Addressing Machine Learning Challenges to Perform Automated Prompting

A report submitted in partial fulfillment of the requirement for the Preliminary Examination of Doctor of Philosophy

by

Barnan Das

School of Electrical Engineering and Computer Science
Washington State University
Pullman WA

November 2012

Abstract

As mankind approaches high life expectancy, a significant portion of the world's population starts aging. Independent living of older adults require a great deal of assistance with everyday activities which become overly burdensome for the caregiver, who may be a family member or a professional. Recent progress in smart environments research and the necessity to help older adults "age-in-place" motivate the design of automated intervention system which can help older adults with activities of daily living (ADLs) by prompting them with effective instructions to complete their activities successfully.

The current work proposes solutions to machine learning problems that occur in smart home sensor network data while addressing the automated prompting challenge. Specifically, the solution to predicting potential prompt situation is modeled by two different machine learning approaches.

The first is a supervised machine learning approach that treats prompt situation as a classification task based on labeled training data of activities and predefined steps corresponding to each activity. However, as there are far lesser "prompt" situations as compared to "no-prompt" situations in reality, it causes an imbalanced distribution of class labels. We propose two novel Gibbs sampling-based minority class over-sampling techniques, RACOG and wRACOG to handle class imbalance. In addition to imbalanced classes, our training data also exhibit ambiguous regions in the data space where the prior probabilities of both classes is approximately equal and thus have even more adverse effect on classification accuracy. We propose solution to this problem, known as overlapping classes, by considering a clustering based under-sampling technique (ClusBUS) that removes majority class examples from the ambiguous regions of data space.

As the former approach cannot predict prompt situation on streaming sensor data, our second approach relies on statistical measures derived from training data to identify prompt situation on test data in real-time. The goal of this approach is to predict sub-activity level prompts, and therefore it is assumed that activity label for sensor events is available in real-time. The current model is capable of identifying two categories of activity errors present in the data by considering the likelihood of a sensor trigger in an activity and the probability of elapsed time of a specific sensor from the

beginning of the activity. Due to the nature of the data and the way it was collected, this approach is still at an early stage and needs more work.

We also explore the potential of the sensor platform on mobile devices such as accelerometer and magnetometer to gather ambient intelligence. We have performed activity recognition of ambulatory movement-based activities such as standing, sitting, walking, running, etc. and explore the possibility of integration of this knowledge with environmental sensors to gain deeper insight into a smart home inhabitant's daily routine.

Training and testing of the proposed models are performed on ADL data obtained by conducting experiments on human participants in an on-campus smart environment testbed.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

As mankind approachers high life expectancy, a significant portion of the world's population starts aging. It privileges an individual to spend more time with loved ones, but at the cost of spending longer periods of old age which is synonymous to "not-so-active" mind and body. The demented mind makes it hard for the individual to lead a prosperous life, especially in first world countries where people prefer to stay independently as long as possible. Before the dementia epidemic hits the world, it is about time to design assistive living tools that would help people "age in place".

Smart environments is a relatively new research area as compared to other established basic sciences research. Being a multi-disciplinary research topic, it involves a good mix of computer scientists, electrical engineers, psycologists, gerontologists and nursing professionals. Most attention of smart environment research has been directed towards health monitoring and activity recognition [1, 2]. Recently, assistive health care systems have started making an impact in society, especially in countries where human care-giving facilities are expensive and a large population of adults prefers an independent lifestyle. According to the studies conducted by the US Census Bureau [3], the number of older adults in the US aged 65+ is expected to increase from approximately 35 million in 2000 to an estimated 71 million in 2030, and adults aged 80+ from 9.3 million in 2000 to 19.5 million in 2030. Moreover, there are currently 18 million people worldwide who are diagnosed with dementia and this number is predicted to reach 35 million by 2050 [4]. These older adults face problems completing both simple (e.g. eating, dressing) and complex (e.g. cooking, taking medicine) Activities of Daily Living (ADLs) [5].

We note that real-world caregivers do not perform all activities for the care recipient, nor do they prompt each step of a task. Instead, the caregiver recognizes when the care recipient is experiencing difficulty with an activity and provides a prompt at that time to help in performing the activity completely. The number of prompts that a caregiver typically provides depends upon the level of cognitive impairment. Worsening of the level of impairment demands an increased number of caregiver duties and thus places a heavier burden on the caregiver. Caregivers for individuals with dementia are also at increased risk for health problems, including higher levels of stress hormones, reduced immune function, slower wound healing, new hypertension, higher serum insulin levels and related markers of diabetes, cardiovascular disease, increased morbidity and premature death [6]. This necessity motivates the need for an automated intervention system which can help older adults with activities of daily living (ADLs) by prompting them with effective instructions to complete their activities successfully. This is a vital step in reducing health risks and alleviating the burden of many caregivers that are helping a large section of the population.

The recent upsurge of statistical learning or machine learning methods have inspired the smart environment researchers to design adaptive systems by leveraging data of an individual's daily activities collected from sensor network placed at their home. Although placing a network of sensors at an older adult's home requires minor modifications of the ambiance, it has been found that the sensor data provide a good deal of intelligence about the individuals lifestyle which can be translated to her cognitive health using machine learning algorithms.

Therefore, the current work proposes solutions to machine learning problems that occur in smart home sensor network data while addressing the automated prompting challenge.

1.1 Problem Definition

In the context of a smart home environment, we [7] define "prompts" as any form of verbal or non-verbal intervention delivered to a user on the basis of time, context or acquired intelligence that helps in successful (in terms of time and purpose) completion of an activity. However, terms like *reminders*, *notifications* and *alerts* have been used by different research groups differently, as these terms tend to depend on the granularity of interventions, methods of

delivery and applications. For example, Kaushik et al. [8] have use all three (reminders, notification and alerts) terms in their user adaptive medication adherence system. In this system, the notifications were delivered when a specific task (taking medication in this case) was recognized. Reminders, which are more proactive, were delivered based on heuristic measure of convenience. On the other hand, alerts were issued preemptively, upon the detection of specific sensor events to prevent missed tasks. In a similar way, different research groups have kept the definition of this nomenclature to their discretion.

The motivation of prompting comes from an enormously common real-life issue of related to cognition and therefore, there has been a wide spectrum of approaches, techniques and devices that have associated to solving the problem. The approaches could be as simple as that of a time-based an alarm clock to a complex as a machine learning technique that learn prompt situations from historic data. Similarly, the prompting devices could a simple time-piece alarm clock to a highly sophisticated smart phone that works ubiquitously and necessary prompts. These approaches and devices have found a wide range of applications. However, the last decade has witnessed a paradigm shift in the fundamental idea of interventions to help people in different activities. While most of the erstwhile prompting technologies focused on delivery of proactive prompts, goal of today's prompting technologies to be unobtrusive and deliver the prompts when crucial.

1.2 Related Work in Prompting Technologies

Thus far a number of different approaches have been considered for prompting. In the following the approaches have been discussed in order of increasing complexity.

Time-Based The simplest example of a time based prompt (or reminder, to be more appropriate in this case) is an alarm clock. It produces a general audible or vibration alert to grab the user's attention when the appointed time arrives. Most of the electronic organizers have this feature. These days Google Calendar [9] is being widely used by people as organizers to set reminders on certain events.

However, time-based prompts are very simple and limited. The prompts cannot be conditioned as per the actions of the user. Even if it could be

adapted according to the users' demands, it the users who need to do it by themselves. This limitation of time-based prompts put this idea in the back seat, in spite of being very reliable in terms of time and precision.

Location-Based Location based prompts are more complex than time-based prompts. They are usually used in association with time-based or context-aware prompts (as discussed next) to provide more precise location related reminders to the users. The level of granularity to which location information is derived, is entirely application dependent. It could be an outdoor geographic location, derived using GPS receivers, or an indoor location such as specific floor or room.

Marmasse et al. [10] did a pioneering work on deliveing proactive location-based reminders and messages with the help of their system comMotion that uses GPS to determine location. Similarly, the impulse project of MIT Media Lab used GPS information to determine what the user "wants" when he is in a certain location. Later Google started providing location services [11, 12] to its users by determining the user's network location. The Place-Its application [13] was designed around the post-it usage metaphor and has the ability to place a reminder message at a physically remote location. It uses mobile phones as the ubiquitous platform to detect user's location and deliver reminders.

However, the research in the area of smart environments [14, 15, 16] has made it possible to determine indoor location of the user by using motion sensors in different rooms in an apartment. Das et al. [17] use the same technological framework to deliver prompts to the inhabitants for activities of daily living using different machine learning techniques.

The Assistive Cognition Project at University of Washington [18, 19] sensed aspects of individual's location and environment (both outdoors and at home) by relying on a wide range of sensors such as GPS, active badges, motion detectors and other ubiquitous computing infrastructure.

Context-Aware Context-aware prompting have been in existing for a long time. It is the oldest and first of its kind technology that can help in solving the problem highlighted in this paper. There has been a humongous number publications in this area. This sections gives a non-exhaustive review of context-aware prompting technologies

The Forget-me-not project [20] was one of the pioneering works in the

area of context-aware reminders. It employs a small PDA like device to associate different items of interest with icons to help users to remember various tasks that need to be attended. Years later, another milestone was set by Dey et al. when they developed CyberMinder [21], a context-aware reminder application based on Context Toolkit [22] that focused on using complex contextual information to determine a prompt situation. The same Context Toolkit architecture was used by Hristova et al. [23] for developing services for ambient assisted living that can support heart rate monitoring, medication prompting, generation of agenda reminders, weather alerts, emergency notification, etc.

Mihailidis et al. [24] proposed the usage of context-aware computing to assist people with dementia for ADLs that needs more privacy such as toileting. Later, the same group developed an artificial intelligence planning based approach that has been discussed next. The Assistive Cognition Project [18, 19] developed novel representation and reasoning techniques for context-aware cognitive assistance that can help Alzheimer's patients carry out multi-step everyday tasks like cooking. Helal et al. [25], with the help of their Smart House project, developed three different application for a smart phone based cognitive assistance. The applications interact with the smart house sensor grid to assist individuals with activities of daily living by means of reminders, orientation, context-sensitive teaching and monitoring. The applications also enhance the level of awareness of the inhabitants by notifying them when certain events happen. Vurgun et al. [26] proposed a statistical reasoning system for determining in what circumstance the subject should be prompted for medication. They evaluate a simple rule-based approach and compare its effectiveness with a Dynamic Bayesian Network representation. HYCARE [27] or hybrid context-aware reminding framework uses a novel scheduling mechanism to handle synchronous time-based and asynchronous event-based reminding services. By conducting an extensive user study, the authors classify the reminding services into four categories and formulates the context-aware reminding rules accordingly. Chang et al. [28] describes an approach to providing distributed cognition support of work engagement for persons with cognitive disabilities. The system is capable of providing unique-to-the-user prompts that are triggered by context.

Other context aware systems [29, 8, 30] have also been proposed recently.

AI Planning Assistance in a smart environment involves not only reminders of certain timely events, but also guidance for more complicated activities of daily living that involves multiple steps. This necessitates the exploration of different approaches in artificial intelligence to accomplish the goal. The applications of artificial intelligence planning based approaches in smart environments have been very limited. There are works by Marquardt et al. for service composition [31] and creating planning domains [32] for smart environments. There are certain groups that have taken some plan recognition techniques to address the problems of prompting in smart environments.

