SENSOR NETWORK FOR EARLY ILLNESS DETECTION IN THE ELDERLY

A Thesis presented to

the Faculty of the Graduate School

at the University of Missouri-Columbia

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

---

by

RAINER DANE A. GUEVARA

Dr. Marjorie Skubic, Thesis Supervisor

JULY 2012

The undersigned, appointed by the dean of the Graduate School, have examined the thesis entitled

SENSOR NETWORK FOR EARLY ILLNESS

DETECTION IN THE ELDERLY

presented by Rainer Dane A. Guevara,

a candidate for the degree of Master of Science,

and hereby certify that, in their opinion, it is worthy of acceptance.

Dr. Marjorie Skubic, Ph.D.

Dr. Marilyn Rantz, Ph.D.

Dr. Mihail Popescu, Ph.D.

Acknowledgements

I would like to thank my advisor Dr. Marjorie Skubic for not only her guidance and support but also for her patience and determination to help me graduate. Whenever I struggled she willingly offered advice, direction, and encouragement. Dr. Skubic introduced me to the Mizzou Eldertech team, the world of computational intelligence, and the importance of eldercare technology.

I would like to thank Dr. Marilyn Rantz for her leadership and continued support through the entirety of the R21 research project and her expert knowledge in gerontology and nursing care coordination.

I want to give a special thanks to Dr. Mihail Popescu, who initially sparked the idea to create an alert feedback loop through email. Dr. Popescu constantly provided insight and alternative strategies for this research.

I would like to thank those who took the time to deal with the glitches in the alert system and iterative design process of the logger: Katy Musterman, Jessica Back, Melissa Koga, Dr. Gregory Alexander, Dr. Myra Aud, Dr. Richelle Koopman, and Dr. Lorraine Phillips. They left valuable feedback that was vital to the development of more sophisticated alerts and will continue to be of importance as this research evolves in the future.

Lastly, I would like to thank Ian Kable for his technical expertise and help with the installation and maintenance of the sensor networks and loggers at TigerPlace.

Table of Contents

List of Figures

List of Tables

Abstract

The independent living facilities at TigerPlace allow the elderly to age in place in a homely setting that fosters independence, autonomy, and privacy while providing some level of aid like shared meals, housekeeping, and other services. When elderly residents fall victim to situations that have long-term health implications, e.g. urinary tract infections, even immediate intervention by a nursing care-provider after the fact may not be timely enough. However, if the nursing care-providers know of certain behavior patterns or specifically changes in those patterns that indicate early signs of illness, then they can make an intervention or take preventative measures ahead of time. This research provides a framework and method for using passive in-home sensor networks to collect sensor data, Early Illness Alert Algorithms to model and detect signs of early illness, single-dimensional alerts that notify nursing care-providers, and collect clinical feedback on alerts from a team of clinical researchers with an expertise in gerontology. The feedback collected provides valuable ground truth that is used to analyze and improve the Early Illness Alert Algorithms. Classification accuracy more than doubles through the application of four machine learning methods for classification and include the Fuzzy Pattern Tree, the Fuzzy K-Nearest Neighbor, the Neural Network, and the Support Vector Machine.

Chapter 1 - Introduction

In the United States, the population of the elderly, or adults 65 years of age and older, is projected at over 80 million people in 2050, more than double the 2010 population of 40 million [1]. Because of this growth, there has been extensive research over the past decade in the field of eldercare technology. The paradigm has shifted from the traditional health care system where health care is administered retroactively, i.e. after critical health events, to a proactive system where technology is used in independent, smart home settings to predict signs of declining health before the elderly get worse. With this paradigm shift various research groups have developed at academic and corporate institutions all over the United States and the world to conduct research and advance the technologies for eldercare.

At the University of Missouri – Columbia, the Eldertech research team began in 2002 as a collaboration of faculty and students from computer and electrical engineering, computer science, nursing, medicine, health management informatics, physical therapy, and social work with the vision of developing and advancing technologies for eldercare. Together with the local independent living facilities at TigerPlace, the team continues to conduct eldercare technology research in a real environment with real data collected from real elderly people, in addition to simulated data collected in the more typical laboratory setting. As one of the specific goals of their research, the team aims to use technology to reduce the rate of functional/cognitive decline in the elderly in independent living to provide for the elderly the highest possible quality of life in the latter part of their lives. Simply put, their goal is to keep the elderly at TigerPlace as active, functionally

independent, and happy for as long as possible. An illustration of this idea is summarized in Figure 1.1 [2].

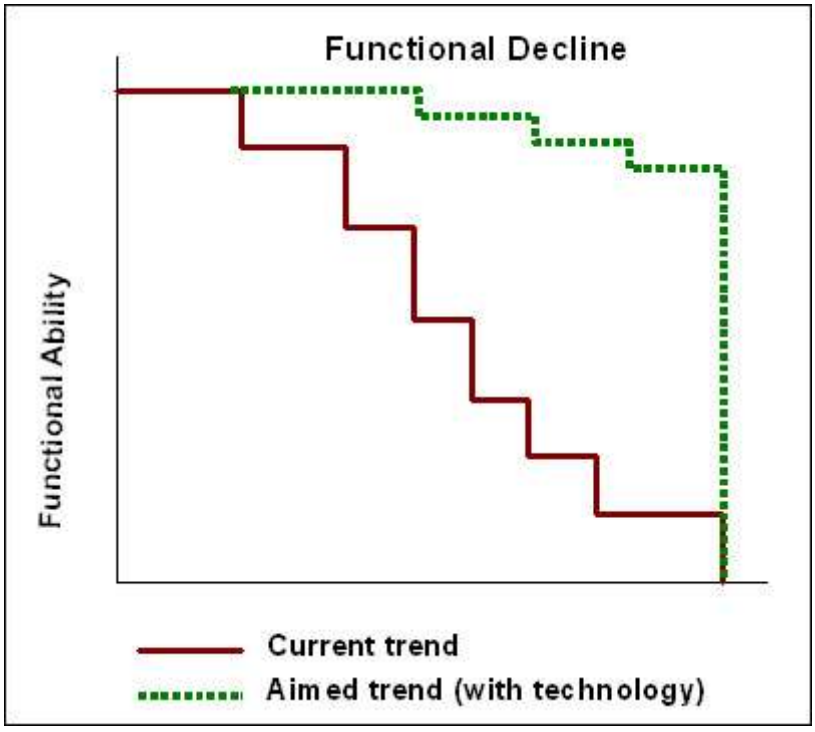

Figure 1.1 Objective of Eldercare Technology

As the elderly age their functional abilities decline whether it is due to long term health conditions like Alzheimer's or dementia or shorter term health conditions like urinary tract infections or isolated events like an injury or fall. Unfortunately, these events can sometimes cause more severe declines in functional ability even though often times they are preventable or at the very least more manageable if caught early enough. A potential solution to help bring about the aimed trend lies in the data from the sensor networks deployed for elderly residents at TigerPlace, whether that data can shed some light on changes in the residents' health, and how responders in the assisted living facility can use that information for the benefit of the resident. In particular, the Eldertech team

wants to create a proactive health care system through the early detection of possible illness in contrast to the reactive healthcare system more typically used today.

1.2 Problem Statement

Elderly residents at TigerPlace have a network of motion sensors, bed sensors, and stove temperature sensors installed in their apartments, along with a personal computer that collects data from the sensors and stores it in a sensor database. Of these residents, over 90% have chronic health conditions and over 60% have more than one chronic health condition. The residents are between 65 and 100 years old. On the Eldertech team is a group of clinical researchers (RNs, Nursing PhDs, and MDs) with expertise on gerontology and aging in place. The clinical researchers also have remote access to visualizations of the sensor data via an internet browser connected to our sensor web interface.

The goal of the research in this thesis is to investigate Early Illness Alerts generated automatically from a passive sensor network with multiple steps: (1) develop the data logging capability, (2) develop the Early Illness Alert Algorithm through an iterative human-centered design process with the help of the clinical researchers, (3) develop a system for capturing feedback on the Early Illness Alerts from the research clinicians, and (4) investigate other pattern recognition methods for generating Early Illness Alerts based on the ground truth created from the feedback. The system consists of an algorithm to model the behaviors of the residents over time, recognize changes in those behaviors, and alert the appropriate responders, e.g. care-providers, as quickly and early as possible to maintain the health and functional ability of the residents. In addition, the clinical researchers review the generated alerts and their conditions and provide

feedback to create a ground truth dataset that can be used to refine the original behavior modeling and change detection algorithm. With the idea that in the future the Eldertech team would want to give clinical researchers or nursing care-providers the ability to customize the algorithm without having to be expert computer engineers or computer scientists, this work presents a Pattern Tree as the refined algorithm. This research also compares the Pattern Tree against other machine classifiers: the Fuzzy K-Nearest-Neighbor, the Neural Network, and the Support Vector Machine.

1.2 Overview

Chapter 2 covers related eldercare technology research, the sensors used in this research, and backgrounds on the Pattern Tree, Fuzzy K-Nearest Neighbor, Neural Network, and Support Vector Machine classifiers. Chapter 3 discusses the main concepts and methods used in this research, including the integrated sensor network framework, the method for modeling behavior and detecting an Early Illness Alert, and how the classifiers from Chapter 2 are tailored for Early Illness Alerts. Chapter 4 presents the experimental results, which include the ground truth alert dataset and results from single dimensional alerts and multi-dimensional alerts using each of the four classifiers. A discussion of the results follows in Chapter 5. Chapter 6 concludes with a summary of the research and opportunities for extensions and future work.

1.3 Contributions

Listed below are the contributions of this research.

1. Development of the CERT Logger
2. Development of the single-dimensional Early Illness Alert Algorithm
3. Development of the Early Illness Alert email mechanism

4. Development of the Early Illness Alert feedback collection mechanism
5. Collection of feedback and generation of Early Illness Alert ground truth
6. Analysis of single dimensional Early Illness Alerts
7. Analysis of four classification methods (Pattern Tree, Fuzzy K-Nearest Neighbor, Neural Network, and Support Vector Machine) for multi-dimensional Early Illness Alerts

Chapter 2 - Related Work and Background

Chapter 2 begins by covering research related to the use of wireless sensor networks in smart home applications for the elderly and how the sensor networks, made up of various types of sensors, have been and continue to monitor the health of elderly residents in various eldercare projects not only in the United States but all over the world. The chapter continues on to describe in depth the sensor technology used in the sensor networks at TigerPlace. Finally, a general methodology of the Pattern Tree, Fuzzy K-Nearest Neighbor, Neural Network, and Support Vector Machine classifiers used in the research are presented.

2.1 Related Work

In pervasive computing, smart homes for health related applications [3] have emerged as proactive solution for maintaining the health of the elderly in independent living environment and have several advantages. Smart homes have the capability to remotely, continuously, and automatically monitor elderly residents using sensor networks [4], assess sensor information to detect abnormal behaviors or emergencies, and notify appropriate caregivers. Often times, these smart home projects model normal behavior patterns of the resident using some strategy for activity recognition [5], which could be as low-level as statistical inference on overall sensor activity levels or as high-level as recognizing the frequency, duration, presence, or absence of common activities of daily living (ADLs) [6]. When changes in or abnormal behaviors occur based on critical thresholds, the smart home system alerts the appropriate personnel. The automated detection and notification of emergencies along with nursing care coordination

allows the resident's family to feel at ease while the elderly resident continues to live independently. On the other hand, smart home applications for eldercare also present challenges like balancing ubiquity and privacy [7] and successfully delivery and acceptance of technology in the elderly community [4]. The Eldertech team has deployed its own version of a smart home in the apartments of elderly residents at TigerPlace. Over the past two decades, other eldercare technology research teams, some in conjunction with continuing care retirement communities (CCRCs) like TigerPlace, have also conducted similar research using combinations of smart homes, sensor networks, activity recognition, and alerts for eldercare.

Some of the earliest work written in the eldercare technology literature in 1995 comes from the multidisciplinary project at the University of New South Wales [8]. The team used sensor networks to continuously monitor several simple ADL parameters – mobility, sleep patterns, cooking, washing, and toilet use – using infrared (IR) sensors, light sensors, temperature sensors, and mechanical/magnetic switches and tested whether they could accurately detect changes in functional health remotely. Their end goal is to develop the system to automatically prompt appropriate, timely and cost-effective intervention.

In 1998, a research project [9] between British Telecom (BT) and Anchor Trust in the United Kingdom investigated and developed Lifestyle Monitoring , a smart home sensor network that used low-cost motion, door, temperature and appliance usage sensors to create normal daily profiles for an elderly resident. If certain events consisting of lack of sensor activity, not getting out of bed in the morning, abnormal use of the fridge, or abnormal room temperature were detected, an alert was sent to a caregiver via telephone.

Also from British Telecom, Majeed and Brown described the "well-being" monitoring of elderly residents a sensor network of door and motion sensors [10]. Sensor data are classified using a system fuzzy of rules to classify into one of size ADLs such as sleeping, preparing or eating food, and receiving visitors. The system was tested with two elderly participants.

In 1999, the team at the Georgia Institute of Technology had begun building Aware Home [11], a smart home laboratory for research in ubiquitous computing for everyday activities. The goal of Aware Home, which is still used today, is to conduct human-centered research and develop context-aware sensor networks.

In 2000, another smart home project conducted at Drexel University used a similar system based on ADL recognition [12]. Their system logged sensor events from motion, heat, vibration, and electric current sensors to continuously monitor and inferred ADLs from sequences of sensor events. The system was fully automated and unobtrusive. They collected data on four ADLs: medication adherence, movement throughout the house, bathroom use, and meal preparation. They tested the system during a 12 day period in the residence of a 71 year old male who lived alone in a CCRC.

In 2002, a study conducted by the Tokyo Medical and Dental University in Japan deployed two sensor networks in the home of 74 year old woman for two years and the home of a 72 year old woman for one year. The sensor networks used motion sensors, magnetic switches, wattmeters, and flame detectors to detect various ADLs including number of door openings, the length of sleep, time absent from house, time spent cooking, and time spent watching television.

During the same year, the interdisciplinary team at the Massachusetts Institute of Technology (MIT) had been developing PlaceLab [13-15], a live smart home laboratory similar to Aware Home to conduct research in context-aware sensing in the home. The PlaceLab interior is formed by 15 prefabricated cabinetry interior components, each of which contains a microcontroller and 25-30 sensors. There are environmental sensors such as floor and ceiling air temperature, humidity, and CO and CO2 sensors. Small sensors are located on every object that people touch and use, such as cabinet doors and drawers, controls, furniture, windows and kitchen containers, to detect on-off, open-close and movement events. Video and audio sensors are also included. In 2004, Bao and Intille [14] studied the use of wearable accelerometers to detect ADLs. They succeeded in detecting 80% of over 20 different ADLs using a decision tree classifier. They summarize their three PlaceLab pilot studies and the datasets they were able to create in [15].

Also in 2004, Honeywell Laboratories developed the Independent LifeStyle Assistant (ILSA), an integrated smart home proposed to incorporate a unified sensor network, activity recognition, and machine learning to generate alerts based on abnormal behaviors [16]. Honeywell prototyped a passive monitoring system and conducted a 6-month study before the system was retired.

During the same year, the Codeblue project at Harvard began exploring the application of wireless sensor network to a range of medical applications, including pre-hospital and in-hospital emergency care, disaster response, and stroke patient rehabilitation. Currently, they are working on Peleton (resource management of ad-hoc sensor networks) [17] and Pixie (resource-aware programming of embedded sensors) [18].

In 2005, a research group in France investigated the use of a sensor network in a smart home to monitor motor activity data (in bed, getting up, getting out, visiting the toilet), which was analyzed statistically to assess changes in occurrence, time, and duration [19]. They discovered interesting trends in the changes that could be used to build an abnormal event diagnosis system to assist elderly living along at home. In 2010, a second group from France investigated the use of Support Vector Machine multimodal classification to detect seven ADLs using features extracted from motion sensors, wearable accelerometers and magnetometers, temperature and hygrometry sensors [20]. They achieved 75% accuracy using a polynomial kernel and 86% using a Gaussian kernel

In 2008, the team at the Intelligent Systems Lab Amsterdam developed a sensor network capable of automatically recognizing activities [21]. Using the hidden Markov model with conditional random fields, they achieved a classification accuracy of 79.4%.

More recently in 2010, a group from Tamkang University in Taiwan used an RFID-based sensor network to develop a behavior modeling and anomaly detection system. Fuzzy C-means clustering is used to create the normal cluster and a threshold is used to indicate outliers or abnormal behaviors. Preliminary results were promising but a more robust method is desired.

Sections 2.1.1-3 describe selected related research projects in more detail.

2.1.1 University of Virginia

In 2005, the SmartHouse project from the University of Virginia (UVA) used wireless sensor networks for health monitoring [22]. Their sensor networks continuously monitored assisted and independent-living elderly residents and consisted of motion sensors, wearable sensors, temperature and luminosity sensors, a bed sensor, a pulse-

oximeter, and an electrocardiogram [23]. They investigated whether simple motion sensors could detect behavioral changes and proposed a mixture model framework to develop a probabilistic model of behavior. The mixture model clusters observations into clusters such as kitchen event or bathroom event and uses the Expectation Maximization algorithm to learn the model parameters and assign the training set to clusters. Results show that simple motion sensors can be used to detect behavioral patterns.

Also in 2005, UVA proposed a fuzzy rule-based inference system to detect ADLs so they could identify abnormal patterns in their ADLs consistent with health issues that developed over time [24]. They wanted to detect possible emergency conditions for alerts. The data came from motion sensors, a bed sensor, and a stove temperature sensor and was collected for 37 days. A period of 17 days was used to train the system of fuzzy rules and 20 days were used to validate. Results showed that the system based on domain knowledge could successfully detect meal preparation and showering ADLs using motion sensors.

In 2008, the group at UVA created AlarmNet [25]. AlarmNet integrated environmental, physiological, and activity sensors in a scalable, flexible architecture. The system supported a two-way flow of data to enable context-aware protocols and used Circadian Activity Rhythm (CAR) analysis to learn activity patterns and aid context-aware sensor power management.

2.1.2 Oregon Health and Science University

The team at Oregon Health and Science University (OHSU) is focused on unobtrusive monitoring and detecting the gradual decline of cognitive function in the elderly. In 2004, an in-home monitoring system was set up to detect key motor changes

preceding cognitive decline [26]. The system was installed in three apartments of different elderly residents and consisted of low-cost motion sensors and contact sensors. They tried to measured sensor activity levels, average walking speed and patterns of activity. They had up to 20 subjects enrolled at a time and data collection lasted for at least eight weeks. Using a two week sliding window, they used Kernel Density Estimation to approximate the distribution of the walking time, trimmed the outliers, and calculated the average walking speed. Results show that this is a good method for estimating the speed of walking. The main problem arises with the presence of visitors, and sensor data has to be filtered out before the walking speed can be processed. In 2011, an update of the research at OHSU indicates that the project has grown and sensor networks have been installed in the homes of 265 elderly residents for an average of 33 months [27].

2.1.3 Eldertech

The Eldertech team at Mizzou have and continue to conduct numerous projects related to the use of sensor networks in a smart home application to monitor the elderly in an independent-living environment [28-31]. Currently, the Eldertech team has over 20 sensor networks collecting data and generating alerts at TigerPlace as well as new sensor networks deployed remotely at Cedar Falls.

2.2 X10 Sensors

X10 refers to the protocol used by sensors in the sensor network. X10 sensors are commercially-available sensors primarily advertised for in-home security and automation, domotics, purposes and use passive IR (PIR) sensing or temperature sensing

as an extension of PIR sensing. The motion sensors used in this research are ActiveHome EagleEye X10 motion sensors. The bed sensors have been previously developed by the University of Virginia (UVA) and use the X10 protocol to encode and transmit its signals [32] . The stove temperature sensors are X10 motion sensors with a thermistor integrated into the sensing element to detect changes in temperature and send a “Temp High” signal when the oven gets hot and “Temp Low” when it cools back down. The advantages of using X10 sensors are their relatively low cost and availability.

2.2.1 X10 Protocol

The X10 protocol is the coded procedure in which X10 sensors transmit data. The protocol is an international and open industry standard for communication between electronic devices used for home automation. X10 primarily uses power line wiring for signaling and control, but also defined is a wireless radio based transport protocol.

All of the sensors in this research use the wireless protocol to send a signal in the form of -House-Code/Unit-Code/Command. For example, A1 ON is a signal where A is the house code, 1 is the unit code, and ON is the command. There are a total of 16 house codes (A-P) and 16 unit codes (1-16). These signals are encoded (discussed further in section 3.3.1) into a one-way stream of four bytes that a W800 X10 receiver decodes and error-checks. Since there is no handshaking involved, a signal is repeated five times over the course of about 104 ms to help ensure transmission to the receiver. This does not, however, protect against the possibility of collision, i.e. multiple sensors sending X10 wireless signals simultaneously, giving the receiver an undecipherable code.

The combination of a house code and unit code determine the address of an X10 sensor, so there are a total of 256 unique address spaces, which limits the number of

sensors that can be configured in close proximity of a receiver without overlapping. In other words, once there are more than 256 motion or stove temperature sensors (bed/chair sensors use a slightly different configuration), new motion or temperature sensors must reuse old address spaces and the deployment needs to be careful about how the addresses are distributed to prevent a sensor in one resident's network from providing false data for another resident with an X10 receiver in close proximity.

The available X10 commands for the motion and temperature sensors are ON and OFF. Currently, the logging system (discussed further in Chapter 3) observes only the ON signals to collect motion hits and stove on hits from the motion and stove temperature sensors, respectively. The available X10 commands for the bed/chair sensors are ALL ON and ALL OFF, which makes them distinguishable from the motion and stove temperature sensors. The X10 protocol does not present a method for encryption. Intruders have the possibility of sniffing X10 wireless packets using their own X10 receivers.

2.2.2 X10 Motion Sensor

The X10 motion sensor shown in Figure 2.1uses a PIR sensing element to detect motion. The manufacturers have designed the MS14A as both an indoor and a weatherproof outdoor sensor. It runs on two AAA batteries and is programmable to any of the 256 unique house-code/unit-code combinations.

Figure 2.1 ActiveHome EagleEye (MS14A) X10 Motion Sensor

The PIR sensing element measures IR waves radiating in its field of view. Consider the sensing element with a normal source temperature in front of it, e.g. a wall. The sensor detects motion when an IR source with another temperature, e.g. a human, passes through its field of view. A Fresnel lens which focuses the IR radiation into the sensing element determines the X10 sensor's field of view. The cone-shaped field of view is illustrated in Figure 2.2.

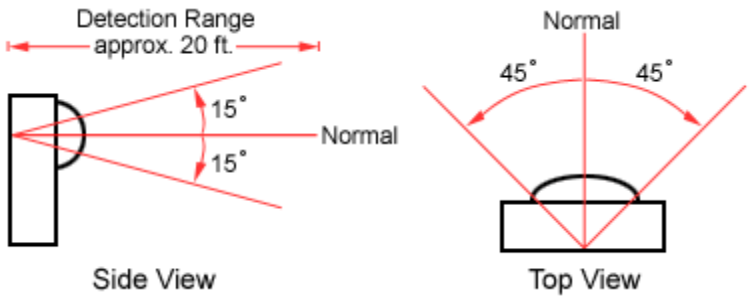

Figure 2.2 X10 Sensing Range (taken from [33])

When triggered, the X10 motion sensor sends an ON signal and waits for at least seven seconds before motion will trigger the sensor to send another ON signal. After 60 seconds of no motion after an ON signal, the sensor sends an OFF signal.

Problems that arise in a production environment with the X10 motion sensors (besides those due to the limitations of the X10 protocol) include loss of data from sensors with low batteries or fallen sensors and false alarm data detected from large non-human bodies of heat, e.g. dogs, cats, heating ducts, or an active laundry dryer. Older sensors are also known to stop sensing or transmitting after extended use. In the worst case, the loss of data for one resident and the presence of false data for another may occur if a motion sensor not only falls off the wall but also loses its batteries in the process. If someone besides the technicians who deploy the sensors finds the sensor and places the batteries back in it, the X10 motion sensor will default to the A1 house/unit code and will send mis-configured signals if there is another resident with that house/unit code nearby. Two mechanisms to counteract the possibility of this worst case scenario are (1) the battery cover screw present in all newer model motion sensors and (2) the elimination of the A1 ON/OFF commands from all sensor network configurations.

2.2.3 UVA Bed Sensor

The pneumatic bed sensor can is diagramed in Figure 2.3. These sensors have pneumatic strips which are installed across a resident's bed if that is where they primarily sleep and along the back side of a resident's recliner if the chair is where they sleep. If the resident sleeps in both, they have a bed sensor installed in each. The bed sensor differs from the X10 motion sensors in that they do not check for motion using PIR sensing but instead use a pneumatic pressure strip to monitor various levels of three kinds of signals[32]. The pneumatic strip captures vibrations from movement, breathing, and the resident's ballistocardiogram, which is the mechanical effect of the heartbeat.

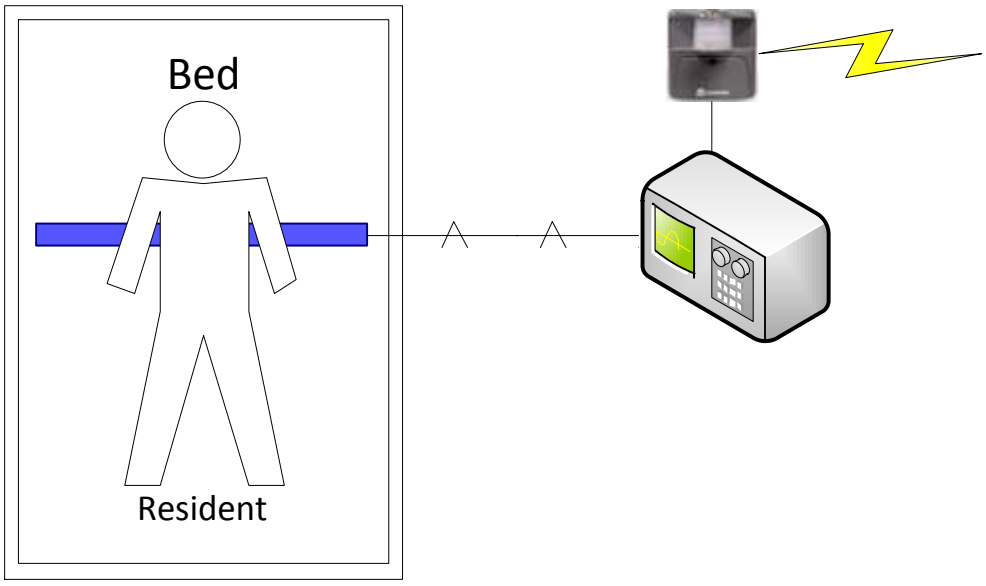

Figure 2.3 Eldertech Pneumatic Bed Sensor

The bed sensor system tracks four levels of bed restlessness (four is the highest level), three levels of bed breathing (low, normal, high), and three levels of bed pulse (low, normal, high). Level 1 bed restlessness indicates that the restlessness signal remains above the movement threshold for up to about three seconds. Levels 2, 3, and 4 correspond to the signal remaining above the threshold for up to about six, nine, and

more than nine seconds, respectively. Low bed breathing indicates detected breathing rates under six breaths per minute. High breathing rates indicate detected breathing rates above 30 breaths per minute. Normal breathing rates indicate detected breathing rates between 6 and 30 breaths per minute. Low pulse rates indicate detected pulses under 30 beats per minute (bpm). High pulse rates indicate detected pulses above 100 bpm and normal pulse rates indicate detected pulses between 30 and 100 bpm. The bed sensor measures pulse rates within ± 10 bpm.

The bed sensor box is connected to a modified X10 motion sensor to transmit the bed restlessness, breathing, and pulse signals through the X10 protocol. The difference between the bed sensor and the other sensors in terms of X10 is that address codes are hard-coded, i.e. they cannot be re-addressed, into the bed sensor and that the bed sensor uses ALL ON and ALL OFF commands as opposed to ON and OFF. The bed sensor does not follow the seven second interval between hits like the motion sensor and instead sends hits as they are detected.

The bed sensor is plugged in as opposed to the other sensors which run on batteries. Maintenance is required, however, whenever the pneumatic strip moves out of place, either by the resident or staff members flipping the mattress. The bed sensor detects respiration and heart beat signals best if the strip rests under the upper torso.

2.2.4 UVA Stove Temperature Sensor

The UVA stove temperature sensor uses a crisp threshold to send two different X10 signals. When the thermistor senses a high enough temperature, the sensor sends an X10 signal representing “Temp On” and continues to send the signal. When the thermistor senses that the temperature has fallen below the threshold, it sends an X10

signal representing “Temp Off”. The thermistor consumes battery power so often times the stove temperature sensor is the first sensor in the apartment that requires new batteries. X10 sensors in the apartment are usually replaced whenever the battery on the stove temperature sensor gets low.

2.3 Pattern Tree

Pattern Trees (PTs) and Pattern Tree Induction have recently been introduced as a new supervised machine learning method [34, 35]. Others [35] have presented alternative learning methods and work related to the Pattern Tree. Similar to a Fuzzy Decision Tree (FDT) [36], Pattern Trees are classifiers with a tree-like structure and make use of fuzzy operators [37].Pattern Trees can achieve a high rate of accuracy while maintaining transparency and providing intuitive interpretation. Figure 2.4shows an example of a Pattern Tree.

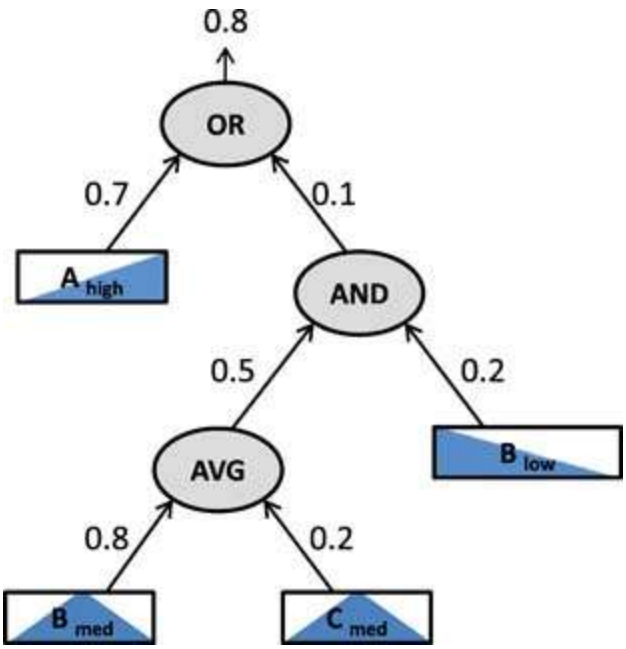

Figure 2.4 Pattern Tree Example (taken from [38])

To classify a sample, FDTs start at the root node and progress down based on expressions on individual features until it ends in one of the leaf nodes, which represents the classification of the given sample. Pattern trees, however, start with the features at the leaf nodes, which aggregate upwards using fuzzy operators until they reach the root node. Pattern Tree Induction methods offer a way to learn the Pattern Tree using training data. These methods, however, incorporate the use of averaging operators – mean, ordered-weighted-average(OWA) [39] – in addition to t-norms, or AND operators, and t-conorms, or OR operators. Using only t-norms and t-conorms, linguistic expressions like "kitchen motion is high AND stove temp is high" or "kitchen motion is high OR stove temp is HIGH" could intuitively describe a two feature Pattern Tree used to indicate if a person is cooking in the kitchen. However, with the addition of averaging operators, it becomes much harder to form an intuitive description of the Pattern Tree.

2.3.1 Leaf Node

Fuzzy Set Theory [37] provides the basis for the leaf nodes in the Pattern Tree. Given a linguistic variable X, and a linguistic term, A, represented by a fuzzy set, $\{(x, u_A(x))\}$ exists in [0,1], the amount that a value x in the universe of discourse X is A is represented by the membership value $u_A(x)$. Consider the linguistic variable X as "bathroom activity at night" and the linguistic term A as "high". Displayed in Figure 2.5, the number of sensor hits in the bathroom can be mapped to its membership in "high bathroom activity at night".

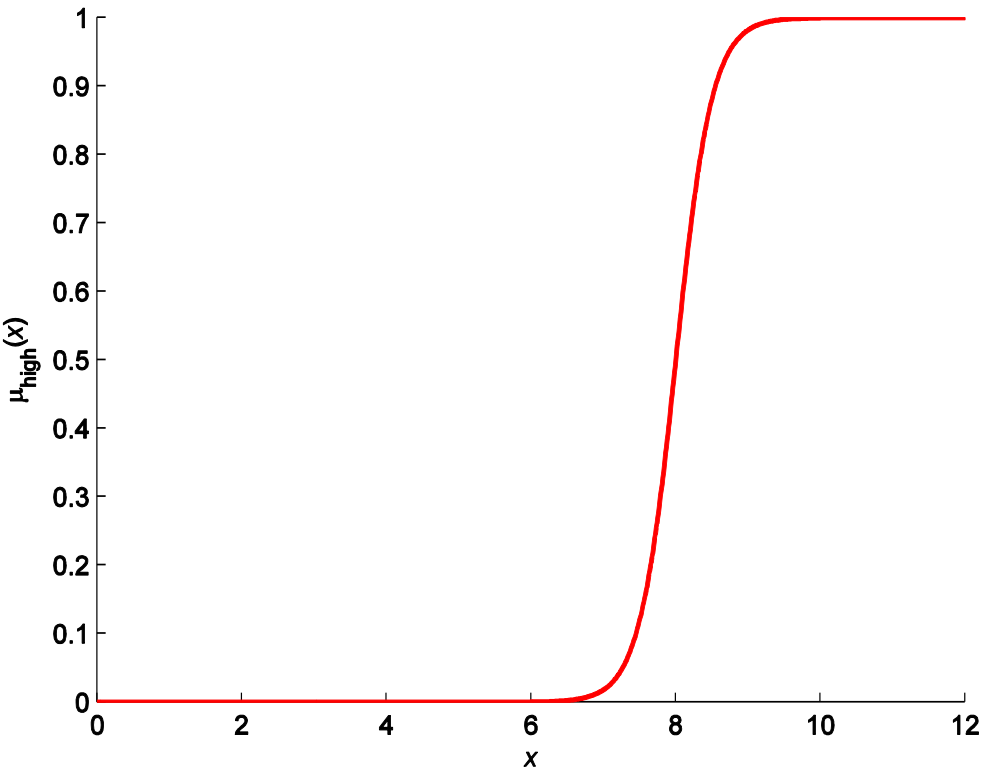

Figure 2.5 Linguistic Term "High"

2.3.2 Operator Node

Operator Nodes in the Pattern Tree aggregate their inputs together into one output, thus allowing the Pattern Tree to traverse upwards from the leaf nodes to the root node. The aggregation operator is a function, f: $[0,1]^2 \rightarrow [0,1]$. Operator nodes belong to generalized logical or arithmetic operators:

- t-norms (OR) and t-conorms (AND) or
- weighted and ordered weighted average (mean)

Table 2.1 and 2.2 list possible T-norms and T-conorms, respectively, with inputs $a, b \in [0,1]$ and output $c \in [0,1]$.

