quickly, as evident by its higher MAE. Although the *adaptable* model also drifted some, it eventually reduced its error level as it learned from new examples.

Another experiment was conducted at the track, but this time, the disturbance was in the form of a box, which was wrapped in aluminum foil, that was placed over the robot for a period of time. Shortly after the robot initiated its link with the OCU, the box was placed over the robot as evident by the sharp decrease in RSSI around prediction number 30. The box was later removed approximately 100 predictions later, as noted by the sharp increase in RSSI. Not long after the box was removed, the figure shows that the *static* model began to drift, while the adaptable model gradually improved its prediction error.

Two other experiments were performed at the park. The one labeled with 'NLOS' introduced an obstruction during the tail end of testing. Specially, an automobile was placed in between the robot and OCU around prediction number 600, and as a result, a slight divergence between the models is evident during this period. On the other hand, the park experiment labeled with 'LOS' did not include any type of forced disturbance. The intent during this experiment was to determine the whether the model may drift naturally over time without any known disturbance. The park was intentionally selected for this test due to it being the most removed from any potential disturbance sources including obstacles, noise, and interference. The results indicate the model will remain relatively stable in the absence of disturbances, and therefore, the time-based replacement of samples only serves as a failsafe measure in case concept drift is not detected using change detection schemes.

The effects of concept drift are evident in Figure 44. Whenever one of the aforementioned disturbances was introduced, the prediction error of the model without concept drift mitigation began to increase more than the other model that occasionally replaced samples. In order to more precisely quantify the impact of not countering concept

drift, paired t-test statistics were calculated between two different models. Table 10 outlines the 95% confidence intervals in the mean difference (μ_d) between the models *with* and *without* concept drift mitigation for the four experiments of Figure 44. As indicated by the sample range provided in the table, the statistics were calculated over the period starting when the disturbances were roughly introduced and until the end of testing. The table emphasizes the importance of concept drift mitigation because it shows that prediction error consistently increases without the countering mechanisms. For instance, during the indoor interference testing, the prediction error difference between the models was about 9% on average.

Table 10

Mean Difference between Models with and without Change Detection & Forgetting Mechanisms

Location	**Sample Range**	**Paired t-test 95% Confidence Interval**
Indoor Res. (w/ Interference)	375 to 545	$0.0650 < \mu_d < 0.0903$
Track (w/ NLOS)	60 to 571	$0.0210 < \mu_d < 0.0342$
Park (w/ NLOS)	615 to 818	$0.0227 < \mu_d < 0.0440$
Park (w/ LOS)	30 to 895	$0.0016 < \mu_d < 0.0056$

Selective Sampling Effectiveness

The next experiment takes a look into the sampling and retraining behavior of the proposed framework. For this experiment, the results corresponds to the same testing event discussed in Figures 42-43, but different metrics from that experiment are provided in Figure 45. The figure uses both y-axes to represent different information. The dashed line corresponds to the left axis and shows the number of examples included in each batch training event. The markers distinguish the reason for each retraining cycle. From the plot, it is evident that the number of samples rapidly increases in the beginning as the robot initially explores the sample space; however, the number of samples begins to saturate near 250 as the robot continued to occupy mostly the same sample space. Interestingly, the robot continued to retrain its model on a relatively frequent basis, even though the total number of samples in the batch tended to saturate. The explanation relates to the model's forgetting mechanisms that replace samples either based on change detection or time. The y-axis on the right-side of Figure 45 shows the evolution of class labels corresponding to each of the classifiers embedded within the ETF model. The three plots show that the percentage breakdown in label types generally corresponds with the position that each classifier assumes within the ensemble (e.g., bottom, middle, or top). In the early stages of sampling, the percentage of labels below the boundaries is relatively low, but eventually, they stabilize near their respective placement along the TPR universe.

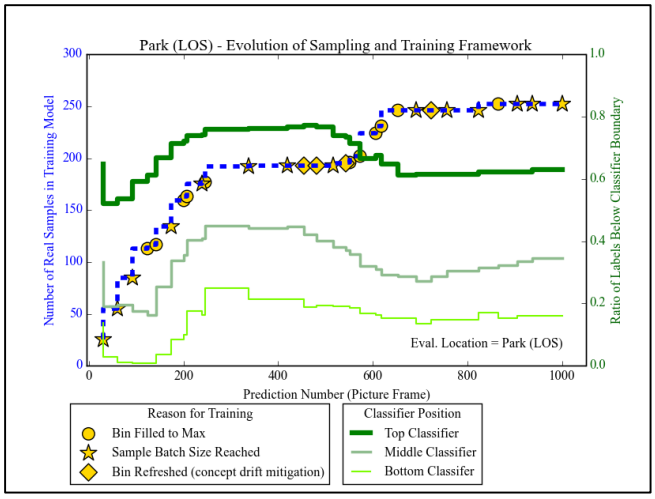

Figure 45. Labeling and retraining behavior of the active and incremental learning framework