The foundation stone of using planning based approaches for cognitive assistance and interventions was laid by Levinson [33]. Levinson developed a hand-held system that uses classical deterministic planning algorithms to compute a best plan for completion of an activity and provides step-by-step guidance through tasks in the form of visual and audio cues. The Autominder System developed by Pollack et al. [34] uses dynamic Bayesian networks as an underlying domain model to coordinate pre-planned events in an attempt to ensure that scheduled tasks are executed without interfering with each other or with other activities, such as watching television. Pineau et. al. [35] uses variant of partially observable Markov decision processes (POMDPs) to design the high level control system for "Nursebot", an artificially intelligent robot designed to assist elderly people with daily activities. The robot primarily provides intelligent reminders regarding specific activities but also engages in a certain degree of social interaction. In the Assistive Cognition Project [19], plan recognition and temporal reasoning techniques were used describe goals and plans of of the users at various levels of abstraction.

Boger et al. [36] proposed a planning system that uses Markov decision processes (MDPs) to determine when and how to provide prompts to a user with dementia for guidance through the activity of hand washing. The hand washing task is divided into five different sequential steps required for proper completion. With the help of video camera, the user's progress through the activity is monitored and a prompt is issues when there is a user regression/departure from the appropriate sequence of steps. This framework was named as the COACH system and extended to have three significant changes [37]: use of marker-less flocking to track the activity, usage of partially observable Markov decision process (POMDP) to model the hand washing guidance problem and the refinement of audio prompts and video demonstrations. Although, this framework was claimed to be generalizable to other ADLs, no

further extension has been done yet.

Machine Learning As mentioned before, activities in a smart environment for which the inhabitants would seek assistance, can be highly complicated. There are certain conditions when planning approaches might as well turn out to be inefficient. In these situations, classical machine learning techniques could be used to predict a prompt situation based on historic data.

In the Assistive Cognition Project [19], dynamic Bayesian networks are used to create predictive models of user behavior from observations. They also devised relational Markov models that can allow states belonging to different types. Rudary et al. [38] integrated temporal constraint reasoning with reinforcement learning to build an adaptive reminder system. It can personalize to a user and adapt to both short and long term changes. Although this approach is useful when there is no direct or indirect user feedback, it relies on a complete schedule of user activities. The Independent LifeStyle Assistant project [39] used machine learning techniques to capture interactions among devices, environment and humans. Patterned behavior profiles were created to build models of what sensor firings correspond to what activities in what order and at what time. Alerts were raised when activities that were probabilistically unlikely, occurred. Also, schedule information for regular activities were learned using machine learning techniques.

Das et al. [40, 17] took the very first approach to solve the prompting problem by using machine learning techniques to learn prompting situations for eight different activities of daily living. In this approach, eight ADLs are subdivided into critical steps. The errors committed by older adult participants while performing ADLs are labeled as prompt situations or steps. Supervised machine leaning models are trained on this data and prompting situations are tested. However, the tests were performed on offline data and not in real-time.

1.3 System Architecture

The CASAS smart home infrastructure is used to replicate near ideal day to day lives of individuals in their homes. The facility is used for a wide spectrum of different goals which have both computer science and psychology focus. These goals include, activity recognition, activity discovery, activity

prediction, functional assessment of participants based on demographic information, user study of machine learning driven solutions to some of the problems, testing a wide variety of sensor platforms, and at last but not the least, the automated prompting task that needs a real-life validation of the effectiveness of machine learning approaches from a technological perspective and effectiveness of prompts from psychological perspective. Although, the data that is used for automated prompting task is collected under a controlled environment, the best effort has been on maintaining a realistic daily life setting.

The test bed, or smart apartment, is located in an on-campus town house apartment. The current sensor system is composed of several different sensor types. There are sensors for motion, ambient light level, temperature, doors, light switches, items, objects, water flow, and power use. A majority of our sensors are now wireless, utilizing a Control4 ZigBee wireless mesh network. There are two types of motion detectors, ceiling mounted and wall mounted. The ceiling mounted motion detectors sense directly below them and have their viewing aperture confined so that they can only sense approximately a four feet diameter area below them. The wall mounted motion detectors are mounted so that they can look out into an area, such as an entire room, and detect any motion within that space. Integrated into the motion detector enclosure are ambient light level sensors. This can be useful for allowing the home to automatically turn on lights where you are to help prevent tripping at night, or illuminating your workspace when enough natural light is no longer available. Temperature sensors are also useful for determining inhabitant behavior, such as thermal preferences, determining when the stove or oven is in use in the kitchen.

To determine the ability of a machine learning algorithm to generate appropriate activity-aware prompts, we performed a study in our smart home with 128 volunteer participants, aged 50+, who are healthy older adults or individuals with mild cognitive impairment. The smart home is a two-story apartment equipped with sensors that monitor motion, door open/shut status, and usage of water, burner, and specific items throughout the apartment.

Clinically-trained psychologists watch over a web camera as the participants perform 8 different activities. The psychology experimenter remotely issues a prompt when they determine that the individual is having difficulty initiating or completing an activity. Sensor events, denoted by the event date, time, sensor identifier, and message, are collected continuously, along with the prompt timings. A human annotator annotates the sensor events with

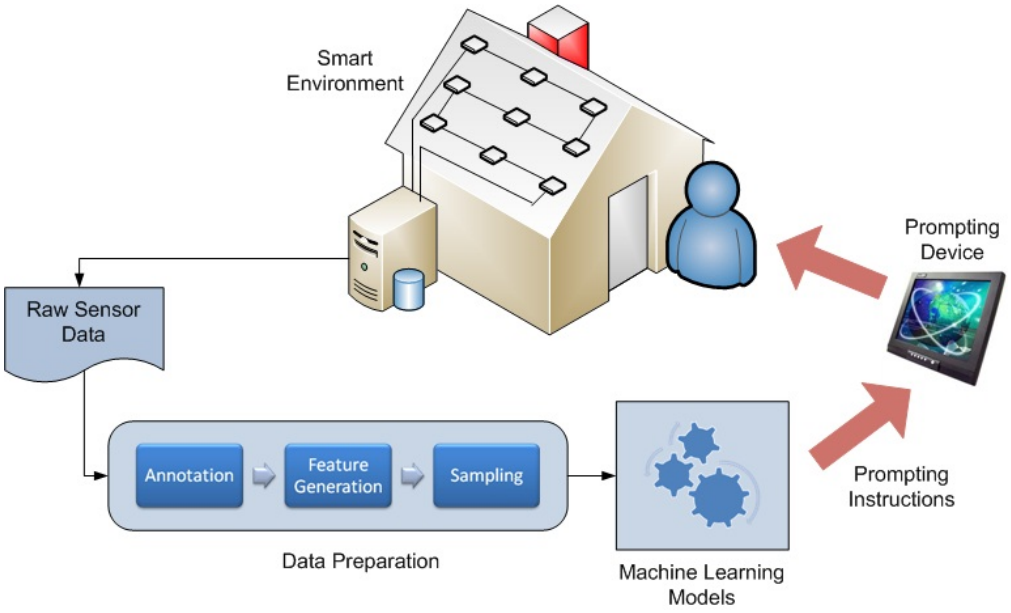

Figure 1.1: System Architecture

corresponding activity and a sub-step that act as the ground truth. Table 1.1 shows a snippet of annotated events. On the basis of ground truth information, the feature vector for every activity sub-step present in the database is generated that consists of temporal, spatial and contextual features. We can view automated prompting as a supervised learning problem in which each activity step is mapped to a "Prompt" or "No-prompt" class label. Thus, automated prompting emulates natural interventions provided by a caregiver.

Date	Time	SensorID	Message	Annotation	Prompt Status
2009-05-11	14:59:54.934979	D010	CLOSE	cooking, substep-3	*prompt*
2009-05-11	14:59:55.213769	M017	ON	cooking, substep-4	*no-prompt*
2009-05-11	15:00:02.062455	M017	OFF	none	*no-prompt*
2009-05-11	15:00:17.348279	M017	ON	cooking, substep-8	*no-prompt*
2009-05-11	15:00:34.006763	M018	ON	cooking, substep-8	*no-prompt*
2009-05-11	15:00:35.487639	M051	ON	cooking, substep-8	*no-prompt*
2009-05-11	15:00:43.028589	M016	ON	cooking, substep-8	*no-prompt*
2009-05-11	15:00:43.091891	M015	ON	cooking, substep-9	*no-prompt*
2009-05-11	15:00:45.008148	M014	ON	cooking, substep-9	*no-prompt*

Table 1.1: Human annotated sensor events

1.4 The Prompting Dataset

The *prompting* dataset, as we would like to call it, has 3980 examples with 17 features. Out of the 17 features, 4 are categorical and the rest are numeric. Table 1.2 gives the attribute name, attribute type, the measurement unit and a brief description. The classification goal is to predict the "class"attribute. The difficulty that is faced for the prompting problem is that the majority of activity steps are "No-prompt" cases and standard machine learning algorithms will likely map all data points to this class, which defeats the purpose of the intervention. Our goal is to design solutions to the class imbalance problem that improve sensitivity for this prompting application.

Name	Data Type	Measure	Description
stepLength	numeric	seconds	Length of step in time
numSensors	numeric		Number of unique sensors involves with the step
numEvents	numeric		Number of sensor events associated with the step
prevStep	nominal	*	Previous step ID
nextStep	nominal	*	Next step ID
timeActBegin	numeric	seconds	Time elapsed since the beginning of the activity
timePrevStep	numeric	seconds	Time difference between the last event of the previous step and first event of the current step
stepsActBegin	numeric		Number of steps visited since the beginning of the activity
livingRoom	numeric		Frequency of motion sensor firing in the living room associated with the current step
diningRoom	numeric		Frequency of motion sensor firing in the dining room associated with the current step
kitchen	numeric		Frequency of motion sensor firing in the kitchen associated with the current step
hallway	numeric		Frequency of motion sensor firing in the hallway associated with the current step
kitchenDoor	numeric		Frequency of all kitchen door (cabinets, microwave, refrigerator) usage associated with the current step
closet	numeric		Frequency of closet usage associated with the current step
kitchenItems	numeric		Frequency of kitchen item sensor firing associated with the current step
activityID	nominal	1,2,3,4,5,6,7,8	Activity ID
stepID	nominal	*	Current step ID
class	nominal	0,1	Binary class representing *prompt* and *no-prompt*

∗ Attributes *prevStep*, *nextStep* and *stepID* have the following nominal values:
1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 1.10, 2.1, 2.2, 2.3, 3.1, 3.2, 3.3, 3.4, 3.5, 3.6, 3.7, 3.8, 4.1, 4.2, 4.3, 4.4, 4.5, 4.6, 4.7, 4.8, 5.1, 5.2, 5.3, 5.4, 5.5, 5.6, 6.1, 6.2, 6.3, 6.4, 6.5, 6.6, 7.1, 7.2, 7.3, 7.4, 7.5, 7.6, 7.7, 7.8, 7.9, 8.1, 8.2, 8.3

Table 1.2: Attribute information of *prompting* dataset

1.5 Proposed Solutions

The current work proposes solutions to machine learning problems that occur in smart home sensor network data while addressing the automated prompting challenge. Specifically, the solution to predicting potential prompt situation is modeled by two different machine learning approaches.

The first is a supervised machine learning approach that treats prompt situation as a classification task based on labeled training data of activities and predefined steps corresponding to each activity. However, as there are far lesser "prompt" situations as compared to "no-prompt" situations in reality, it causes an imbalanced distribution of class labels. We propose two novel Gibbs sampling-based minority class over-sampling techniques, RACOG and wRACOG to handle class imbalance. In addition to imbalanced classes, our training data also exhibit ambiguous regions in the data space where the prior probabilities of both classes is approximately equal and thus have even more adverse effect on classification accuracy. We propose solution to this problem, known as overlapping classes, by considering a clustering based under-sampling technique (ClusBUS) that removes majority class examples from the ambiguous regions of data space.