Table 2.1 Fuzzy Operators: T-norms

Name	Definition	Notes
Minimum	$\boldsymbol{c = \min(a, b)}$	Standard AND
Algebraic	$\boldsymbol{c = ab}$	Algebraic AND
Yager	$\boldsymbol{c = \max\left(0, 1 - ((1-a)^w + (1-b)^w)^{\frac{1}{w}}\right)}$	$\boldsymbol{0 < w < \infty}$

Table 2.2 Fuzzy Operators: T-conorms

Name	Definition	Notes
Maximum	$\boldsymbol{c = \max(a, b)}$	Standard OR
Algebraic	$\boldsymbol{c = a + b - ab}$	Algebraic OR
Yager	$\boldsymbol{c = \min\left(1, (a^w + b^w)^{\frac{1}{w}}\right)}$	$\boldsymbol{0 < w < \infty}$

Minimum and maximum are the standard operators for logical AND and OR, respectively. The algebraic T-norm is more pessimistic than the minimum, i.e. $ab \leq \min(a, b)\ \forall a, b \in [0,1]$, as shown in the contour plots of Figure 2.6. Similarly, the algebraic T-conorm is more optimistic than the standard maximum, i.e. $a + b - ab \geq \max(a, b)\ \forall a, b \in [0,1]$, as shown in the contour plots of figure 2.8

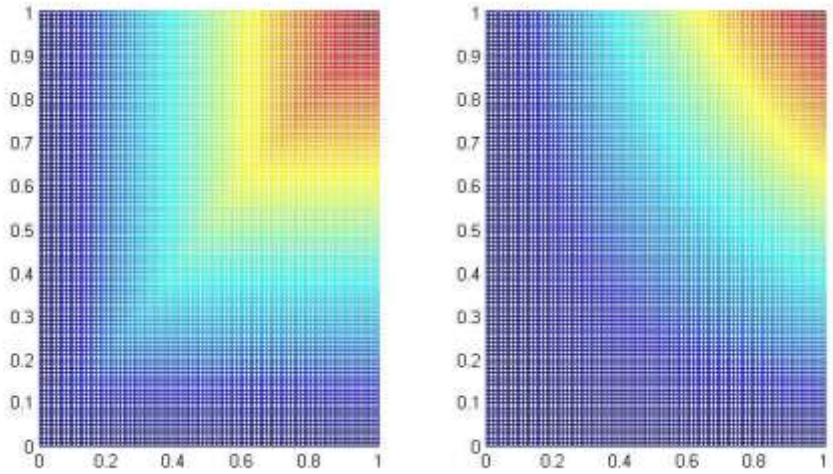

Figure 2.6 (a) Minimum, (b)Algebraic AND

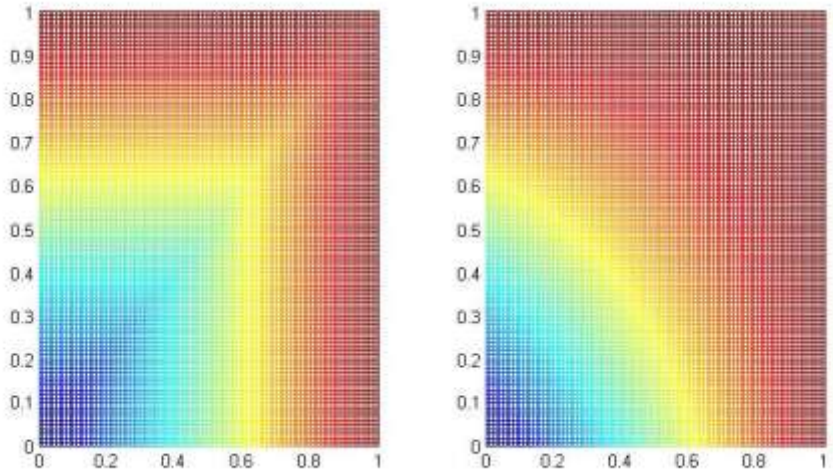

Figure 2.7 (a) Maximum, (b) Algebraic OR

The Yager operators [40] are useful because the parameter *w* defines the degree of pessimism or optimism of the T-norm or T-conorm, respectively. In other words, AND nodes can give less value to the output and OR nodes can give more value. Figure 2.9 illustrates the effect of *w* on the contour of the Yager OR.

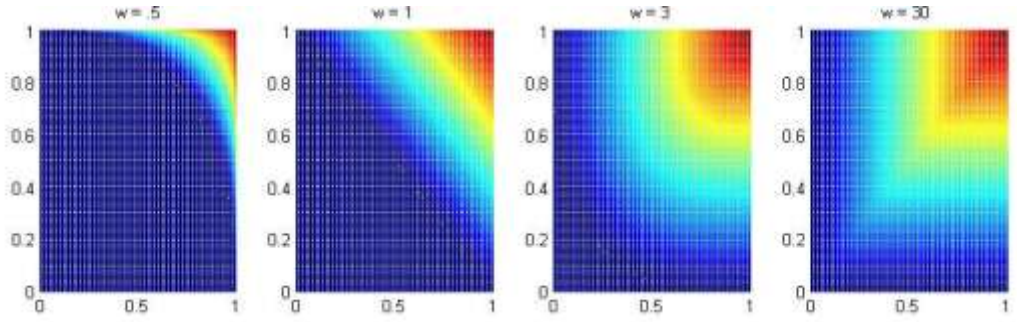

Figure 2.8 Yager T-conorm. w = 0.5, 1, 3, 30

2.3.3 Classification

After using the fuzzy operators to aggregate up the tree, the output of the root node represents a degree or confidence in [0,1] that the given feature belongs to a particular class. In a one-class or two-class classification problem, a single Pattern Tree representing one of the two classes will suffice. Samples with a Pattern Tree output greater than a specified threshold classify as class one and samples with outputs below the threshold classify as class two. In a multi-class classification problem, a set of Pattern Trees is required for classification, one Pattern Tree for each class. Each Pattern Tree provides an output for a given sample. The sample belongs to the class with the highest Pattern Tree output.

2.4 Fuzzy K-Nearest-Neighbor

K-Nearest-Neighbor (kNN) [41] is a classification method based on locating close training examples in the feature space. kNN classification is instance-based or lazy learning, where training samples are stored in memory until classification. Figure 2.10 shows an example of kNN classification, where the algorithm finds the nearest *k* examples, based on a distance measure, and classifies the unknown sample as the class

containing the majority of the *k* examples. Possible distance measures include the Euclidean and Mahalanobis distance [42].

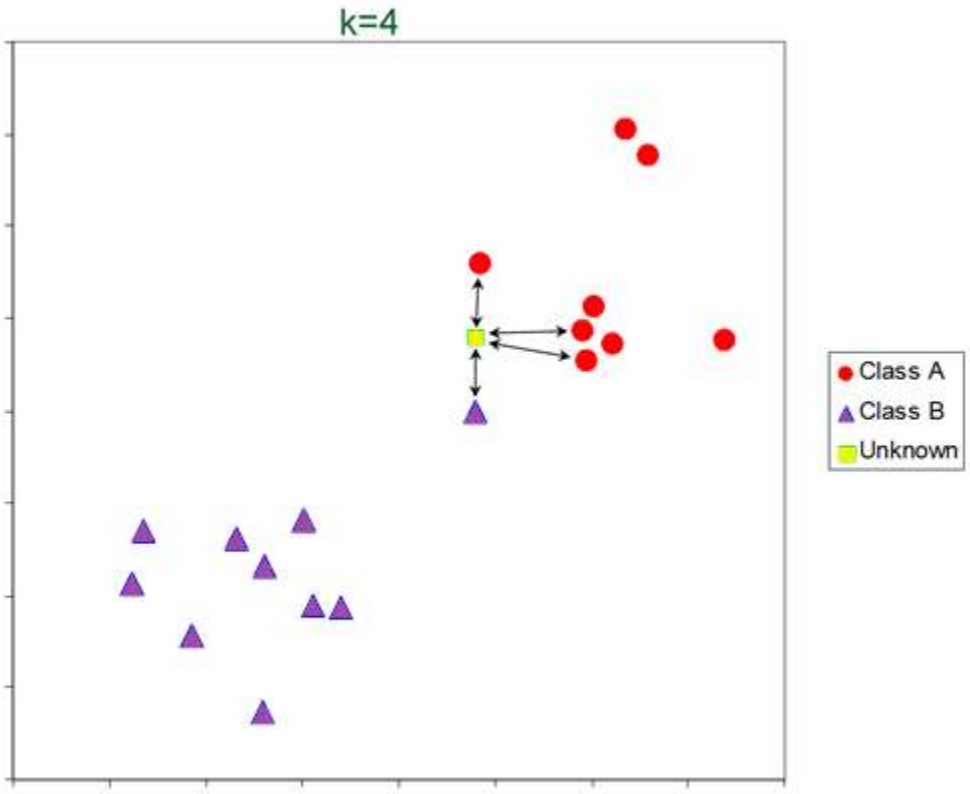

Figure 2.9 kNN Classification

Fuzzy K-Nearest-Neighbor[43] classification is an extension of the crisp classification, (full memberships in a single class) in the kNN to soft classifications (partial memberships in multiple classes). Partial membership in a class gives the degree to which a sample belongs to that class. For instance, whereas the crisp kNN would give a sample x a membership of 1 in class one and 0 in all other classes, the fuzzy kNN could give the same x a membership of 0.9 in class one, 0.05 in class two, and 0.05 in class three. In this situation it would make sense to classify x as class 1. However, consider the situation that the fuzzy kNN gives partial memberships of 0.55, 0.42, and 0.03 for a given

sample. In this situation, it is not so clear that the sample belongs to class 1, and further investigation may be required. The fuzzy kNN is capable of giving this information, while the crisp kNN is not. The algorithm for the fuzzy kNN is as follows.

Algorithm: Fuzzy K-Nearest Neighbor

BEGIN

Input x, an unknown sample
Set $K, 1 \leq K \leq n$ $(No. of\ training\ samples)$
Initialize i = 1
DO UNTIL (K-nearest neighbors to x found)
Compute distance from x to x_i
IF ($i \leq K$)
Include x_i in the set of K-nearest neighbors
ELSE IF (x_i closer to x than any previous nearest neighbor)
Delete the farthest of the K-nearest neighbors
Include x_i in the set of K-nearest neighbors
END IF
END DO UNTIL
Initialize i = 1
DO UNTIL (x assigned membership in all classes)
**Compute $u_i(x)$ using (#)
Increment i
END DO UNTIL
END

$$** \ u_i(x) = \frac{\sum_{j=1}^{K} u_{ij}\left(\left||x-x_j|\right|^{\frac{-2}{m-1}}\right)}{\sum_{j=1}^{K}\left(\left||x-x_j|\right|^{\frac{-2}{m-1}}\right)}$$

2.5 Neural Network

A Neural Network (NN) [44] is a pattern recognition tool inspired by the structure and/or functional aspects of biological neural networks. NNs are typically defined by three parameters:

1. The connection pattern between layers of neurons
2. The learning process to update weights of the connections

3. The activation function that feeds the weighted connections forward to the output

An example Neural Network is shown in figure 2.11. The NN consists of three input nodes, one hidden layer with four hidden nodes, and two output nodes. At each hidden node and output node, the output of all nodes in the previous layer are weighted by their connections to the node, summed together, put through the activation function of the node. Several common activation functions include the hyperbolic tangent and the sigmoid functions.

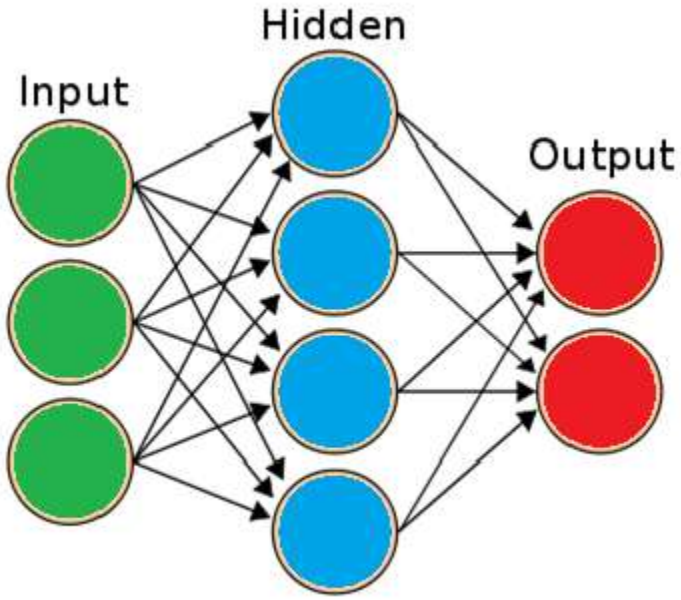

Figure 2.10 Neural Network Structure

Several supervised learning methods for updating the Neural Network weights from training data have been developed and include scaled gradient back propagation [45]. This method uses the mean squared-error (MSE) between all outputs of each training sample and the training samples target class and then uses gradient descent to move towards a better local optimum and hopefully towards a global optimum.

2.6 Support Vector Machine

The Support Vector Machine (SVM) [46] is a machine learning algorithm that classifies supervised data by building a maximum-margin hyperplane between either the feature space of the data (linear boundary) or the kernel-space of features put through a kernel function (non-linear boundary). Examples of some linear and non-linear SVMs are shown in figure 2.12.

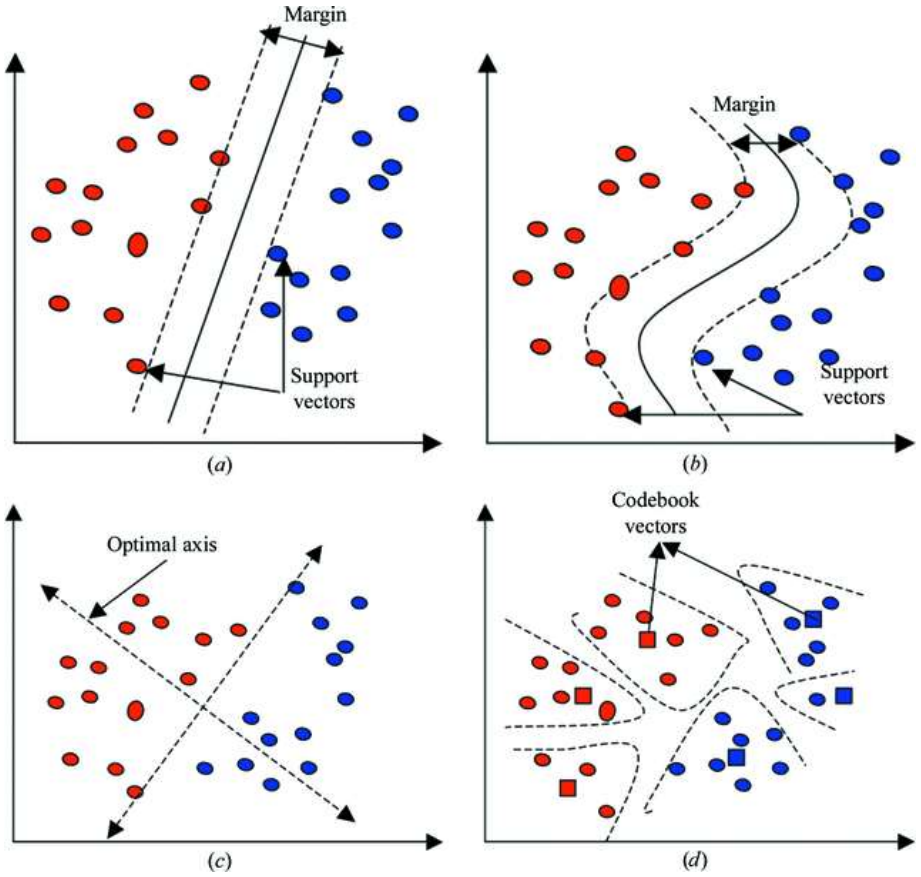

Figure 2.11 Example Support Vector Machines (taken from [47])

To build a decision boundary, Support Vector Machines start with either the primal or dual form of the Lagrange multiplier-based optimization problem, and then use quadratic programming to solve for the Lagrange multipliers. The values of the multipliers reveal which feature vectors act as the support vectors, i.e. the samples that

define the decision boundary or boundaries. SVMs have become a standard tool for comparing new supervised learning methods.

Chapter 3 - Methodology

Chapter 3 covers the design and implementation of the Early Illness Alert research, from the detection of passive infrared sensors to the proactive prevention of health declines for elderly residents in a home environment, as well as every process in between. Section 3.1 defines important terms that will be used in remaining chapters and provides details on the Eldertech team's integrated system of hardware (sensor networks, PCs, servers) and software (CERTLogger, MySQL databases, web interface) and how the individual pieces fit together to generate alerts from passive sensors and capture alert feedback from clinical researchers. Section 3.2 explains the methodology behind the Early Illness Alert email algorithm and how it calculates and detects change to generate single-dimensional alerts. Section 3.3 discusses the way in which feedback for these single-dimensional alerts is aggregated to define a basis for clinical relevance, i.e. good versus poor alerts, which provides a ground truth dataset applicable to supervised learning techniques and classification. Section 3.4 outlines how four multi-dimensional classifiers can use the single-dimensional features combined with the ground truth dataset to more than double the predictive power of the single-dimensional early illness alerts.

3.1 System Overview

In a collaborative effort to advance research in the field of eldercare technology, specifically in the development of novel early illness alerts, the Eldertech team uses an integrated system of sensor networks, data loggers, database servers, email alerts, a web portal, and feedback as illustrated in figure 3.1. Human activity data comes in through passive sensors, which send the data to the logger, which logs the data to the server,

which stores data for viewing in the web interface and generating future alerts, which notify clinical researchers, who leave feedback through the web interface, which sends the feedback back to the server for further analysis. The system operates around the clock, allowing simultaneous collection, viewing, and reviewing of sensor and alert data.

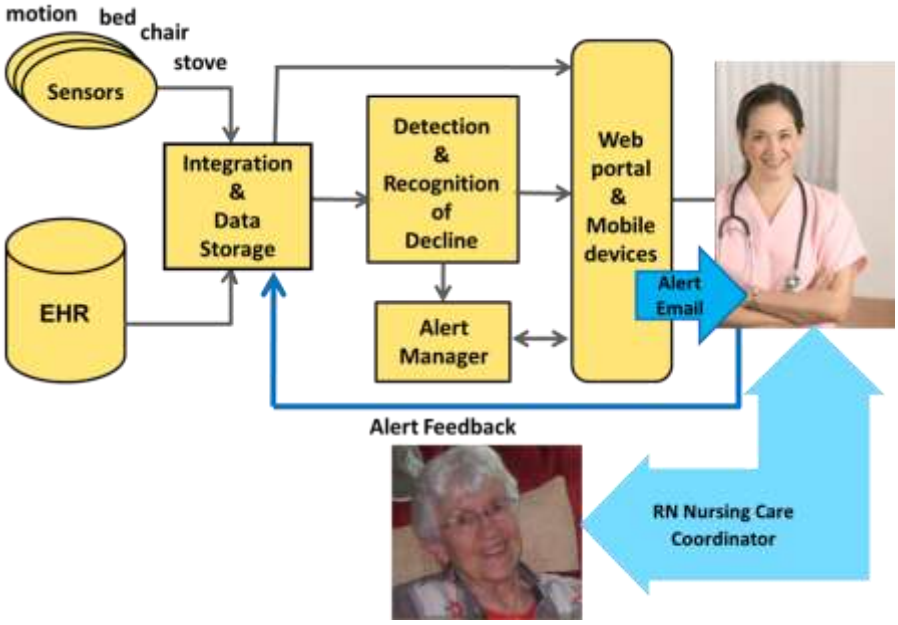

Figure 3.1 Integrated System

Between 20-25 sensor networks (one per apartment) exist at any one time, and they are setup and taken down periodically as residents move in and out of the apartments. A PC with the CERTLogger software accompanies each sensor network and all of the PCs connect and upload data to a main server at TigerPlace. The server transfers the data to a remote server, which also houses the web server running the web interface.

There are several key terms used for the remaining chapters that should be distinguished. The term *user ID* designates a unique identification number to a sensor

network that has been setup in an apartment at TigerPlace. It does not refer to the user of logging software and is not necessarily associated with a single resident (one such sensor network accounts for an elderly couple living in the same apartment). The *sensor type* defines the type of sensor sending an X10 signal, e.g. "Motion" refers to an X10 motion sensor and "BedMovement1" refers to bed restlessness level 1 from the bed sensor. An *X10 signal* represents the wireless signal sent by an X10 sensor and is a combination of an X10 house code (A-P), X10 unit code (1-16), and X10 command (ON, OFF, ALL ON, ALL OFF). For a single user ID, each X10 signal within that sensor network is mapped using a *sensor ID*. *Sensor location* is self-explanatory. A *sensor event* is the unique firing of an X10 signal at a specific time. Given the appropriate sensor network mappings, a sensor event logged in the database can be traced back to a unique physical sensor.

The term *Early Illness Alert (EIA) (single-dimensional)* refers to a notification the CERTLogger software emails to nursing clinicians when the Early Illness Alert Algorithm has detected a significant change in a sensor network's alert parameter. An *alert parameter* represents either a single X10 signal or a group of X10 signals and how they are aggregated together on a daily basis (the Early Illness Alert Algorithm also has a set of algorithm parameters that should not be confused with alert parameters). An *alert period* defines the time period each day for which to calculate an alert parameter. An *alert type* refers to the kind of change the Early Illness Algorithm detects, i.e. increase or decrease. To tie the alert terms together, consider an EIA that notifies the nursing clinician that User ID 3000 has an increase in nighttime bathroom activity. The alert parameter is bathroom activity, the alert period is nighttime, and the alert type is increase. A *single-dimensional EIA feature* is simply an EIA with some quantity representing the

amount of that alert type, e.g. by how much was there an increase. These single-dimensional EIA features can be combined into a multi-dimensional EIA by replacing the Early Illness Alert Algorithm with a multi-dimensional classifier.

3.1.1 Eldertech Sensor Networks at TigerPlace

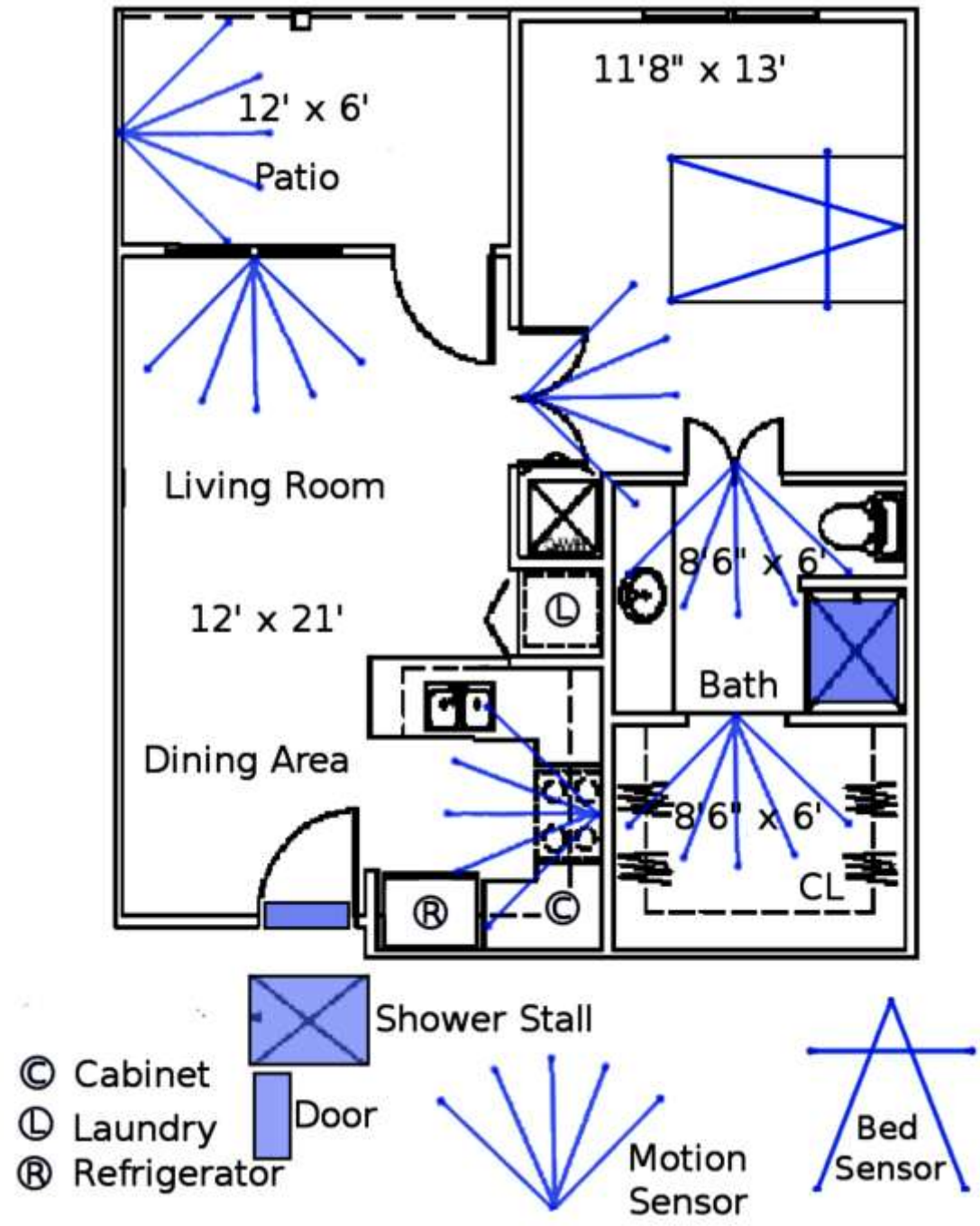

Figure 3.2 Sensor Network Layout

An example sensor network layout is shown in figure 3.2. These sensor network layouts are created for every apartment with a sensor network at TigerPlace and are

stored in the sensor database. Members of the Eldertech team can view the layouts online through the sensor web interface. The motion sensor cones designate the approximate location and orientation of the X10 motion sensors. The shower stall, front door, and washer and dryer (L) usually house an X10 motion sensor. The sensors for the shower stall and front door face downwards and the sensor for the washer and dryer is positioned beside but not facing outwards.

Typically, TigerPlace sensor networks consist of a combination of X10 motion sensors, bed sensors, and stove temperature sensors, which are described in Chapter 2. The exact number and location of each type of sensor usually depends on the apartment type and any living conditions designated by the resident(s). For instance, some apartments contain multiple bathrooms and hence multiple bathroom motion sensors. Apartments that have a washer and dryer may house the appliances in the bathroom, while others house the appliances in a living room closet, so the laundry motion sensor would have different locations in each sensor network. Some residents sleep in reclining chairs as opposed to a bed, so their bed sensor is setup in the chair as opposed to a bed. Several of the apartments house an elderly couple, so their sensor network has two bed sensors located in the same bed on their designated side.

The sensor network layout gives information on where and what type of sensors belong to which apartments. Thus, the sensor network information is stored across multiple tables in a MySQL database to define a group of entities and their relationships, which will be discussed further in Section 3.1.3. In short, the senor network mapping in table 3.1 is how the sensor network layout is stored in the database so software from other parts of the integrated system can access and use that information. Each row in the

mapping contains a unique signal that the sensor network could send to the logger. For each row which has a sensor type “Motion” each unique sensor ID designates a unique X10 motion sensor and the sensor location defines where it is located. The bed sensor consists of the rows whose sensor type contains either “Pulse”, “BedMovement”, or “Breathing”. The stove temperature sensor consists of the rows with sensor type “TempHigh” and “TempLow”. Sensor IDs 31-34 may be confusing because their sensor type actually designates the specific location of an X10 motion sensor in the kitchen. It may be wise to revise these confusing mappings in the database. The sensor types “Bed Motion” and “ExitFall” were originally designed by UVA and are not used by the Eldertech team’s software.

Table 3.1 Sensor Network Mapping

Sensor ID	X10 Signal	Sensor Type	Sensor Location
1	B1 on	Motion	Front Door
2	A2 on	Motion	Office
3	A3 on	Motion	Living Room
5	A4 on	Motion	Den
6	A5 on	Motion	Bathroom
7	A6 on	Motion	Shower
8	A7 on	Motion	Closet
10	D1 ALL ON	Bed Motion	Bed
11	D1 ALL OFF	ExitFall	Bed
12	D2 ALL ON	Pulse1	Bed
13	D2 ALL OFF	Pulse2	Bed
14	D3 ALL ON	Pulse3	Bed
15	D3 ALL OFF	BedMovement1	Bed
16	D4 ALL ON	BedMovement2	Bed
17	D4 ALL OFF	BedMovement3	Bed
18	D5 ALL ON	BedMovement4	Bed
19	D5 ALL OFF	Breathing1	Bed
20	D6 ALL ON	Breathing2	Bed
21	D6 ALL OFF	Breathing3	Bed
28	A12 ON	Motion	Kitchen
29	A13 OFF	TempHigh	Kitchen
30	A13 ON	TempLow	Kitchen
31	A8 ON	Cup Cabinet	Kitchen
32	A9 ON	Plate Cabinet	Kitchen
33	A10 ON	Silverware Drawer	Kitchen
34	A11 ON	Refrigerator	Kitchen
35	A14 on	Off Chair	Office
36	A14 off	On Chair	Office
37	A15 ON	Motion	Laundry
38	A16 ON	Motion	Bedroom

3.1.2 Eldertech Logger

The Eldertech logger, illustrated in figure 3.3, is an Asus EEE PC that runs the Eldertech CERTLogger © software in the background to collect and store data from the sensor networks.

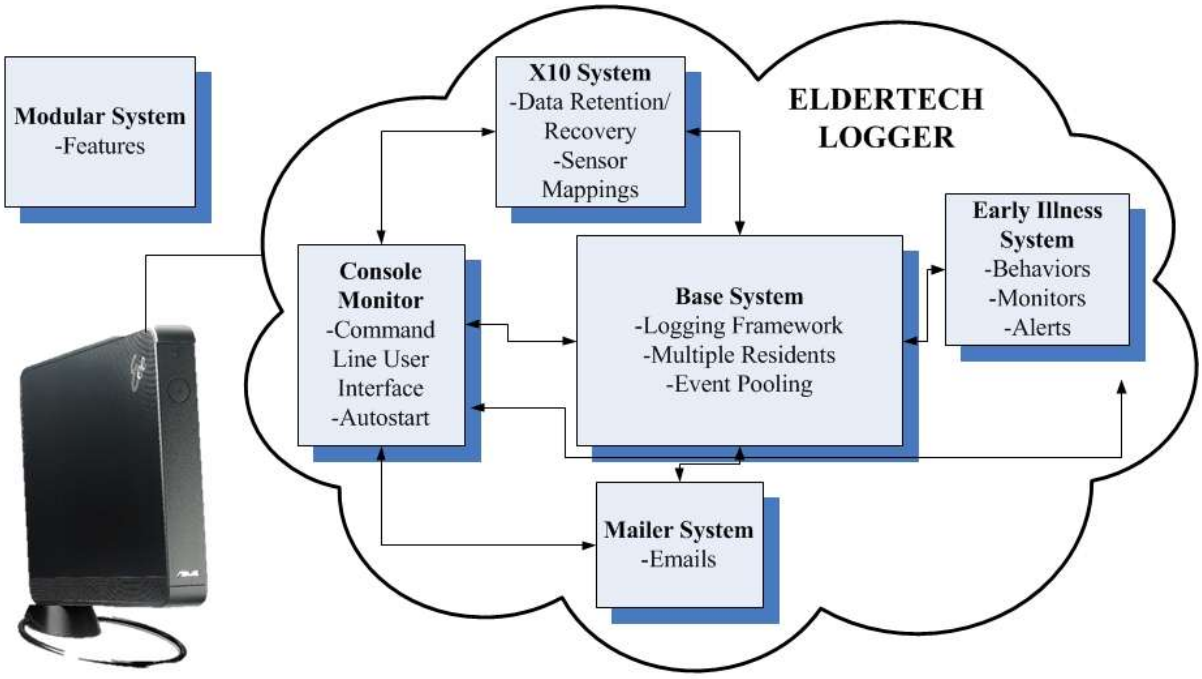

Figure 3.3 Eldertech Logger

The source code for CERTLogger is written in Java (the latest version is Java 6) and builds upon the framework originally developed by Huggard [48]. The software design is modular, allowing the possibility to develop and integrate new systems and features if needed in the future. As indicated in figure 3.3, five modules exist in CERTLogger: a Console Monitor and four Systems. Conceptually, the Systems are independent from each other but collectively make up the functionality of CERTLogger. Each System is an object that holds the current state, or configuration, of that part of CERTLogger. A user can modify the configuration of each System, and hence the overall function of CERTLogger, through a Control that is associated with each System. The different parts of CERTLogger were developed using this System-Control relationship so that future developers could program new CERTLogger features by first defining a System which holds a new set of configurations as well as the functions to perform new tasks, while separating the Control which holds the input commands a user can use to tell

the new System to perform those tasks. The developers could then simply add these new Systems and Controls to the list of Systems and Controls CERTLogger loads at runtime. In the immediate future, the Eldertech Team will require CERTLogger to have the capability of interacting with new sensors embedded with the ZigBee protocol [49].