The primary objective of the selective sampling scheme was to minimize the number of label requests to only those necessary to maintaining sufficient accuracy. To evaluate the framework's labeling efficiency, the data captured during the experiments of Figure 40 were utilized. Specifically, the types of labeling requests made by the robot, as well as their respective proportions, were investigated. The results of this investigation are summarized in Table 11. In general, the percentage breakdown among the types of requests is variable largely due to the activity of the change detection mechanism, or the amount of time the robot remained active in each scenario. For instance, the park experiment, which was exposed to the least amount of noise and interference, had the fewest number of change detection requests (i.e., ~25%). On the other hand, potential sources of interference were more likely during the other experiments; hence, the probable explanation why roughly 30% or more of the labels requested at these sites were due to change detection. Operating time also factored into the breakdown of the label requests. The robot operated the longest at the park and inside the academic building, and in these

cases, the robot requested over a quarter of its labels based on time expiration; in contrast, less than 8% of the requests were due to time in the briefer experiments. Ideally, the 'None' category would represent a significant portion of the percentages in Table 11 because that would indicate higher efficiency in conserving labeling resources. Given the findings in Table 11, some recommendations to boost labeling efficiency can be formed, and these suggestions, along with others, are provided in the next and final chapter.

Table 11

Breakdown of Labeling Requests based on Active Learning Framework

		Park	Res. Neighborhood	Academic Bldg.	Track
Label Request Type	Bin Not Full	29.2%	39.4%	19.8%	45.0%
	RSSI Change	14.6%	7.8%	16.1%	32.0%
	SQ Change	10.6%	23.5%	19.7%	12.6%
	Time Expired	27.3%	7.8%	34.3%	5.4%
	None	18.3%	21.5%	10.1%	5.0%

Conclusion

In this chapter, an online and incremental learning framework was developed and evaluated. It incorporated novel active learning principles to reduce labeling expenses. Several of these concepts, including the active learning aspect, have yet to be explored in the domain of LQ prediction. A primary objective of this chapter was to compare the online prediction accuracy of the novel ETF model with respect to GLR. The findings show that the ETF model performs slightly better when the sample space has not been fully explored and the model is somewhat underdeveloped. These finding differ slightly from the last chapter, which was based on offline training with more complete datasets. In other words, the online experiments conducted in this chapter exposed the advantage that the ETF model offers during the early stages of LQ prediction. Also, in the process of evaluating the proposed model, the value of incorporating synthetic samples was demonstrated. In particular, these synthetic samples were incorporated in the unexplored feature space so that the robot would be assisted with its initial predictions as it initially entered these regions. This concept of adding synthetics to the training batches has also never been studied in the context of LQ prediction. In addition to the accuracy comparisons between ETF and GLR, the effectiveness of the proposed selective sampling and incremental learning framework was also evaluated. The results showed that under diverse conditions the proposed model effectively recovers from concept drift given its various forgetting mechanisms. Furthermore, the evaluation revealed that the framework also conserves labeling resources to varying degrees, depending upon the operating conditions. In the next chapter, several opportunities for future work related to the framework are offered to the research community.

CHAPTER VII

CONCLUSIONS AND FUTURE WORK

Conclusions

This dissertation was dedicated to advancing the state of the art in terms of estimating or predicting wireless LQ at layers above the physical. The practicality of the problem was realized through the application of the LQ system on a real-world robotic system. Although the problem was framed around robotic networks, the wireless LQ prediction system proposed in this dissertation could be abstracted and used in a number of other upper layer applications.

The concept of making proactive decisions at higher layers based on periodic LQ assessments was put into practice using a novel radio-switching concept for robots. During that evaluation, fuzzy logic was introduced as an intuitive and robust means of weighting multiple input metrics for decision making purposes in a LQ environment. However, the radio-switching controller in Chapter IV lacked robust adaptability, similar to the other fuzzy LQ systems mentioned in the literature.

The issue was addressed by introducing a novel way of making Mamdani, in addition to T-S fuzzy systems, adaptable using machine learning concepts. The technique used a unique ensemble configuration to perform fuzzification, while leaving the other fuzzy processing steps intact. The new approach offers a straightforward way of making fuzzy systems adaptable when the prepositional logic embedded within the rule based does not need to change, but instead, the classification of the inputs into linguistic values requires tuning due to concept drift. The idea of the ensemble-to-fuzzy architecture is extensible to other domains besides LQ prediction. Wireless LQ is a streaming and

dynamic process, yet many researchers treat it as a static or single-iteration learning process as evident by the number of non-adaptable LQ systems in the literature. Prior to this dissertation, the existing state-of-the-art LQ prediction systems depended solely upon learning one sample at a time. Furthermore, up to this point, no research has introduced a comprehensive framework for incrementally adapting learned LQ relationships in a progressive batch-style fashion. Furthermore, there has yet to be any effort within the research community to reduce the overhead associated with labeling samples. This dissertation bridges these gaps by introducing active learning strategies, as well as an iterative batch-learning framework, for predicting LQ in wireless networks.