As the former approach cannot predict prompt situation on streaming sensor data, our second approach relies on statistical measures derived from training data to identify prompt situation on test data in real-time. The goal of this approach is to predict sub-activity level prompts, and therefore it is assumed that activity label for sensor events is available in real-time. The current model is capable of identifying two categories of activity errors present in the data by considering the likelihood of a sensor trigger in an activity and the probability of elapsed time of a specific sensor from the beginning of the activity. Due to the nature of the data and the way it was collected, this approach is still at an early stage and needs more work.

We also explore the potential of the sensor platform on mobile devices such as accelerometer and magnetometer to gather ambient intelligence. We have performed activity recognition of ambulatory movement-based activities such as standing, sitting, walking, running, etc. and explore the possibility of integration of this knowledge with environmental sensors to gain deeper insight into a smart home inhabitant's daily routine.

Training and testing of the proposed models are performed on ADL data obtained by conducting experiments on human participants in an on-campus smart environment testbed.

Chapter 2

Imbalanced Class Distribution

2.1 Introduction

With a wide spectrum of industries drilling down into their data stores to collect previously unused data, data is being treated as the "new oil". Social media, healthcare, and e-commerce companies are refining and analyzing collected data to leverage their insights with the goal of creating new products and/or adding value to existing ones. As machine learning techniques mature and are used to tackle scientific problems in the current deluge of data, a variety of challenges arise due to the nature of the underlying data. One well-recognized challenge is the imbalanced class distribution problem, where one of the target classes (the *minority* class) is under-represented in comparison with the other classes (the *majority* class or classes).

While many datasets do not contain a perfectly uniform distribution among class labels, some distributions contain less than 10% of the data points in the minority class, which is considered imbalanced. Because the goal of supervised learning algorithms is to optimize prediction accuracy for the entire set of available data points, most approaches ignore performance on the individual class labels. As a result, the highest-performing algorithm will label all points from an imbalanced dataset as members of the majority class despite the fact that all of the minority data points are incorrectly classified. In many problem domains, such as cancerous cell identification [41], oil-spill detection [42], fraud detection [43], keyword extraction [44], and text classification [45], identifying members of the minority class is critical, sometimes more so than achieving optimal overall accuracy for the majority class.

The imbalance class distribution problem has vexed researchers for over a decade and has thus received focused attention. Among the common techniques that have been investigated such as resampling (primarily oversampling), cost-sensitive learning, kernel-based learning and feature selection, resampling methods remain at the forefront due to their ease of implementation. Liu et al. [46] articulate a number of reasons to prefer resampling to other methods. First, because resampling occurs during preprocessing, the approach can be combined with others such as cost sensitive learning, without changing the algorithmic anatomy [47]. Second, theoretical connections between resampling and cost-sensitive learning indicate that resampling can alter the misclassification costs of data points [48]. Third, empirical evidence demonstrates nearly identical performance between resampling and cost-sensitive learning techniques [49, 50].

Although both under- and over-sampling techniques have been improving over the years, we focus our attention on oversampling because it is well suited for the application that motivates our investigation (discussed in detail in Section 1.4). Also, as a major portion of the class imbalance problems involve an absolute rarity of minority class data points [51], undersampling of majority class examples[1] to achieve an acceptable class imbalance is not advisable. However, we use an undersampling method, RUSBoost [52], as a proof of concept to compare with the proposed Gibbs sampling-based oversampling techniques: RACOG and wRACOG.

Existing oversampling approaches [53, 54] that add synthetic data to alleviate class imbalance problem typically rely on spatial location of minority class samples in the Euclidean feature space. These approaches harvest *local* information of individual minority class samples to generate new synthetic samples that are assumed to belong to the minority class as well. Although this approach may be acceptable for data sets where a crisp decision boundary is known to exist between the two classes, spatial location-based synthetic oversampling are not suitable for data sets that have certain degree of overlap between the minority and the majority classes. Therefore, a better idea is to exploit *global* information of minority class samples, which can be done by considering the probability distribution of minority class while synthetically generating new minority class samples.

In this paper, we exploit advances in the area of Markov chain Monte

[1]The terms: *data points*, *examples*, *samples*, and *instances* have been used synonymously in the paper and all refer to training examples

Carlo (MCMC) methods to introduce two Gibbs sampling-based oversampling approaches, called RACOG and wRACOG, to generating and strategically selecting new minority class data points. Specifically, the proposed algorithms generate new minority class data points via Gibbs sampling by exploiting the joint probability distribution and interdependencies of random variables. While RACOG selects samples from the Markov chain generated by the Gibbs sampler using a predefined *lag*, wRACOG selects those samples that have the highest probability of being misclassified by the existing learning model.

We validate our approach using five UCI datasets from a variety of scientific application domains, carefully modified to exhibit class imbalance, and one new application domain dataset[2] with inherent extreme class imbalance. The proposed methods, RACOG and wRACOG are compared with two existing oversampling techniques, SMOTE [53] and SMOTEBoost [54] and an undersampling technique RUSBoost [52] as a proof of concept.

We report Sensitivity, G-mean, and Area Under ROC Curve (AUC-ROC) as the performance measures. SMOTEBoost performs the best among all the existing resampling methods. However, RACOG and wRACOG show statistically significant ($p < 0.05$) improvements over SMOTEBoost. In the remainder of the paper, we detail our approach and perform experiments to evaluate the proposed methods.

2.2 Related Work

Learning from class imbalanced datasets is a niche, yet critical area in supervised machine learning due to its increased prevalence in real world problem applications. Over the last decade, a number of different review articles [55, 56, 57, 58] have appeared in leading journals supporting the fact that this area continues to be of importance among machine learning researchers.

Due to the pervasive nature of the imbalanced class problem, a wide spectrum of related techniques has been proposed over the last decade. One of the most common solutions that has been investigated is cost sensitive learning (CSL). CSL methods counter the underlying assumption of classical classification algorithms that all errors are equal by introducing customized cost factors that describe the costs for misclassifying any particular data point. By assigning a sufficiently high cost to minority sample points, the

[2]http://ailab.wsu.edu/casas/datasets/prompting.zip

algorithm may devote sufficient attention to these points to learn an effective class boundary. The works by Elkan [48] and Maloof [50] provide theoretical foundations and algorithms that verify the effectiveness of CSL methods for imbalanced learning problems. Moreover, various empirical studies [49, 59] have also validated the effectiveness of CSL approaches. In addition, the concepts of CSL have been coupled with existing learning methods to boost their performance. Cost Sensitive Dataspace Weighting with Adaptive Boosting [60] takes advantage of iterative updating of the weight distribution function of AdaBoost by introducing cost items in the weight updating strategy. Cost sensitive decision trees [61] use a cost-sensitive fitting approach to make choices for parameters including the decision threshold, the split criteria at each node and selection of alternative pruning schemes. CSLs have also been used with neural networks [62] to combine the cost factor with the probabilistic estimate of the output values and the learning rate. However, some empirical studies [63] show that between CSL and other methods of handling class imbalance such as resampling, there is no clear winner. Moreover, CSL approaches have certain drawbacks that limit their application in many situations. Firstly, the misclassification costs are often unknown or need to be painstakingly determined for each application. Secondly, not all learning algorithms have cost sensitive implementation.

A second direction is to adapt the underlying classification algorithm to consider imbalanced classes, especially, using kernel-based learning methods. Since kernel-based methods provide state-of-the-art techniques for many machine learning applications, using them to understand the imbalanced learning problem has attracted increased attention. The kernel classifier construction algorithm proposed by Hong et al. [64] is based on orthogonal forward selection (OFS) and a regularized orthogonal weighted least squares (ROWLSs) estimator. This algorithm optimizes generalization in the kernel-based learner by introducing two components that deal with imbalanced data for binary data sets. The first component integrates the concepts of leave-one-out cross validation and the area under curve the ROC curve to develop an objective function as a selection mechanism of the optimal kernel model. The second component takes advantage of the cost-sensitivity of the parameter estimation cost function in the ROWLS algorithm to assign greater weights to erroneous training examples in the minority class. Wu et al. [65] proposed kernel-boundary alignment (KBA) algorithm for adjusting the SVM class boundary. KBA is based on the idea of modifying the kernel matrix generated by a kernel function according to the imbalanced

data distribution. Another interesting kernel modification technique is the k-category proximal support vector machine (PSVM) [66] with Newton refinement proposed by Fung et al. This method transforms the soft-margin maximization paradigm into a simple system of k-linear equations for either linear or non-linear classifiers, k being the number of classes.

Probably the most common approach, however, is to resample, or modify the dataset in a way that balances the class distribution. Determining the ideal class distribution is an open problem [60] and in most cases it is handled empirically. Naive resampling methods include oversampling the minority class by duplicating existing data points and undersampling the majority class by removing chosen data points. However, random over-sampling and under-sampling increases the possibility of overfitting and discarding useful information from the data, respectively.

An intelligent way of oversampling is to synthetically generate new minority class samples. Synthetic minority class oversampling technique, or SMOTE [53], is a powerful method and has shown great deal of success in various application domains. SMOTE oversamples the minority class by taking each minority class data point and introducing synthetic examples along the line segments joining any or all of the k-minority class nearest neighbors. In addition, adaptive synthetic sampling techniques have been introduced that take a more strategic approach to selecting the set of those minority class samples on which synthetic oversampling should be performed. For example, Borderline-SMOTE [67] generates synthetic samples only for those minority class examples that are "closer" to the decision boundary between the two classes. ADASYN [68], on the other hand, uses the density distribution of the minority class samples as a criterion to automatically decide the number of synthetic samples that need to be generated for each minority example by adaptively changing the weights of different minority examples to compensate for the skewed distribution. Furthermore, class imbalance problems associated with *intra-class* imbalanced distribution of data in addition to *inter-class* imbalance can be handled by cluster-based oversampling (CBO) proposed by Jo et al. [69].

On the other hand, the information loss incurred by random undersampling can be overcome by calculative and strategic methods to carefully remove minority class samples. Liu et al. [70] proposed two informed undersampling techniques: EasyEnsemble and BalancedCascade. EasyEnsemble is an ensemble learning system that randomly samples out subsets from the majority class. Multiple classifiers are built on a combination of each of

the majority class subsets with the minority class samples. On the other hand, BalancedCascade develops an ensemble of classifiers that carefully selects the majority class instances to be under-sampled by forming a cascade of hypotheses. For data sets that have overlapping minority and majority classes [71] in addition to class imbalance, data cleansing techniques are used to alleviate the problem of class imbalance. Data cleansing is usually done by cleaning up unwanted overlapping between classes by removing pairs of minimally distanced nearest neighbors of opposite classes, popularly known as Tomek links [72]. One sided selection (OSS) [73] and SMOTE+Tomek [74] are some representative works in this area. However, removing Tomek links changes the distribution of the data. Therefore, it might not be beneficial in situations when there is a dense overlap between the classes, because this approach might lead to overfitting the learning hypothesis on this dataset.

Furthermore, there are a number of ensemble learning approaches [75] that combine the powers of the resampling and ensemble learning such as AdaBoost [76] and Bagging [77]. These approaches include, but are not restricted to, SMOTEBoost [54], RUSBoost [52], IVotes [78] and SMOTE-Bagging [79].

Empirical studies [63] have shown that approaches such as cost sensitive learning or kernel-based learning are not quite suitable for class imbalanced data sets that have "rare" minority class samples. In fact, any form of under-sampling witnesses the same problem. We are interested in applying supervised learning techniques to a dataset that includes rare minority class samples. While oversampling is a natural solution to this problem, existing oversampling techniques such as SMOTE, Borderline-SMOTE and SMOTEBoost generate new data samples that are spatially close to existing minority class examples in the Euclidean feature space. Ideally new data samples should be representative of the entire minority class and not just be drawn from local information. Therefore, in this paper we focus on satisfying the criteria of *global* representation of the minority class by generating multivariate samples from the joint probability distribution of the underlying random variables or attributes.