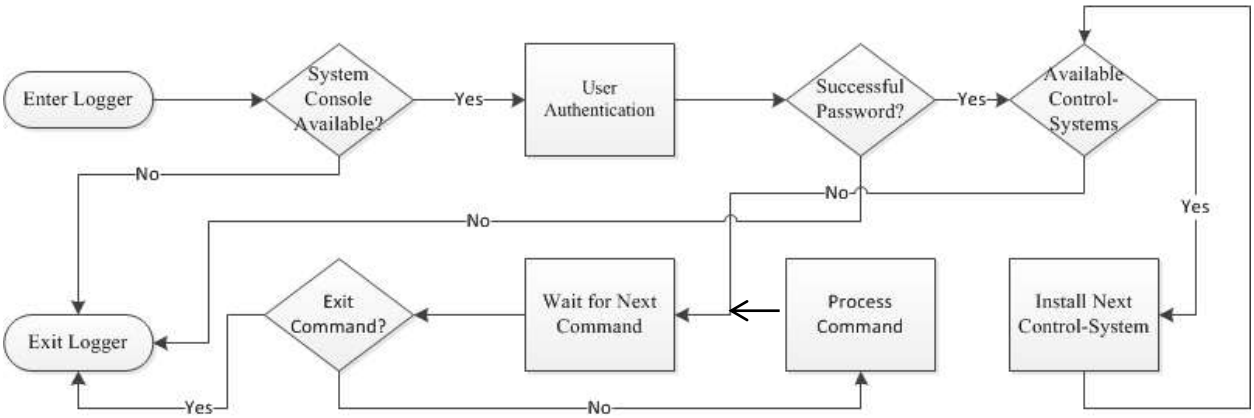

Figure 3.4 Logger Start Process

The entry point of CERTLogger is the command line of the operating system on which it is installed. Java programs run regardless of the operating system as long as the correct Java runtime environment exists on the operating system, so CERTLogger can be deployed on Windows or Linux OS. Once the java command to start CERTLogger executes, the program's main thread sets up the logger environment according to the process flow chart shown in figure 3.4.

First the main thread checks to see whether or not the executing operating system is capable of loading the java console. If possible, the program launches the Console Monitor system, which asks the user for an authentication password. If the user enters the correct password, the main thread will continue to install any configured (hardcoded) Controls. When installed, the Control also sets up the associated System. Once all

Controls and associated Systems set up (the system allows them to run on their own threads if needed), the main thread acknowledges the user and awaits commands to execute. The main thread delegates user commands to the correct Control, which parses the command and executes it if possible. If the main thread recognizes an exit command, the Controls are uninstalled one by one, saving any System configurations if necessary, and finally the program terminates.

The Command Line User Interface (CLI) allows a user to interact with the CERTLogger software, more specifically the individual Systems. The Eldertech logger PCs run the Fedora Linux OS, which makes the command line convenient to remotely connect to the PCs and manage the loggers remotely with just an internet connection and a secure shell client. The CLI passes user commands to the corresponding Control, which parses commands similar to that of the typical Linux command line environment.

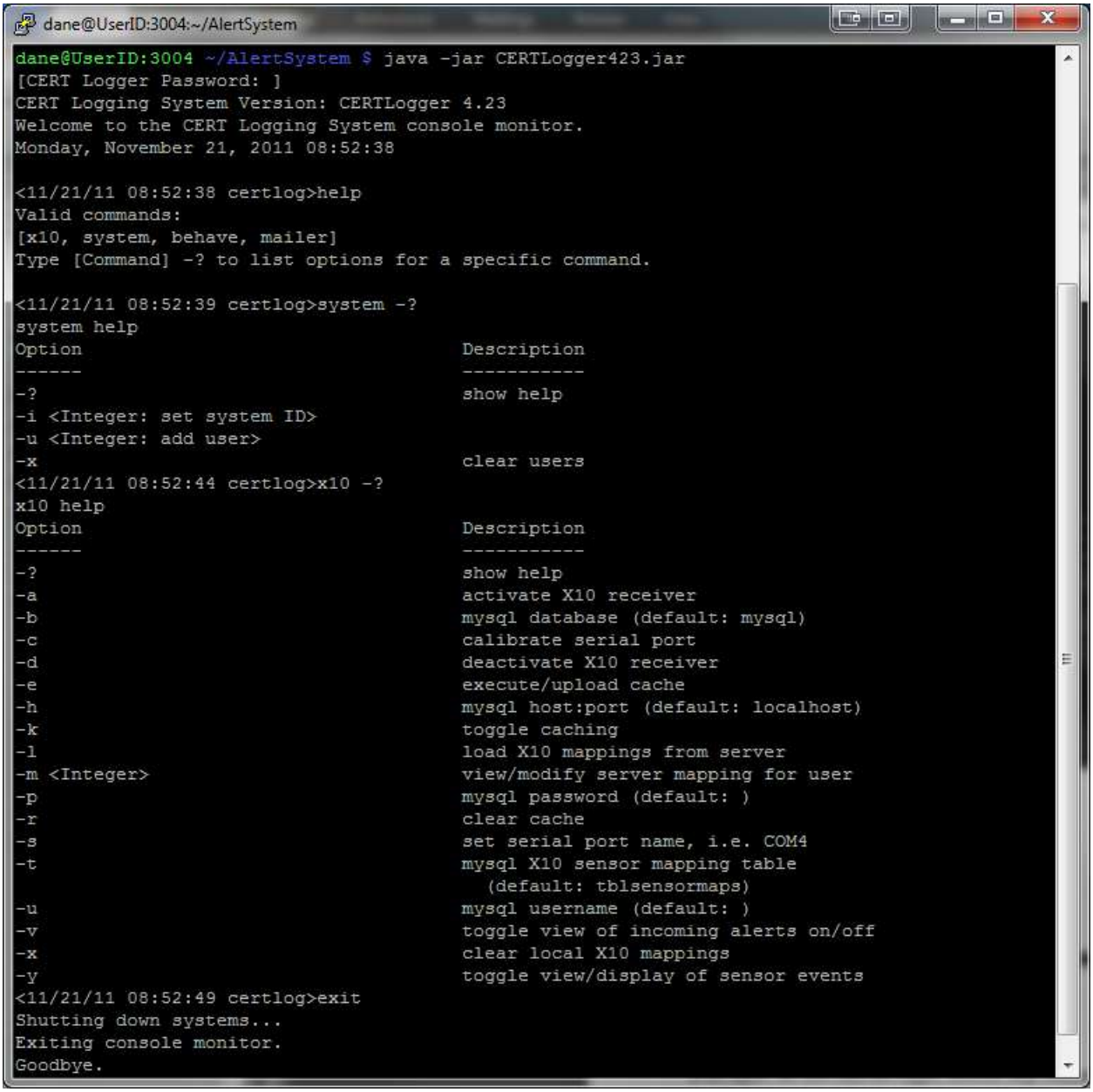

Figure 3.5 Eldertech Logger CLI

The typical CERTLogger CLI environment is shown in figure 3.5, in which the user starts the logger, executes help commands, and exits the system. The parsing libraries used in the Controls not only allow developers to incorporate new commands, but the libraries also have a methodical way of incorporating a help file for new

commands. A manual of commands for each of the existing Controls for CERTLogger can be found in Appendix A.

When a user launches the logger from the shell of the Linux command line environment, he also has the option of sending in commands for the logger to execute when it starts. This becomes useful when Eldertech logger is also setup with a Linux script to automatically launch CERTLogger when the PC turns on. Whenever the power goes off or for whatever reason, CERTLogger will start up and perform any specified tasks when the PC restarts. For instance, CERTLogger can immediately download sensor network mappings and start receiving and storing sensor data without manual startup by a user.

The Base System is the building block System of CERTLogger and is the only completely independent System module. All other Systems require an existing Base System to which they can connect. It is in charge of loading and piecing together modules from the original framework. It holds a unique system ID as well as user IDs for each of the sensor networks with which it is associated. In other words, the Base System holds the identity of the Eldertech logger, in case multiple Eldertech loggers need to interact with one another in the future. Base System tasks set the logger system ID and the addition or deletion of User IDs. On shutdown, the main thread saves the system ID and user IDs in the Base System configuration file.

With a Base System in place, an X10 System loaded on startup can connect and access the CERTLogger's user ID information. With the user ID information, the X10 System can retrieve and store the sensor network mappings for each user ID. The CERTLogger can monitor multiple sensor networks if necessary. Once sensor network

mappings are downloaded, the user can command the X10 System to configure the MySQL storage database and the port location of the X10 receiver, which is connected to the PC via serial port (in this case it was serial to serial-to-USB to USB) as shown in figure 3.6.

Figure 3.6 X10 Receiver Connection

Once the mappings, the storage database, and the X10 port are configured, the user can command the X10 System to activate the X10 receiver. The X10 System creates a new thread of execution. This new thread, the receiver thread, decodes all incoming X10 signals using the following algorithm.

Algorithm: Decode Raw X10 Signals

BEGIN (Triggered by serial port event in Java serial event listener thread)

Given last encoded stream $L = \{L_1, L_2, L_3, L_4\}$
Check encoded byte stream $Q = \{Q_1, Q_2, \dots, Q_n\}$
IF ($n \geq 4$) Decode next sensor event
$B = \{Q_1, Q_2, Q_3, Q_4\}$ (Get first four bytes from encoded byte stream)
$Q = \{Q_5, \dots, Q_n\}$ (Remove first four bytes from encoded byte stream)
IF ($B_1 == (BinaryNot(B_2) \cap 0\text{xFF}) AND\ B_3 == (BinaryNot(B_4) \cap 0\text{xFF})$)
This is a valid four-byte sequence without errors
IF ($B == L$) This is a duplicate event (up to four duplicates)
RETURN
END IF
Decode sensor event
$E_1 = reverse(B_3)$
$E_2 = reverse(B_1)$
**House code $H = HOUSECODES(E_2 \cap 0\text{x}0\text{F})$
Unit code $U = bitshiftright2(0\text{x}00000020 \cap E_2)$
$\cup\ bitshiftleft1(0\text{x}00000002 \cap E_1)$
$\cup\ bitshiftright3(0\text{x}00000018 \cap E_1)$
$+1$
Command C
IF ($(E_1 \cap 1) > 0$
IF ($(E_1 \cap 4) > 0$)
$C = "ALL\ OFF"$
ELSE
$C = "ALL\ ON"$
END IF
ELSE
IF ($((E_1 \cap 4) > 0$)
$C = "OFF"$
ELSE
$C = "ON"$
END IF
END IF
END IF
END IF
Decoded X10 signal $\{H, U, C\}$
END
** $HOUSECODES(i)$ is defined in table 3.2

Table 3.2 X10 House Codes

i	HOUSECODE	*i*	HOUSECODE
1	M	9	N
2	E	10	F
3	C	11	D
4	J	12	L
5	O	13	P
6	G	14	H
7	A	15	B
8	I	16	J

The receiver thread adds a timestamp to a successfully decoded X10 signal to create a sensor event. Sensor events get passed to the Base System, where they get passed onto any sensor event listeners using the Observer-Listener Model [50]. The X10 System is a sensor event listener, so the sensor events eventually traverse back to the X10 System. Once received, the X10 system checks if the X10 signal of the sensor event matches any record in its sensor network mappings. If there is a match, the X10 System logs the timestamp of the sensor event with the appropriate user ID and sensor ID from its mappings to the preconfigured storage database. The X10 System also has a configuration that allows the local cache of Sensor Events. The default configuration is set to on. The process flow diagram for configuring the X10 System to receive and log X10 sensor events for elderly residents is illustrated in figure 3.7. On shutdown, the main thread saves the X10 serial port, the mapping database location (not the sensor network mappings), and the storage database location.

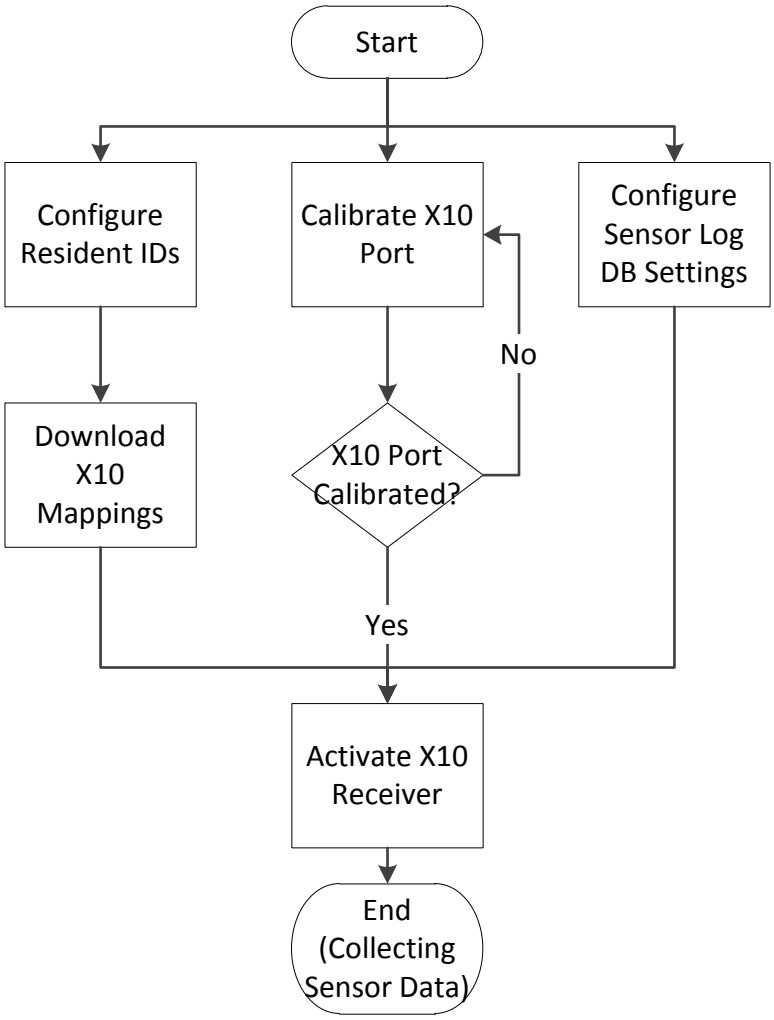

Figure 3.7 X10 Receiver Setup Process Flow

The Early Illness Alert System maintains configuration information for processing single-dimensional EIAs, discussed in detail in section 3.2. The Early Illness Control allows the user to load EIA configurations from a file. The System only loads EIA configurations for sensor networks defined in the Base System. The Control also allows the user to add or modify email monitors, or those that receive the alerts, for each of the user IDs in the Base System. When an alert is generated for a particular user ID, the system constructs an html email in a mail event. The list of email monitors for that user ID provides the email addresses to send an alert for that sensor network. Like sensor

events, the mail events get distributed via the Observer-Listener Model in the Base System to any mail event listeners. Lastly, the user can set up EIAs to process daily. Upon shutdown, the main thread saves the list of email monitors for each resident ID.

The Mailer System is a mail event listener that listens for mail and forwards the email to a simple mail transfer protocol (SMTP) server. The Early Illness Alert System could have been developed to simply send alert emails itself, however the design of the Mailer System as a separate module allows existing or future Systems the capability to send emails. For instance, the Eldertech team may decide to start sending emails whenever the logging of X10 data stops. A developer could then modify the X10 System to check for periods of no sensor events and pass mail events to the Mailer System accordingly. Also, new Systems may need the ability to send emails in the future. On shutdown, the main thread saves the STMP server configuration for the Mailer System.

The four Systems (Base, X10, Early Illness Alert, Mailer) along with the Console Monitor and Controls, allow for a modularized CERTLogger, maintaining the interoperability of the different Systems straightforward addition of new ones. The logger software is developed in Java, which implements its own garbage collection thread, helping to eliminate the problem of memory leaks. The Base System allows for a multi-resident logger as opposed to just a single-resident logger. The X10 System allows local caching of sensor data, providing a means of data retention and recovery. The Early Illness Alert System provides a means of processing sensor data from the logger and the Mailer System gives the software the capability of sending emails.

3.1.3 Sensor Database

The Eldertech sensor database stores mapping information for the TigerPlace sensor networks as well as sensor data collected from the sensor networks. The sensor database was originally designed in accordance with the UVA loggers [23]. There are five main entities that exist for the sensor configurations and sensor data: rooms, sensor types, location types, sensor maps, and sensor events. The entity relationship diagram in figure 3.8 illustrates the main attributes (columns) of the tables representing these entities and the relationship between them.

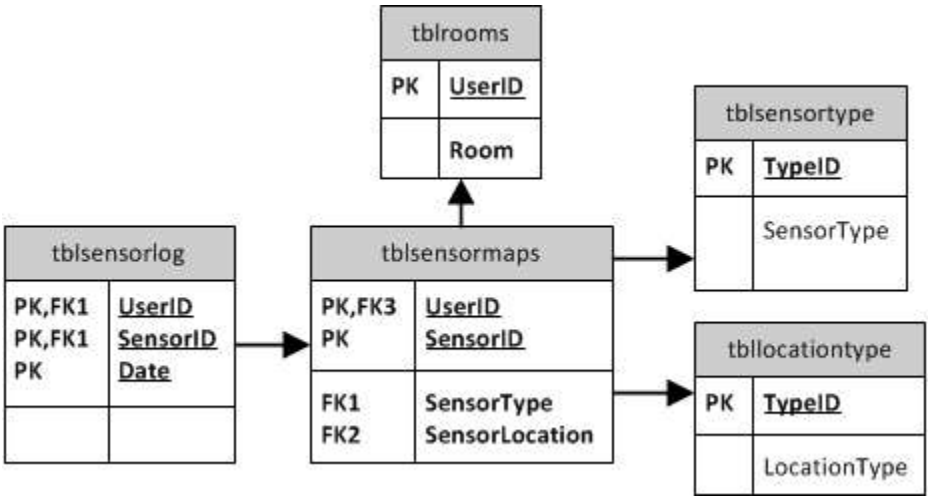

Figure 3.8 Sensor Database ER Diagram

An entry in tblrooms consists of a sensor network's user ID and the apartment number for that sensor network, e.g. sensor network 9999 in room 111. An entry in tblsensortype has a numeric type ID and the name of the sensor type, e.g. "Motion" or "BedMovement1". An entry in tbllocationtype, similar to a tblsensortype, has a numeric type ID and the name of the location, e.g. "Living Room" or "Bathroom". An entry in tblsensorlog represents a unique sensor event and is identified by its three fields: user ID, sensor ID, and date. The user ID combined with the sensor ID identifies a physical sensor

while the date field indicates the time at which the sensor fired. Tblsensorlog is by far the largest table in the sensor database, as it holds over six years of ongoing sensor data for over 40 unique residents. Entries in tblsensormaps tie the other four entities together, providing the sensor type name and location name of a single sensor for a single resident. Before a new sensor network is deployed, an entry for the new sensor network must be added to tblrooms and entries for each of the sensors in the sensor network must be configured in tblsensormaps. These mappings are kept in the central storage location along with the sensor data so they remain consistent and both the Eldertech loggers and the sensor web Interface can store and retrieve sensor data and sensor mappings to and from the same place.

3.1.4 Sensor Web Interface

The sensor web interface [51] operates as a web portal that can be used to access and visualize sensor data from the TigerPlace residents. As illustrated in figure 3.9, the sensor web interface allows a user to view apartment layouts, display activity and sensor maps, and plot sensor data for an individual over a specified timeframe.

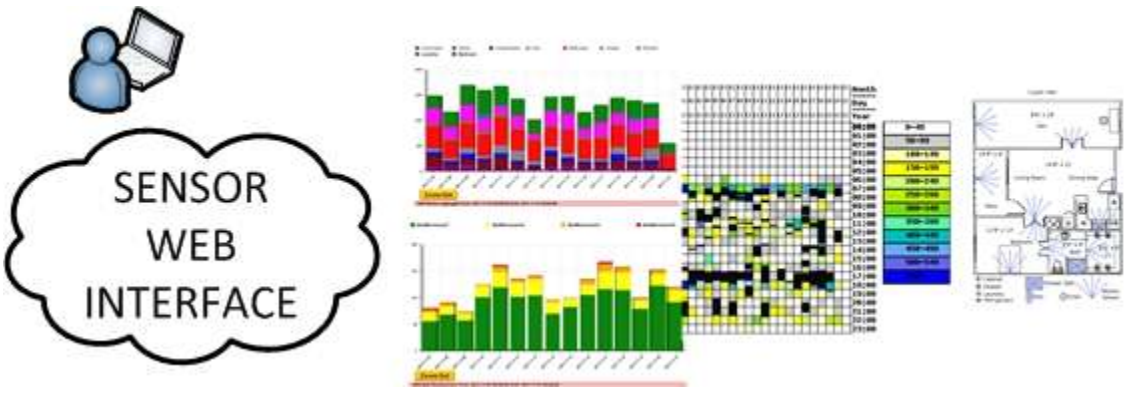

Figure 3.9 Sensor Web Interface

The bar plots include the number of sensor events for each motion sensor per day, the number of individual levels of bed restlessness events per day, the number of individual levels of breathing sensor events per day, and the number of individual levels of pulse sensor events per day. The web interface user also has the option to specify one of three periods of the day: midnight to midnight, midnight to 6 a.m., or 8 a.m. to 8 p.m., which correspond to the alert periods. Visualizations of more sophisticated algorithms including density maps [52], time out of the apartment [29], bathroom visits, and time in bed are also available through the interface. The web interface also integrates information from the alert database into the sensor data visualizations.

3.1.4 Alert Database, Feedback Loop, Feedback Database

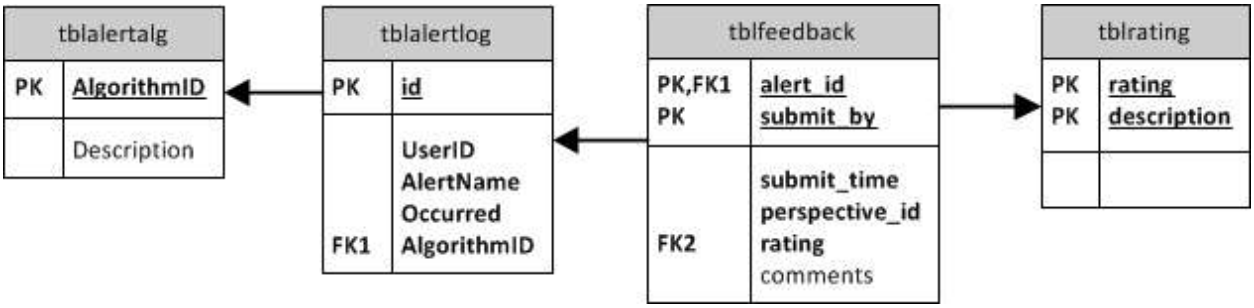

Figure 3.10 Alert Database ER Diagram

Similar to the sensor database, the alert database stores alert data from previously generated EIAs as well as clinical feedback data collected for those EIAs. As shown in figure 3.10, the database consists primarily of four entities: an alert event: an alert period, an alert feedback, and an alert rating.

When CERTLogger generates a single-dimensional EIA, it also stores information for that alert in the tblalertlog table. The alert id is a new and unique numeric identifier assigned to each EIA as they are generated. The user ID associated with that

alert indicates from which sensor network that EIA was generated. The alert name indicates the alert parameter that was previously defined in 3.1 and will be further discussed in section 3.2. One such alert name or alert parameter is “Bathroom Activity”. The occurred field indicates the date on which the logger generated an EIA. Entries in tblperiod define the alert periods for which the EIAs are generated. Table 3.3 lists the periods this research uses; they are discussed in further detail in Section 3.2.

Table 3.3 Alert Periods

Period ID	Description
1	Full Day (12 a.m. – 12 a.m.)
2	Nighttime (10 p.m. – 6 a.m.)
3	Nighttime (12 a.m. – 6 a.m.)
4	Daytime (8 a.m. – 8 p.m.)

The logger sends alert emails, such as the one in figure 3.11, to clinical researchers and TigerPlace care-providers. Two links per email are embedded within the email alerts. The first link takes the clinical researcher or care-provider to the sensor web interface, automatically pulling up the sensor data that generated that EIA. The second link is the feedback loop and allows the clinical researcher or care-provider to provide feedback data via the sensor web interface.

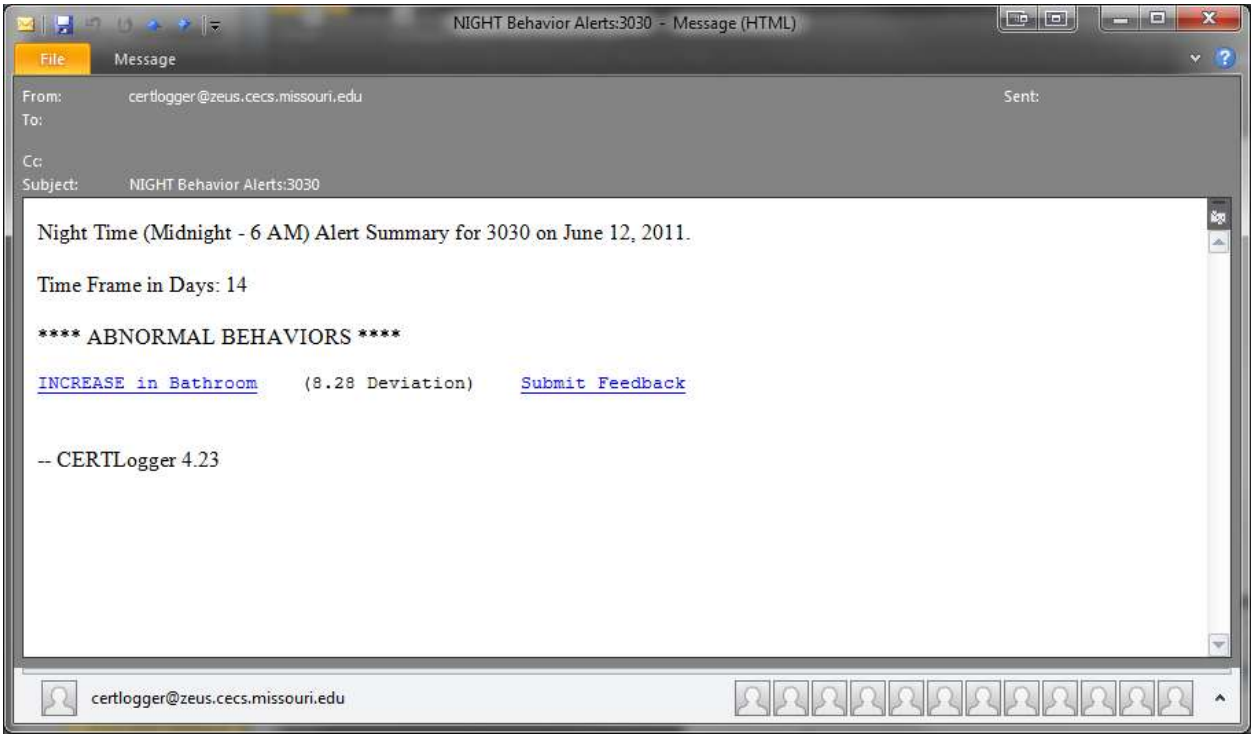

Figure 3.11 Example Email Alert

Figure 3.11 shows the feedback portion of the sensor web interface. The clinical researchers or care-providers submit a feedback rating as well as comments for the alert. The tblfeedback table holds the feedback entries, which consist of an alert ID field to indicate the alert with which the feedback is associated and a submit-by field to indicate the researcher or care-provider who left the feedback. The elderly residents in this study each have more than one clinical researcher or TigerPlace care-provider monitoring them; hence the same alert may have multiple feedback responses.

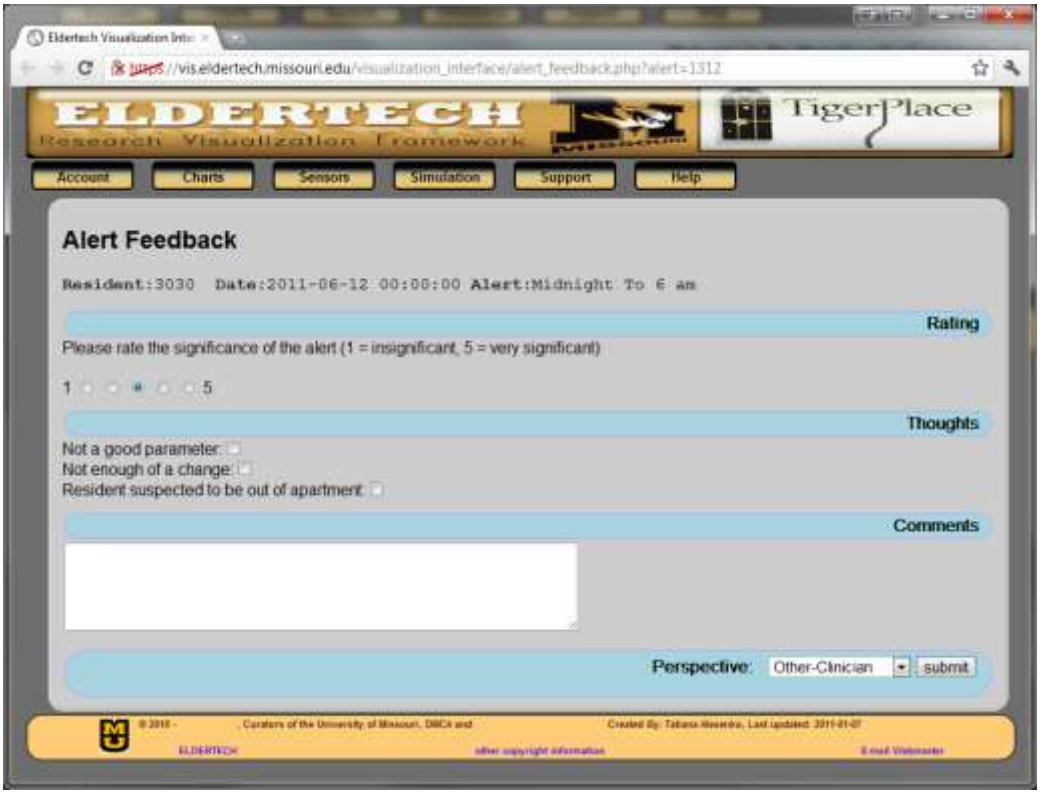

Figure 3.12 Feedback Loop Web Page

The submit time field is the timestamp for when a reviewer submits feedback. The perspective ID indicates whether the reviewer is a clinical researcher or a TigerPlace care-provider. The feedback rating indicates their expert rating on a 1-5 scale, described in table 3.4, and comments hold any feedback comments they leave behind. How this feedback is used for further analysis of more sophisticated EIA algorithms is discussed in further detail in section 3.3.

Table 3.4 Alert Feedback Ratings

Rating	Description
1	Clinically irrelevant
2	Less clinically relevant
3	Somewhat clinically relevant
4	Clinically relevant
5	Very clinically relevant

3.2 Detecting Early Illness

This section covers the techniques this research uses to detect and generate single-dimensional EIAs and the rationale behind them. When the design of the Early Illness Alert Algorithm first began, the main idea was whether or not patterns or changes in the sensor data could have indicated the onset of a critical health junction in an elderly resident's life at TigerPlace. There were already several documented incidents where the sensor data before a resident's hospitalization changed noticeably. How to detect this change in the sensor data became the key concept of the Early Illness Alert Algorithm. The initial idea was to take a continuous sliding window of sensor data and build a "normal" model for each of the initial alert parameters, and then determine when any of the sensor signals deviate greatly from their normal. This idea of modeling normal or how to take a sliding window or which alert parameters to define and how to calculate them were configured and reconfigured several times before finalizing an Early Illness Alert Algorithm that went into production. Even after the CERTLogger began generating and sending out real-time EIAs there were many more false alarms than hits, thus the design of the Early Illness Alert Algorithm became an Interdisciplinary Human-Centered

Iterative design process in which the Eldertech team met weekly to discuss EIA problems, propose solutions, and redesign the algorithm.

3.2.1 Single-Dimensional Early Illness Alert Algorithm

The Single-Dimensional Early Illness Alert Algorithm operates as follows: Given a sliding window L, calculate the alert parameter values $\rho(t_i, P)$ for a given alert period P for each day t_i in the sliding window. With at least S valid alert parameter values in the sliding window, calculate the sample mean and sample standard deviation of the valid values to model "normal" for a given alert parameter at a particular period in the day. If the alert parameter value calculated for the day after the sliding window deviates from the sample mean of the sliding window by enough sample standard deviations $m_{increase}$ or $m_{decrease}$of the window, generate either an increase or decrease EIA.

Algorithm: Single-Dimensional Early Illness Alert

BEGIN (Triggered at 12 AM and 6 AM daily)

Given the sliding window length in days, L
Given alert parameter function $\rho(t, P)$, where t is a date and P is an alert period
Given alert period, $P \in \{Full\ Day, Nighttime, Daytime\}$
Given valid alert parameter value threshold, T
Given the sample size threshold, S
Given deviation thresholds $m_{Increase}$ and $m_{Decrease}$
Alert parameter values, $\boldsymbol{\rho} = \{ \rho_1, \rho_2, \ldots, \rho_L\} = \{0, 0, \ldots, 0\}$
FOR (n = 1 to L)
**Calculate $\rho_n = \rho(t_n, P)$
END FOR
$\boldsymbol{V} = \{v \in \boldsymbol{\rho} \mid v > T\}$
IF ($Size(\boldsymbol{V}) \geq S$)
Sample mean $\mu = mean(\boldsymbol{V})$
Sample standard deviation $\sigma = std(\boldsymbol{V})$
CASE ($\rho(t_{n+1}, P) - \mu \geq \sigma \cdot m_{Increase}$)
Send Increase EIA
CASE $\mu - \rho(t_{n+1}, P) \geq \sigma \cdot m_{Decrease}$
Send Decrease EIA
END IF
END
** The alert parameter functions for each alert parameter are defined in table 3.5

The original EIAs are single-dimensional because each alert represents either an increase or a decrease in a single alert parameter and for a single alert period. There are currently 10 alert parameters (counting the chair sensor equivalent of the bed sensor) and 3 different alert periods for 2 different alert types (increase and decrease), resulting in 60 kinds of EIAs. The alert parameters and how they are calculated on a daily basis are listed in table 3.5.