Future work

The work presented in this dissertation is the first step in a significant paradigm shift for LQ prediction at higher layers. As a result, there certainly remain opportunities to push the domain forward in the future. One focus area could strive to improve prediction accuracy. To maximize accuracy, any future work in the field must be solidly grounded on the known theoretical properties of wireless communication. Often times, important factors, such as channel coherence time (T_c), are overlooked. Although this work presented and was mindful of the theoretical approximation of T_c, future work could more closely study whether samples are paired within the time constraints of T_c. Similarly, future LQ systems should present whether the upper layer application is truly able to sample its features, make some proactive decision, and complete its transmission all within the narrow time constraints of T_c; otherwise, prediction accuracy will suffer and LQ estimation is the next best option for the application.

Another approach that may lead to improved accuracy is an expanded feature set. Although it is challenging to find available and informative features at higher layers, there may be some other good options that were overlooked in this work. Besides using only readily accessible features, future work may also consider making driver and system modifications in order to gain access to raw features embedded within the lower layer protocols. For instance, features indicating contention associated with the shared wireless medium, or raw radio features that are unfiltered by the driver software may likely prove beneficial.

In addition to accuracy, other desirables such as labeling efficiency and adaptability could be improved by modifying the adaptive labeling scheme. For instance, one suggestion is to incorporate an online loss estimation unit [104]. The control unit could be used to monitor prediction error while the system is online and to provide a recent indication into system's prediction performance. In essence, the system component could be used to trigger a temporary increase in sampling and retraining whenever the online predictive error exceeds some acceptable limit or historical average. This approach would likely be a more efficient means of reacting to change than the time-based failsafe used as one of the labeling mechanism in this dissertation. Therefore, it is recommended that the slow concept drift mitigation strategy discussed in this dissertation be replaced with a change detection scheme, such as the aforementioned online loss unit, which is more correlated to the event of concept drift.

Similarly, efficiency and adaptability could also be improved by investigating other types of active learning schemes besides the strategies outlined in this dissertation. Techniques such as uncertainty sampling and maximum disagreement are some

possibilities. Another option, similar to the fast concept drift strategy discussed in this dissertation, is to leverage the information provided by unlabeled samples. More specifically, various techniques within the field of semi-supervised learning [57], such as label propagation, may also prove effective at significantly reducing the costs associated with labeling samples.

Future work could also investigate the possibility of expanding the prediction system into a holistic network model or routing protocol. In this dissertation, LQ was defined as the state of the link for an individual sender and receiver pair. However, in multi-hop networks, routing protocols often look beyond a single hop and look at the conditions all the way to the destination. Therefore, future work may focus on propagating individual LQ estimates so that routing engines could holistically weight different paths to the destination. For example, the next hop decision could be a combination of weighting short-term predictions, based on the connected next hop options, as well as longer-term LQ estimates for the subsequent hops to the destination.

Finally, the overarching concept of the ensemble-to-fuzzy (ETF) model could be abstracted to other problem domains besides LQ prediction. There are no known limitations in the ETF model that may prevent it from being applied to other fields. Theoretically, it can be used on any type of prediction problem where it is feasible or makes sense to divide the range of the target variable into fuzzy sets. Its fuzzy-based structure makes it applicable to many problems because the concept of fuzzy sets tends to reflect many natural-occurring phenomena and the way humans tend to perceive problems [32, 34]. Furthermore, the fuzzy architecture also provides an intuitive interface for scientists and engineers to frame problems and to tweak system performance. Therefore, the ETF

model may prove beneficial in a range of problems requiring predictive models to be learned from data.

REFERENCES

[1] P. L. Yu and B. M. Sadler, "Received signal strength gradient estimation for mobile networks," in *Military Communications Conference*, 2010, pp. 519-523.

[2] J. N. Twigg, J. Fink, P. L. Yu, and B. M. Sadler, "Efficient Base Station Connectivity Area Discovery," *The International Journal of Robotics Research,* July 30, 2013 2013.

[3] F. Zeiger, N. Kraemer, M. Sauer, and K. Schilling, "Challenges in realizing ad-hoc networks based on wireless LAN with mobile robots," in *Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks and Workshops*, 2008, pp. 632-639.

[4] W. Zhigang, Z. MengChu, and N. Ansari, "Ad-hoc robot wireless communication," in *IEEE International Conference on Systems, Man and Cybernetics*, 2003, pp. 4045-4050 vol.4.

[5] P. Maxwell, D. Larkin, and C. Lowrance, "Turning Remote-Controlled Military Systems into Autonomous Force Multipliers," *Potentials, IEEE,* vol. 32, pp. 39-43, 2013.

[6] M. A. Hsieh, A. Cowley, V. Kumar, and C. J. Taylor, "Maintaining network connectivity and performance in robot teams," *Journal of Field Robotics,* vol. 25, pp. 111-131, 2008.

[7] C. J. Lowrance and A. P. Lauf, "Adding transmission diversity to unmanned systems through radio switching and directivity," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014)*, 2014, pp. 3788-3793.