2.3 RACOG and wRACOG

Our proposed approaches to oversampling for imbalanced class distributions is based upon Gibbs sampling. Gibbs sampling is rooted in image processing

and was introduced by Geman and Geman (1984). The family of Markov chain Monte Carlo (MCMC) methods, of which Gibbs sampling is a descendant, originated with the Metropolis algorithm [80, 81]. Because Gibbs sampling exploits global properties of the minority class and not just local distance between specific data points, the technique has found application in several domains [82].

2.3.1 Standard Gibbs Sampling

The Gibbs sampler generates a sequence of data samples from the joint probability distribution of two or more random variables. Each data sample is a node in a first order Markov chain, which is defined as a collection of M random variables. Start with the data samples $z^{(1)}, ..., z^{(M)}$, such that the conditional independence property given in Equation 2.1 holds true for $m \in 1, .., M$.

$$P(z^{(m+1)}|z^{(1)}, .., z^{(m)}) = P(z^{(m+1)}|z^{(m)}) \tag{2.1}$$

A Markov chain is defined by specifying the probability distribution of the initial variable $P(z^{(0)})$ and the conditional probabilities of the subsequent variables (transition probabilities). Thus, the marginal probability of a variable of interest can be expressed in terms of the marginal probability of the previous variable and the transition probability from the previous variable to the current variable (see Equation 2.2).

$$P(z^{(m+1)}) = \sum_{z^{(m)}} P(z^{(m+1)}|z^{(m)})P(z^{(m)}) \tag{2.2}$$

The goal of Gibbs sampling is to generate a Markov chain whose samples converge to the target distribution (see Figure 2.1).

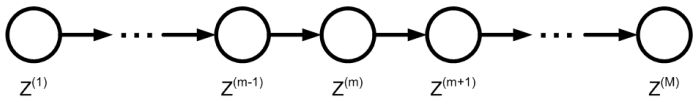

Figure 2.1: A Markov chain

The approach is applicable in situations where the random variable Z has at least two dimensions ($z =< z_1, .., z_k >, k > 1$). At each sampling step, the

algorithm considers univariate conditional distributions where each of the random variables but one is assigned a fixed value. Rather than picking the entire collection of attribute values at once, a separate probabilistic choice is made for each of the k dimensions, where each choice depends on the values of the other $k-1$ dimensions and the previous value of the same dimension. Such conditional distributions are easier to model than the full joint distribution. Table 2.1 shows the algorithm for the standard Gibbs sampler.

Implementations of Gibbs samplers are traditionally dependent on two factors. The first is the number of sample generation iterations that are needed for the samples to reach a stationary distribution (i.e., when the marginal distribution of $Z(n)$ is independent of n). Typically, in order to avoid the estimates being contaminated by values at iterations before this point (referred to as the ***burn-in***), earlier samples are discarded. The second factor is that a sample generated during one iteration is highly dependent on the previous sample. This correlation between successive values, or *auto-correlation*, is avoided by defining a suitable ***lag***, or number of consecutive samples to discard from the Markov chain following each generated sample that is accepted.

Algorithm: Gibbs Sampling
1: $z^{(0)} =< z_1^{(0)}, ..., z_k^{(0)} >$
2: **for** $t = 1$ to T
3: **for** $i = 1$ to k
4: $z_i^{(t+1)} \sim P\left(Z_i \| z_1^{(t+1)}, ..., z_{i-1}^{(t+1)}, z_{i+1}^{(t)}, ..., z_k^{(t)}\right)$
5: **end**
6: **end**

Table 2.1: Gibbs Sampler

2.3.2 The RACOG Algorithm

The proposed RApidy COnverging Gibbs sampler (RACOG) uses Gibbs sampling at its core to generate new minority class samples from the distribution of the minority class. RACOG enhances standard Gibbs sampling by offering an alternative mechanism for choosing the values of $Z^{(0)}$ that help generate samples which rapidly converge to the target minority class distribution, and

by alleviating the attribute independent assumption imposed by the Gibbs sampler.

Conventionally, the starting values of the Gibbs sampler is any randomly chosen value in the state space of attributes. This approach takes a high burn-in period and an extremely high iterations for the sampler to converge with the target distribution. On the other hand, RACOG chooses the minority class data points as the set of initial samples and runs the Gibbs sampler for every minority class example. The total number of iterations for the Gibbs sampler is restricted by the desired *minority*:*majority* class distribution. Thus, RACOG produces multiple Markov chains, each starting with a different minority class data point, instead of one very long chain as done in conventional Gibbs sampling. As the initial samples of RACOG are chosen directly from the minority class distribution, it helps in achieving faster convergence of the generated samples with the minority class distribution (a convergence test performed in Section 2.5 validates this claim).

There are arguments in the literature about the pros and cons of single long Markov chain and multiple shorter chain approaches. Geyer [83] argues that a single long chain is a better approach because, if long burn-in periods are required, or if the chains have high autocorrelations, using a number of shorter chains may result in chains that are not long enough to be of any value in representing the minority class. However, single long chain requires very high number or iterations to converge with the target distribution. Our experiments show that the argument made by Geyer does not hold when multiple chains are generated with the minority class data points as the initial samples of the Gibbs sampler.

Now, the conditional distribution in Step 4 of the Gibbs Sampling algorithm (Table 2.1) can be simplified as shown in Equation 2.3. The numerator of Equation 2.3 represents the joint probability of all attribute values, while the denominator is a normalizing factor.

$$P\left(Z_i|z_1^{(t+1)},...,z_{i-1}^{(t+1)},z_{i+1}^{(t)},...,z_k^{(t)}\right)=\frac{P\left(z_1^{(t+1)},...,z_{i-1}^{(t+1)},z_i^{(t)},z_{i+1}^{(t)},...,z_k^{(t)}\right)}{P\left(z_1^{(t+1)},...,z_{i-1}^{(t+1)},z_{i+1}^{(t)},...,z_k^{(t)}\right)} \tag{2.3}$$

Datasets which have extremely rare minority class points and sufficiently high number of random variables (or, attributes), the joint probability distribution of the attributes would have peaks in the distributions for the joint

attribute values present in the dataset. On the other hand, the probability of joint occurrences of other attribute values is extremely low due to insufficient number of such minority class samples. Because the Gibbs Sampling algorithm imposes an independence assumption on the variable attributes, the sampling performed in Step 4 of the algorithm disallows exploration of the entire space of attributes and is therefore less likely to generate points that are consistent with such minority class distribution. On the other hand, considering a full joint distribution will be prone to error because there are insufficient minority points to accurately estimate the probabilities that are used in Equation 2.3.

The RACOG algorithm explores the state space of attributes more thoroughly by factoring the large-dimensional joint distribution into a *directed acyclic graph* (DAG) that imposes explicit dependencies between attributes. The probability of a particular data point, x, represented as a collection of attributes $\{x_i : 1 \leqslant i \leqslant M\}$, would then be computable as:

$$P(x) = \prod_i P(x_i | x_{parents(i)}) \tag{2.4}$$

Traditionally, the DAG is constructed by learning a Bayesian network which uses search techniques, such as hill climbing or simulated annealing. An alternative, less computationally expensive approach, is to employ the Chow-Liu algorithm (1968) described in Table 2.2, to construct a Bayesian tree of dependencies by reducing the problem of constructing a maximum likelihood tree to that of finding a maximal weighted spanning tree in a graph. RACOG employs this approach to finding a Bayesian tree among attribute dependencies. This allows each attribute (but the root) to have exactly one parent attribute on which its value depends [84]. Figure 2.2 shows Bayesian trees formed from the *abalone* and *car* datasets. Attributes *length* and *lug_boot* are chosen as the roots of the Bayesian trees for *abalone* and *car*, respectively. The number of children for the nodes in the tree gives us an insight into the importance of that attributes in the dataset. In other words, the greater the number of children in a node, the greater its importance. Equation 3 can now be represented as:

$$P\left(Z_i | z_1^{(t+1)}, ..., z_{i-1}^{(t+1)}, z_{i+1}^{(t)}, ..., z_k^{(t)}\right) = \frac{P\left(z_{root} \prod_x P(z_x | z_{parents(x)})\right)}{P\left(z_1^{(t+1)}, ..., z_{i-1}^{(t+1)}, z_{i+1}^{(t)}, ..., z_k^{(t)}\right)} \tag{2.5}$$

Algorithm: Chow-Liu Dependence Tree Construction
1: Compute the mutual information between each pair of variables, $i \neq j$: $I_P(X_i; X_j) = \sum_{x,y} P(x,y) log \frac{P(x,y)}{P(x),P(y)}$
2: Build a complete undirected graph with variables in X as vertices and the weight of an edge connecting X_i and X_j by $I_P(X_i; X_j)$.
3: Build a maximum weighted spanning tree.
4: Transform the resulting undirected tree to a directed one by choosing a root variable and setting the direction of all edges to be outward from it.

Table 2.2: Chow-Liu Dependence Tree Construction

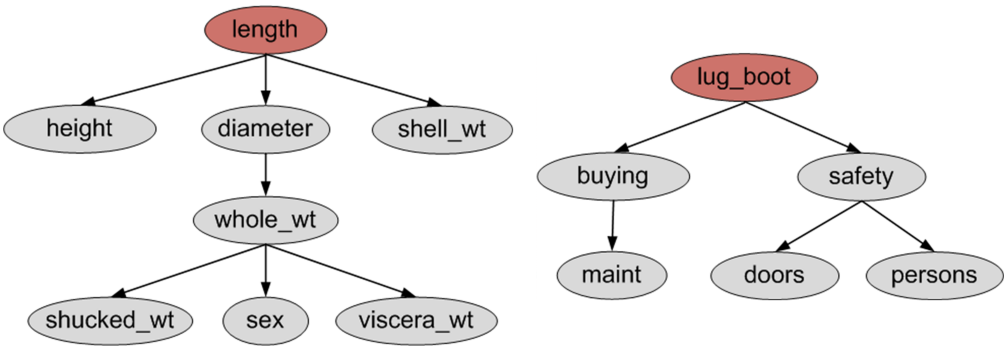

Figure 2.2: Bayesian trees for (*left*) abalone and (*right*) car datasets

Sampling $Z_i^{(t+1)}$ from $P\left(Z_i | z_1^{(t+1)}, ..., z_{i-1}^{(t+1)}, z_{i+1}^{(t)}, ..., z_k^{(t)}\right)$ is done by performing random sampling from the distribution of the state space (all possible values) of attribute i. Table 2.3 summarizes the RACOG algorithm.

While the RACOG algorithm offers improvements over traditional Gibbs sampling, this approach tends to add redundant data points due to the inherent sampling methodology, which does not necessarily help in learning a generalized model. This problem can be alleviated by considering the usefulness of the generated samples towards learning a generalized model. In the next section, we introduce a wrapper enhancement to the RACOG algorithm that addresses this issue.

2.3.3 The wRACOG algorithm

The enhanced RACOG algorithm, named wRACOG, is a wrapper-based technique over RACOG utilizing Gibbs sampling as the core data sampler.

Algorithm: RACOG

Input: *minority* = minority class data points; N = *minority* size;
k = *minority* dimensions; β = burn-in period; α = lag;
T = total number of iterations
Output *new_samples* = new minority class samples

```
1: Construct Bayesian tree BT using Chow-Liu algorithm
2: for n = 1 to N
3:     Z^(0) = minority(n)
4:     for t = 1 to T
5:         for i = 1 to k
6:             Simplify P(Z_i | z_1^(t+1), ..., z_{i-1}^(t+1), z_{i+1}^(t), ..., z_k^(t)) using BT
               and Equation 2.3
7:             z_i^(t+1) ~ P(S_i) where S_i is the state space of attribute i
8:         if t > β AND t%(α) = 0
9:             new_samples = new_samples + Z
```

Table 2.3: The RACOG algorithm

However, the purpose of introducing wRACOG is to decouple the concept of *burn-in* and *lag* associated with sample selection from conventional Gibbs sampling. By performing iterative training on the dataset with newly generated samples, wRACOG selects those samples from the Markov chain that have the highest probability of being misclassified by the learning model generated from the previous version of the dataset.