Table 3.5 Alert Parameter Modeling

Alert Parameter	Calculation of the Parameter Value $\rho(t, P)$, given a day *t* and period *P*
Bathroom Activity	Sum sensor events from bathroom motion, shower, and laundry sensors
Bed Restlessness	$\boldsymbol{x_{bedrest}} = \sum_{i=1}^{4} \boldsymbol{i} * \boldsymbol{x_i}$, where x_i is the no. of bed restlessness level *i* hits
Bed Breathing Low	No. of bed breathing 1 sensor events
Bed Breathing Normal	No. of bed breathing 2 sensor events
Bed Breathing High	No. of bed breathing 3 sensor events
Bed Pulse Low	No. of bed pulse 1 sensor events
Bed Pulse Normal	No. of bed pulse 2 sensor events
Bed Pulse High	No. of bed pulse 3 sensor events
Kitchen Activity	Sum kitchen motion sensor (kitchen, fridge, etc.) events and TEMPON
Living Room Activity	No. of living room sensor events
Chair Sensor Activity	Equivalent to bed sensor behaviors above

The initial alert parameters consisted of combining sensors in similar areas of the apartment, aggregating the bed restlessness in a manner which puts higher weight on higher levels of restlessness, and keeping the multiple levels of bed breathing and bed pulse separate. Although it was apparent that the clinical researchers favored the significance of certain parameters over others, e.g. bathroom activity over bed breathing normal, all of the alert parameters were initially being generated to help identify which parameters made more sense over time.

In addition to multiple alert parameters, multiple alert periods were incorporated into the Early Illness Alert Algorithm to distinguish between different times of the day. Initially, only 24-hour periods were implemented but further discussion with the clinical researchers revealed that more valuable information may be found in many alert parameters during the night for several reasons. For one, the sensor data is more reliable because it is less likely to incorporate sensor events fired from visitors. Second, resident behaviors at night, especially when they are erratic or suddenly start occurring, can be picked up by the nightly alert parameters and usually indicate a need for intervention with the elderly resident. Generating alerts for the daytime parameter also helped to reveal how less informative daytime EIAs tended to be.

3.2.1.1 Modeling Normal and Detecting Change

The normal distribution is a very basic model and in many situations may not completely model "normal" behaviors in a senor network. Imagine an elderly resident who exhibits periodic behavior. For instance, he may do laundry every two weeks. A mixture of Gaussians model would actually model the behavior better. Regardless of its simplicity, however, the normal distribution model allows the Early Illness Alert Algorithm to do a decent job of indicating visibly significant increases and decrease in alert parameters. When the clinical researchers review the alerts with sensor data in a timeline view on the sensor web interface, they can identify right away where and why the alert generated, allowing them to focus more attention on whether the EIA is significant from a clinical perspective as opposed to whether there was indeed a significant change.

3.2.1.2 Sliding Baseline L

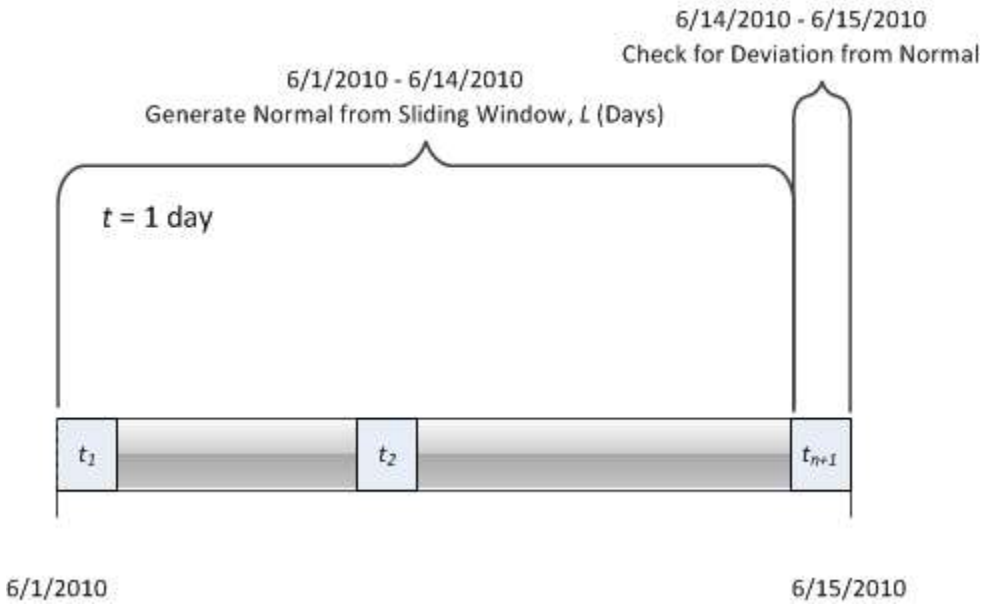

Figure 3.13 Early Illness Algorithm Sliding Window

In order to calculate either a mean or a standard deviation, the algorithm requires a set of training samples. Although it is possible to pick a static range of time to calculate the mean and standard deviation for each periodic behavior for each elderly resident, it is less practical. First, a suitable range of time must be hand-picked and may be tedious to find. Second, the range of time selected may represent a "normal" behavior pattern that is different from the resident's current normal, i.e. a resident's "normal" changes over time. For convenience in terms of implementation and to account for slower changes in a resident's normal behavior, the early illness algorithm uses a sliding baseline. The sliding baseline spans a time frame of two weeks, or 14 days.

3.2.1.3 Standard Deviation Threshold $m_{increase}$ and $m_{decrease}$

The standard deviation thresholds for increases (m_{increase}) and decreases (m_{decrease}) indicate the number of standard deviations an alert parameter value must be

outside of the sliding baseline mean before an increase or decrease alert is generated for that alert parameter. With a constant sliding baseline size, the standard deviation-based threshold has a direct effect on the number of EIAs that are generated. As a result, the clinical researchers require a standard deviation threshold that is low enough to generate early illness alerts in situations where they are desirable but high enough so that they are not overwhelmed with false alarms. A retrospective study analyzed the effect of the standard deviation threshold for residents during a time period where early illness alerts would have proved useful. The number of alerts as well as the time of alerts vs. the standard deviation threshold was analyzed, and the clinicians decided on a threshold that minimized the false alarms while maintaining the desired early illness alerts. A threshold of 4 standard deviations for both m_{increase} and $m_{\text{decrease}}$was chosen initially.

As the iterated user-centered design of the early illness algorithm progressed, so did the values of the m_{increase} and m_{decrease}. It became apparent that the clinicians were not receiving enough decrease alerts, i.e. periodic behaviors that decrease below 4 standard deviations of the mean, or bed sensor alerts, i.e. bed restlessness, breathing, and pulse alerts. As a result the threshold for all decrease periodic behaviors became 2.5 standard deviations and the threshold for increases in bed sensor periodic behaviors became 3 standard deviations.

3.2.1.4 Filtering Malfunctioning Sensors and Extended Absence T

One typical problem with real world sensor data is missing data. Batteries may fail. Sensors may fall off the walls or degenerate and cease to function. With the bed sensor, sometimes the pneumatic strip gets misplaced, the pneumatic strip needs replacement, or the resident sleeps in an awkward position that is not directly on top of

the transducer. Sometimes someone, possible staff or the resident, removes the bed sensor without notifying the Eldertech team. Another possibility is that the resident may simply be away from their apartment for an extended period of time. Missing data are bound to occur at TigerPlace. As a result, it is important to handle missing data accordingly, because not only can the missing data skew calculations, but it can also generate numerous false alarms from residents on vacation.

To filter out these events, a valid parameter value threshold T filters out periodic behavior values on days below that threshold from the sliding window. This filter may result in a training set that is less than the L days in the sliding baseline. If there are not at least S valid parameter values to calculate a normal distribution, or the sensor parameter value for the new day is less than T, then the algorithm will not generate an EIA. T was initially set to 0 to simply eliminate days where the periodic behaviors are absent. However, with the presence of TigerPlace staff there is a possibility of setting off a few sensor events while the resident is away. Thus, T was increased to 10.

3.2.1.5 Summary of Algorithm Changes

The Early Illness Alert Algorithm first began generating EIAs in March of 2010. At this time, the set of EIAs being generated consisted of two alert periods, full day (midnight to midnight) and nighttime (10 pm – 6 am). The sliding window was 14 days long, the valid parameter value was set to 0, the valid sample size threshold was set to 0, and the standard deviation threshold was set to 4. The first EIA to receive clinical feedback from the clinical researchers was recorded on April 1st on 2010.

Further discussions with the nursing clinicians brought about the need to include alert types to differentiate between increases and decreases in alert parameters. The

nursing clinicians also wanted to modify the nighttime period to begin at midnight and add a daytime period from 8 am to 8 pm. These changes were implemented in July of 2010.

After 6 months of reviewing EIAs, the nursing clinicians argued that receiving a full day EIA while also receiving either a nighttime or daytime EIA for the same user ID, alert parameter, and date was repetitive because often times a nighttime or daytime alert would imply a corresponding full day alert. Although not necessarily true in all cases, this repetitiveness occurred frequently enough to merit a change in November of 2010.

After a year of EIAs, there were considerably more alerts for increases generated than those for decreases. The Eldertech team decided that in order to build a more inclusive dataset, the decrease standard deviation threshold in the Early Illness Alert Algorithm should be lowered. The tradeoff is that the algorithm may generate more false alarms. The valid sample size threshold was increased at this point to make sure that at least 10 days-worth of alert parameter values were available before generating an EIA.

A few months later, the influx of more EIAs, specifically false alarms, overwhelmed the nursing clinicians and they decided it would be wise to raise the decrease standard deviation slightly. The increase standard deviation for EIAs with alert parameters coming from the bed sensor was also lowered to generate more bed restlessness, bed pulse, and bed breathing EIAs. The valid parameter value threshold was increased to 10 to help filter out noisy sensor data. A total of 254 days' worth of alerts that occurred between November 1st, 2010 and July 13th, 2011 were used in the analyses in the remaining chapters.

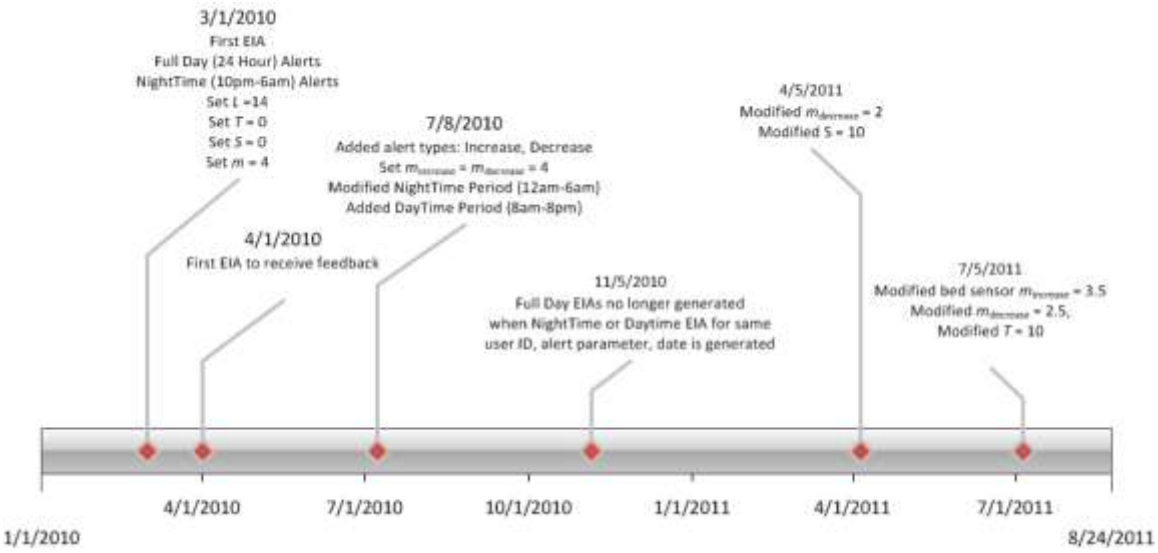

Figure 3.14 Early Illness Alert Algorithm Timeline

3.3 Clinically Relevant Early Illness Alerts

With the ability to continually and conveniently collect clinical feedback via automatically generated EIAs, the alert database becomes a valuable tool for further analysis of the Early Illness Alert algorithm. With quantitative ratings available, ground truth can be created to separate the alerts the nursing clinicians view as having clinical relevance and merit an intervention and those they view as false alarms, or simply good versus poor alerts.

Presented in this work are five Early Illness Alert ground truth datasets. The first four datasets (listed later in Table 4.2) were formed from multiple alert periods of a single alert parameter, while the fifth dataset (EIAFULL) was formed from multiple alert periods of the four alert parameters from the first four datasets. The five datasets allowed the analysis of individual EIAs (single-dimensional, single alert period, single alert parameter), a fusion of EIAs from the same parameter (multi-dimensional, multi alert period, single alert parameter) and finally a fusion of EIAs from multiple alert parameters

(multi-dimensional, multi alert period, multi alert parameter). The four alert parameters involved were initially selected because they had the EIAs that received the most feedback and had EIAs in both the good and poor alert categories. The steps to build the ground truth datasets are outlined in the following algorithm.

Algorithm: Build Early Illness Alert Ground Truth Dataset

```
BEGIN
    Select alert parameter(s)
    Create an empty dataset, Y, whose elements represent a unique user ID and date (alert
    day)
    FOR EACH alert parameter
        FOR EACH alert period
            FOR EACH EIA a with ≥ 1 clinical ratings
                **Filter out inconsistent/questionable ratings based on feedback
                comments
                Aggregate remaining ratings by taking the mean
                SWITCH
                CASE mean ≥ 3
                    Categorize a as a good alert
                CASE mean ≤ 3
                    Categorize a as a poor alert
                CASE mean == 3
                    Exclude a
                END SWITCH
                IF G does not contain the alert day y from a
                    Make a new element y in Y
                END IF
            END FOR EACH
        END FOR EACH
    END FOR EACH
    FOR EACH alert day y in the dataset Y
        Categorize y as a good alert day if at least one a in y is categorized a good alert
    END FOR EACH
END
**Alerts ratings were manually checked for questionable ratings, e.g. ratings left that did
not match comments or ratings left when the sensor web interface was unavailable
```

One element of the ground truth dataset represents a unique user ID and date. In other words, each individual sensor network on any given date presents an opportunity to send a single EIA based on the change in sensor data on that day. The feedback from the nursing clinicians are aggregated together using the mean of their clinical ratings to categorize each EIA for each alert period and alert parameter into good alerts or poor alerts or they may be excluded when they are borderline clinically relevant. Many of the EIAs on a given alert day may not indicate any clinical relevance, but if at least one EIA is clinically relevant, then an alert should notify nursing clinicians or staff to intervene on that alert day. Thus, at least one rated EIA categorized as a good alert is needed to categorize an alert day in the ground truth dataset as a good alert sample. After the ground truth dataset is built, it consists of labeled samples in one of two categories, good and poor alert days. This dataset can be used in supervised learning two-class or one-class classification methods. In the one-class case, the poor alert days would constitute the normal health samples and the good alert days (abnormal health days) would make up the outliers.

Alert ratings were manually checked for questionable ratings. Sometimes the comments left by a rater does not match how they rated the alert, so these ratings are filtered before generating the ground truth. At times, a rater would leave an alert rating even when the sensor web interface was unavailable or malfunctioning (indicated by the rater comments). Those ratings were filtered as well.

The clinical feedback was collected from two groups: the clinical staff consisting of nurses that work at TigerPlace and meet with residents on a daily basis, and the clinical researchers who do not. In this research, only the clinical feedback from the

clinical researchers is considered because there was confusion in the clinical staff group that led to inaccuracies in their feedback ratings during the time period of the alerts and feedback collection.

3.4 Multi-Dimensional Early Illness Algorithms

With ground truth datasets available, the supervised learning methods in Chapter 2 can be applied to build multi-dimensional Early Illness Alerts (MEIAs) which are superior to the single-dimensional EAIs in correctly classifying an alert day, i.e. the multi-dimensional methods can generate more accurate EIAs in the future. The single-dimensional EIAs are combined to create feature vectors that quantify each labeled alert day in a multi-dimensional feature space, where these supervised learning methods can build decision boundaries. This section outlines how the methods discussed in general in chapter 2 are applied to the improvement of Early Illness Alerts.

3.4.1 Features

Whereas the single-dimensional Early Illness Alerts output a binary value (either there is an alert or there is not) each feature for the Multi-Dimensional Early Illness Alerts outputs a quantitative value. Since only increase alerts are considered, the value is a measure of how much an alert feature is an increase in that alert parameter, with higher values corresponding to higher increases. The same sliding window is used to generate alert parameter values for each day in the sliding window, and values below the valid parameter value threshold are ignored, leaving only the samples above. If enough values remain, then two statistical parameters – the sample mean and sample standard deviation – for those values are computed. The mean is subtracted from the new day's parameter

value and that difference is divided by the standard deviation, leaving the value for the alert feature. This algorithm is highlighted as follows.

Algorithm: Computation of An Early Illness Alert Feature

BEGIN

Given the sliding window length in days, L
Given alert parameter function $\rho(t, P)$, where t is a date and P is an alert period
Given alert period, $P \in \{Full\ Day, Nighttime, Daytime\}$
Given valid alert parameter value threshold, T
Given the sample size threshold, S
Alert parameter values, $\boldsymbol{\rho} = \{ \rho_1, \rho_2, \ldots, \rho_L\} = \{0, 0, \ldots, 0\}$
FOR (n = 1 to L)
*Calculate $\rho_n = \rho(t_n, P)$
END FOR
$\boldsymbol{V} = \{v \in \boldsymbol{\rho} \mid v > T\}$
IF $(Size(\boldsymbol{V}) \geq S)$
$\mu = mean(\boldsymbol{V})$
$\sigma = std(\boldsymbol{V})$
**Output $x_i = \frac{(\rho(t_{n+1}, P) - \mu)}{\sigma}$
ELSE
**Output $x_i = 0$
END IF
END
*The alert parameter function can be found in table 3.5
**i refers to the a unique combination of alert parameter and alert period

For example, if the sample mean and sample standard deviation of the sliding window is computed as 200 sensor events and 50 sensor events, respectively, and the alert parameter value of the next day is 350, then the feature value for the given i is 3. The change from the sliding baseline is an increase of 3 standard deviations. Note that EIAs that would normally generate an increase alert would have positive feature values and decrease alerts would have negative values. Since the number of decrease EIAs at the start of this analysis was much smaller than that of the increases, the decrease EIAs were

excluded entirely. If an output cannot be computed because of either of the two threshold requirements, the feature value defaults to 0.

Out of all the alert parameters listed in table 3.5 and the alert periods listed in table 3.4, 4 alert parameters were chosen along with 3 alert periods resulting in a total of 12 EIA features, so using all of the available features resulted in 12-dimensional classifiers. A subset of those EIA features, based on recommendations from the clinical researchers, was also used to create 6-dimensional versions of the classifiers. Two values of the valid alert parameter value threshold T were used throughout the time the clinical researchers were submitting feedback for the EIAs in this study, beginning with $T = 0$ and ending with $T = 10$. Thus, different T-configurations were used to create different 12 and 6-dimensional feature spaces. The final T-configurations for the 12-dimensional and 6-dimensional feature spaces to compare the different classifiers were chosen based on experimental results from the Pattern Tree classifier.

A major concern with pattern recognition methods that use feature vectors to train their classifier is proper feature normalization. The methods implicitly assign more weight to those features that have larger ranges and features with smaller ranges fall out of the picture. For the EIAs, feature normalization is a concern because some EIAs may have considerably higher increases than others. The original idea behind fusing multiple EIAs was to not only send an alert if there is a large increase in one important alert parameter, but also if there were smaller increases in multiple important alert parameters. The features need to be normalized in a way that keeps larger increases from completely outweighing smaller features, while also keeping smaller increases in a feature that has a larger range of increases from becoming too small after normalization. Thus, the

Gaussian-based normalization scheme shown in figure 3.15 is used to preprocess the feature values before they are presented to the multi-dimensional classifiers.

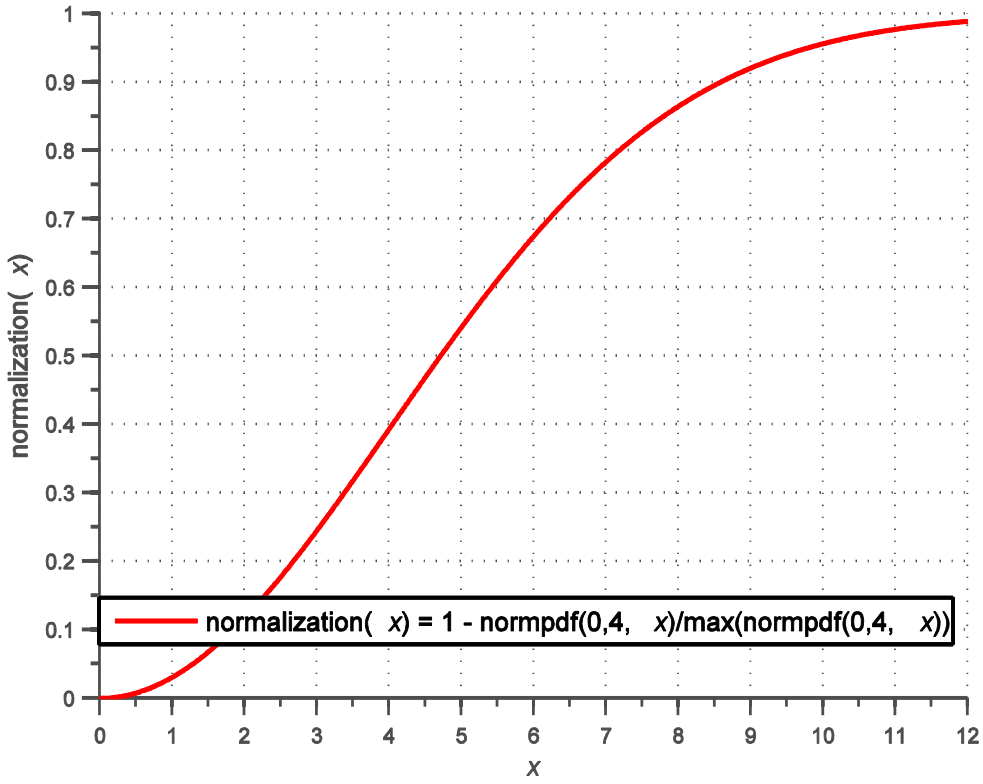

Figure 3.15 Feature Value Normalization Function

The normalization function remains fairly linear between increases of 1 and 6 standard deviations and then slowly approaches the limit value 1 as the increases gets much larger. The function maintains the monoticity of the feature values, emphasizing smaller to mid-range increases while limiting the weight of much higher increases. This function is exactly the same as the inverted Gaussian increase membership function defined for the Pattern Tree discussed in the next section, which will allow a clearer comparison between the decision boundaries built by the Pattern Tree's OR operator and the boundaries built by the K-Nearest Neighbor, Neural Network, and Support Vector Machine. This is the

same as taking the fuzzified inputs (based on the inverted Gaussian) features to the Pattern Tree as inputs to the other classifiers.

3.4.2 Pattern Tree Early Illness Detection

The Pattern Tree presents an intriguing method for generating MEIAs because of the intuitive nature of its tree structure and its applicability to an expert system. Although there are top down and bottom-up induction methods for learning the pattern tree structure and how it could build a classifier for Early Illness Alerts, having an expert system with domain knowledge from experienced clinical researchers provides an opportunity to simply build a Pattern Tree using domain knowledge. That would mean a classifier potentially robust enough for a variety of situations without having to train the classifier with training data.

The experiments in this study cover only the surface of Pattern Tree classification, using only a single OR operator (Yager t-conorm) to combine linguistic variable inputs (individual EIAs) that have a single linguistic term (increase). The Yager OR operator has only a single operator parameter, *w*, that determines how more optimistic the Yager OR is compared to the MAX OR operator. The *w* parameter was initially set to 3 although it was varied in one experiment to observe its effects. The original membership function used to model an EIA's increase was covered as the feature normalization function in the previous section, although several other membership functions were used as well. The general shape of these membership functions are shown in figure 3.16.

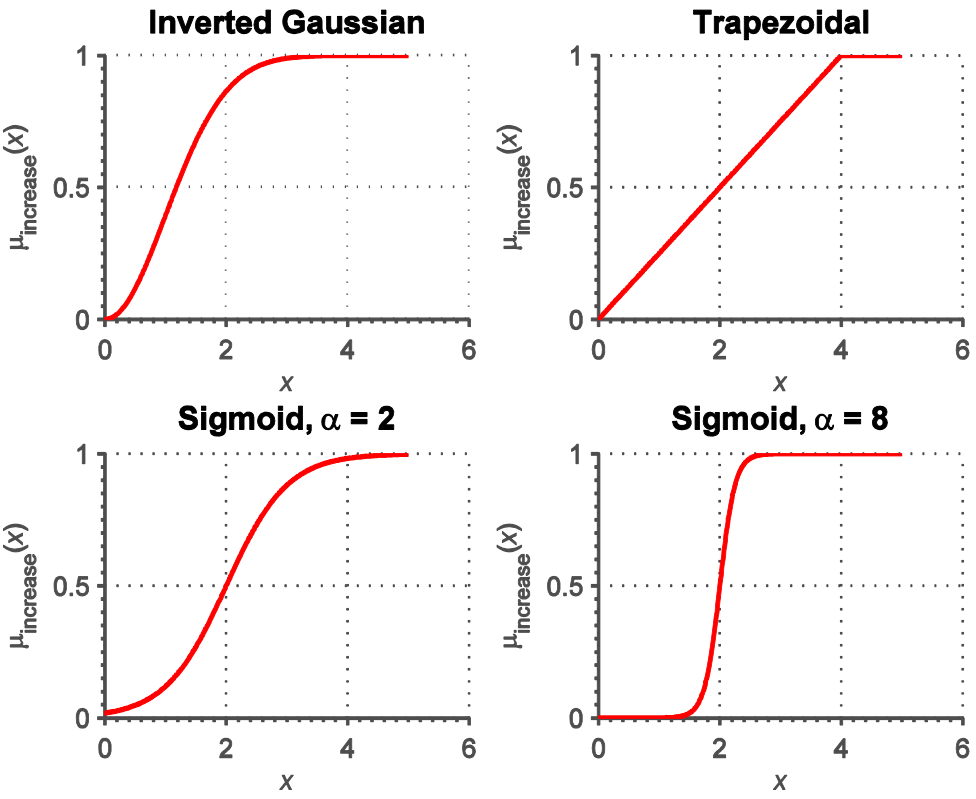

Figure 3.16 Membership Functions for Linguistic Term Increase

The first investigation with multi-dimensional Pattern Trees combined the three different EIAs from each of the three alert periods for a single alert parameter to investigate the relative importance of each alert parameter, and during which period(s) they are more clinically relevant. A second set of experiments involved using all 12 features with different *T*-configurations in a 12-dimensional Pattern Tree that intuitively functions as "if bathroom activity for the full day increases or bathroom activity at night increases, or…, or bed restlessness for the full day increases, or…, … then send an Early Illness Alert." Essentially, the Pattern Tree functions the same as the 12 individual EIAs do collectively, and the accuracy of the Pattern Tree should not improve significantly. However, if the 12 features are reduced to 6 features, based on recommendations from the nursing clinicians on which features tend to be more clinically relevant, a better

Pattern Tree can be built, even with just a simple Yager OR operator. This process is somewhat like a feature selection process for creating a linear classifier. The 6 features chosen are shown as a Pattern Tree in figure 3.17. Similar to the 12-dimensional case, experiments with different *T*-configurations were also performed on the 6-dimensional Pattern Tree and Principal Component Analysis (PCA) [53] visualizations of the two feature spaces were compared.

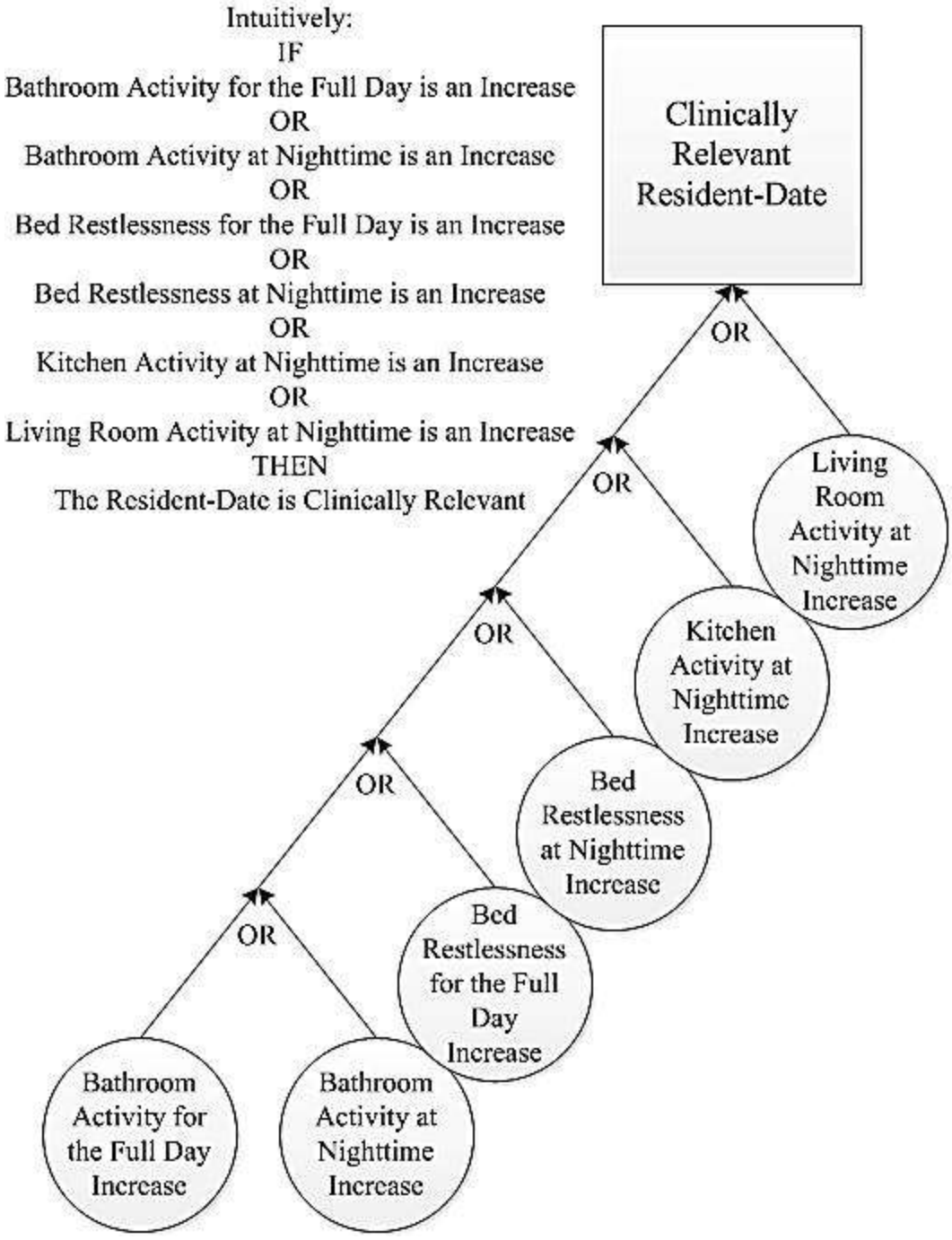

Figure 3.17 Multi-Dimensional Pattern Tree

Additional experiments were performed on the 6-dimensional Pattern Tree using different membership functions, using crisp membership functions, and varying the Yager parameter.

3.4.3 Fuzzy K-Nearest-Neighbor Early Illness Detection

The EIA features were normalized as described in Section 3.4.1 and combined into a feature vector as input to the Fuzzy K-Nearest Neighbor algorithm. Two sets of FkNNs, 12-dimensional and 6-dimensional, using the standard Euclidean distance as a dissimilarity measure, were trained and tested using 10-fold cross validation and ROC curve analysis. In each fold of the cross validation, 90% of both the good and poor alert days from the EIAFULL dataset were used for training with the remaining 10% used for testing. Each set of FkNN experiments varied *K* using 1, 3, 5, 7, 9, and 11. The ROC curve was formed taking the output of the test samples of each cross fold FkNN that exists in [0, 1] and varying the decision threshold from 0 to 1. A FkNN MATLAB library [41]was used to perform the experiments.

3.4.4 Neural Network Early Illness Detection

Two sets of 10-fold cross validation experiments were also performed to build 12-dimensional and 6-dimensional Neural Network classifiers. Both sets of NNs used 5 hidden nodes and one output node, with all weighted connections free to change. The feature vectors were normalized as described in Section 3.4.1. The feed-forward network used the symmetric s-shaped sigmoid activation function at each hidden node and the output node. After each training sample, the NNs were updated using scaled gradient back propagation. Each fold randomly split the EIAFULL dataset into approximately 70% training data, 15% validation data, and 15% testing data. The training data was used to update the NNs and after each epoch, a sum of squared error (SSE) function was computed using the validation data. The NNs were trained while the SSE using the validation set decreased and terminated when the SSE increased for several epochs. The

outputs of the NNs using the test dataset from each fold were combined together and a decision threshold was varied to perform ROC curve analysis on the 12-dimensional and 6-dimensional NNs. The MATLAB Neural Network library was used to build the NNs.

3.4.5 Support Vector Machine Early Illness Detection

Similar to the FkNN experiments, the Support Vector Machine (SVM) experiments tested both 12-dimensional and 6-dimensional SVMs using 10 fold cross validation with 90% training data and 10% training data in each fold. The feature vectors were normalized prior to training as described in Section 3.4.1. In addition to using two different feature spaces, two SVM kernels, Linear and Radial Basis Function (RBF), were tested. Allowing the SVM decision boundaries to vary resulted in a total of four sets of ROC curve analyses. The MATLAB SVM library was used to build the SVM classifiers.