[8] J. N. Twigg, J. R. Fink, P. L. Yu, and B. M. Sadler, "RSS gradient-assisted frontier exploration and radio source localization," in *IEEE International Conference on Robotics and Automation (ICRA)* 2012, pp. 889-895.

[9] P. L. Yu, J. N. Twigg, and B. M. Sadler, "Radio signal strength tracking and control for robotic networks," SPIE Defense, Security, and Sensing, 2011, pp. 803116-803116-12.

[10] H. G. Nguyen, N. Pezeshkian, A. Hart, A. Burmeister, K. Holz, J. Neff*, et al.*, "Evolution of a radio communication relay system," in *Proc. SPIE 8741, Unmanned Systems Technology XV*, 2013, pp. 87410H-8.

[11] T. Samad, J. S. Bay, and D. Godbole, "Network-Centric Systems for Military Operations in Urban Terrain: The Role of UAVs," *Proceedings of the IEEE,* vol. 95, pp. 92-107, 2007.

[12] N. Pezeshkian, J. D. Neff, and A. Hart, "Link Quality Estimator for a Mobile Robot," in *9th Int. Conf. on Informatics in Control, Automation and Robotics (ICINCO)*, Rome, Italy, 2012.

[13] E. Strickland, "Fukushima's next 40 years," *Spectrum, IEEE,* vol. 51, pp. 46-53, 2014.

[14] W. H. Robinson and A. P. Lauf, "Resilient and efficient MANET aerial communications for search and rescue applications," in *2013 International Conference on Computing, Networking and Communications (ICNC)*, 2013, pp. 845-849.

[15] G. Gaertner and V. Cahill, "Understanding link quality in 802.11 mobile ad hoc networks," *Internet Computing, IEEE,* vol. 8, pp. 55-60, 2004.

[16] A. Goldsmith, *Wireless communications*. Cambridge ; New York: Cambridge University Press, 2005.

[17] B. Sklar, *Digital communications* vol. 2: Prentice Hall NJ, 2001.

[18] Y. Mostofi, A. Gonzalez-Ruiz, A. Gaffarkhah, and L. Ding, "Characterization and modeling of wireless channels for networked robotic and control systems - a comprehensive overview," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 4849-4854.

[19] A. M. Tulino, A. Loz, and S. Verdú, "MIMO capacity with channel state information at the transmitter," in *IEEE Eighth International Symposium on Spread Spectrum Techniques and Applications*, 2004, pp. 22-26.

[20] N. Baccour, A. Koubaa, L. Mottola, M. A. Zuniga, H. Youssef, C. A. Boano*, et al.*, "Radio link quality estimation in wireless sensor networks: a survey," *ACM Transactions on Sensor Networks (TOSN),* vol. 8, p. 34, 2012.

[21] W. Stallings, *Data and Computer Communications*, Eighth ed., 2006.

[22] C. Renner, S. Ernst, C. Weyer, and V. Turau, "Prediction accuracy of link-quality estimators," in *Wireless Sensor Networks*, ed: Springer, 2011, pp. 1-16.

[23] S. Biaz, Q. Bing, and J. Yiming, "Improving Expected Transmission Time Metric in Multi-Rate Multi-Hop Networks," in *IEEE Consumer Communications and Networking Conference*, 2008, pp. 533-537.

[24] T. Liu and A. E. Cerpa, "Temporal Adaptive Link Quality Prediction with Online Learning," *ACM Transactions on Sensor Networks (TOSN),* vol. 10, p. 46, 2014.

[25] A. Wapf and M. R. Souryal, "Measuring Indoor Mobile Wireless Link Quality," in *IEEE International Conference on Communications*, 2009, pp. 1-6.

[26] V. Kolar, S. Razak, P. Mähönen, and N. B. Abu-Ghazaleh, "Link quality analysis and measurement in wireless mesh networks," *Ad Hoc Networks,* vol. 9, pp. 1430-1447, 11// 2011.

[27] A. Vlavianos, L. K. Law, I. Broustis, S. V. Krishnamurthy, and M. Faloutsos, "Assessing link quality in IEEE 802.11 Wireless Networks: Which is the right metric?," in *IEEE 19th International Symposium on Personal, Indoor and Mobile Radio Communications*, 2008, pp. 1-6.

[28] E. F. Flushing, J. Nagi, and G. A. Di Caro, "A mobility-assisted protocol for supervised learning of link quality estimates in wireless networks," in *2012 International Conference on Computing, Networking and Communications (ICNC)*, 2012, pp. 137-143.

[29] M. Kudelski, L. M. Gambardella, and G. A. Di Caro, "A mobility-controlled link quality learning protocol for multi-robot coordination tasks," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, 2014, pp. 5024-5031.

[30] G. A. Di Caro, M. Kudelski, E. F. Flushing, J. Nagi, I. Ahmed, and L. M. Gambardella, "Online supervised incremental learning of link quality estimates in wireless networks," in *IEEE 12th Annual Mediterranean Ad Hoc Networking Workshop (MED-HOC-NET)*, 2013, pp. 133-140.