While the RACOG algorithm generates minority class samples for a fixed (predefined) number of iterations, the wRACOG algorithm keeps on fine tuning its hypothesis at every iteration by adding new samples until there is no further improvement with respect to a chosen performance measure. As our goal is to improve the performance of classifiers on the minority class, wRACOG keeps on adding new samples until there is no further improvement in *sensitivity* (true positive rate) of the *wrapper* classifier (the core classifier that retrains at every iteration) of wRACOG. This process acts as the "stopping criterion" for the wRACOG algorithm. However, the choice of performance measure for the stopping criterion is application dependent.

At each iteration of wRACOG, new minority class samples are generated by the Gibbs sampler. The model learned by the *wrapper* classifier on the enhanced set of samples produced in the previous iteration is used to make

predictions on the newly generated set of samples. Those samples that are misclassified by the model are added to the existing set of data samples and a new model is trained using the *wrapper* classifier. Meanwhile, at each iteration the trained model is used to perform prediction on a held out *validation* set and the sensitivity of the model is recorded. Generation of new samples stops once the standard deviation of sensitivities over the past iterations falls below a threshold. As the wRACOG algorithm might end up running many iterations, the standard deviation of sensitivities is calculated over a fixed number of most recent iterations stored in *slide_win*. Currently, the 10 most recent iterations of wRACOG are considered to calculate the standard deviation of sensitivities; while the threshold has been fixed to 0.02. These values are determined by performing an empirical study on the datasets described in this paper. The wRACOG algorithm is summarized in Table 2.4.

Algorithm: wRACOG

Input: *train* = training dataset enhanced at each iteration with new samples; *validation* = validation set on which trained model is tested at every iteration to track improvements; *wrapper* = classifier that is retrained on the enhanced dataset at every iteration; *slide_win* = sensitivities of previous iterations; *threshold* = threshold of standard deviation of sensitivities over *slide_win*

Output *new_train* = final hypothesis encoded in the oversampled training set

1: Build *model* by training *wrapper* on *train*
2: Run Gibbs sampler on all minority class samples simultaneously
3: **do**
4: Perform prediction on newly generated samples using *model*
5: Add misclassified samples to form *new_train*
6: Train *model* on *new_train* using *wrapper*
7: Perform prediction on *validation* set using trained *model* and add sensitivity to *slide_win*
8: **while** $\sigma(slide\_win) \geqslant threshold$

Table 2.4: The wRACOG algorithm

Although wRACOG is similar to existing boosting techniques (such as AdaBoost) in the way misclassified samples are assigned higher weights to

ensure they get selected during random sampling in the next boosting iteration, there are a number of major distinctions. Firstly, while in traditional boosting, both training and prediction are performed on the the same set of data samples, wRACOG trains the current hypothesis on the data samples from the previous iteration and performs prediction only on the newly generated samples from the Gibbs sampler. Secondly, there is no concept of changing weights of the samples before resampling, as the newly generated samples are directly added to the existing set of samples. Thirdly, wRACOG does not use weighted voting of multiple hypotheses learned at every iteration. Instead, it employs multiple iterations to fine tune a single hypothesis. We hypothesize that by applying this approach we can reduce the generation of redundant samples to converge more closely to the true distribution of the minority class, and also reduce the overhead of generating multiple hypotheses as is employed by traditional boosting techniques.

2.4 Experimental Setup

The goal of the current work is to design algorithms that effectively classify data points from all classes, even with imbalanced class distribution. We hypothesize that Gibbs sampling, particularly with additional modelling of attribute dependencies, will yield improved results for problems that exhibit class imbalance. To validate our hypothesis, we compare the results for alternative sampling algorithms using the datasets summarized in Table 2.5.

2.4.1 Alternative Sampling Approaches

While reporting the results for the experiments, six different methods are evaluated in order to validate the proposed hypothesis. The first method, named *Baseline*, relies upon the baseline dataset without any sampling or preprocessing. The evaluation is performed using the classifiers discussed in Section 2.4.3. The rest of the methods discussed in this section preprocess the baseline dataset and the *Baseline* approach is used to evaluate and compare their performance.

SMOTE [53] SMOTE is an oversampling approach in which the minority class is oversampled by creating "synthetic" examples based on spatial location of the data points in the Euclidean feature space. Oversampling is

performed by considering each minority class data point and introducing synthetic examples along the line segments joining any or all of the k-minority class nearest neighbors. The k-nearest neighbors are randomly chosen depending upon the amount of oversampling required. Synthetic data points are generated in the following way: First, the difference between the data point under consideration and its nearest neighbor is computed. This difference is multiplied by a random number between 0 and 1, and it is added to the data point under consideration. This results in the selection of a random point in the Euclidean space, along the line segment between two specific data points. Consequently, by adding diversity to the minority class, this approach forces the decision boundary between the two regions to be crisper. However, as SMOTE does not rely on the probability distribution of the minority class as a whole, there is no guarantee that the generated samples belong to the minority class, especially when the samples from the majority and minority classes overlap [71].

In our experiments, we have used the publicly available implementation of SMOTE available with the Weka API. Although there is no ideal class distribution for effective classification of examples from all classes, we use SMOTE to oversample the minority class and attain a 50:50 class distribution, which is considered near optimal [85].

SMOTEBoost [54] By using a combination of SMOTE and a standard boosting procedure, SMOTEBoost tries to better model the minority class by providing the learner not only with the minority class examples that were misclassified in the previous boosting iteration but also with a broader representation of those instances achieved by SMOTE. The inherent skewness in the updated distribution towards majority class points at every iteration of the boosting procedure is rectified by introducing SMOTE to increase the number of minority class points according to the distribution learned from the previous iteration. Thus, introduction of SMOTE increases the number of minority class samples for the learner and focuses on these cases in the distribution at each boosting round. In addition to maximizing the margin for the skewed class dataset, this procedure also increases the diversity among the classifiers in the ensemble because at each iteration a different set of synthetic samples is produced.

A SMOTEBoost implementation is not currently available. As a result, we implemented the algorithm using MATLAB and made it available at the

MATLAB CENTRAL File Exchange website[3]. This implementation considers 10 boosting iterations as the experiments performed by Seiffert et al. [52] suggest that there is no significant improvement between 10 and 50 iterations of AdaBoost. However, if the *error rate* or *pseudo loss* of the learning hypothesis in the boosting procedure exceeds 0.5 for more than a specified number of consecutive iterations, the boosting iteration is broken and rolled back to the state before *pseudo loss* > 0.5 was encountered. Chawla et al. had used RIPPER [86], a divide-and-conquer strategy-based rule learner, as the weak learner in the boosting procedure. Our implementation uses the same classifiers (mentioned in Section 2.4.3) that have been used to evaluate the performance of the sampling techniques.

Although iterative learning of the weak learner by the boosting procedure attempts to form a stronger hypothesis which better classifies minority class data points, the quality of the generated samples is still dependent on the spatial location of minority class samples in the Euclidean feature space, as is done by SMOTE. Moreover, SMOTEBoost executes SMOTE ten times (for ten boosting iterations), generating $10 \times (\#\textit{majority class samples} - \#\textit{minority class samples})$ samples and thus making it highly computationally expensive.

RUSBoost [52] RUSBoost is very similar to SMOTEBoost, but claims to achieve better classification performance on the minority class data points by random under-sampling (RUS) of majority class examples. Although this method results in a simpler algorithm with a faster model training time, it is not able to achieve favorable performance (explained later in Section 2.5) as claimed by Seiffert et al., especially when the datasets have an absolute rarity of minority class examples. RUSBoost is used as an example of an under-sampling technique for comparing the performance of under-sampling approaches with the oversampling techniques, which is the primary focus of this paper.

We also implemented RUSBoost in MATLAB and made it available at the MATLAB CENTRAL File Exchange website[4]. The number of boosting iterations and types of weak learners used for the boosting procedure are same as that of the SMOTEBoost implementation. However, as most of the data sets under consideration have an absolute rarity of minority class examples,

[3]http://www.mathworks.com/matlabcentral/fileexchange/37311

[4]http://www.mathworks.com/matlabcentral/fileexchange/37315

the class imbalance ratio has been set to 35:65 (minority:majority). The choice of class distribution is based on the empirical investigations performed by Khoshgoftaar et al. [87] on datasets with rare minority class samples. Khoshgoftaar et al. empirically verified that a 35:65 class distribution would result in better classification performance than a 50:50 class distribution when examples from one class are extremely rare, but the examples of the other class(es) are plentiful.

The proposed approaches, RACOG and wRACOG are compared with the aforementioned alternative sampling techniques. RACOG oversamples the minority class to achieve a 50:50 class distribution. Therefore, the total number of iterations is fixed and is determined on the basis of (*#majority class samples −#minority class samples*), *burn-in* and *lag*. A *burn-in* period of 100 and a *lag* of 20 iterations is chosen as the convention in the literature [88] to avoid autocorrelation among the samples generated by the Gibbs sampler. As mentioned earlier, wRACOG's sample selection strategy continues to add samples to the minority class until the standard deviation of sensitivity over the 10 recent iterations fall below an empirically determined threshold. The classifiers presented in Section 2.4.3 are used as *wrapper* classifiers and are trained on an enhanced dataset at every iteration of wRACOG.

2.4.2 Datasets

We evaluate the applicability of our algorithm for *prompting* dataset and other five additional real-world datasets from the UCI repository that exhibit the class imbalance problem: *abalone*, *car*, *nursery*, *letter*, and *connect-4*. Characteristics of these datasets are summarized in Table 2.5. The real-valued attributes were transformed into discrete sets by using a simple binning technique. The multi-class datasets were converted into binary class by choosing a particular class label as *minority* class and the rest of the class labels together as *majority* class.

2.4.3 Classifiers for Performance Evaluation

We choose four most common classifiers in machine learning to evaluate the performance of the proposed methods and other sampling approaches. The Weka implementations of these classifiers found with the Weka Java API are integrated into our implementations in MATLAB. The parameter values that we use for the classifiers in these experiments are listed in Table 2.6.

Dataset	Size	Dim	% Min. Class	Description
Prompting	3,980	17	3.7437	Description provided above.
Abalone	4,177	8	6.2006	Predicting age of large sea snails, abalone, from its physical measurements.
Car	1,728	6	3.9931	Evaluating car acceptability based on price, technology and comfort.
Nursery	12,960	8	2.5463	Nursery school application evaluation based on financial standing, parents' education and social health.
Letter	20,000	16	3.7902	Classifying English alphabets using image frame features.
Connect-4	5000	42	10.0000	Predicting first player's outcome in connect-4 game given 8-ply positions information .

Table 2.5: Description of selected datasets

All the experiments are performed using 5-fold cross validation.