Chapter 4 - Experimental Results and Analysis

This chapter presents the experimental results from both the single-dimensional EIAs and the MEIAs using the four classification methods (PT, FkNN, NN, and SVM) in conjunction with the most recently available EIAFULL ground truth dataset. First, Section 4.1 summarizes the Early Illness Alerts dataset and the ground truth generated using clinical feedback. Section 4.2 shows the performance of each of the EIA features individually. Section 4.3 follows with experimental results using multi-dimensional algorithms, beginning first with the Pattern Tree, then progressing through the Fuzzy K-Nearest Neighbor, Neural Network, and Support Vector Machine. Comparisons between the four classification methods are also presented.

4.1 Early Illness Alert Dataset

A summary of the number of EIAs that have been generated from the sensor networks are listed in Table 4.1, totaling over 700 EIAs from November 1st of 2010 to July 13th of 2011.

Table 4.1 Summary of Single-Dimensional EIAs

Alert Parameter	Alert Period			Alert Type		Total
	Full Day Time	Night Time	Day	Increase	Decrease	
Bathroom	27	54	77	140	18	158
Bed Restlessness	34	55	42	119	12	131
Bed Breath Low	24	27	13	48	16	64
Bed Breath High	0	0	1	1	0	1
Bed Pulse Low	20	13	21	44	10	54
Bed Pulse High	2	0	5	7	0	7
Chair Restlessness	1	0	4	4	1	5
Chair Pulse Low	0	0	1	1	0	1
Kitchen	28	28	79	107	28	135
Living Room	35	34	100	142	27	169
Total	171	211	343	613	112	725

Using the feedback from the clinical researchers, 5 subsets of ground truth data were created, as listed in table 4.1. The EIAFULL dataset consists of four alert parameters for three alert periods (i.e., 12 EIA features), and 299 samples. Alerts with an average alert rating greater than three were considered good alerts. Alerts with an average alert rating less than three were considered poor alerts. Alerts with an average rating of three were ignored. An alert day is considered a good alert (abnormal) day if at least one alert on that day was a good alert. An alert day is considered a poor alert (normal) otherwise. The ignored column indicates the number of alert days that were excluded from

Table 4.2 EIA Ground Truth (# alert days)

Alert Parameter	Good	Poor	Ignored	Total	Name
Bathroom Activity	42	43	16	85	EIABA
Bed Restlessness	57	21	18	78	EIABR
Kitchen Activity	7	63	17	70	EIAK
Living Room Activity	17	85	21	102	EIALR
Total	116	183		299	EIAFULL

4.2 Single-Dimensional Early Illness Alert Results

This section presents the accuracy (computed as the sum of true positives and true negatives, the sum divided by the sum of all the samples) analyses of each individual EIA with both $T = 0$ and $T = 10$ against the ground truth subset that corresponded to the EIAs alert parameter. For instance, the bathroom activity for the full day, bathroom activity at night, and bathroom activity for the full day EIA were all tested against the EIABA ground truth dataset. Figures 4.2.1 through 4.2.4 show the ROC curves of each alert parameter using its corresponding ground truth and figure 4.2.5 shows the ROC curve of each EIA feature versus the full ground truth dataset.

4.2.1 Bathroom Activity

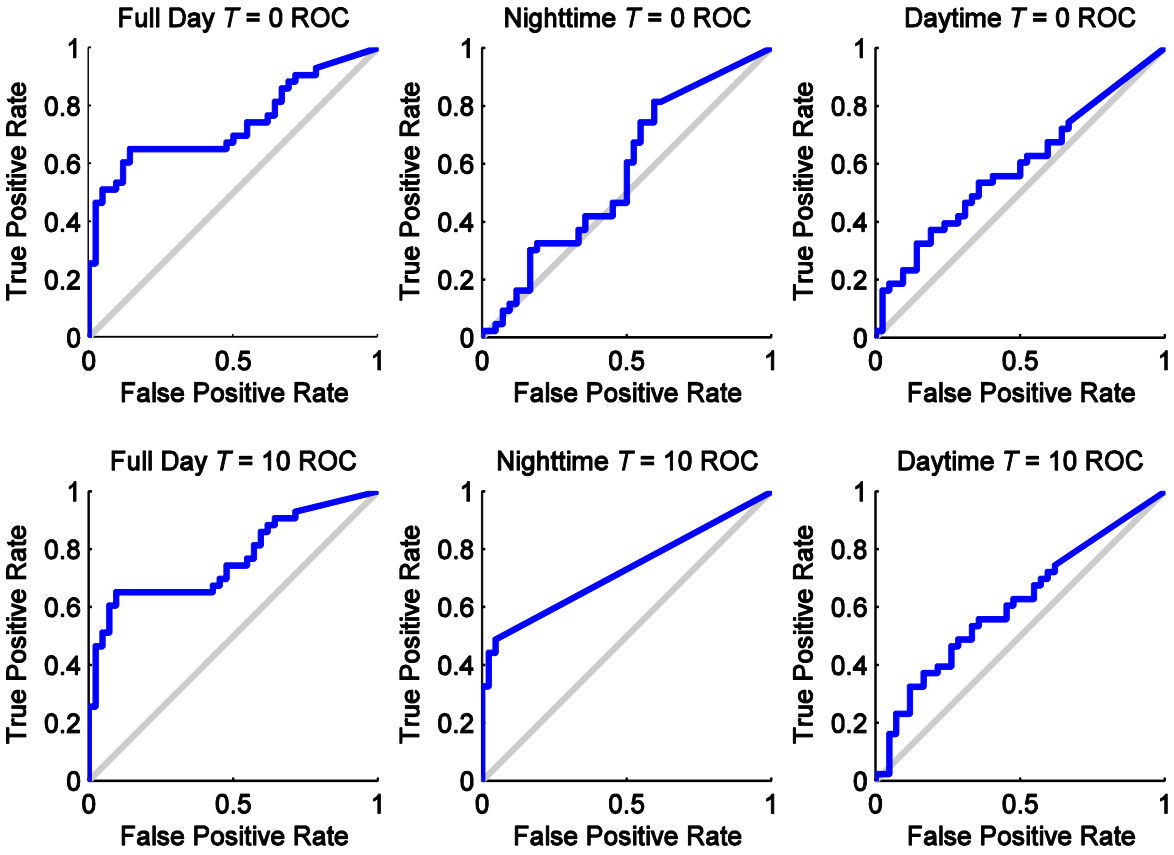

Figure 4.1 Bathroom Activity EIA ROCS on EIABA

Individually, the full day period for the bathroom activity alert parameter performed the best for both values of the threshold. The nighttime period for $T = 10$ also performed better than the chance line.

4.2.2 Bed Restlessness

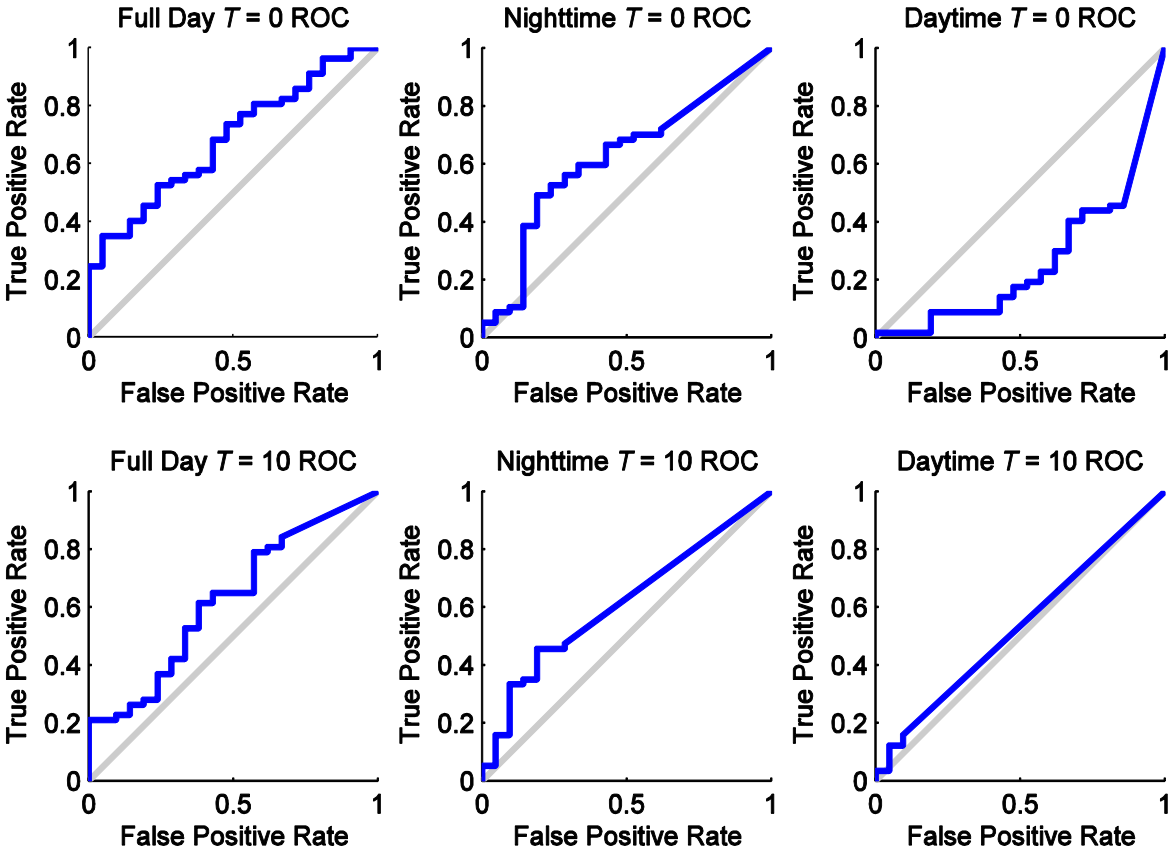

Figure 4.2 Bed Restlessness EIA ROCs on EIABR

There is not much of an advantage in the bed restlessness alert parameters when they are taken individually versus the EIABR dataset. The full day for $T = 0$ performed the best in terms of AUC. The daytime period for $T = 0$ could be reversed to create a better ROC curve although it does not make sense to look for no increases in daytime bed restlessness as an alert condition.

4.2.3 Kitchen Activity

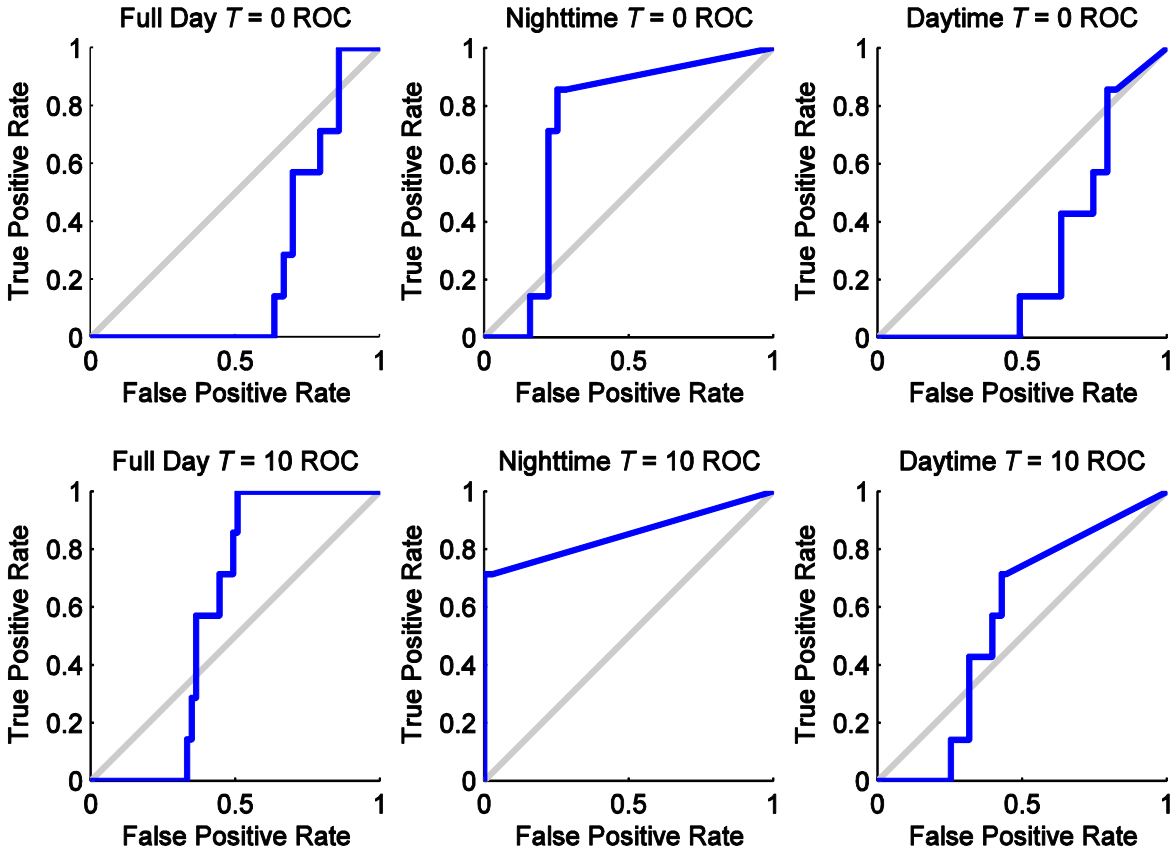

Figure 4.3 Kitchen Activity EIA ROCs on EIAK

The nighttime kitchen activity alert parameter had the best ROC curves, which is expected based on discussions with the clinical researchers. The ROC curves are much steeper, likely due to the disproportional number of poor kitchen activity alerts versus good kitchen activity alerts.

4.2.4 Living Room Activity

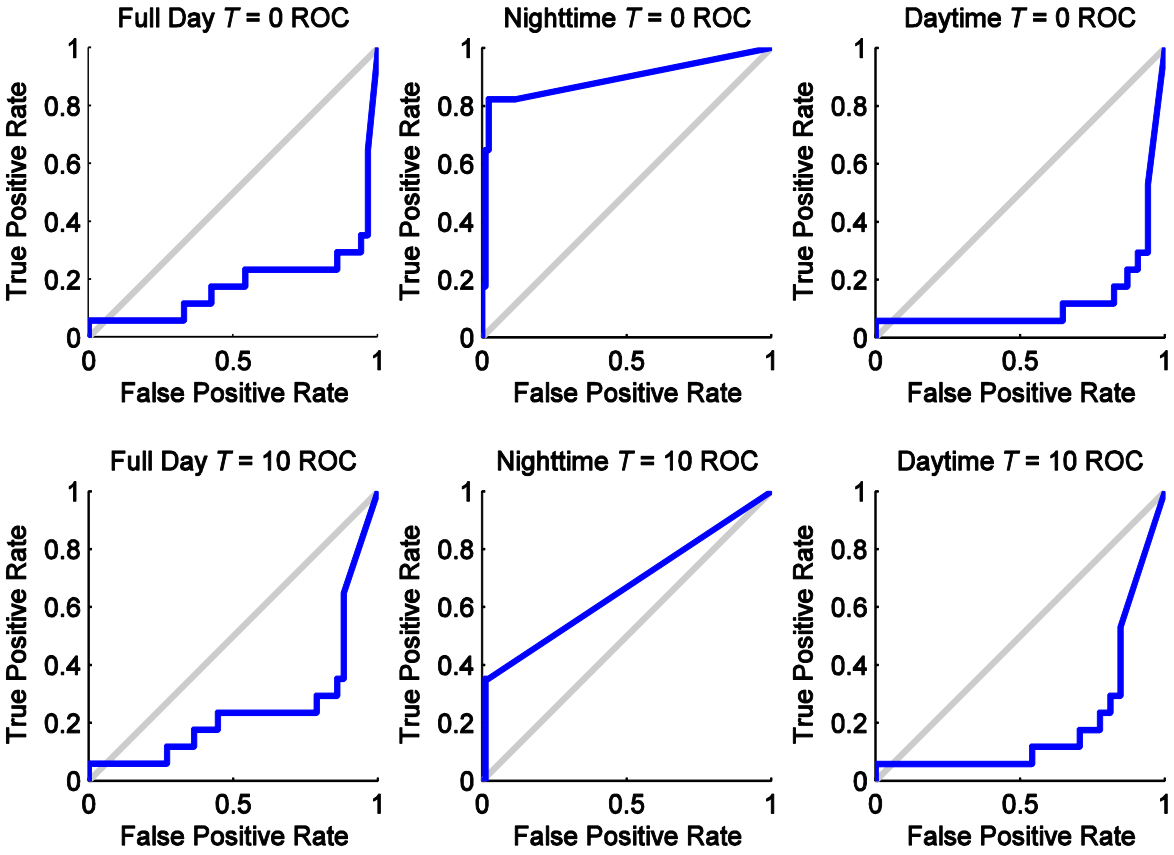

Figure 4.4 Living Room Activity EIA ROCs on EIALR

The living room activity at night with $T = 0$ generated the best ROC curve for the living room alert parameter versus EIALR. The daytime ROC curves could be inverted to generate very good ROC curves, but again, it does it make sense to look for an absence in increases in living room activity during the day as an alert condition.

4.2.5 Single-Dimensional Early Illness Alerts vs. Full Dataset

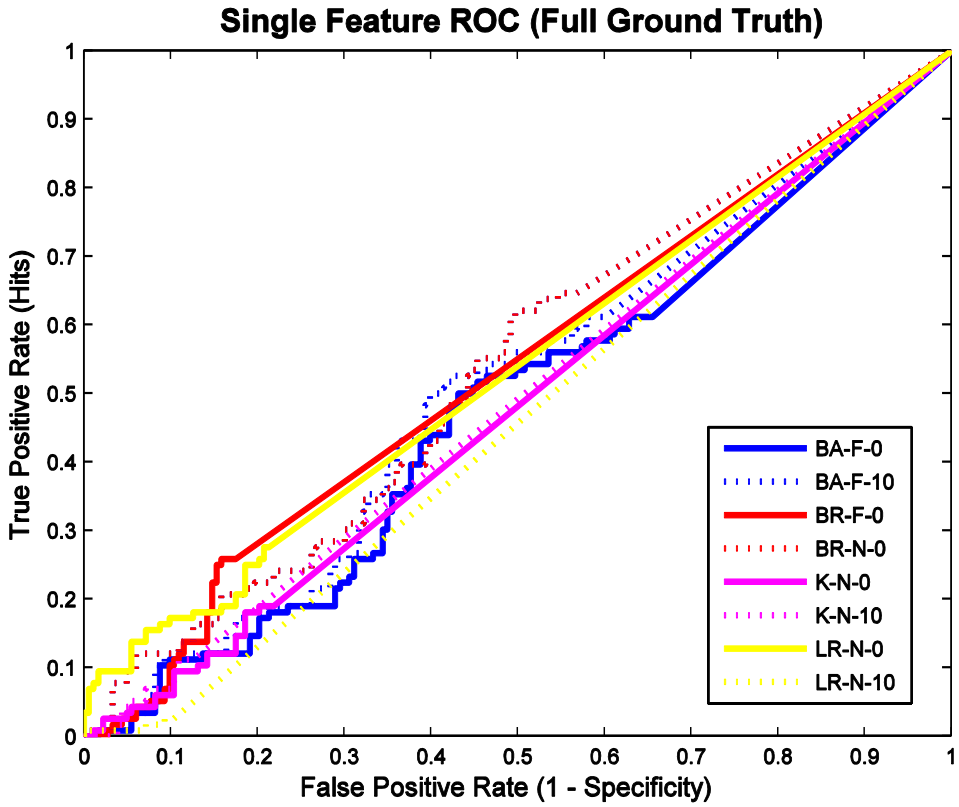

Figure 4.5 Single-Dimensional EIAs on EIAFULL

As indicated by the indecisive ROC curves, the individual alerts just by themselves do not do a good job of correctly classifying alert days.

4.3 Multi-Dimensional Algorithms

This section presents the experimental results from the multi-dimensional Pattern Tree, Fuzzy K-Nearest Neighbor, Neural Network, and Support Vector machine classifiers.

4.3.1 Pattern Tree Results

For the Pattern Tree experiments, the Yager OR operator with $w = 3$ was used to combine features into a multi-dimensional classifier. This default value was selected because it was experimentally the optimal value.

4.3.1.1 Multi-Dimensional, One Alert Parameter

The EIAs from a single alert parameter were combined in one of 7 possible combinations of alert periods against the alert parameters ground truth data subset to determine which, if any, combination for that alert parameter performed more accurately. T was varied between the two values 0 and 10. The ROC curve results are displayed in figures 4.6-13. The cyan lines represent the combination of the full day OR night time periods. The magenta represents the combination of the full day OR day time periods. The yellow represents the combination of the night time or day time periods. The black represents the OR of all three periods.

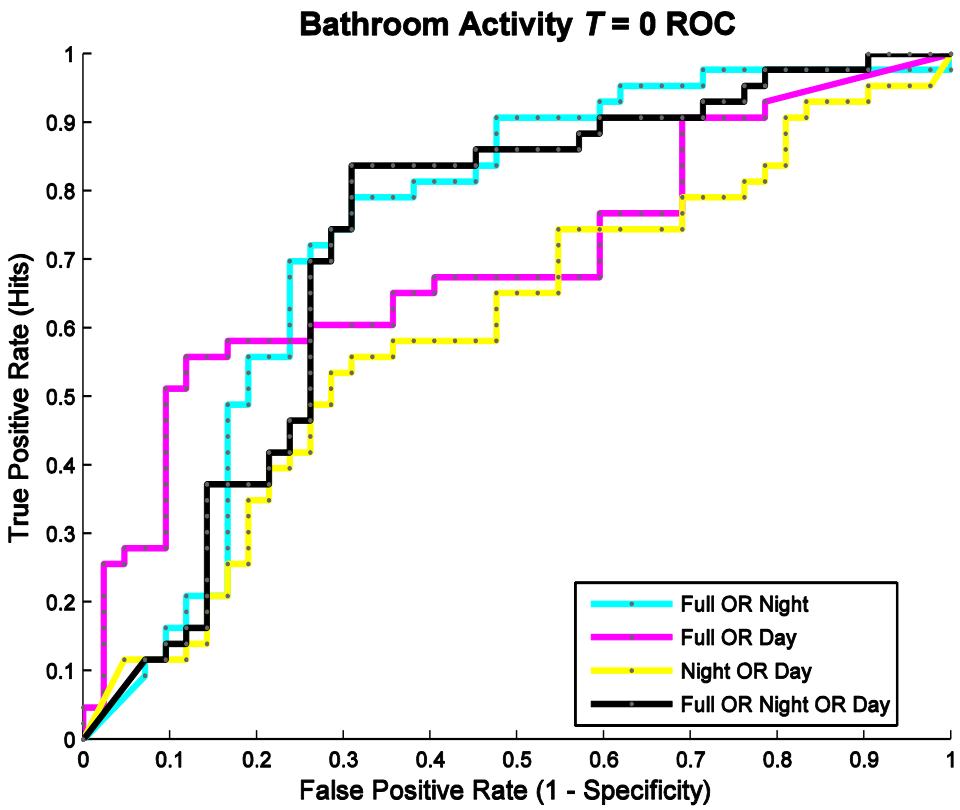

Figure 4.6 Multi-Dimensional Bathroom Activity on EIABA, T = 0

Overall, combining two or more bathroom parameters for $T = 0$ improves the classification above the chance line. The combination of full day or nighttime edges out combining all three periods based on the AUC.

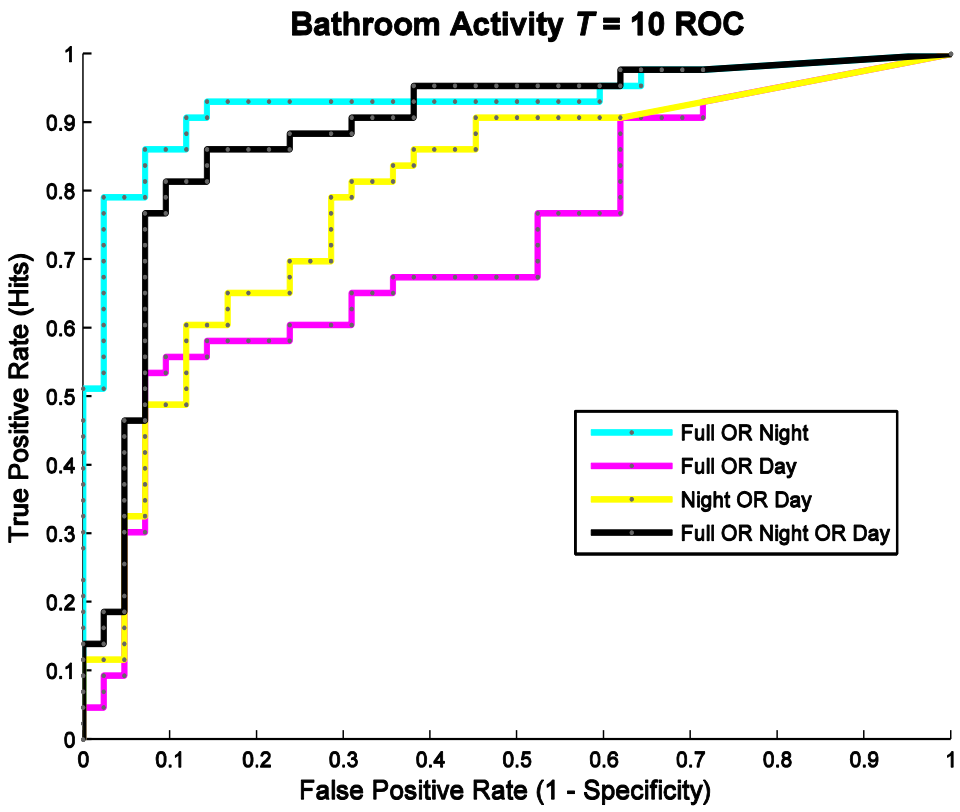

Figure 4.7 Multi-Dimensional Bathroom Activity on EIABA, T = 10

The ROC curves for the combined bathroom parameters with $T = 10$ produce two very good classifiers versus the EIABA dataset. Combining the full day with the nighttime period produces the best ROC curve.

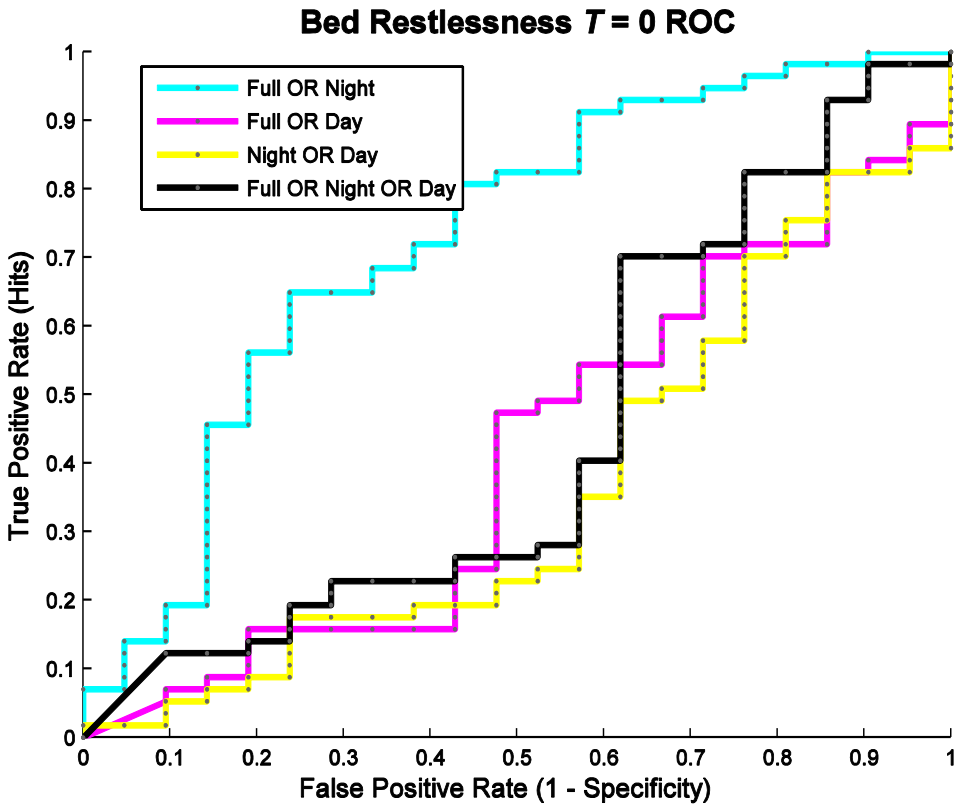

Figure 4.8 Multi-Dimensional Bed Restlessness on EIABR, T = 0

For the bed restlessness feature with $T = 0$, only the full day combined with nighttime ROC curve improves over the chance line (without inverting the curve).

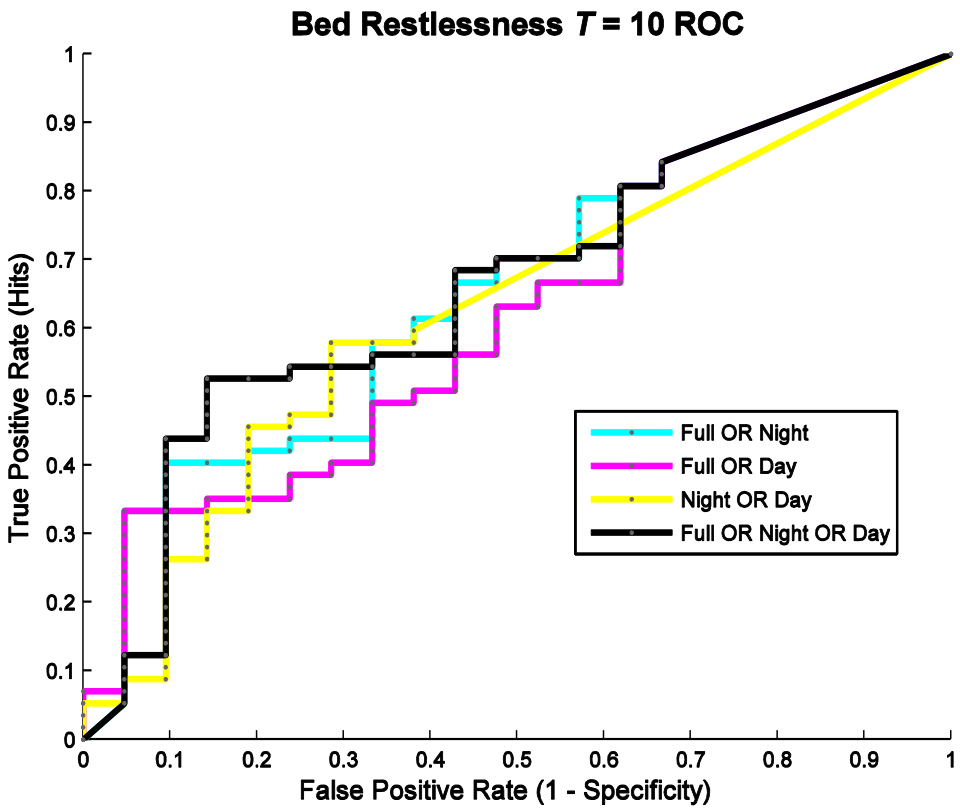

Figure 4.9 Multi-Dimensional Bed Restlessness on EIABR, T = 10

With $T = 10$, the ROC curves are slightly improved over the no chance line, with the combination of all three performing the best based on AUC.

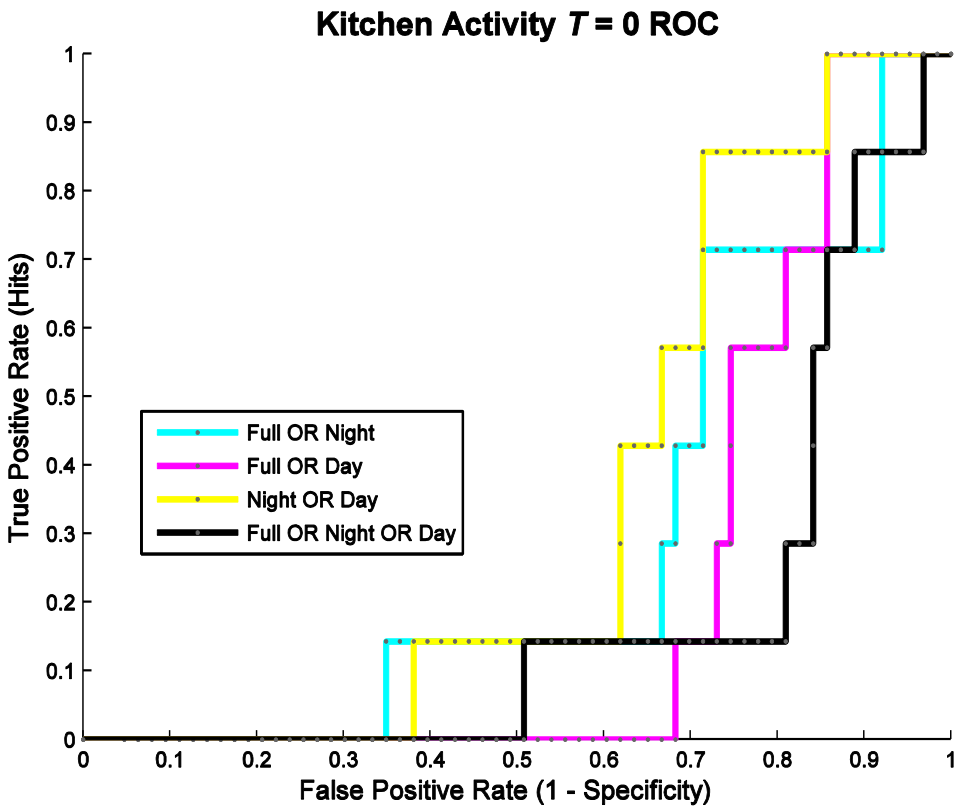

Figure 4.10 Multi-Dimensional Kitchen Activity on EIAK, T = 0

Since it does not make sense intuitively to invert the ROC curves for combination of the kitchen activity parameters, combining the parameters performs worse than just using the kitchen activity at night by itself.