[31] Y. Mostofi, M. Malmirchegini, and A. Ghaffarkhah, "Estimation of communication signal strength in robotic networks," in *IEEE International Conference on Robotics and Automation*, 2010, pp. 1946-1951.

[32] M. Kantardzic, *Data mining: concepts, models, methods, and algorithms*: John Wiley & Sons, 2011.

[33] L. Zadeh, "Fuzzy Sets," *Information and Control,* vol. 8, pp. 338-353, 1965.

[34] J. H. Lilly, *Fuzzy control and identification*. Hoboken, N.J.: Wiley, 2010.

[35] O. Cordón, "A historical review of evolutionary learning methods for Mamdani-type fuzzy rule-based systems: Designing interpretable genetic fuzzy systems," *International Journal of Approximate Reasoning,* vol. 52, pp. 894-913, 2011.

[36] P. Domingos, "A few useful things to know about machine learning," *Communications of the ACM,* vol. 55, pp. 78-87, 2012.

[37] E. Hüllermeier, "Fuzzy sets in machine learning and data mining," *Applied Soft Computing,* vol. 11, pp. 1493-1505, 2011.

[38] Y. S. Abu-Mostafa, M. Magdon-Ismail, and H.-T. Lin, *Learning from data*: AMLBook, 2012.

[39] T. M. Mitchell, *Machine Learning*: McGraw-Hill, Inc., 1997.

[40] D. H. Wolpert, "The lack of a priori distinctions between learning algorithms," *Neural computation,* vol. 8, pp. 1341-1390, 1996.

[41] M. Bowles, *Machine Learning in Python: Essential Techniques for Predictive Analysis*: John Wiley & Sons, 2015.

[42] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*: Prentice Hall, 2010.

[43] W. Chu, M. Zinkevich, L. Li, A. Thomas, and B. Tseng, "Unbiased online active learning in data streams," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2011, pp. 195-203.

[44] X. Zhu, P. Zhang, X. Lin, and Y. Shi, "Active learning from data streams," in *Seventh IEEE International Conference on Data Mining* 2007, pp. 757-762.

[45] B. Settles, "Active learning literature survey," *University of Wisconsin, Madison,* vol. 52, p. 11, 2010.

[46] S. Dasgupta, "Two faces of active learning," *Theoretical computer science,* vol. 412, pp. 1767-1781, 2011.

[47] R. E. Walpole, R. H. Myers, S. Myers, and K. Ye, "Probability and statistics for scientists and engineers," ed: Prentice Hall, 1993.

[48] M. Raju, T. Oliveira, and D. P. Agrawal, "A practical distance estimator through distributed RSSI/LQI processing—An experimental study," in *IEEE International Conference on Communications (ICC)*, 2012, pp. 6575-6579.

[49] K. Dantu, P. Goyal, and G. Sukhatme, "Relative bearing estimation from commodity radios," in *IEEE International Conference on Robotics and Automation*, 2009, pp. 3871-3877.

[50] M. E. M. Campista, P. M. Esposito, I. M. Moraes, L. H. M. K. Costa, O. C. M. B. Duarte, D. G. Passos*, et al.*, "Routing Metrics and Protocols for Wireless Mesh Networks," *Network, IEEE,* vol. 22, pp. 6-12, 2008.

[51] J. Zhang, K. Tan, J. Zhao, H. Wu, and Y. Zhang, "A practical SNR-guided rate adaptation," in *IEEE 27th Conference on Computer Communications*, 2008.

[52] C. Bouras, V. Kapoulas, K. Stamos, N. Stathopoulos, and N. Tavoularis, "Power management for wireless adapters using multiple feedback metrics," in *International Wireless Communications and Mobile Computing Conference*, 2014, pp. 262-267.

[53] C. J. Lowrance and A. P. Lauf, "An efficient fuzzy-based power control scheme for ad hoc networks," in *Wireless Telecommunications Symposium (WTS), 2015*, 2015, pp. 1-8.

[54] N. Baccour, A. Koubâa, H. Youssef, M. Ben Jamâa, D. do Rosário, M. Alves*, et al.*, "F-LQE: A Fuzzy Link Quality Estimator for Wireless Sensor Networks," in *Wireless Sensor Networks*. vol. 5970, J. Silva, B. Krishnamachari, and F. Boavida, Eds., ed: Springer Berlin Heidelberg, 2010, pp. 240-255.

[55] IEEE, "IEEE Std 802.11," in *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, ed, 2012.

[56] M. R. Souryal, J. Geissbuehler, L. E. Miller, and N. Moayeri, "Real-time deployment of multihop relays for range extension," presented at the Proceedings of the 5th international conference on Mobile systems, applications and services, San Juan, Puerto Rico, 2007.

[57] O. Chapelle, B. Schlkopf, and A. Zien, *Semi-Supervised Learning*: The MIT Press, 2010.

[58] J. Zhou, "Impact of wireless link quality across communication layers," TU Delft, Delft University of Technology, 2010.

[59] M. Malmirchegini and Y. Mostofi, "On the spatial predictability of communication channels," *IEEE Transactions on Wireless Communications,* vol. 11, pp. 964-978, 2012.