2.4.4 Performance Measures

Performance measures are critically important to assess classification performance. By convention, in a binary classifier, the minority and majority classes is referred as *positive* and *negative* classes respectively. Traditionally, the most frequently used performance measures are *Accuracy* and *Error Rate*. Following this convention, *Accuracy* and *Error Rate* can be defined as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}; ErrorRate = 1 - Accuracy \quad (2.6)$$

These conventional performance measures give a very high level idea about the classifier's performance on a given data set. However, in a class

Classifier	Parameter Values
C4.5 Decision Tree	Confidence factor = 2, Minimum # instances per leaf = 2
SVM	Kernel = RBF, RBF kernel $\gamma = 0.01$
k-Nearest Neighbor	k =5, Distance measure = Euclidean
Logistic Regression	Log likelihood ridge value = $1 \times 10^{(-8)}$

Table 2.6: Classifiers and parameter values

imbalance scenario this can be deceiving. *Accuracy* and *Error Rate* are ineffective for evaluating classifier performance in a class imbalanced dataset as they consider different types of classification errors as equally important. For example, in an imbalanced dataset with 5% minority class, a random prediction of all the test instances being negative, will give an accuracy of 95%. However, in this case the classifier could not correctly predict any of the minority class points. Therefore, in order to provide comprehensive assessment of the imbalanced learning problem, we need to either consider metrics that can report the performance of classifier on two classes separately or not let the effect of class imbalance get reflected in the metric. In the following, performance metrics that are capable of measuring the effectiveness of the classifiers in the presence of class imbalance is described.

Sensitivity Sensitivity is a measure of completeness, i.e., the portion of the positive class examples that were predicted correctly. It is not sensitive to data distribution and does not give any insight on the number of examples that have been incorrectly predicted as positive. Sensitivity is represented as:

$$Sensitivity = \frac{TP}{TP + FN} \tag{2.7}$$

Specificity Like sensitivity, specificity is also a measure of completeness, but for the negative class, i.e., the portion of the negative class examples that were predicted correctly. It is represented as:

$$Specificity = \frac{TN}{TN + FP} \tag{2.8}$$

G-mean G-mean [89] evaluates the degree of inductive bias in terms of a ratio of positive accuracy (i.e., Sensitivity) and negative accuracy (i.e.,

Specificity). It is given by:

$$G - mean = \sqrt{\frac{TP}{TP + FN} \times \frac{TN}{TN + FP}} = \sqrt{Sensitivity \times Specificity} \tag{2.9}$$

ROC Curves and AUC-ROC The graphical representation of ROC assessment techniques [90, 91] helps in visualizing the benefits (TP Rate) and costs (FP Rate) of the classifier, defined as:

$$TPRate = \frac{TP}{TP + FN}; FPRate = \frac{FP}{FP + TN} \tag{2.10}$$

On the other hand, the area under ROC curve (AUC-ROC) [92] gives a single measure of a classifier's performance for evaluating which model is better on average.

2.5 Results

The primary limitation of classifiers that model imbalanced class datasets is in achieving desirable prediction accuracy for minority class instances. That is, the sensitivity is typically low assuming that the minority class is represented as the positive class. The proposed approaches place emphasis on on boosting the sensitivity of the classifiers while maintaining a strong prediction performance for both of the classes, which is measured by G-mean. However, we understand that the choice of performance measure that needs to be boosted when dealing with class imbalanced dataset is tightly coupled with the application domain. Moreover, finding a trade-off between improving classifier performance on the minority class in isolation and on the overall dataset should ideally be left at the discretion of the domain expert.

We compare the sensitivity of this C4.5 decision tree on all six approaches in Figure 2.3. From the figure it is quite evident that both RACOG and wRACOG perform better than the other methods. Also, there is not too much performance variability for RACOG and wRACOG over the five cross validation. RUSBoost fails by performing nowhere close to the oversampling techniques. The poor performance of RUSBoost can be attributed to the rarity of minority class samples in the datasets under consideration. When the minority class samples are already rare, random under-sampling of majority

class to achieve a 35:65 class distribution (as done by RUSBoost) at each iteration of RUSBoost, makes the majority class samples rare as well. Thus, the weak leaner of RUSBoost does not learn anything on the majority class which increases the error rate of the hypothesis at every iteration. Both SMOTE and SMOTEBoost are good contenders, although, SMOTEBoost outperforms SMOTE in most of the cases. Although wRACOG performs better than RACOG on four datasets, they perform equally well for the *car* and *nursery* datasets. We verify the statistical significance of these improvements using a *Student's t test*. RACOG and wRACOG do exhibit significant ($p < 0.05$) performance improvement over SMOTEBoost. The statistical significance of the performance improvements of RACOG and wRACOG over SMOTEBoost for alternative classifiers are reported in bold in Appendix B.

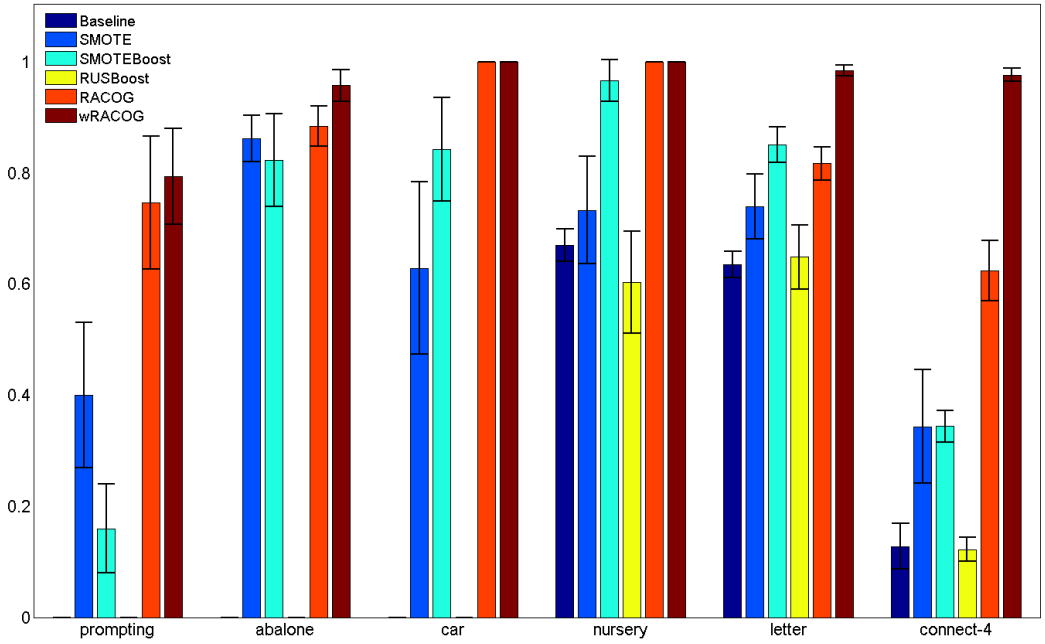

Figure 2.3: Sensitivity for C4.5 Decision Tree

Because the sampling techniques boost the minority class, there is a tendency for the false positive rate to increase. However, the increase in false positive rate is not very significant and therefore it does not affect the G-mean scores. Figure 2.4 reports the G-mean scores of C4.5 decision tree on all the methods when tested with the six datasets. Clearly, RACOG and

wRACOG result in a superior performance over SMOTE and SMOTEBoost. However, SMOTEBoost is a very strong contender.

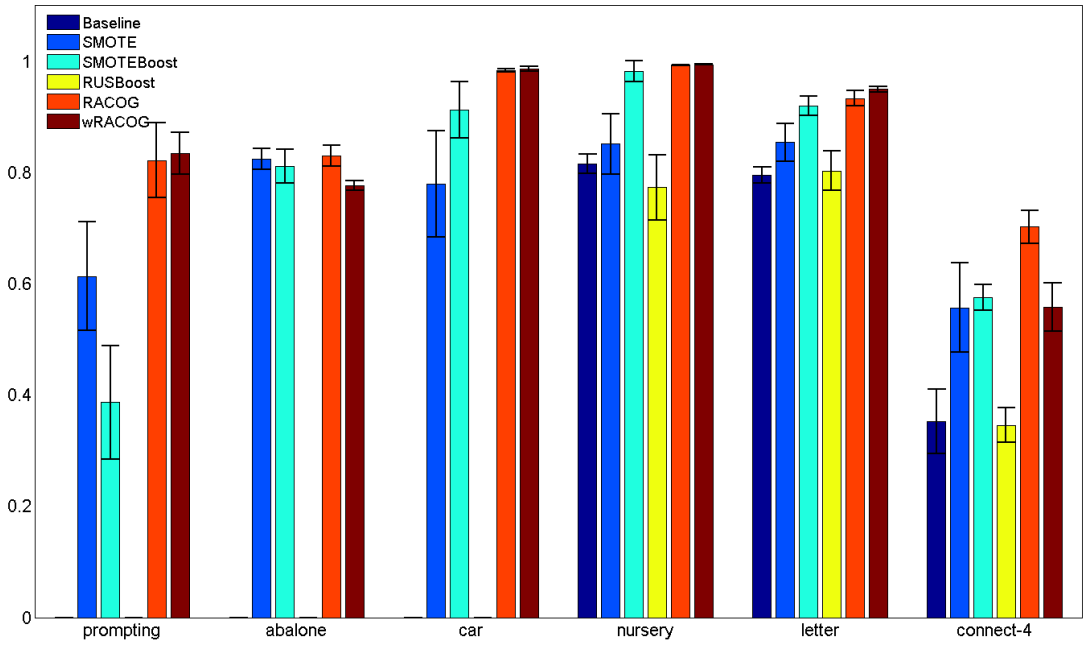

Figure 2.4: G-mean for C4.5 Decision Tree

In Figure 2.5, we plot the ROC curves produced by the different approaches on each of the 6 datasets when evaluated with a C4.5 decision tree. For the *prompting*, *abalone* and *car* datasets, Baseline and RUSBoost do not perform any better than random prediction. The performance of RACOG and wRACOG are clearly better than SMOTE and SMOTEBoost for the *prompting* dataset. However, there is no clear winner among them for the *abalone*, *car* and *nursery* datasets. The AUC-ROCs are reported in Table 2.7. For the *letter* and *connect-4* datasets, the AUC-ROC for SMOTEBoost is higher than RACOG and wRACOG. However, no statistically significant improvement of RACOG and wRACOG was found over SMOTEBoost based on AUC-ROC. Also, as there is no clear winner between RACOG and wRACOG on any of the performance measures we do not conduct any statistical significance test between them.

Convergence Diagnostic Test In the current work we explore the strengths of Markov chain Monte Carlo techniques, specifically Gibbs sampling, to

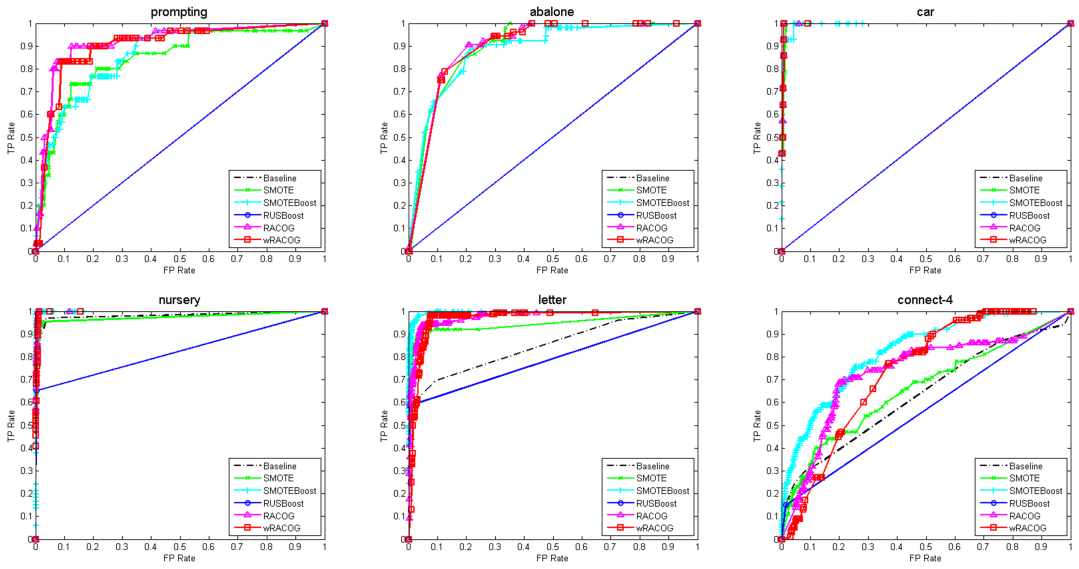

Figure 2.5: ROC curves produced by C4.5 Decision Tree

Dataset	Baseline	SMOTE	SMOTEBoost	RUSBoost	RACOG	wRACOG
prompting	0.5000 ± 0.0000	0.8681 ± 0.0359	0.8546 ± 0.0334	0.5000 ± 0.0000	0.8851 ± 0.0207	0.8717 ± 0.0404
abalone	0.5000 ± 0.0000	0.8693 ± 0.0272	0.8718 ± 0.0262	0.5000 ± 0.0000	0.8646 ± 0.0233	0.8727 ± 0.0208
car	0.5000 ± 0.0000	0.9595 ± 0.0320	0.9957 ± 0.0025	0.5000 ± 0.0000	0.9879 ± 0.0027	0.9904 ± 0.0067
nursery	0.9849 ± 0.0077	0.9515 ± 0.0219	0.9999 ± 0.0001	0.7999 ± 0.0456	0.9968 ± 0.0005	0.9967 ± 0.0005
letter	0.8745 ± 0.0056	0.9340 ± 0.0153	0.9865 ± 0.0080	0.8226 ± 0.0286	0.9659 ± 0.0102	0.9707 ± 0.0026
connect-4	0.6215 ± 0.0310	0.6907 ± 0.0335	0.8317 ± 0.0140	0.5552 ± 0.0092	0.7333 ± 0.0181	0.7115 ± 0.0282