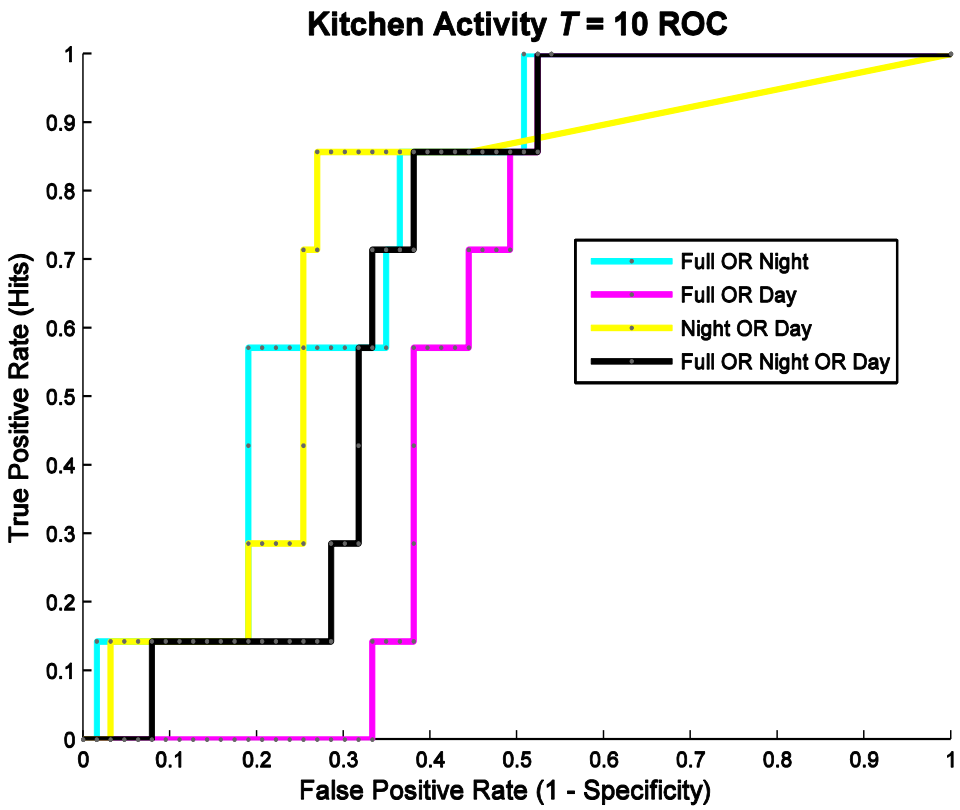

Figure 4.11 Multi-Dimensional Kitchen Activity on EIAK, T = 10

However, with $T = 10$, the combinations of kitchen activity periods results in ROC curve AUCs above 0.5, with the exception of the combination of the full day or night. Not including the nighttime parameter results in a poor ROC curve versus the EIAK dataset.

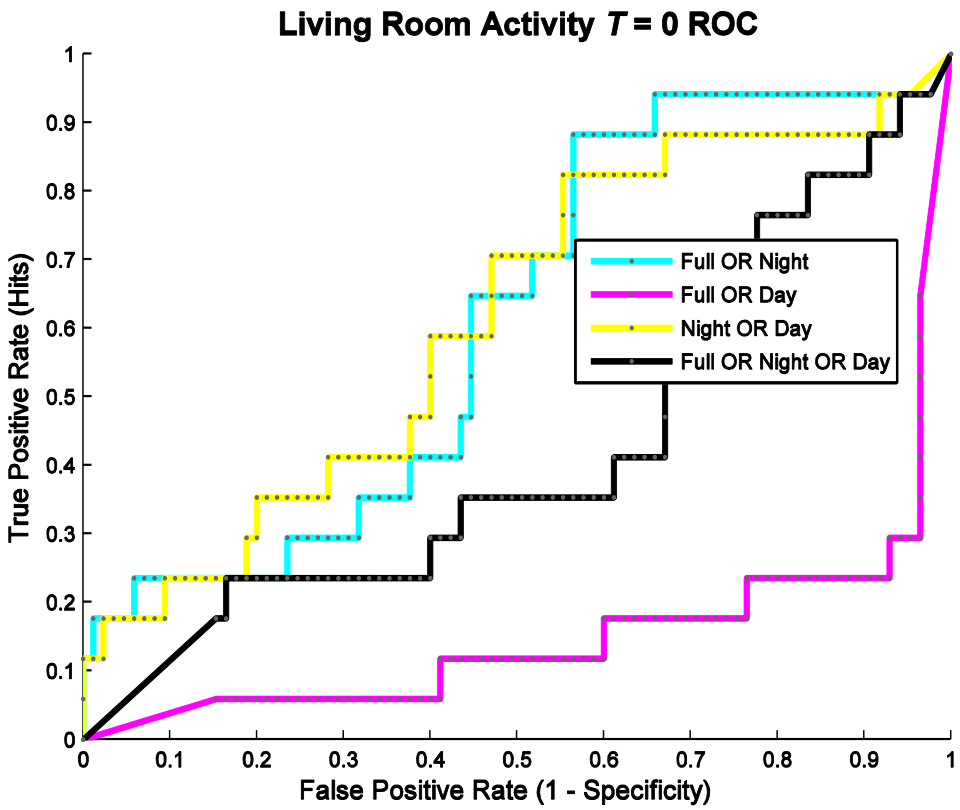

Figure 4.12 Multi-Dimensional Living Room Activity on EIALR, T = 0

The combination of living room periods that include the nighttime period perform at or better than the chance line, while the one combination without the nighttime period performs very poorly (without inverting).

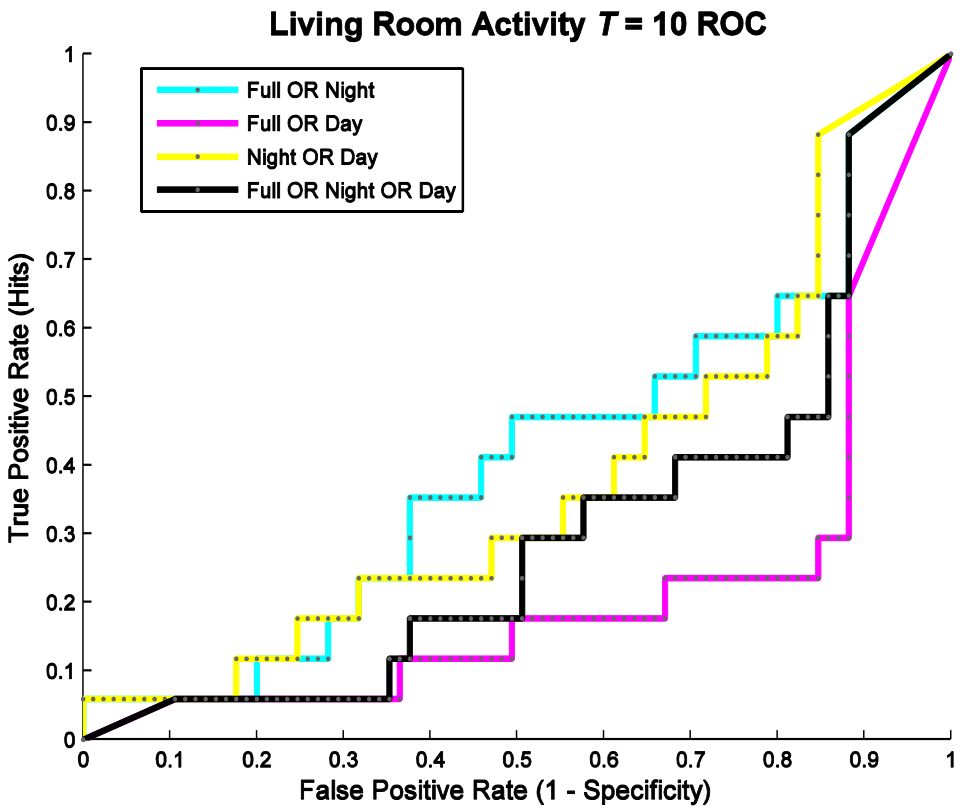

Figure 4.13 Multi-Dimensional Living Room Activity on EIALR, T = 10

With $T = 10$, all of the combinations of living room periods perform poorly (without inverting).

4.3.1.2 Multi-Dimensional, Multiple Alert Parameters

As previously mentioned in chapter 3, the combination of all 12 features using the Yager OR should intuitively present accuracy results similar to the EIAs that are already being generated. However, in order to compare 12-dimensional versions of each of the four classification methods, the proper *T*-configuration had to be selected and it was chosen based on the ROC curve performance, using both the maximum operating point accuracy and the area under the curve (AUC), of the Pattern Tree with each *T*-configuration. The first two configurations, listed in table 4.3, simply keep *T* constant for

each alert parameter, while the remaining four represent the best four remaining configurations. There are a total of 16 *T*-configurations if *T* is kept constant for the EIAs within each alert parameter. Figure 4.14 shows the results of the 12-dimensional PT with different configurations.

Table 4.3 *T* Configuration for 12-Feature Fuzzy Pattern Trees

Configuration	Bathroom Activity	Bed Restlessness	Kitchen Activity	Living Room Activity	Area Under Curve
1	0	0	0	0	0.456
2	10	10	10	10	0.568
3	0	0	10	10	0.552
4	0	10	0	0	0.427
5	10	0	10	10	0.513
6	10	0	10	0	0.604

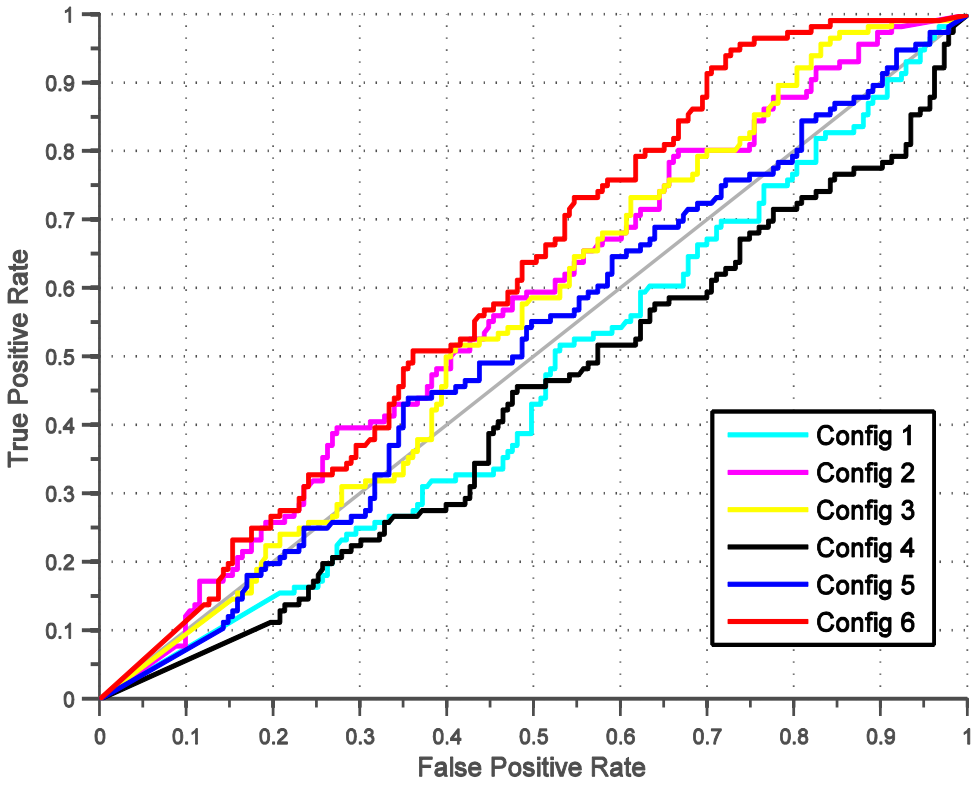

Figure 4.14 ROC Curves for 12-Feature FPTs, various *T* Configurations

Configurations 3-6 all have the same maximum accuracy, which actually comes from the trivial case where all alert days are simply classified into the poor category. Nonetheless configuration 6 had the highest AUC and is chosen as the 12-dimensional *T*-configuration of choice. Table 4.4 shows the confusion matrix for this Pattern Tree's maximum, yet trivial, operating point.

Table 4.4 Confusion Matrix for 12-Feature FPT, Configuration 6, at Max Accuracy

# Samples	183	**Actual**	
Accuracy	**0.6120**	Relevant	Irrelevant
Predicted	Relevant	0	0
	TPR/FPR	**0.0000**	**0.0000**
	Irrelevant	116	183
	FNR/TNR	**1.000**	**1.000**
		116	183

Table 4.5 *T* Configuration for 6-Feature Fuzzy Pattern Trees

Configuration	**Bathroom Activity**	**Bed Restlessness**	**Kitchen Activity**	**Living Room Activity**	**Maximum % Accuracy**
1	0	0	0	0	75.59
2	10	10	10	10	78.91
3	10	0	10	10	83.61
4	10	10	10	0	81.61
5	10	10	0	10	75.25
6	10	0	10	0	85.62

From discussions with the clinical researchers, the following 6 features determined to provide the most clinically relevant EIAs were: bathroom activity at night, bathroom activity during the day, bed restlessness at night, bed restlessness during the day, kitchen activity at night, and living room activity at night. So for the 6-dimesional classifiers, a *T*-configuration analysis similar to the 12-dimesional case was performed to select the value of *T* for each of the four alert parameters involved. The configurations are listed in table 4.5 while the ROC curves for each configuration are shown in figure 4.15.

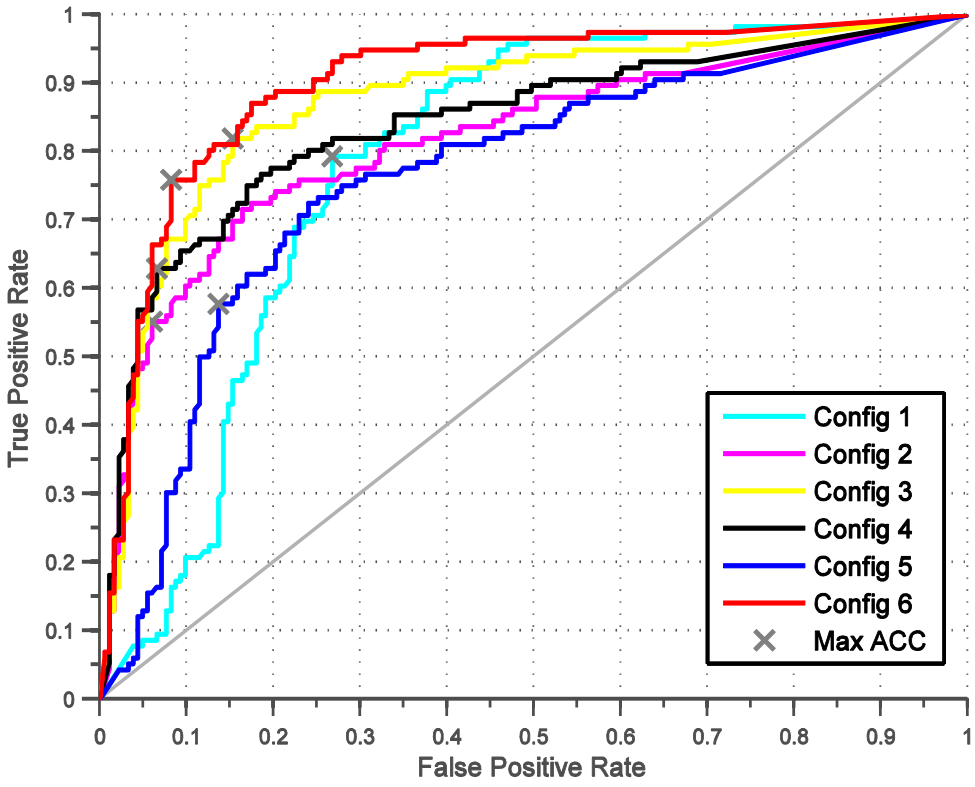

Figure 4.15 ROC Curves for 6-Feature FPTs, various *T* Configurations

Configuration 6 had the maximum accuracy at 85.62% and the confusion matrix for this operating point is listed in table 4.6.

Table 4.6 Confusion Matrix for 6-Feature FPT, Configuration 6, at Max Accuracy

# Samples	256	**Actual**	
Accuracy	**0.8562**	Relevant	Irrelevant
Predicted	Relevant **TPR/FPR**	88 **0.7586**	15 **0.0820**
	Irrelevant **FNR/TNR**	28 **.2414**	168 **0.9180**
		116	183

4.3.1.2 Input Space Visualization

To compare the considerable improvement gained from using 6 features instead of all 12 features for the Pattern Tree, as well as provide insight on how the other three classification methods would create decision boundaries, PCA visualizations of both the 12-dimensional and 6-dimensional features spaces were generated. Prior to the PCA in both cases, the features were normalized using the normalization function in chapter 3. The visualizations are shown in figures 4.16-20.

Note that for the 12 feature PCA feature space there may be a hyperplane that separates the two classes to some degree, although it may be hard to see. With the 6 feature PCA feature space there is a much more defined boundary. The poor alert days or normal days are centered around the origin and the good alert days or abnormal days move away from the origin.

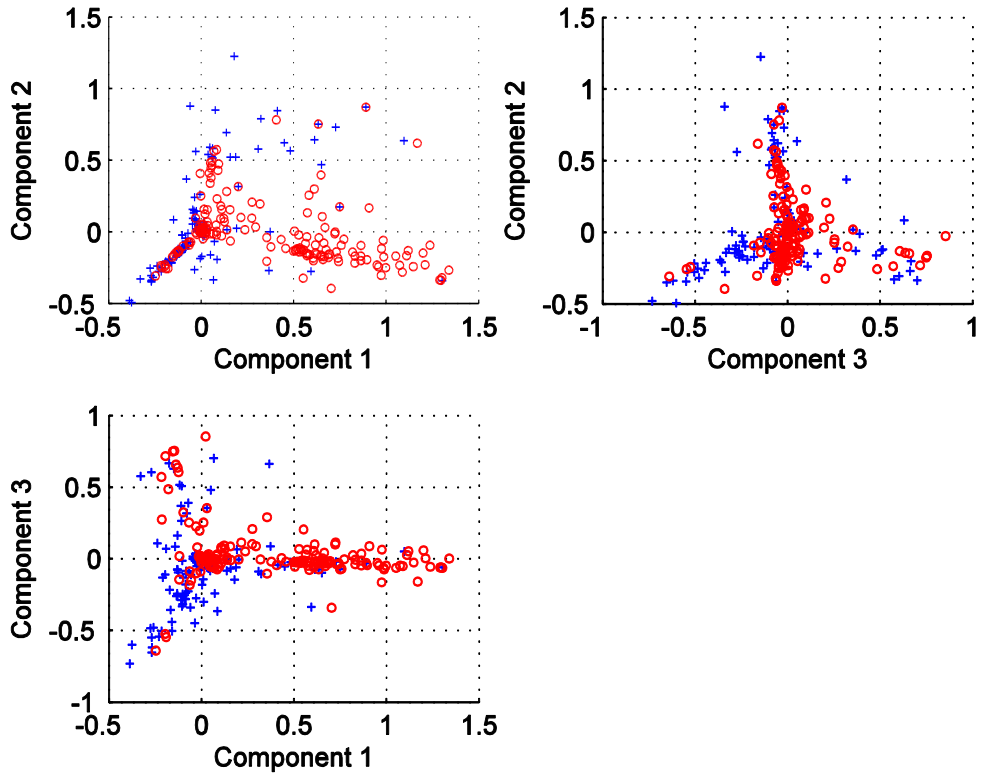

Figure 4.16 2D Plots of 12-Feature PCA

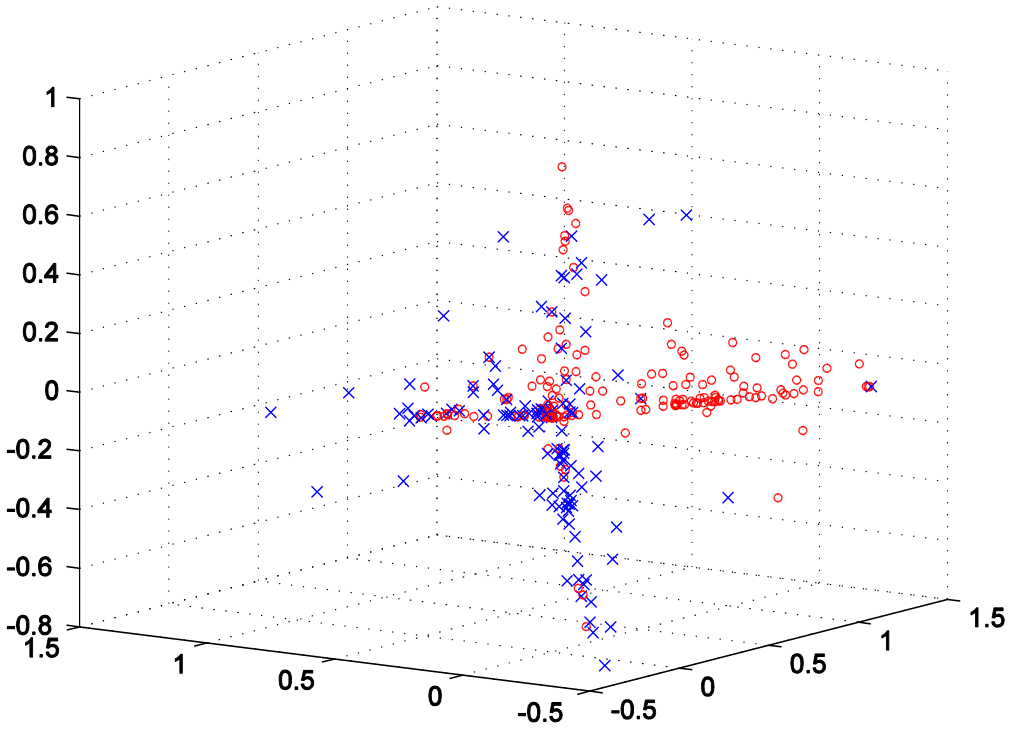

Figure 4.17 3D Plot of 12-Feature PCA

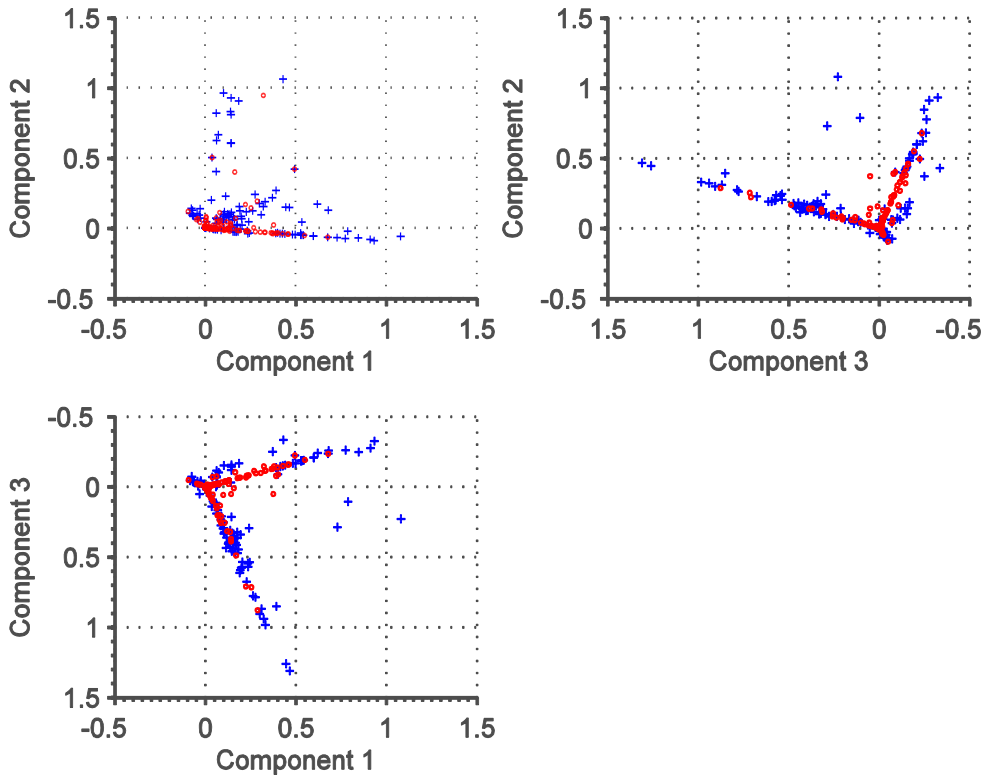

Figure 4.18 2D Plots of 6-Feature PCA

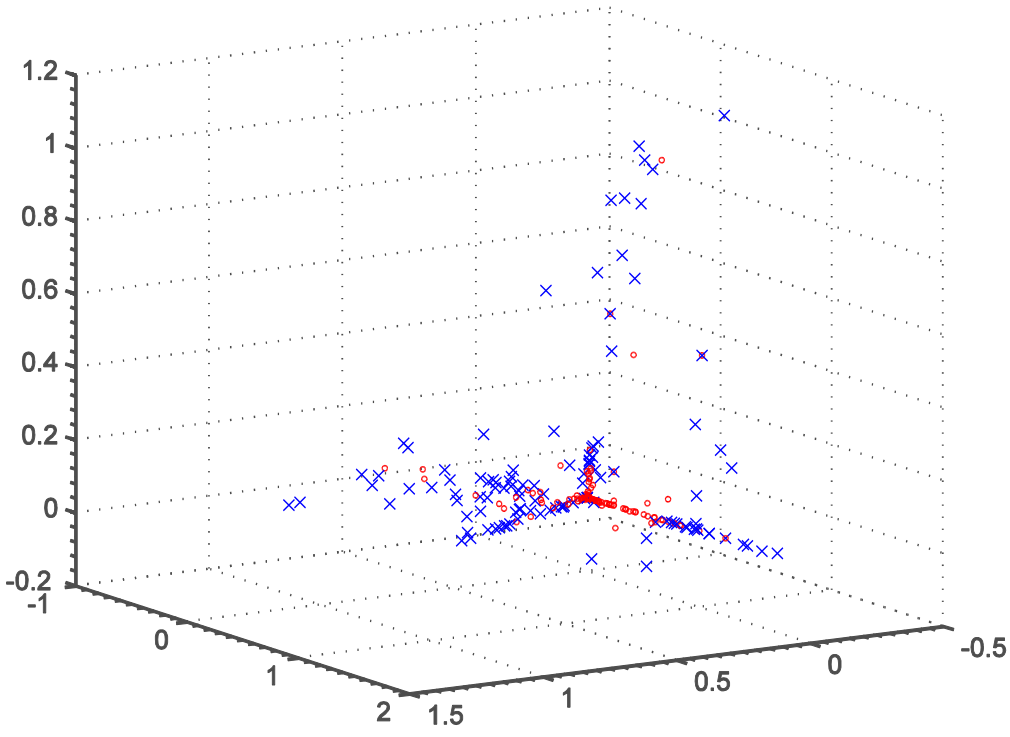

Figure 4.19 3D Plot of 6-Feature PCA

4.3.1.3 The Membership Function

As alternative to the default inverted Gaussian membership function to define an increase in an EIA, the trapezoidal and various versions of the sigmoid function were also used for the 6-dimesional Pattern Tree classifier. The ROC curves are shown in figure 4.20.

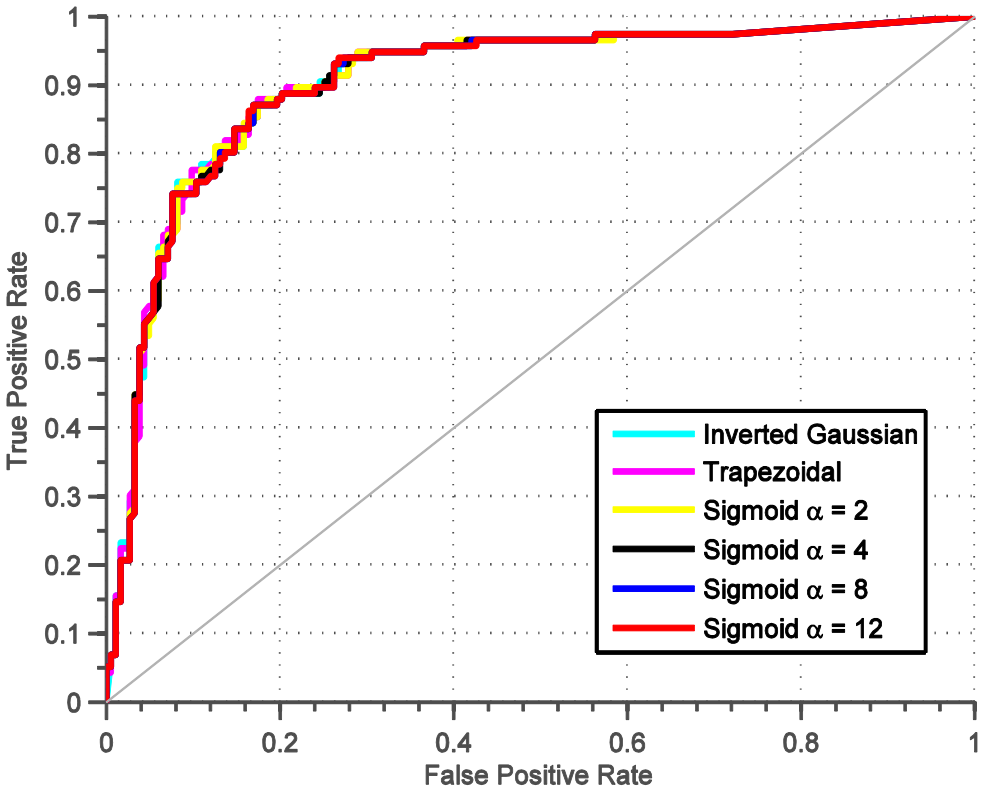

Figure 4.20 6-Feature FPT ROC vs. Fuzzy Membership Functions

There is not a clear advantage between the use of different membership functions, so long as they are monotonically increasing.

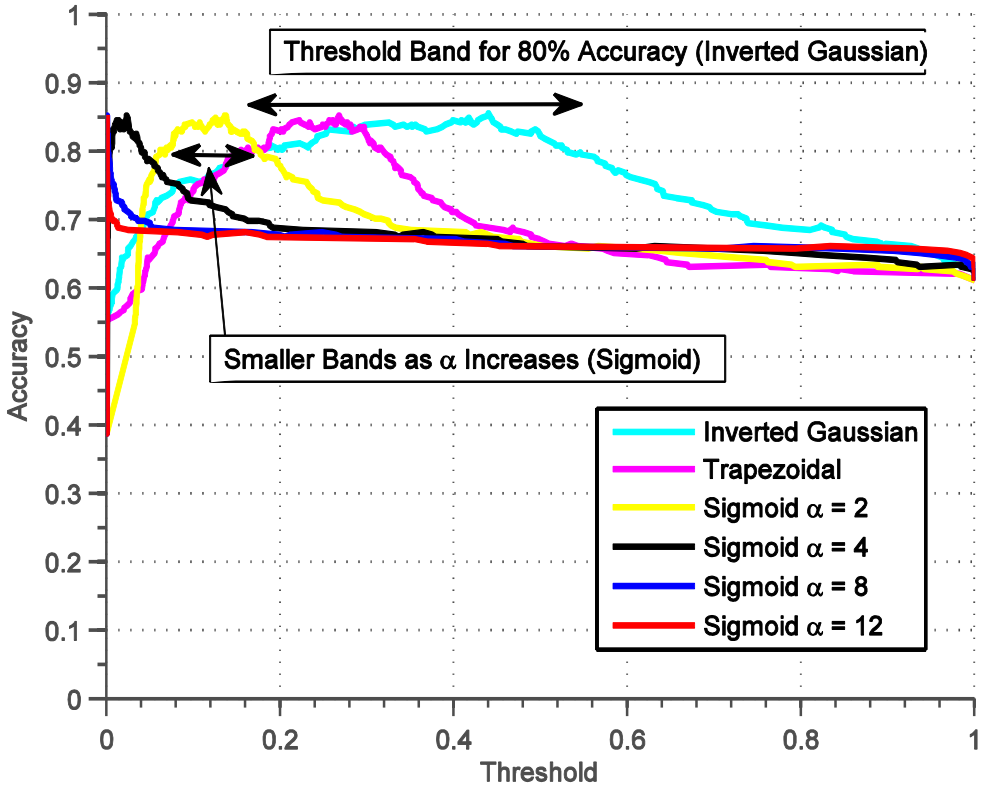

Figure 4.21 6-Feature PT Accuracy vs. Decision Threshold for Fuzzy MFs

Because it was difficult to see considerable differences between the ROC curves for different membership functions, an accuracy vs. threshold plot was generated for each of the functions. These curves help illustrate the flexibility of the Pattern Tree for each membership function to vary the operating point threshold between TPR and FPR while maintaining high accuracy.