[60] T. Wee Lum, H. Peizhao, and M. Portmann, "SNR-Based Link Quality Estimation," in *IEEE 75th Vehicular Technology Conference (VTC Spring)*, 2012, pp. 1-5.

[61] D. S. De Couto, D. Aguayo, J. Bicket, and R. Morris, "A high-throughput path metric for multi-hop wireless routing," presented at the Proceedings of the 9th annual international conference on Mobile computing and networking, San Diego, CA, USA, 2003.

[62] D. S. De Couto, D. Aguayo, J. Bicket, and R. Morris, "A high-throughput path metric for multi-hop wireless routing," *Wireless Networks,* vol. 11, pp. 419-434, 2005.

[63] R. Draves, J. Padhye, and B. Zill, "Routing in multi-radio, multi-hop wireless mesh networks," presented at the Proceedings of the 10th annual international conference on Mobile computing and networking, Philadelphia, PA, USA, 2004.

[64] S. Kim, O. Lee, S. Choi, and S.-J. Lee, "Comparative analysis of link quality metrics and routing protocols for optimal route construction in wireless mesh networks," *Ad Hoc Netw.,* vol. 9, pp. 1343-1358, 2011.

[65] Q. Bing, S. Biaz, and S. Fangyang, "Accurate Assessment of Link Loss Rate in Wireless Mesh Networks," in *Seventh International Conference on Information Technology: New Generations (ITNG)*, 2010, pp. 862-866.

[66] K. Kyu-Han and K. G. Shin, "On Accurate and Asymmetry-Aware Measurement of Link Quality in Wireless Mesh Networks," *IEEE/ACM Transactions on Networking,* vol. 17, pp. 1172-1185, 2009.

[67] T. Anh Tai and K. Myung Kyun, "Characteristics of ETX Link Quality Estimator Under High Traffic Load in Wireless Networks," in *IEEE International Conference on High Performance Computing and Communications & Embedded and Ubiquitous Computing*, 2013, pp. 611-618.

[68] H. Zhang, A. Arora, and P. Sinha, "Link estimation and routing in sensor network backbones: Beacon-based or data-driven?," *IEEE Transactions on Mobile Computing,* vol. 8, pp. 653-667, 2009.

[69] NS-3. Available: http://www.nsnam.org/

[70] T. Ye, K. Xu, and N. Ansari, "TCP in wireless environments: problems and solutions," *Communications Magazine, IEEE,* vol. 43, pp. S27-S32, 2005.

[71] M. H. Alizai, H. Wirtz, G. Kunz, B. Grap, and K. Wehrle, "Efficient online estimation of bursty wireless links," in *IEEE Symposium on Computers and Communications (ISCC)*, 2011, pp. 191-198.

[72] A. Becher, O. Landsiedel, G. Kunz, and K. Wehrle, "Towards short-term wireless link quality estimation," *Hot Emnets,* 2008.

[73] D. Giustiniano, D. Malone, D. J. Leith, and K. Papagiannaki, "Estimating link quality in 802.11 WLANs," 2007.

[74] D. Wu, P. Djukic, and P. Mohapatra, "Determining 802.11 link quality with passive measurements," in *IEEE International Symposium on Wireless Communication Systems*, 2008, pp. 728-732.

[75] H. Zhang, L. Sang, and A. Arora, "Comparison of data-driven link estimation methods in low-power wireless networks," *IEEE Transactions on Mobile Computing,* vol. 9, pp. 1634-1648, 2010.

[76] C. Reis, R. Mahajan, M. Rodrig, D. Wetherall, and J. Zahorjan, "Measurement-based models of delivery and interference in static wireless networks," *SIGCOMM Comput. Commun. Rev.,* vol. 36, pp. 51-62, 2006.

[77] M. Senel, K. Chintalapudi, D. Lal, A. Keshavarzian, and E. J. Coyle, "A Kalman Filter Based Link Quality Estimation Scheme for Wireless Sensor Networks," in *IEEE Global Telecommunications Conference*, 2007, pp. 875-880.

[78] M. R. Souryal, L. Klein-Berndt, M. E. Miller, and N. Moayeri, "Link assessment in an indoor 802.11 network," in *IEEE Wireless Communications and Networking Conference*, 2006, pp. 1402-1407.

[79] L. Verma, K. Seongkwan, C. Sunghyun, and L. Sung-Ju, "Reliable, Low Overhead Link Quality Estimation for 802.11 Wireless Mesh Networks," in *Sensor, Mesh and Ad Hoc Communications and Networks Workshops, 2008. SECON Workshops '08. 5th IEEE Annual Communications Society Conference on*, 2008, pp. 1-6.

[80] G. Judd, X. Wang, and P. Steenkiste, "Efficient channel-aware rate adaptation in dynamic environments," in *Proceedings of the 6th international conference on Mobile systems, applications, and services*, 2008, pp. 118-131.