Table 2.7: AUC-ROC for C4.5 Decision Tree

generate new minority class samples. A major issue for the successful implementation of any MCMC technique is to determine the number of iterations required for the generated samples to converge to the target distribution.

Researchers in econometrics [93] use formal methods such as the Raftery-Lewis test [88] to make this determination. Given outputs from a Gibbs sampler, the Raftery-Lewis test provides the answer to: *how long to monitor the chain of samples?* Here, one specifies a particular quantile q of the distribution of interest (typically 2.5% and 97.5%, to give a 95% confidence interval), an accuracy σ of the quantile, and a power $1-\beta$ for achieving this accuracy on the specified quantile. These parameters are used to determine the burn-in (M), the total number of iterations required to achieve the desired accuracy for the posterior (N), the appropriate lag (k), and the number of iterations (N_{min}) that would be needed if the samples represented an

independent and identically distributed (iid)[5] chain, which is not true in our case because of the autocorrelation structure present in the Markov chain of generated samples.

The diagnostic was designed to test the number of iterations and burn-in needed, by first running and testing a shorter pilot chain. In practice, it is also used to test any normal Markov chain generated by a Gibbs sampler to see if it satisfies the results that the diagnostic suggests. These output values can be combined to calculate *i-stat* defined as follows:

$$i - stat = \frac{M + N}{N_{min}} \tag{2.11}$$

i-stat measures the increase in the number of iterations due to dependence in the sequence (Markov chain). Raftery and Lewis indicate that *i-stat* is indicative of a convergence of the sampler if the value does not exceed 5.

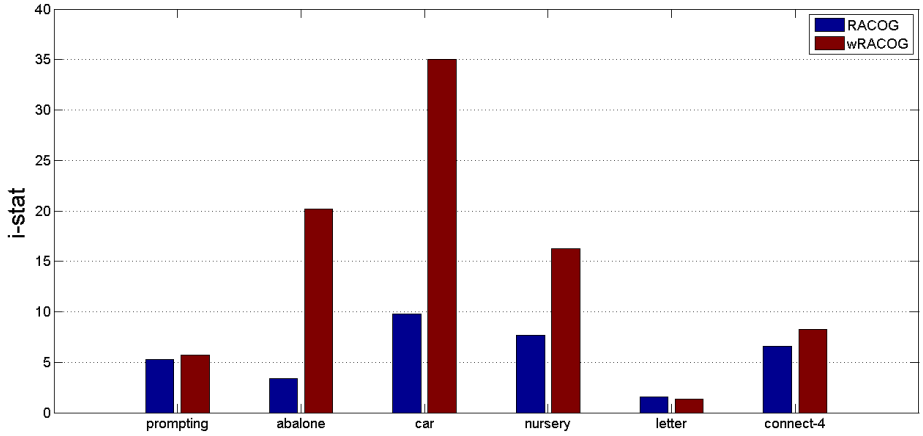

Figure 2.6: *i-stat* values for RACOG and wRACOG

From Figure 2.6 it is evident that convergence is almost achieved by both RACOG and wRACOG for the *prompting* and *letter* datasets. However, the wRACOG *i-stat* value is greater than the RACOG value for the remainder of the datasets. Although this indicates that wRACOG could not converge to the target minority class distribution for the *abalone*, *nursery* and *car*

[5]An *iid* sequence is a very special kind of Markov chain; whereas a Markov chain's future is allowed (but not required) to depend on the present state, an *iid* sequence's future does not depend on the present state at all.

datasets, we have seen that the wRACOG sensitivity and G-mean are better or at par with RACOG. Therefore, one explanation is that convergence here (not necessarily probabilistic) is subjective as it is tightly coupled with the application.

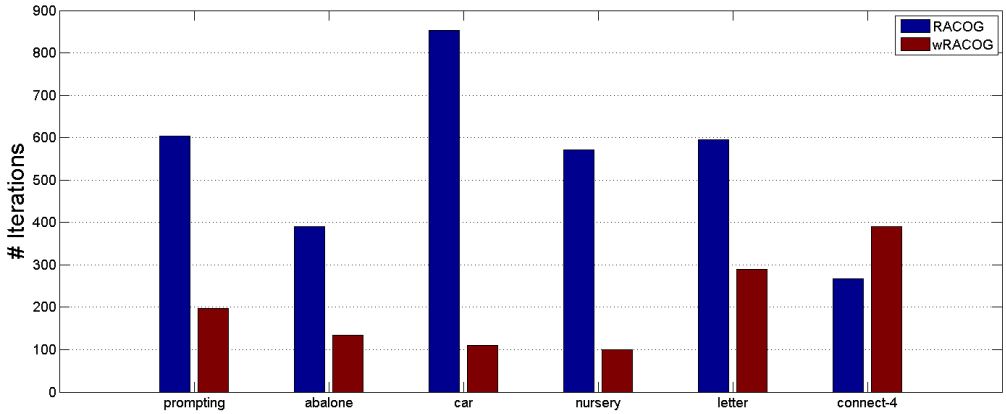

Figure 2.7: Comparison of total number of iterations required by different methods to achieve given *i-stat*

We further analyze the convergence of RACOG and wRACOG in terms of the number of iterations the Gibbs sampler undergoes to achieve the *i-stat* value reported in Figure 2.6. From Figure 2.7 we notice that wRACOG generates a much fewer (except for *connect-4* dataset) number of iterations ($\sim 63\%$) than RACOG. This explains the poor performance of wRACOG in terms of probabilistic convergence on the Raftery-Lewis test. However, the sensitivity and G-mean score are not affected by the reduction in the number of iterations.

The number of samples added to the baseline datasets by the different oversampling algorithms for achieving the reported performance is also an important parameter to analyze. Ideally, we would want to obtain a high performance by adding as less number of samples as possible. Figure 2.8 illustrates the number of samples added by the different approaches. As the number of samples added by SMOTEBoost is far higher than other methods, we present log_{10} values of the *number of added samples* so that the comparison could be better represented in the plot. SMOTE and RACOG try to achieve a 50:50 class distribution and thus add samples accordingly. SMOTEBoost

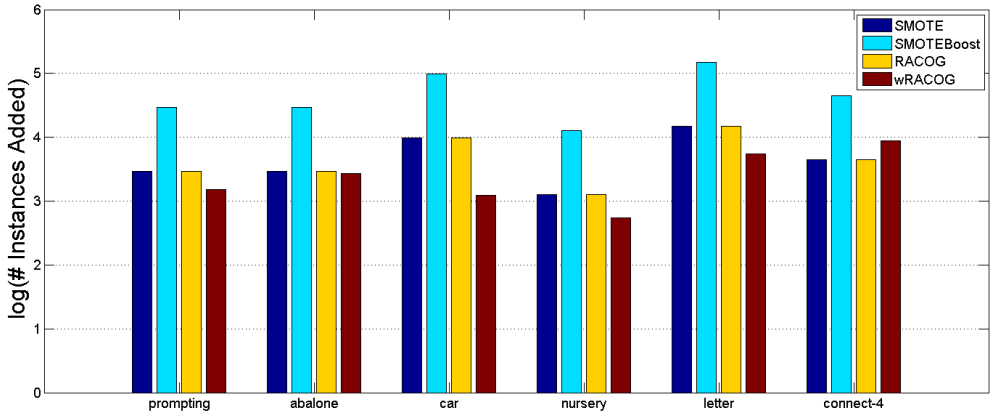

Figure 2.8: Comparison of *log*(number of instances added) by different methods

requires ten boosting iterations to produce ten hypotheses on which weighted voting is performed while prediction. The hypothesis learned at each iteration of SMOTEBoost has the form: $h_t : X \times Y \longrightarrow [0, 1]$, and therefore stores the instances generated by SMOTE at every iteration. Hence, SMOTEBoost adds ten times the number of samples added by SMOTE. On the other hand, wRACOG requires a fraction ($\sim 56\%$) of the number of samples generated by SMOTE and RACOG, and ($\sim 5.6\%$) of the number of samples generated by SMOTEBoost, to obtain superior performance. We attribute this behavior to wRACOG's sample selection methodology which ensures the diversity in the samples that are added.

2.6 Summary

In this chapter, we propose two Gibbs sampling-based algorithms for generating new minority class samples for class imbalanced datasets. While conventional Gibbs sampling uses joint probability distribution of random variables (attributes) to generate samples, we do not find this approach to be suitable for class imbalanced datasets which (mostly) contains rare minority class samples. As a solution, we use a Bayesian tree-based approach to impose dependencies among attributes. Both the proposed approaches, RACOG and wRACOG, use Gibbs sampling at their core. However, they

differ in the sample selection strategy. RACOG runs the Gibbs sampler for a predetermined number of iterations and selects samples from the Markov chain generated by the Gibbs sampler using predefined *burn-in* and *lag*. On the other hand, wRACOG selects samples that have the highest probability of being misclassified by the existing learning model. It keeps on adding samples unless there is no further improvement in sensitivity over a predefined number of most recent iterations.

Experiments with RACOG and wRACOG on a wide variety of datasets and classifiers indicate that the algorithm is able to attain higher sensitivity than other methods, while maintaining higher G-mean. This supports our hypotheses that generating new samples by considering the global distribution of minority class points is a good approach for dealing with class imbalance. The motivation to focus mainly on improving sensitivity comes from the our application domain in pervasive computing. However, we ensure that the performance of the classifiers on both the classes is not hampered.

In the future, we want to experiment with alternative methods to the Bayesian tree to see if attribute dependencies could be captured more accurately. As the current Bayesian tree approach is computationally lightweight, we would ensure that this property is preserved while experimenting with alternative approaches. Although the current stopping criteria for the wRACOG algorithm is based on the standard deviation of sensitivity for the 10 most recent iterations, we would like to perform a study on the effects of standard deviation threshold and the number of iterations in the past that needs to be considered. Also, we would propose RACOGBoost by combining the advantages of boosting with that of RACOG.