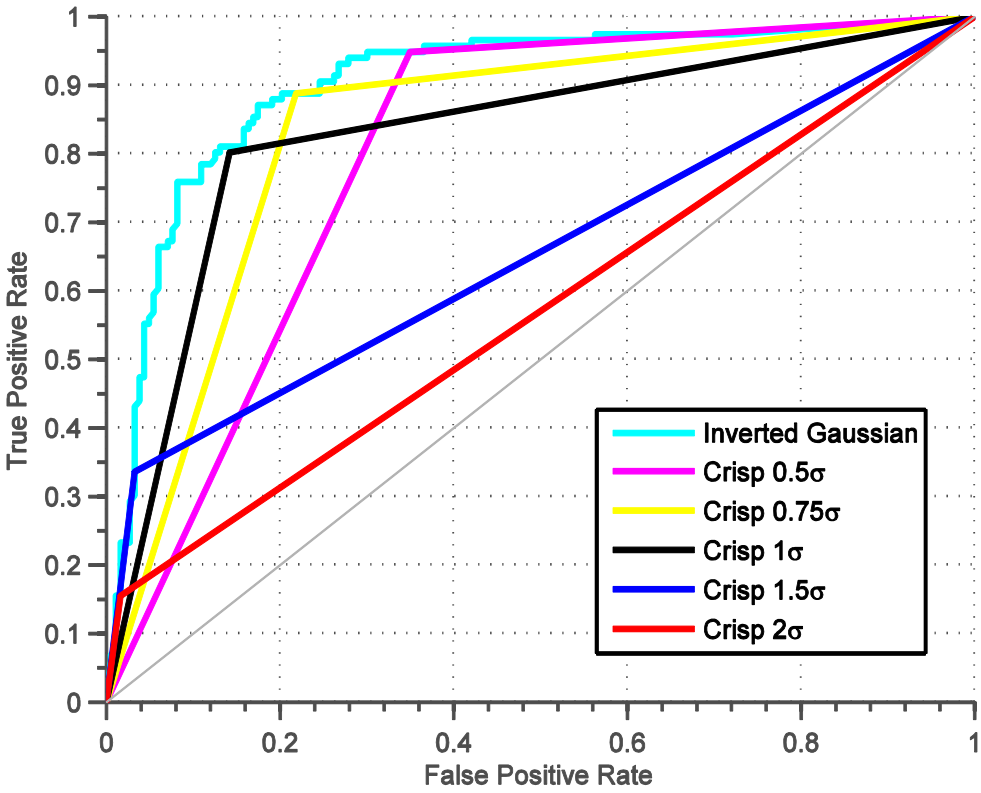

Figure 4.22 6-Feature PT ROCs, Crisp MFs

The Pattern Tree can easily be turned into a crisp classifier by using a crisp membership function, i.e. a membership function that jumps from 0 to 1 at a specified threshold. The ROC curves in figure 4.22 illustrate the crispness of these Pattern Trees because their ROC curves only consist of one operating point, determined by the specified threshold from the crisp membership function.

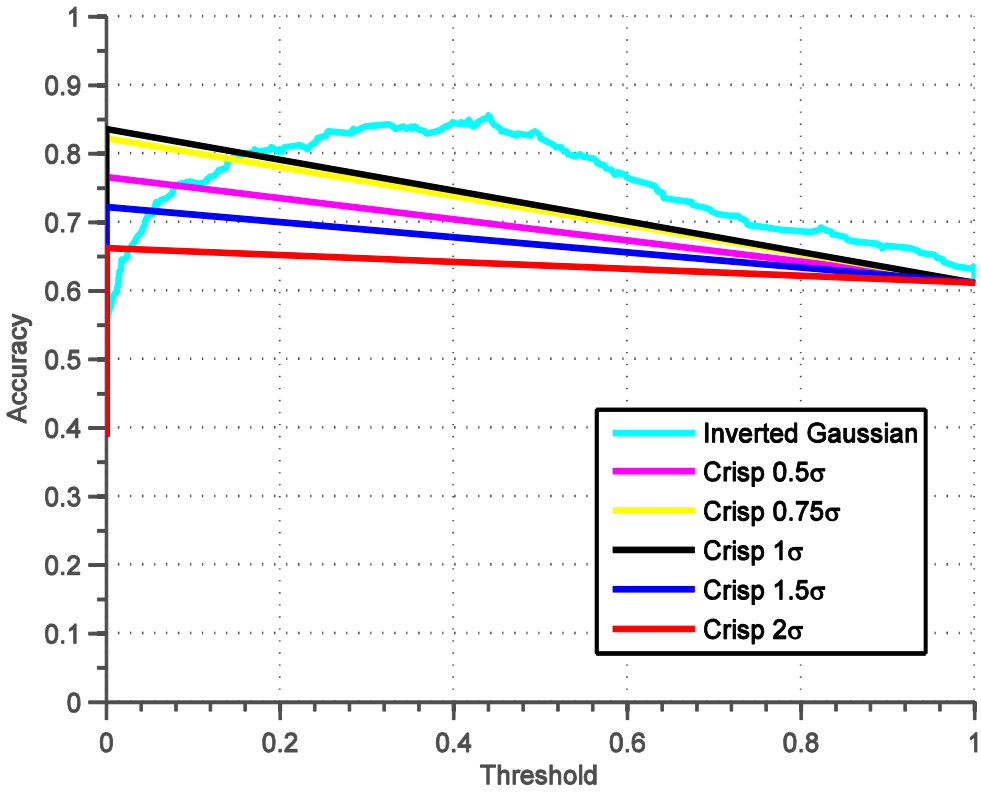

Figure 4.23 6-Feature PT Accuracy vs. Decision Threshold, Crisp MFs

The crispness of the Pattern Trees are also illustrated in the accuracy vs. threshold plot of figure 4.23, where there is exists no flexibility in varying an operating point threshold.

4.3.1.4 The Yager T-Conorm Parameter w

The Yager parameter *w* was value varied to see its effect on the ROC curve of the Pattern Tree. The results are shown in figure 4.24. Figure 4.25, which shows the effect of the Yager parameter on Pattern Tree maximum accuracy and AUC, illustrates why a default value of 3 was originally chosen.

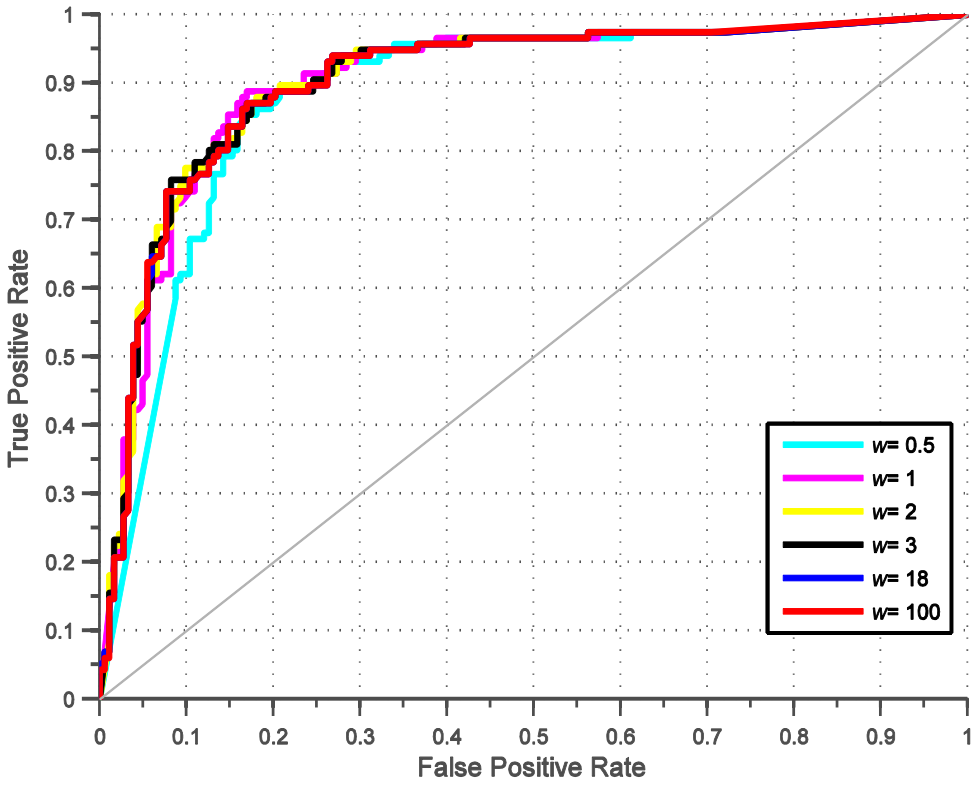

Figure 4.24 ROC Curves for PT6 on EIAFULL, various *w*

There is some decline in classification performance as *w* drops below 1 and approaches 0. Besides that, however, there is not much of an improvement over values lower than infinity, where the Yager OR degenerates into the standard MAX.

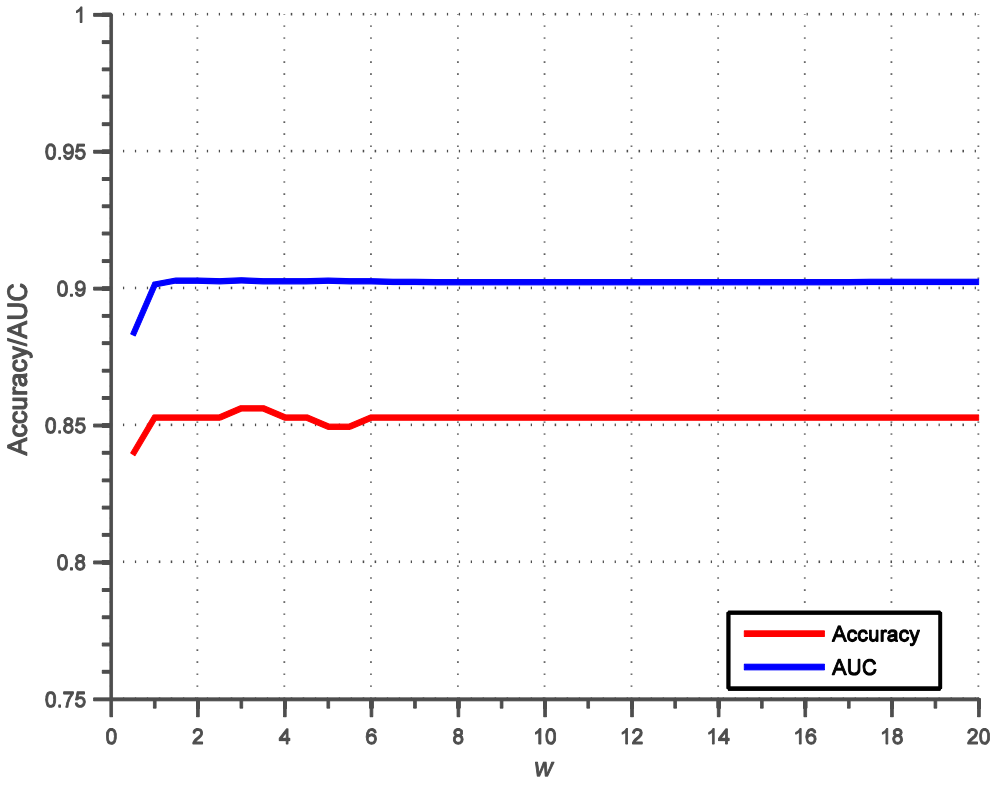

Figure 4.25 PT6 on EIAGTFULL, Accuracy and ROC AUC vs. *w*

Once again, the Yager OR shows little advantage over the standard MAX in terms of accuracy or AUC.

.3.2 Fuzzy K-Nearest-Neighbor Results

The subplots for the ROC curves of the 12-dimensional FkNN classifiers are shown in figure 4.26, along with the most accurate operating point of each ROC. Along the top row are the ROC curves for k = 1, 3, and 5 respectively, and along the bottom are the ROC curves for k = 7, 9, and 11. The Fuzzy 11-Nearest Neighbor performed the most accurately, and its operating point is shown in the confusion matrix in table 4.7. The actual values from ground truth span across the columns while the predicted values from the FkNN span down the rows. Similarly, the results for the 6-dimensional FkNN

classifiers are shown in figure 4.27 and table 4.8. Overall, the ROCs for the 12 feature FkNN classifiers perform slightly better than the FkNNs using 6 features.

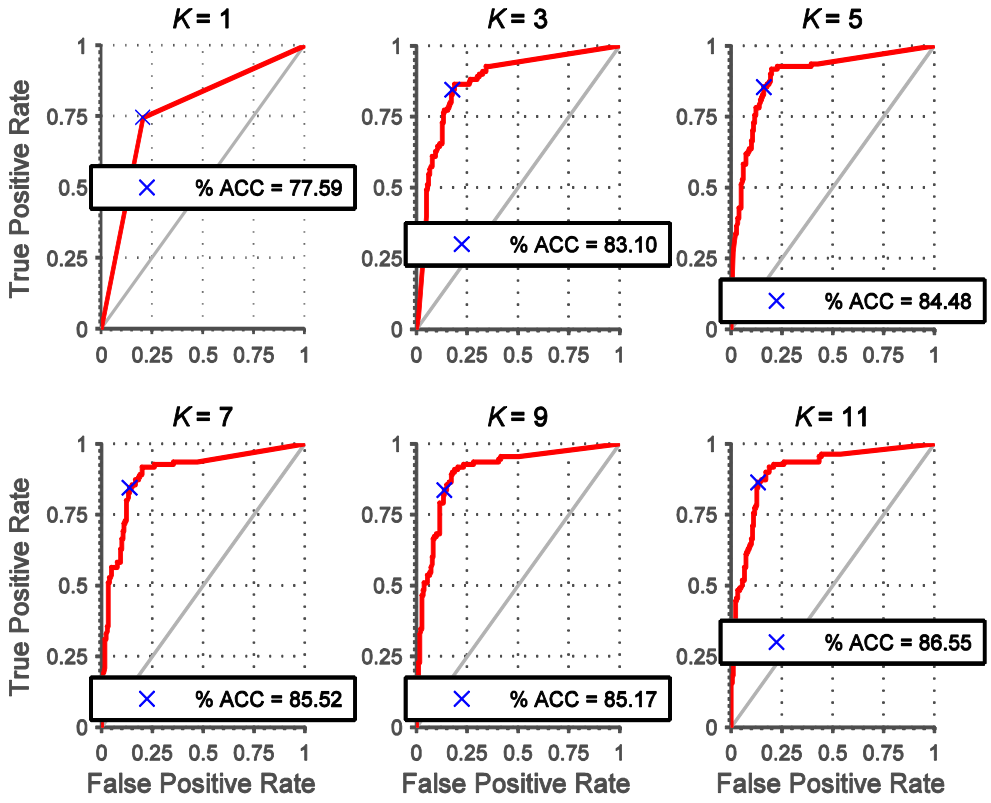

Figure 4.26 ROC Curves for 12-Feature FkNN, various *K*

Table 4.7 Confusion Matrix for 12-Feature FkNN, *K* = 11, at Maximum Accuracy

Samples	251	**Actual**	
Accuracy	**0.8655**	Relevant	Irrelevant
Predicted	Relevant	95	24
	TPR/FPR	**0.8636**	**0.1333**
	Irrelevant	15	156
	FNR/TNR	**0.1364**	**0.8667**
		110	180

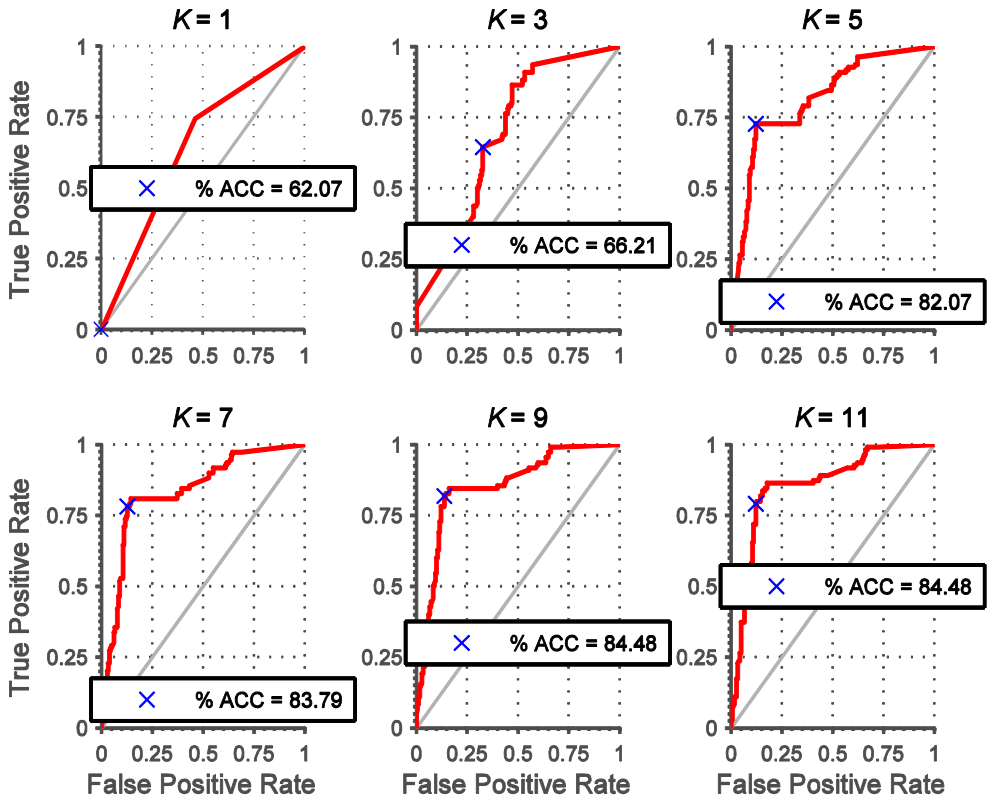

Figure 4.27 ROC Curves for 6-Feature FkNN, various K

Table 4.8 Confusion Matrix for 6-Feature FkNN, *K* = 11, at Maximum Accuracy

Samples	245	**Actual**	
Accuracy	**0.8448**	Relevant	Irrelevant
Predicted	Relevant	87	22
	TPR/FPR	**0.7909**	**0.1222**
	Irrelevant	23	158
	FNR/TNR	**0.2091**	**0.8778**
		110	180

4.3.3 Neural Network Results

Figure 4.28 illustrates how the MATLAB Neural Network toolbox trains a neural network using a training and validation dataset. Figure 4.29 and table 4.9 show the ROC curve for the 12-dimensional NN after 10-fold cross validation and the most accurate operating point, respectively. Note that because the training, validation, and test datasets are randomly chosen each time, the ROC curve differs every time the experiment runs. Figure 4.30 and table 4.10 show similar results for the 6-dimensional NN.

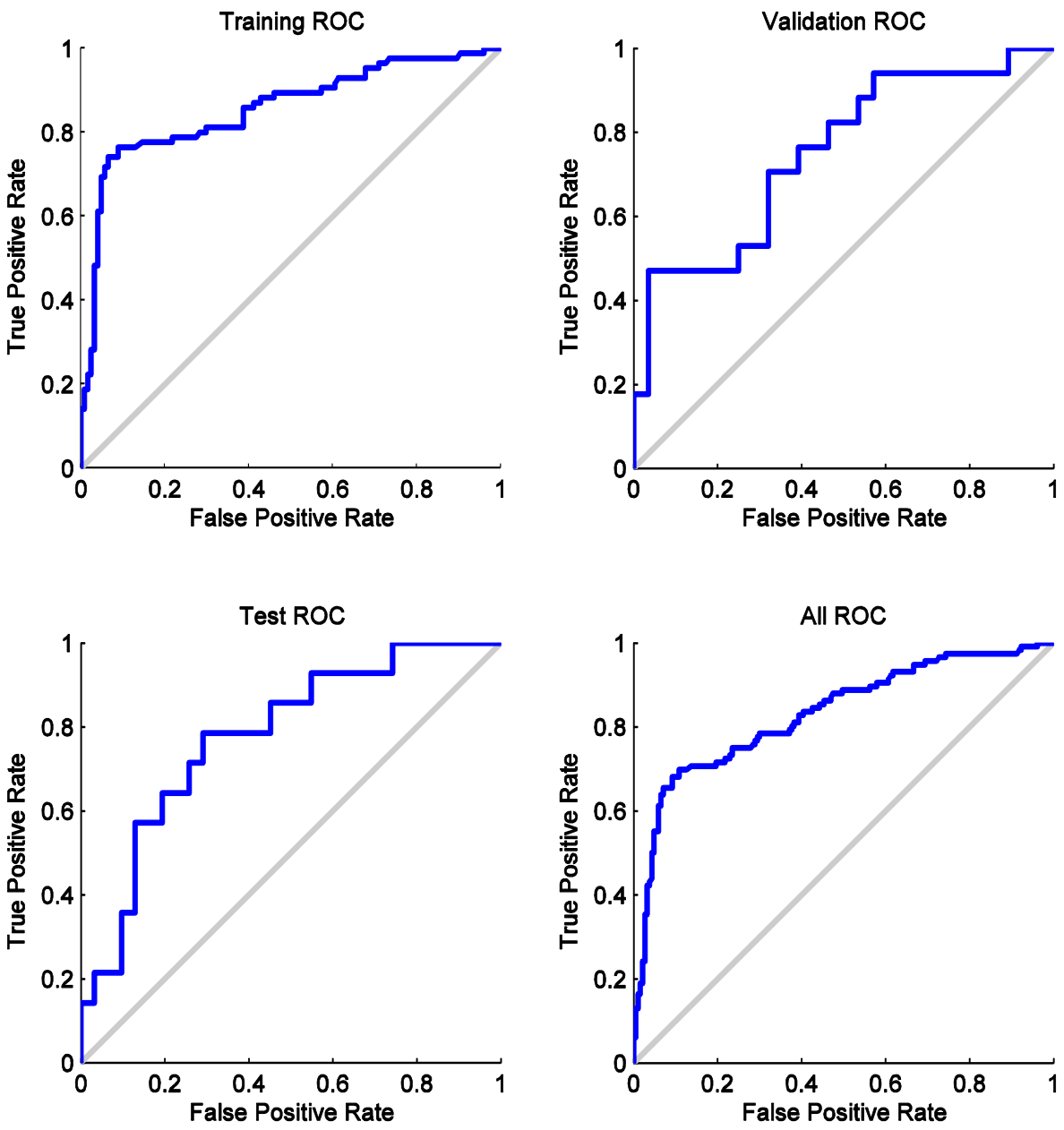

Figure 4.28 ROC Curves for Neural Network Training

The top left ROC indicates resubstitution performance on the training data. The top right indicates performance on the validation set, which is used to determine termination. The bottom left ROC is the one that is eventually taken into account into the cross validation.

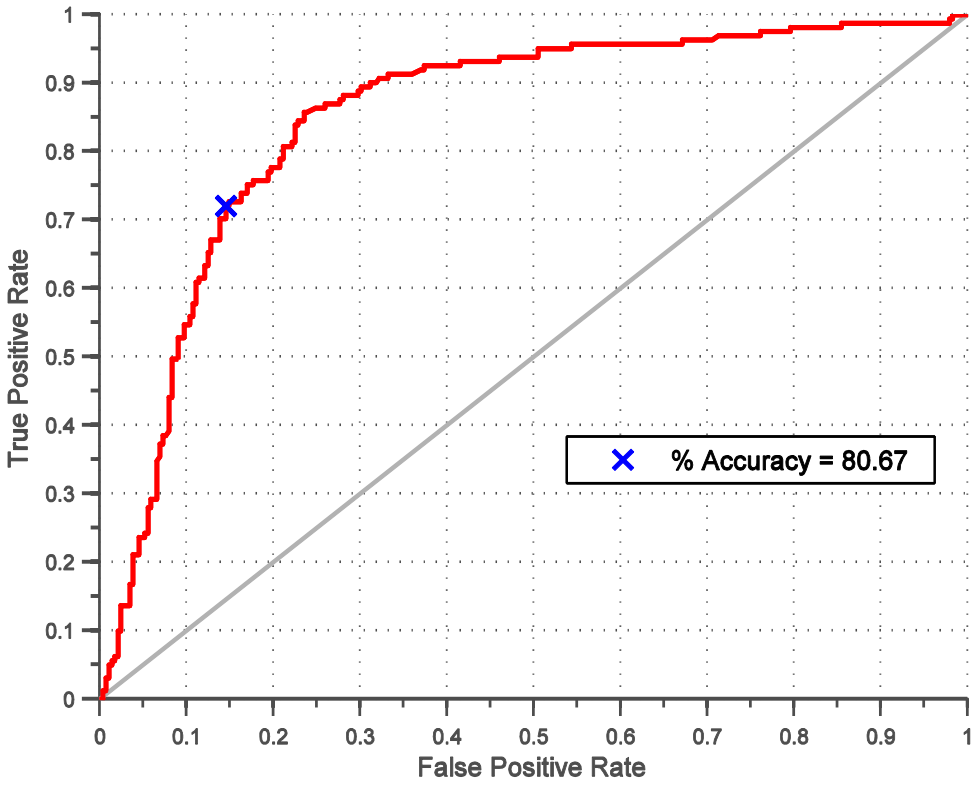

Figure 4.29 ROC Curve for 12-Feature NN

The 12 feature neural network outperforms the FPT but not the FkNN.

Table 4.9 Confusion Matrix for 12-Feature NN at Maximum Accuracy

# Samples	363	**Actual**	
Accuracy	**0.8067**	Relevant	Irrelevant
Predicted	Relevant	116	42
	TPR/FPR	**0.7205**	**0.1453**
	Irrelevant	45	247
	FNR/TNR	**0.2795**	**0.1073**
		161	289

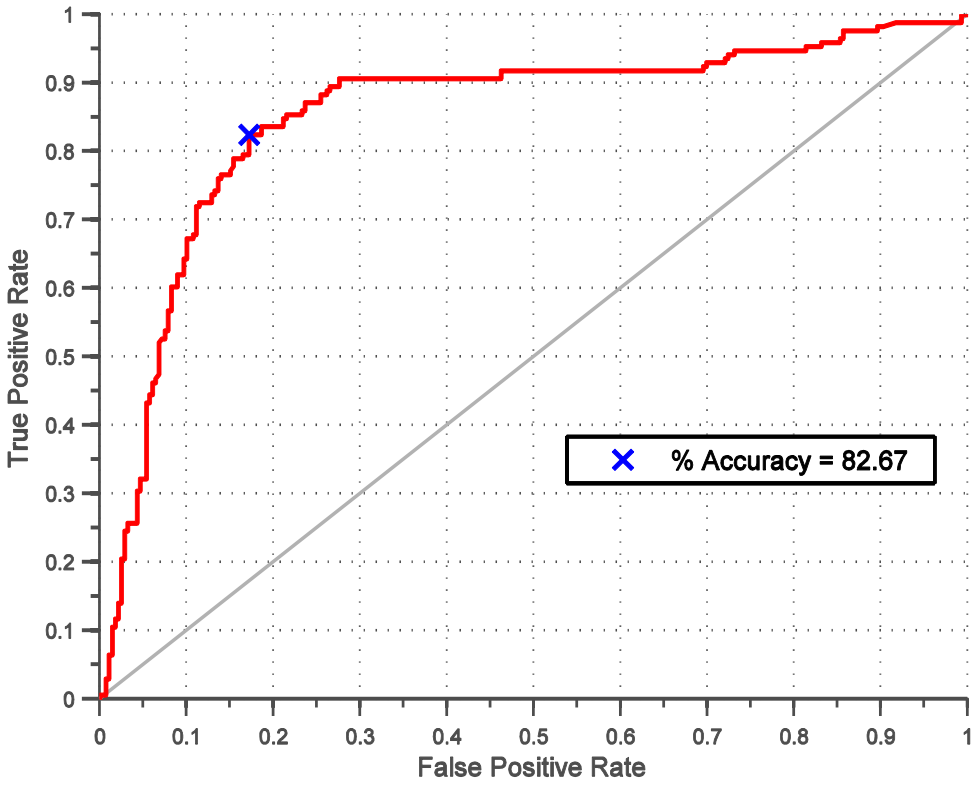

Figure 4.30 Average ROC curve for 6-Feature NN

The 6 feature NN does not outperform either the FPT or the FkNN.

Table 4.10 Confusion Matrix for 6-Feature NN Maximum Accuracy

# Samples	372	**Actual**	
Accuracy	**0.8267**	Relevant	Irrelevant
Predicted	Relevant	141	48
	TPR/FPR	**0.8246**	**.1720**
	Irrelevant	30	231
	FNR/TNR	**0.1754**	**0.8280**
		171	279

4.3.4 SVM Results

The ROC curve and confusion matrix for the 12-D and 6-D Linear SVMs are presented first, followed by the 12-D and 6-D RBF SVMs. Each confusion matrix following a ROC curve represents the operating point with the maximum accuracy indicated by the blue "x."

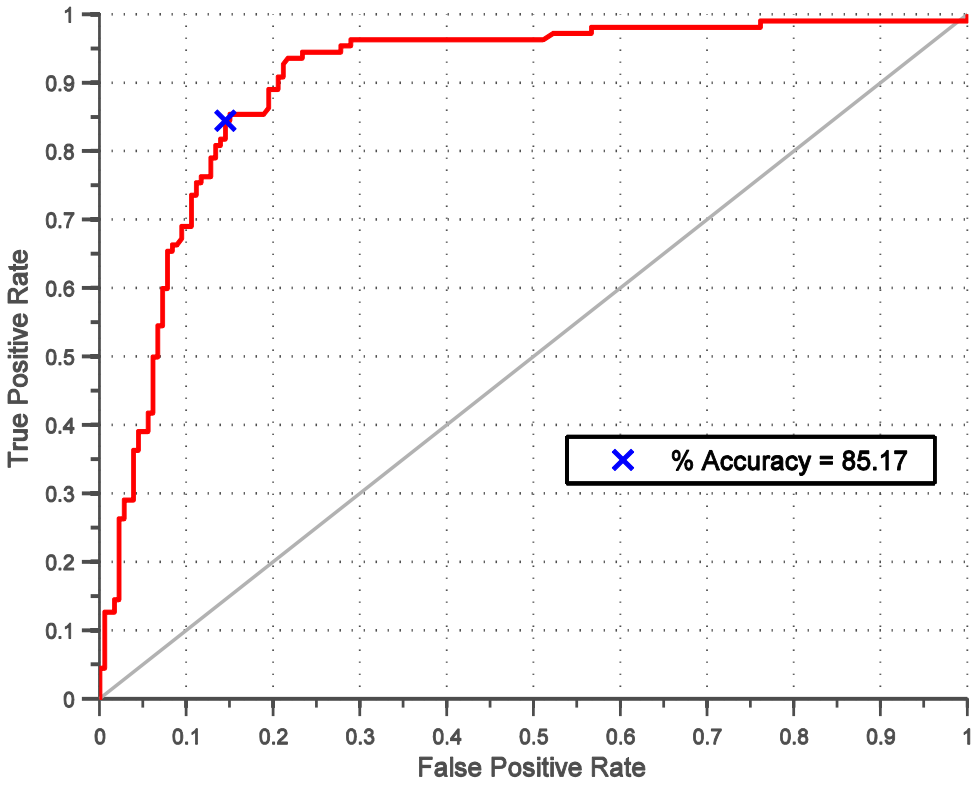

Figure 4.31 ROC Curve for 12-Feature SVM with Linear Kernel

The linear 12 feature SVM performs almost as well as the FkNN and outperforms the FPT and the NN.

Table 4.11 Confusion Matrix for SVMLIN12 at Maximum Accuracy

# Samples	247	**Actual**	
Accuracy	**0.8517**	Relevant	Irrelevant
Predicted	Relevant	93	26
	TPR/FPR	**0.8455**	**0.1444**
	Irrelevant	17	154
	FNR/TNR	**0.1545**	**0.8556**
		110	180

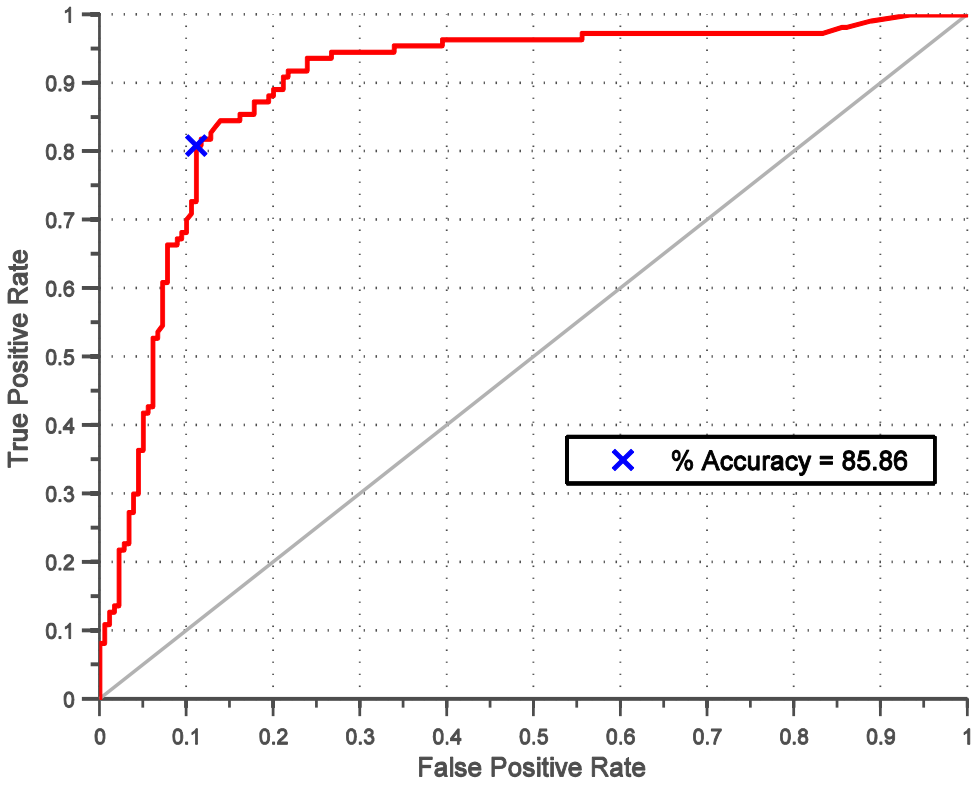

Figure 4.32 ROC Curve for 6-Feature SVM with Linear Kernel

The linear 6 feature SVM outperforms the other 6 feature classifiers.