[81] J. Zhou, M. Jacobsson, E. Onur, and I. Niemegeers, "An Investigation of Link Quality Assessment for Mobile Multi-hop and Multi-rate Wireless Networks," *Wireless Personal Communications,* vol. 65, pp. 405-423, 2012/07/01 2012.

[82] C. A. Boano, M. A. Zuniga, T. Voigt, A. Willig, and K. Römer, "The Triangle Metric: Fast Link Quality Estimation for Mobile Wireless Sensor Networks," in *Proceedings of 19th International Conference on Computer Communications and Networks*, 2010, pp. 1-7.

[83] J. Ko and M. Chang, "MoMoRo: Providing Mobility Support for Low-Power Wireless Applications," *Systems Journal, IEEE,* vol. PP, pp. 1-10, 2014.

[84] G. Zhi-Qiang, W. Qin, L. Mo-Han, and H. Jie, "Fuzzy Logic Based Multidimensional Link Quality Estimation for Multi-Hop Wireless Sensor Networks," *Sensors Journal, IEEE,* vol. 13, pp. 3605-3615, 2013.

[85] L. Liu, Y. Fan, J. Shu, and K. Yu, "A link quality prediction mechanism for wsns based on time series model," in *7th International Conference on Ubiquitous Intelligence & Computing and Autonomic & Trusted Computing*, 2010, pp. 175-179.

[86] K. Farkas, T. Hossmann, F. Legendre, B. Plattner, and S. K. Das, "Link quality prediction in mesh networks," *Computer Communications,* vol. 31, pp. 1497-1512, 2008.

[87] K. Farkas, T. Hossmann, L. Ruf, and B. Plattner, "Pattern matching based link quality prediction in wireless mobile ad hoc networks," in *Proceedings of the 9th ACM international symposium on Modeling analysis and simulation of wireless and mobile systems*, 2006, pp. 239-246.

[88] P. Millan, C. Molina, E. Medina, D. Vega, R. Meseguer, B. Braem*, et al.*, "Tracking and predicting link quality in wireless community networks," in *IEEE 10th International Conference on Wireless and Mobile Computing, Networking and Communications*, 2014, pp. 239-244.

[89] A. S. Cacciapuoti, M. Caleffi, L. Paura, and M. Rahman, "Link quality estimators for multi-hop mesh network," in *Euro Med Telco Conference (EMTC), 2014*, 2014, pp. 1-6.

[90] M. Caleffi and L. Paura, "Bio-inspired link quality estimation for wireless mesh networks," in *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks & Workshops*, 2009, pp. 1-6.

[91] E. F. Flushing, M. Kudelski, L. M. Gambardella, and G. A. Di Caro, "Spatial prediction of wireless links and its application to the path control of mobile robots," in *9th IEEE International Symposium on Industrial Embedded Systems (SIES)*, 2014, pp. 218-227.

[92] Y. Wang, M. Martonosi, and L.-S. Peh, "Predicting link quality using supervised learning in wireless sensor networks," *ACM SIGMOBILE Mobile Computing and Communications Review,* vol. 11, pp. 71-83, 2007.

[93] T. Liu and A. E. Cerpa, "Data-driven link quality prediction using link features," *ACM Transactions on Sensor Networks (TOSN),* vol. 10, p. 37, 2014.

[94] H.-N. Dai, K.-W. Ng, M. Li, and M.-Y. Wu, "An overview of using directional antennas in wireless networks," *International Journal of Communication Systems,* vol. 26, pp. 413-448, 2013.

[95] D. Gesbert, M. Kountouris, R. W. Heath, C. Chan-Byoung, and T. Salzer, "Shifting the MIMO Paradigm," *Signal Processing Magazine, IEEE,* vol. 24, pp. 36-46, 2007.

[96] W. E. Combs and J. E. Andrews, "Combinatorial rule explosion eliminated by a fuzzy rule configuration," *IEEE Transactions on Fuzzy Systems,* vol. 6, pp. 1-11, 1998.

[97] K. Srinivasan, P. Dutta, A. Tavakoli, and P. Levis, "Understanding the causes of packet delivery success and failure in dense wireless sensor networks," presented at the 4th International Conference on Embedded Networked Sensor Systems, Boulder, Colorado, USA, 2006.

[98] J. Barker, "You Believe You Understand What You Think I Said: The Truth About 802.11 Signal And Noise Metrics," *Document D100201,* 2004.

[99] K. Srinivasan and P. Levis, "RSSI is Under Appreciated," in *Proceedings of the Third Workshop on Embedded Networked Sensors*, 2006.

[100] IEEE, "IEEE Standard for Local and metropolitan area networks--Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)," ed, 2011.

[101] L. Liu, J. Li, J. Shu, Z. Wu, and Y. Chen, "CCI-based link quality estimation mechanism for wireless sensor networks under perceive packet loss," *Journal of Software,* vol. 5, pp. 387-395, 2010.

[102] L. Huang and T.-H. Lai, "On the scalability of IEEE 802.11 ad hoc networks," presented at the Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking and computing, Lausanne, Switzerland, 2002.

[103] Google. Available: https://www.google.com/maps/

[104] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Computing Surveys (CSUR),* vol. 46, p. 44, 2014.

[105] scikit-learn: Machine Learning in Python [Online]. Available: http://scikit-learn.org/stable/

[106] Weka 3: Data Mining Software in Java [Online]. Available: http://www.cs.waikato.ac.nz/ml/weka/

[107] Broadcom. *BCM2835*. Available: http://www.broadcom.com/products/BCM2835

[108] scikit-learn: Support Vector Machines [Online]. Available: http://scikit-learn.org/stable/modules/svm.html

[109] J. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," *Advances in large margin classifiers,* vol. 10, pp. 61-74, 1999.

[110] R. N. Lichtenwalter and N. V. Chawla, "Adaptive Methods for Classification in Arbitrarily Imbalanced and Drifting Data Streams," in *International Workshops on New Frontiers in Applied Data Mining*, T. Theeramunkong, C. Nattee, P. J. L. Adeodato, N. Chawla, P. Christen, P. Lenca*, et al.*, Eds., ed Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 53-75.

[111] I. Zliobaite, A. Bifet, B. Pfahringer, and G. Holmes, "Active learning with drifting streaming data," *IEEE Transactions on Neural Networks and Learning Systems,* vol. 25, pp. 27-39, 2014.

[112] M. A. Maloof and R. S. Michalski, "Incremental learning with partial instance memory," *Artificial intelligence,* vol. 154, pp. 95-126, 2004.

[113] S. Markovitch and P. D. Scott, "The role of forgetting in learning," in *5th International Conference on Machine Learning*, 1988, pp. 459-465.

[114] H. Nakayama and K. Yoshii, "Active forgetting in machine learning and its application to financial problems," in *IEEE-INNS-ENNS International Joint Conference on Neural Networks*, 2000, pp. 123-128.

CURRICULUM VITAE

NAME: Christopher J. Lowrance

ADDRESS: Department of Computer Engineering and Computer Science
J.B. Speed School of Engineering
University of Louisville
Louisville, KY 40292

EDUCATION & TRAINING:

Ph.D. Candidate, Computer Science and Engineering
University of Louisville
2013-2016

Graduate Certificate, Telecommunications Security and Electronic Warfare
The George Washington University
2009

M.S., Electrical Engineering
The George Washington University
2007-2008

B.S., Electrical Engineering
Virginia Military Institute
1996-2000

APPOINTMENTS: Assistant Professor
Department of Electrical Engineering and Computer Science
United States Military Academy (USMA)
West Point, NY 10996
2009-2012 and 2016-2019

PUBLICATIONS:

Journals

Maxwell, P., Larkin, D., Lowrance, C., "The Joint Cooperative Unmanned Systems Initiative: Turning Remote Controlled Military Systems Into Autonomous Force Multipliers." IEEE Potentials, Nov 2013.

Conferences

Abdelwahab, O., Bahgat M., Lowrance, C.J., Elmaghraby, A., "Effect of Training Set Size on SVM and Naïve Bayes for Twitter Sentiment Analysis." IEEE International Symposium on Signal Processing and Information Technology, Dec. 2015.

Lowrance, C.J., Lauf, A. P., "An Efficient Fuzzy-based Power Control Scheme for Ad hoc Networks." Wireless Telecommunications Symposium, April 2015.

Lowrance, C.J., Abdelwahab, O., Yampolskiy, R.V., "Evolution of a Metaheuristic for Aggregating Wisdom from Artificial Crowds." 17th Portuguese Conference on Artificial Intelligence, Sept 2015.

Lowrance, C.J., Lauf, A. P., "Adding Transmission Diversity to Unmanned Systems through Radio Switching and Directivity." IEEE/RSJ International Conference on Intelligent Robots and Systems, Sept 2014.

Fernandes, J.,Hammond, P., Nelson, C., Starck-King, N., Lowrance, C.J., and Sadowski, R.W., Joint Cooperative Unmanned Systems Initiative: U.S. Military Academy 2012 Ground Segment Development," Poster presentation at the Ground Robotics Capabilities Conference, Mar 2012.

Viall, K., Lowrance, C., Bronikowski, S., "Thayer Quiz Method: Replacing Homework with Frequent Quizzes in Engineering Classes." 41st ASEE/IEEE Frontiers in Education Conference, Rapid City, SD, October 12-15, 2011.

Bronikowski, S., Lowrance, C., Viall, K., "Lather, Rinse, Repeat: The Effect of Replacing Homework with Periodic Quizzes in Engineering Courses." American Society for Engineering Education, ASEE Middle Atlantic Section Spring Conference, Farmingdale, NY, April 29-30, 2011.

Lowrance, C., "An Efficient Teaching Technique for Engineering." ASEE Spring 2010 Mid-Atlantic Section Spring Conference, Lafayette College, Easton, PA. April 16-17, 2010.

PROFESSION SOCIETIES:

IEEE, IEEE Computer Society, Eta Kappa Nu (HKN), Tau Beta Pi, Phi Kappa Phi