Table 4.12 Confusion Matrix for SVMLIN6 at Maximum Accuracy

Samples	249	**Actual**	
Accuracy	**0.8586**	Relevant	Irrelevant
Predicted	Relevant	89	20
	TPR/FPR	**0.8091**	**0.1111**
	Irrelevant	21	160
	FNR/TNR	**0.1909**	**0.8889**
		110	180

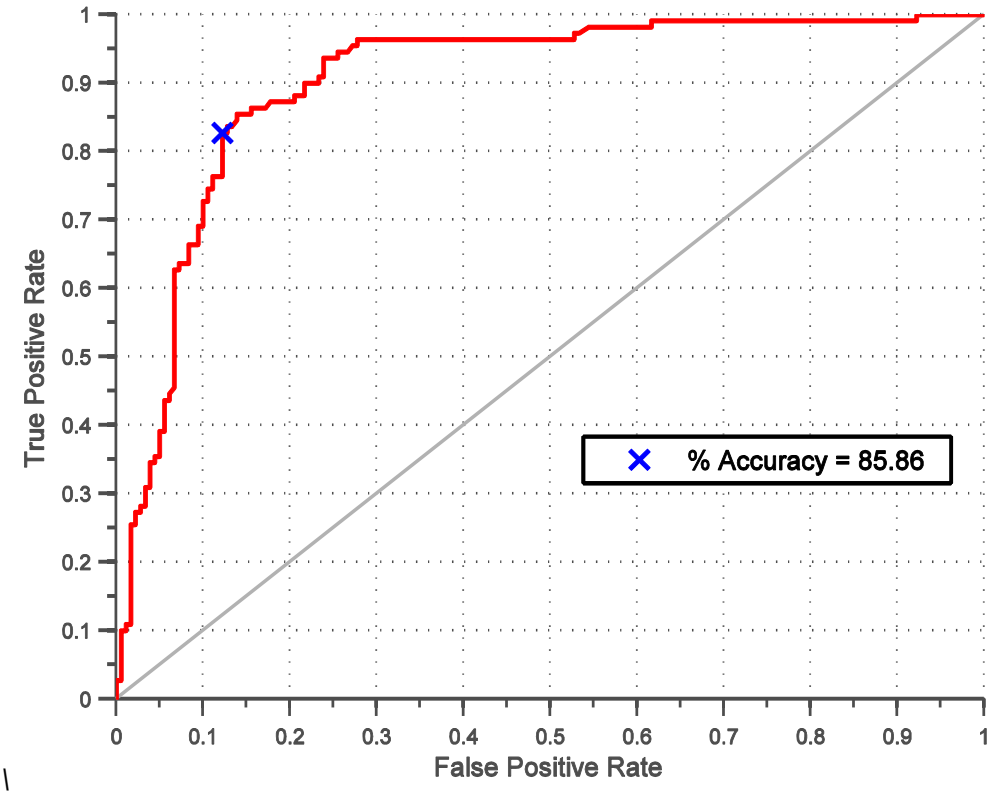

Figure 4.33 ROC Curve for 12-Feature SVM with RBF Kernel

The 12 feature SVM with the RBF kernel performs slightly better than the linear 12 feature SVM.

Table 4.13 Confusion Matrix for SVMRBF12 at Maximum Accuracy

Samples	249	**Actual**	
Accuracy	**0.8586**	Relevant	Irrelevant
Predicted	Relevant	91	22
	TPR/FPR	**0.8273**	**0.1222**
	Irrelevant	19	158
	FNR/TNR	**0.1727**	**0.8778**
		110	180

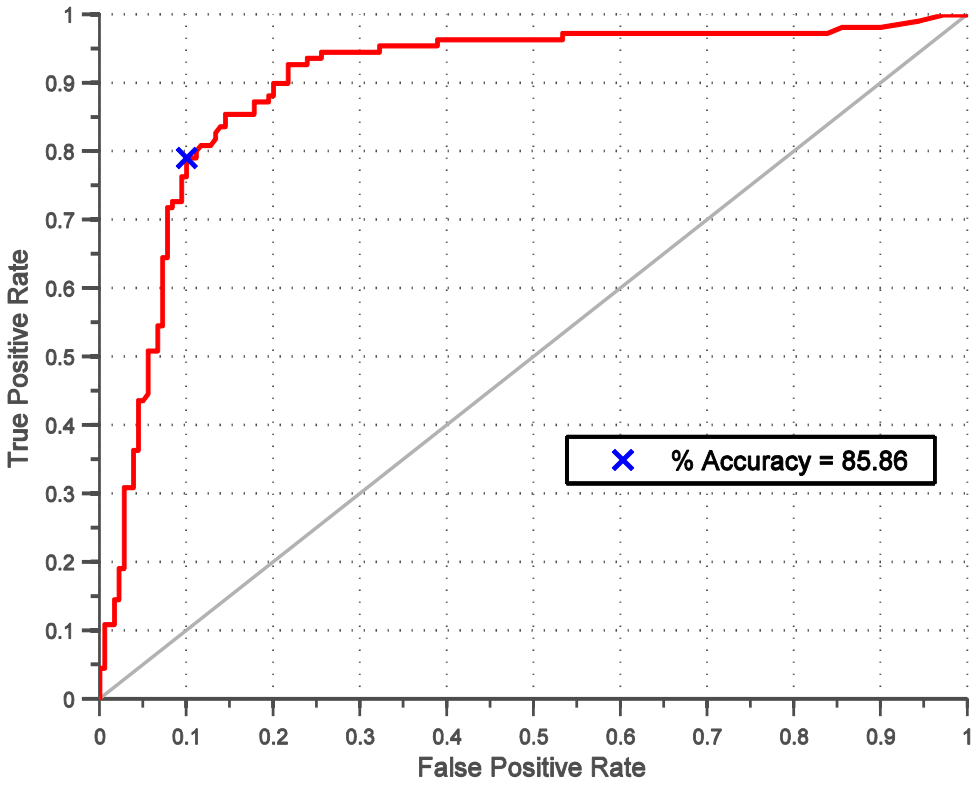

Figure 4.34 ROC Curve for 6-Feature SVM with RBF Kernel

The 6 feature SVM with the RBF kernel performs comparably with the linear version and outperforms the other three classifiers.

Table 4.14 Confusion Matrix for SVMRBF6 at Maximum Accuracy

Samples	249	**Actual**	
Accuracy	**0.8586**	Relevant	Irrelevant
Predicted	Relevant	87	18
	TPR/FPR	**0.7909**	**0.1000**
	Irrelevant	23	162
	FNR/TNR	**0.2091**	**0.9000**
		110	180

4.3.5 Classifier Comparison

Finally, this subsection compares the ROC curve performance of the four classification methods. Figures 4.35 shows the ROC curves for the best 12-dimensional classifier from each method (both SVMs, Linear and RBF kernels, are included). Table 4.15 compares their training methods, accuracy, TPR, and FPR. Figure 4.36 and table 4.16 do the same for the 6-dimensional classifiers.

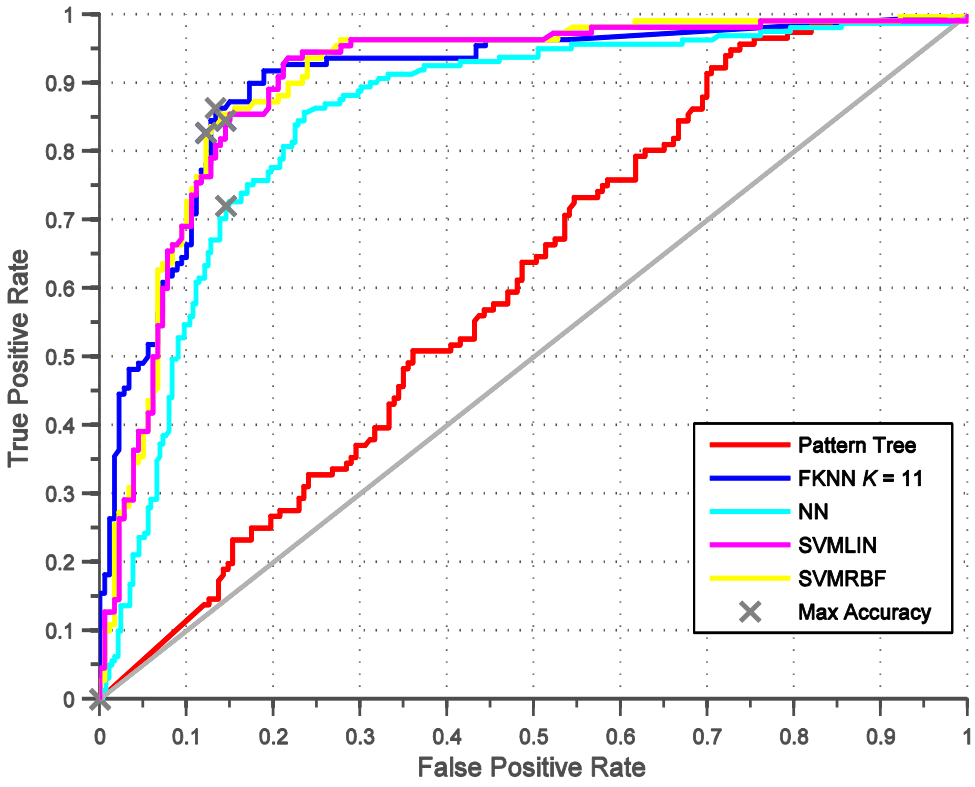

Figure 4.35 ROC Curves for 12-Dimensional Classifiers

The three trainable methods result in good ROC curves while the FPT, which relies on domain knowledge, performs poorly. Using all available alert features, the FkNN performs the best.

Table 4.15 Comparison of 12-Dimensional Classifiers

Classifier	Training	Testing	Accuracy	TPR	FPR
Pattern Tree	None	299 Samples	0.6120	0.0000	**0.0000**
FkNN *K*=11	90% Training	10% Testing	**0.8655**	**0.8636**	0.1333
NN	70% Training, 15% Validation	15% Testing	0.8067	0.7205	0.1453
SVMLIN	90% Training	10% Testing	0.8517	0.8455	0.1444
SVMRBF	10% Testing	10% Testing	0.8586	0.8273	0.1222

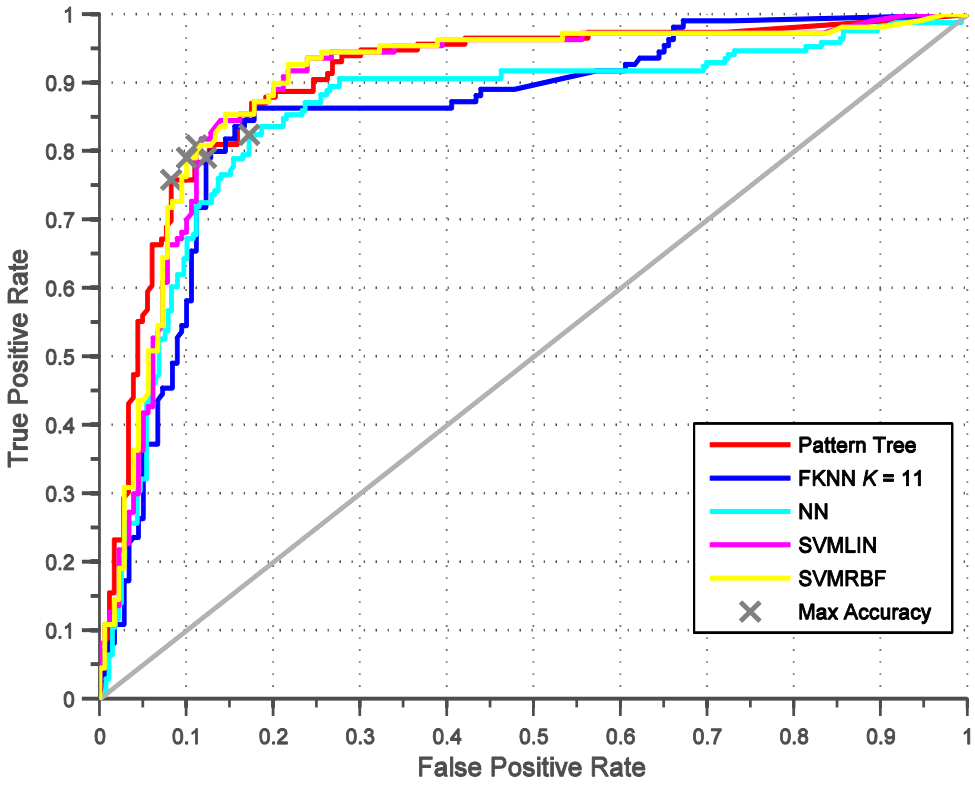

Figure 4.36 ROC Curves for 6-Dimensional Classifiers

Using only 6 features, the SVMs and the FPT perform the best at correctly classifying alert days.

Table 4.16 Comparison of 6-Dimensional Classifiers

Classifier	Training	Testing	Accuracy	TPR	FPR
Pattern Tree	None	299 Samples	0.8562	0.7586	**0.0820**
FkNN *K*=11	90% Training	10% Testing	0.8448	0.7909	0.1222
NN	70% Training, 15% Validation	15% Testing	0.8267	**0.8246**	0.1720
SVMLIN	90% Training	10% Testing	**0.8586**	0.8091	0.1111
SVMRBF	10% Testing	10% Testing	**0.8586**	0.7909	0.1000

The table summarizes the maximum accuracy points of each of the

Chapter 5 - Discussion

5.1 Single-Dimensional Features

Out of the single-dimensional EIA experiments, the 0-threshold living room activity parameter at night had the best performance in terms of AUC, with the 0-threshold and 10-threshold kitchen activity parameter at night coming next, followed by the 10-threshold and then 0-threshold bathroom activity alert parameter for the full day. Interestingly enough, the bathroom activity EIAs at night did not discriminate as well even though the clinical researchers mentioned numerous times that bathroom EIAs at night tended to be more clinically relevant. One reason the kitchen activity and living room activity parameter ROC curves looked so well is that there were very few good EIAs in those two categories compared to the number of poor EIAs, and most of the EIAs for those two categories at night were rated good. The bathroom activity and bed restlessness EIAs had a more even distribution of good and poor alerts. Regardless, when any of the EIA features are used versus the EIAFULL dataset, they are all essentially indiscriminate. This simply reiterates the fact that a single EIA feature cannot account for alert days when another EIA feature would generate a clinically relevant alert. Multiple EIA features must be used in order to account for more of the possible clinically relevant situations.

5.2 Multi-Dimensional Classifiers

In general, the multi-dimensional classifiers more than doubled the 38.80% accuracy (from EIAFULL in table 4.2) of the ensemble of single-dimensional EIAs. In every case except for the 12-dimensional Pattern Tree, the classifiers would have

generated MEIAs that would have been over 80% accurate. That indicates accurate decision boundaries can be created from the multi-dimensional EIA feature space and other machine learning algorithms outside of the four in this study would perform comparably.

5.2.1 Pattern Tree

The Pattern Tree was used to make an in-depth investigation of the feature space. Its intuitive nature provides the ability to interpret the feature space in a way that makes sense logically and linguistically. For instance, it makes sense to the nurses that combining four different alert parameters at night with a couple during the entire day, i.e. bathroom activity at night or bed restlessness at night or kitchen activity at night or living room activity at night or bathroom activity for the full day or bed restlessness for the full day, would provide a good decision boundary for the feature space instead of describing the boundary as a Euclidean distance from the origin.

From the multi-dimensional, single alert parameter Pattern Trees, the combination of bathroom activity increases at night and during the full day ($T = 10$) discriminated the best between good and poor alert days, with the same combination except with bed restlessness ($T = 0$) performing second best in terms of AUC. This matches more closely with what the clinical researchers recommended. Combining alert periods for the kitchen activity and living room activity alert parameters resulted in worse ROC curves than the ROC curves from the nighttime version of those alert parameters. This indicates that those alert parameters are more clinically relevant at night, which also agrees with what the clinical researchers have suggested.

At first, the 12-dimesional Pattern Trees from figure 4.35 may seem like they improved the accuracy of the EIAs to 61.20%, but that is only because their maximum operating point is the trivial case where all alert days are classified as poor, and since there are more poor alert days than good, the accuracy increases implicitly. Configuration 6 has an AUC of 0.604, but even though the AUC is above the chance AUC of 0.5, it is not enough to merit the 12-dimensional Pattern Tree as a viable option for generating MEIAs.

The 6-dimensional case, however, proves much more effective, raising the accuracy to 85.62%. This performance is very interesting because the Pattern Tree was built with just an OR operator and recommendations from the clinical researchers. This is essentially a classifier that has been built as an expert system without any training data. This changes the machine learning problem from worrying about what learning parameters to set to asking experts how to aggregate features. A linguistic response could be used to create an appropriate Pattern Tree.

The PCA visualization provides very good insight on the failure of the 12-dimensional and success of the Yager OR-based 6-dimensional Pattern Tree. From the 6-dimensional visualization, many of the poor alert days cluster around the origin and generally the good alert days are farther from the origin. The Yager OR, which is essentially a distance measure, can create a hyper-sphere decision boundary that would classify this feature space accurately. On the other hand, the 12-dimensional PCA does not do a good job of clustering one group and spreading out the other, so the 12-dimensional Pattern Tree does not classify very accurately.

The membership function experiments present some interesting effects of the crispness of the membership function to the OR-based Pattern Tree's threshold flexibility. Although the ROC curves are very similar for the fuzzy membership functions and allow for many operating points, the Pattern Tree with a crisp membership function degenerates into a crisp PT with only one operating point. As the membership functions become more crisp, i.e. they approach a vertical jump from 0 to 1 at a single threshold (from the inverted Gaussian to the trapezoidal to the sigmoid to the crisp), the threshold bands in figures 4.22 and 4.23 decrease.

Varying the Yager parameter *w* also has little effect on the ROC curves of the 6-dimensional Pattern Tree except when *w* becomes very small. Regardless, the small bump in the AUC curve at $w = 3$ illustrates why the original value was chosen for the Pattern Tree experiments. Although the Yager OR provides a parameter to make a more optimistic aggregation than the standard MAX OR, the advantages are not apparent in Pattern Trees using only Yager OR operators.

5.2.2 Fuzzy K-Nearest-Neighbor

Out of the 12-dimensional classifiers, the Fuzzy 11-Nearest Neighbor had both the best accuracy (86.55%) and best true positive, or hit, rate. For the 6-dimensional case however, it only outperformed the Neural Network in terms of accuracy. The FkNN would seem like a viable option for MEIAs provided that enough training samples exist and that they cover much of the feature space for both clinically relevant and poor alert days. The simplicity of implementing and training the FkNN is definitely an advantage over the other classifiers and with the relatively small sizes of current EIA ground truth datasets, memory space should not be a concern at this point.

5.2.3 Neural Network

In terms of accuracy, the Neural Network was only superior to the trivial 12-dimensional Pattern Tree in terms of accuracy, although its most accurate 6-dimensional operating point had a higher TPR than those of the other 6-dimensional classifiers. Heuristically, Neural Networks should be trained with a number of samples greater than 10 times the number of network connections. Thus in the 12-dimensional case, where 5 hidden nodes results in 60 network connections, the Neural Network would need at least 600 training samples to consistently find a good solution. There are only 299 samples total in the ground truth dataset and only 70% of those are used for training in each fold, so even for the 6-dimensional neural network the number of training samples is suboptimal. Regardless, the Neural Network was still able to achieve at least 80% accuracy in both feature spaces.

5.2.4 Support Vector Machine

In the 6-dimensional case, both the Linear and RBF SVMs attained the highest accuracy of 85.86% compared to the other three classification methods. Both the 12-D and 6-D SVMs were able to find both a good linear decision boundary and an RBF boundary.

5.3 Early Illness Alerts Dataset

Although preliminary analyses have been possible with the 299 samples from the EIAFULL ground truth dataset, it would be more beneficial to include not only more alert days, but also more alert parameters. This will come in time as more EIAs are generated each day and more of the rarer EIAs receive clinical feedback. So far the

ground truth datasets that have been built are based on the ratings from the clinical researchers only. It will be worth investigating ground truth based on the ratings of not only the TigerPlace care providers, but also ground truth based on the combined ratings of clinical researchers and TigerPlace care providers. Specifically with the feedback from the care providers, more complex Pattern Trees that use AND or NOT operators in addition to the OR operators will be viable, as the care providers can provide better clues on situations that may be relevant as well as ones that need to be left out.

Chapter 6 - Conclusion

This thesis presented a framework and methods to detect early signs of illness in older adults living in an independent living facility, generate alerts, collect clinical alert feedback, and build ground truth for further analyses. The ground truth for Early Illness Alerts was successfully built and used to build four kinds of classifiers based on different machine learning algorithms. Even with a suboptimal framework with sensor networks that generate coarse, noisy sensor data, combining Early Illness Alert features into multi-dimensional features allowed four different kinds of classifiers to achieve over 80% classification accuracy of clinically relevant alert days, more than double the 38.08% from the individual EIAs. The 6-dimensional Pattern Tree was able to achieve 85.56% with no training data. The Fuzzy K-Nearest Neighbor is simple to apply so long as enough of the feature space is covered by the training samples. The Neural Network achieved over 80% accuracy in spite of a small training dataset. The Support Vector Machine successfully found accurate hyperplanes using both Linear and Radial Basis Function kernels.

Overall, this research has shown promise in developing a remote, continuous, and automated integrated sensor network for the smart home application of eldercare health monitoring and alerting. The results have demonstrated that by using a sliding window, early changes in general behaviors, as opposed to more specific and complicated ADLs, actually provide clinically relevant insight on an elderly residents behavior and can notify caregivers in a timely manner. Moreover, combining changes in multiple alert parameters allows multiple supervised pattern recognition methods to correctly classify alert days

into good (abnormal) and poor (normal) alert days, so that in the future, EIAs will be more accurate.

This research touches on just the surface of possibilities for using sensor networks to detect early illness alerts. Only the alert ratings from the clinical researchers on a small subset of the available sensor and alert data have been used to test the multidimensional classifiers. A larger alert ground truth dataset that also includes the ratings from the TigerPlace staff can be incorporated into further testing. It will be worth exploring more complicated Fuzzy Pattern Trees that use both AND and OR operators with features other than increases in different alert parameters. Those other features could also be used for the other classifiers. Other methods like a System of Fuzzy Rules could be explored using the sliding window approach. This work provides a basis for further investigation in integrated sensor networks and early illness alerts.

References

[1] U. S. Census Bureau, Populations Division. *National population projections, population pyramids*. Available: http://www.census.gov/population/www/projections/natchart.html

[2] M. J. Rantz, K. D. Marek, M. Aud, H. W. Tyrer, M. Skubic, G. Demiris, and A. Hussam, "A technology and nursing collaboration to help older adults age in place," *Nurs Outlook,* vol. 53, pp. 40-5, Jan-Feb 2005.

[3] G. Demiris and B. Hensel, "Technologies for an aging society: a systematic review of "smart home" applications," *Yearb Med Inform,* vol. 2008, pp. 33-40, 2008.

[4] J. Stankovic, Q. Cao, T. Doan, L. Fang, Z. He, R. Kiran, S. Lin, S. Son, R. Stoleru, and A. Wood, "Wireless sensor networks for in-home healthcare: Potential and challenges," 2005, pp. 2-3.

[5] Q. Yang, "Activity recognition: linking low-level sensors to high-level intelligence," 2009, pp. 20-25.

[6] M. Wallace, "Katz index of independence in activities of daily living (ADL)," *image,* vol. 1, p. 2, 2007.

[7] R. Beckwith, "Designing for ubiquity: The perception of privacy," *Pervasive Computing, IEEE,* vol. 2, pp. 40-46, 2003.

[8] B. Celler, W. Earnshaw, E. Ilsar, L. Betbeder-Matibet, M. Harris, R. Clark, T. Hesketh, and N. Lovell, "Remote monitoring of health status of the elderly at home. A multidisciplinary project on aging at the University of New South Wales," *International journal of bio-medical computing,* vol. 40, pp. 147-155, 1995.

[9] N. M. Barnes, N. H. Edwards, D. A. D. Rose, and P. Garner, "Lifestyle monitoring-technology for supported independence," *Computing & Control Engineering Journal,* vol. 9, pp. 169-174, 1998.

[10] B. A. Majeed and S. J. Brown, "Developing a well-being monitoring system—Modeling and data analysis techniques," *Applied Soft Computing,* vol. 6, pp. 384-393, 2006.

[11] C. Kidd, R. Orr, G. Abowd, C. Atkeson, I. Essa, B. MacIntyre, E. Mynatt, T. Starner, and W. Newstetter, "The aware home: A living laboratory for ubiquitous computing research," *Cooperative buildings. Integrating information, organizations, and architecture,* pp. 191-198, 1999.

[12] A. P. Glascock and D. M. Kutzik, "Behavioral telemedicine: A new approach to the continuous nonintrusive monitoring of activities of daily living," *Telemedicine journal,* vol. 6, pp. 33-44, 2000.

[13] S. S. Intille, "Designing a home of the future," *Pervasive Computing, IEEE,* vol. 1, pp. 76-82, 2002.

[14] L. Bao and S. Intille, "Activity recognition from user-annotated acceleration data," *Pervasive Computing,* pp. 1-17, 2004.

[15] S. Intille, K. Larson, E. Tapia, J. Beaudin, P. Kaushik, J. Nawyn, and R. Rockinson, "Using a live-in laboratory for ubiquitous computing research," *Pervasive Computing,* pp. 349-365, 2006.

[16] K. Z. Haigh, L. M. Kiff, J. Myers, V. Guralnik, C. W. Geib, J. Phelps, and T. Wagner, "The independent lifestyle assistanttm (ilsa): Ai lessons learned," 2004.
[17] J. Waterman, G. W. Challen, and M. Welsh, "Peloton: Coordinated resource management for sensor networks," 2009, pp. 9-9.
[18] K. Lorincz, B. rong Chen, J. Waterman, G. Werner-Allen, and M. Welsh, "Pixie: An operating system for resource-aware programming of embedded sensors," 2008.
[19] M. Chan, E. Campo, and D. Esteve, "Assessment of activity of elderly people using a home monitoring system," *Int J Rehabil Res,* vol. 28, pp. 69-76, Mar 2005.
[20] A. Fleury, M. Vacher, and N. Noury, "SVM-based multimodal classification of activities of daily living in Health Smart Homes: sensors, algorithms, and first experimental results," *IEEE Trans Inf Technol Biomed,* vol. 14, pp. 274-83, Mar 2010.
[21] T. Van Kasteren, A. Noulas, G. Englebienne, and B. Kröse, "Accurate activity recognition in a home setting," 2008, pp. 1-9.
[22] T. S. Barger, D. E. Brown, and M. Alwan, "Health-status monitoring through analysis of behavioral patterns," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on,* vol. 35, pp. 22-27, 2005.
[23] M. Alwan, S. Dalal, D. Mack, S. Kell, B. Turner, J. Leachtenauer, and R. Felder, "Impact of monitoring technology in assisted living: outcome pilot," *Information Technology in Biomedicine, IEEE Transactions on,* vol. 10, pp. 192-198, 2006.
[24] S. Dalal, M. Alwan, R. Seifrafi, S. Kell, and D. Brown, "A rule-based approach to the analysis of elders' activity data: detection of health and possible emergency conditions," 2005.
[25] G. Virone, A. Wood, L. Selavo, Q. Cao, L. Fang, T. Doan, Z. He, and J. Stankovic, "An advanced wireless sensor network for health monitoring," 2006, pp. 2-4.
[26] T. L. Hayes, M. Pavel, and J. A. Kaye, "An unobtrusive in-home monitoring system for detection of key motor changes preceding cognitive decline," *Conf Proc IEEE Eng Med Biol Soc,* vol. 4, pp. 2480-3, 2004.
[27] J. A. Kaye, S. A. Maxwell, N. Mattek, T. L. Hayes, H. Dodge, M. Pavel, H. B. Jimison, K. Wild, L. Boise, and T. A. Zitzelberger, "Intelligent Systems For Assessing Aging Changes: home-based, unobtrusive, and continuous assessment of aging," *J Gerontol B Psychol Sci Soc Sci,* vol. 66 Suppl 1, pp. i180-90, Jul 2011.
[28] F. Wang, E. Stone, W. Dai, T. Banerjee, J. Giger, J. Krampe, M. Rantz, and M. Skubic, "Testing an in-home gait assessment tool for older adults," 2009, pp. 6147-6150.
[29] S. Wang, "Change Detection for Eldercare Using Passive Sensing," PhD Research, Electrical and Computer Engineering, University of Missouri - Columbia, 2011.
[30] M. J. Moore, "PIR sensing array for fall detection," M.S. Research, Computer Science, University of missouri - Columbia, 2011.
[31] D. Heise, L. Rosales, M. Skubic, and M. Devaney, "Refinement and evaluation of a hydraulic bed sensor," 2011, pp. 4356-4360.

[32] D. C. Mack, M. Alwan, B. Turner, P. Suratt, and R. A. Felder, "A passive and portable system for monitoring heart rate and detecting sleep apnea and arousals: Preliminary validation," 2006, pp. 51-54.
[33] *MS16A ACTIVEEYE INSTRUCTIONS*. Available: http://www.x10.com/instructions/ms16a_inst.htm
[34] Z. Huang, T. D. Gedeon, and M. Nikravesh, "Pattern trees induction: A new machine learning method," *Fuzzy Systems, IEEE Transactions on,* vol. 16, pp. 958-970, 2008.
[35] R. Senge, Hu, x, and E. llermeier, "Top-Down Induction of Fuzzy Pattern Trees," *Fuzzy Systems, IEEE Transactions on,* vol. 19, pp. 241-252, 2011.
[36] C. Olaru and L. Wehenkel, "A complete fuzzy decision tree technique," *Fuzzy sets and systems,* vol. 138, pp. 221-254, 2003.
[37] D. DuBois and H. M. Prade, *Fuzzy sets and systems: theory and applications* vol. 144: Academic Pr, 1980.
[38] *Wiley Online Library*. Available: http://media.wiley.com/wires/WIDM/WIDM34/nfig003.jpg
[39] R. R. Yager, "On ordered weighted averaging aggregation operators in multicriteria decisionmaking," *Systems, Man and Cybernetics, IEEE Transactions on,* vol. 18, pp. 183-190, 1988.
[40] R. R. Yager, "On a general class of fuzzy connectives," *Fuzzy sets and systems,* vol. 4, pp. 235-242, 1980.
[41] J. M. Keller, "A Fuzzy-Nearest Neighbor Algorithm JAMES M. KELLER, MICHAEL R. GRAY, and JAMES A. GIVENS, JR," *IEEE Transactions on Systems, Man, and Cybernetics,* vol. 15, p. 581, 1985.
[42] G. McLachlan, "Mahalanobis distance," *Resonance,* vol. 4, pp. 20-26, 1999.
[43] T. Cover and P. Hart, "Nearest neighbor pattern classification," *Information Theory, IEEE Transactions on,* vol. 13, pp. 21-27, 1967.
[44] J. A. Anderson and J. Davis, *An introduction to neural networks* vol. 1: MIT Press, 1995.
[45] M. F. Møller, "A scaled conjugate gradient algorithm for fast supervised learning," *Neural networks,* vol. 6, pp. 525-533, 1993.
[46] J. A. K. Suykens and J. Vandewalle, "Least Squares Support Vector Machine Classifiers," *Neural Processing Letters,* vol. 9, pp. 293-300, 1999.
[47] E. Hüllermeier, "Fuzzy machine learning and data mininga," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery,* vol. 1, pp. 269-283, 2011.
[48] C. Huggard, "Eldercare Technology Logging & Alerting Framework/API," M.S. Research, Computer Science, University of Missouri - Columbia, 2008.
[49] S. Ondrej, B. Zdenek, F. Petr, and H. Ondrej, "ZigBee Technology and Device Design," in *Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies, 2006. ICN/ICONS/MCL 2006. International Conference on*, 2006, pp. 129-129.
[50] E. Gamma, *Design patterns: elements of reusable object-oriented software*: Addison-Wesley Professional, 1995.

[51] G. L. Alexander, B. J. Wakefield, M. Rantz, M. Skubic, M. A. Aud, S. Erdelez, and S. A. Ghenaimi, "Passive Sensor Technology Interface to Assess Elder Activity in Independent Living," *Nursing Research,* vol. 60, p. 318, 2011.
[52] S. Wang, M. Skubic, and Y. Zhu, "Activity density map dis-similarity comparison for eldercare monitoring," *Conf Proc IEEE Eng Med Biol Soc,* vol. 2009, pp. 7232-5, 2009.
[53] L. I. Smith, "A tutorial on principal components analysis," *Cornell University, USA,* vol. 51, p. 52, 2002.

Appendix A – CERTLogger Manual

The CERTLogger software collects and logs sensor data from an Eldertech sensor network and logs the data to a back-end database for storage. CERTLogger can also be configured to automatically cache backup data, and process and email daily Early Illness Alerts.

Latest version: **CERTLogger 4.2.3**

Installation Requirements

- Windows XP or Higher, Linux, Mac OSX or higher
- Java Version : Java 6 or later http://java.com/en/download/index.jsp
- Install rxtxserial.dll http://rxtx.qbang.org/wiki/index.php/Installation
- Connect W800RF32A X10 wireless receiver to PC
- Properly setup serial port for receiver and serial port address

Installation Instructions

1. Save a copy of the latest version of the logger (CERTLogger*.*.*.jar) to the desired directory
2. Open a command line console (CMD in Windows, terminal in OSX or Linux)
3. Type java –jar 4.*.*.*.jar [-optional startup commands]
4. Enter password

Program Startup

At the startup command line interface, type help to list available control modules (commands)

Control Modules

The control module name indicates the module command, e.g. *system*. The options below the module are possible actions for that command, e.g. *-i*. For example, to change the current CERTLogger's system ID, type *system -i ####*.

SYSTEM

- i <integer>	Set system ID
- u <integer>	Add user ID

X10

- a	Activate x10 receiver
- b	Set MySQL storage database
- c	Calibrate serial port
- d	Deactivate X10 receiver
- e	Execute/Upload sensor data cache
- h	Set MySQL Host:Port
- k	Toggle sensor caching
- l	Load X10 mappings from server
- m <integer>	View/modify server mapping for user
- p	Set MySQL password
- r	Clear cache
- s	Set serial port name, i.e. COM4
- t	Set MySQL X10 sensor mapping table
- u	Set MySQL username
- v	Toggle view of incoming X10 signals
- x	Clear local X10 mappings
- y	Toggle view/display of sensor events

BEHAVE

- D	Process daytime / 24 hour alerts only
- E	Add email monitor for all users

- M	Log and email alerts
- N	Check nighttime alerts only
- X	Clear email monitors
- a	Activate daily Early Illness Alerts
- b	Set MySQL database
- c	Check MySQL connection
- d	Deactivate Early Illness Alerts
- e <MM/dd/yyyy>	Process alerts until this date
- h	Set MySQL host:port
- k [MM/dd/yyyy]	Check alerts an a specific date
- l [File]	Load Early Illness Alerts configuration file
- n <Integer>	Apply to a specific user (sensor network)
- p	Set MySQL password
- s <MM/dd/yyyy>	Process alerts start on this date
- t	Set MySQL sensor log table
-u	Set MySQL username
-x	Clear Early Illness Alert configurations

MAILER

- h	Set SMTP mailer Host:Port
- t	Send test email