Chapter 3

Class Overlap Problem

3.1 Introduction

In spite of making significant progress in the area of class imbalance problem, researchers are facing new emerging class imbalance challenges that make the problem harder to solve with existing techniques. Consider a network intrusion detection system. Network intruders these days have become smart enough to disguise their identity as legitimate users. This is a binary class learning problem where data points may appear as valid examples of both *positive* and *negative* classes. The same situation exists in problem domains such as, credit card fraud detection, character recognition or automated prompting in smart environments [40] where samples from different classes have very similar characteristics. The minor differences present between the samples of two different classes are usually difficult to capture in the feature vector proposed by the domain expert. Therefore, there is a growing algorithmic need to deal with this issue. In this chapter, automated prompting in smart environments has been considered as the application domain for the evaluation of the proposed approach. Although this chapter focuses on the automated prompting problem, our approach is easily extensible for other problem domains which have datasets of a similar nature.

From a supervised learning perspective, the goal of the prompting system is to classify an activity step (represented as a training example) either as a *prompt* step or a *no-prompt* step. Thus, it is a binary classification problem. As, in a realistic setting, there are very few situations that would require a prompt as opposed to situations that would not, the number of training

examples for *prompt* class is extremely low as compared to *no-prompt* class. This makes the data inherently class imbalanced. Moreover, the features that represent each activity step are insufficient to draw crisp boundaries between these classes for some regions in the data space that is ambiguous. This causes the occurrence of overlapping classes in addition to the inherent presence of imbalance class distribution. It can be argued that, if the proposed featured are incapable of capturing the representative properties of the training examples that can help in proper distinction of the two classes, why not engineer new features that can add more distinctive representation of the classes? The best answer to this question is that the lack of infrastructural requirements in a realistic setting restricts the addition of new features.

3.2 Problem Definition

The class overlap problem [94] occurs when there are ambiguous regions in the data space where there are approximately the same number of training examples from both classes. Conceptually, ambiguous regions can be visualized as regions where the prior probabilities for both classes is approximately equal and thus makes it difficult or impossible to distinguish between the two classes. This is because, it is difficult to make a principled choice of where to place the class boundary in this region since it is expected that the accuracy will be equal to the proportion of the volume assigned to each class. Figure 3.1 illustrates the difference between normal data with crisp class boundaries and data with overlapping classes.

Overlapping classes when combined with class imbalance causes the resultant problem to be harder to solve than solving them independently. It has been seen in some cases that identifying the overlapping region in the data space and getting rid of those instances makes the data linearly separable. This idea is implemented in approaches such as, SMOTE+Tomek [74] and Tomek+CNN [72] among few others. But, in some cases an additional problem of rare training examples of the positive class makes class overlap in imbalanced class distribution even more difficult.

As mentioned in the previous section, in a smart environment setting class overlap occurs due to insufficient number of features that can differentiate between the *prompt* class and the *no-prompt* class. The presence of class overlap problem in prompting data is confirmed by a 3-dimensional data visualization using Principal Component Analysis (PCA)-based [95] di-

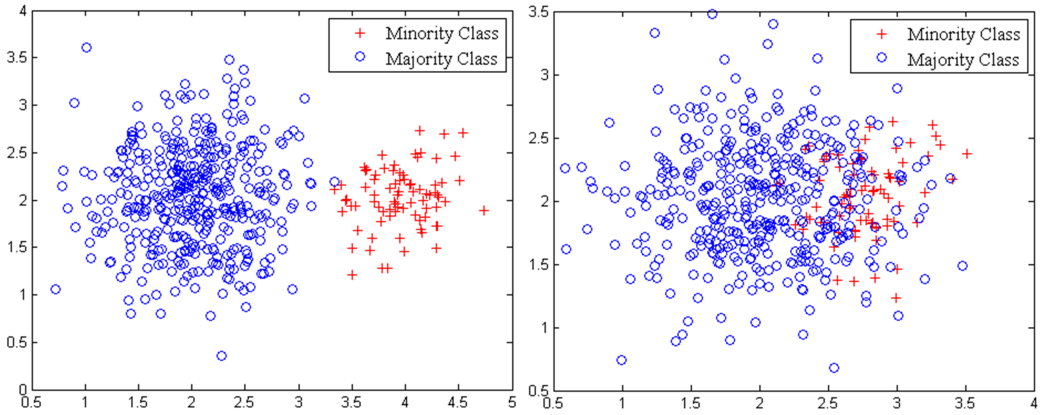

Figure 3.1: Data without overlap (left), data with overlap (right)

mensionality reduction on the attributes. Figure 3.2 shows the 3-dimensional PCA plot of the prompting data. The figure clearly shows that the *prompt* class examples are highly embedded in *no-prompt* class examples.

The class overlap problem in imbalanced class data can be subdivided into a three-step sequential problem as shown in a schematic view in Figure 3.3. First, it is important to identify the regions of overlap in the data space. However, it is a major obstacle in studying overlap. Once the overlapping regions are successfully identified, the training examples in this region should be handled with a separating, merging or discarding scheme [96]. The next step is to perform the learning using different machine learning algorithms. The proposed approach is a preprocessing technique which under-samples negative class examples in the overlapping region and thus creates a bias for the learning model towards the positive class.

3.3 Related Work

While there has not been significant work in dealing with the class overlap problem in combination with imbalanced class distribution, the problem of overlapping classes or ambiguous data has been widely studied in isolation [97, 98, 99, 100], particularly in the areas of character recognition and document analysis [101], text classification, automated image annotation, credit card fraud detection, and drug design [102].

There have been several systematic and extensive investigations to study

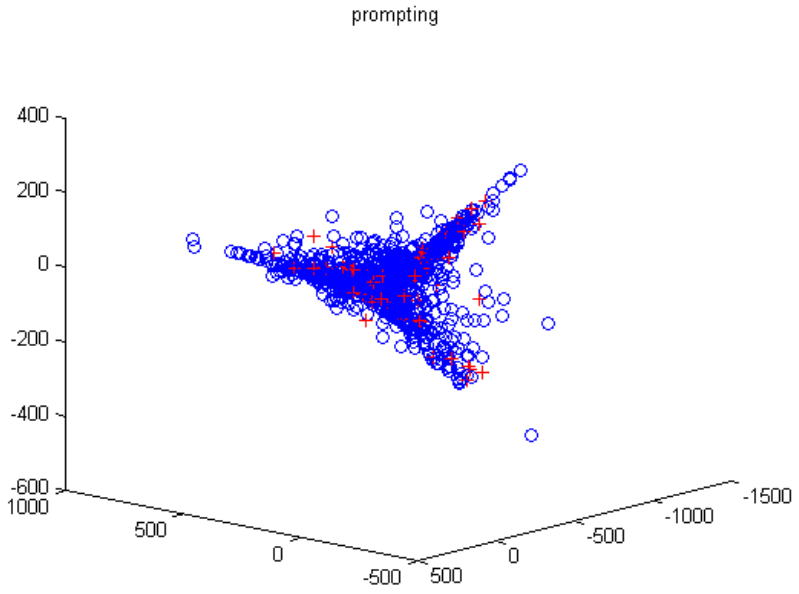

Figure 3.2: 3D PCA plot for Prompting data

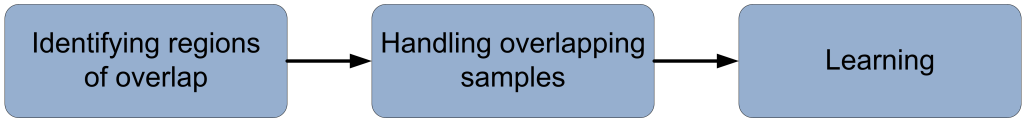

Figure 3.3: Steps taken to address class overlap

the nature of classifiers when they are faced with the class overlap problem in addition to an imbalanced class problem. Prati et al. [103] give a vivid illustration of the cause of imbalanced class distribution posing a problem in the presence of high degree of class overlap. They show that overlap aggravates the problem of imbalance and is sufficient to degrade the performance of the classifier on its own. The same authors report the performance of different balancing strategies on artificial datasets in [104]. Garcia et al. [105] analyze the combined effects of class imbalance and class overlap on instance-based classification. This work is extended [106] by using several performance measures to see which one of them captures the degraded performance more accurately.

A major obstacle in solving class overlap problem in data is identification

of ambiguous or overlapping regions. However, this issue has been addressed to some extent by the approaches that deal with class overlap problem in isolation. Tang et al. [99] proposed a k-Nearest Neighbor based approach to identify ambiguous regions in the data space. Trappenberg et al. [97] took a very similar approach to identify ambiguous regions. Visa et al. [107] perform a fuzzy set representation of the concept and thus incorporate overlap information in their fuzzy classifiers. In addition, Xiong et al. [96] use one-class classification algorithm Support Vector Data Description (SVDD) to capture the overlapping regions in real-life datasets which have imbalanced class distribution as well.

Once the overlapping region of the data space is identified, the obvious next step is to handle the training examples that belong to this region. Xiong et al. [96] propose that the data with the presence of class overlapping can be modeled with three different schemes: discarding, merging and separating. The *discarding scheme* ignores that data in the overlapping region and just learns on the rest of the data that belongs to the non-overlapping region. SMOTE + Tomek Links [108] uses the discarding scheme to improve classification performance of protein annotations in bioinformatics. While the discarding scheme works satisfactorily for datasets that have ample number of training examples from both classes, it would perform drastically when applied to datasets which have absolute rarity in data.

The *merging scheme* merges the data in the overlapping region into a new class. A two-tier classification model is built on the data. The upper tier classifier focuses on the whole data with an additional class which represents the overlapping region. The lower tier classifier on the other hand focuses on the data that belongs to the overlapping region. Trappenberg et al. [97] proposed a scheme that refers to the overlapping region class as *IDK* (I don't know) and do not attempt to predict the original class of this data. The authors argue that, although this scheme loses some prediction of data, a drastic increase of confidence can be gained on the classification of the remaining data. Hashemi et al. [98] take a very similar approach to address the issue.

In the *separating scheme*, the data from overlapping and non-overlapping regions are treated separately to build the learning models. Tang et al. [99] proposed a multi-model classifier named Dual Rough Support Vector Machine (DR-SVM) which combines SVM and kNN under rough set technique. kNN is used to extract boundary patterns or overlapping regions. Two different SVMs are then trained for the overlapping and non-overlapping regions.

But, the classification result will show whether a pattern lies in overlapping region. Although, the classification of a test example as belonging to overlapping and non-overlapping region depends on the goal of the application problem, this methodology would involve an additional domain expert knowledge to determine the class of the test example. Thus, this scheme is not suitable for applications where it is a requirement of the system to determine the class of the test example and has no room for additional domain expert intervention.

All of the aforementioned schemes either consider the overlapping data as noise or just avoid making a decision on their original classes so that the confidence of prediction on the remaining data could be increased. This approach of partially "*avoiding the problem*" rather than proposing a solution is not appropriate for many real-life problem domains where it is absolutely necessary for the system to take a decision with certainty (often due to a time-critical nature) rather than waiting for the domain expert intervention. For example, in the problem domain of intrusion detection where attackers can disguise themselves as legitimate users, high traffic of attackers necessitates to take a time-critical decision on the authenticity of the user.

In this chapter we take a preprocessing approach (the following section describes the proposed approach) similar to the discarding scheme to deal with the overlapping training examples. Instead of designating the boundary points as noises, our approach considers them as crucial for decision making in the overlapping region. The minority class points in the overlapping region are retained and the majority class points are discarded to make a clear distinction between the minority class points in the overlapping region and the rest of the dataset.

3.4 Cluster-Based Under-Sampling(ClusBUS)

By performing a hypothesis testing, Denil et al. proved [109] that overlap and imbalance are not two independent factors. They have very strong coupling when it comes to the problem of imbalanced class distribution. Denil et al. showed that if overlap and imbalance levels are too high, good performance cannot be achieved regardless of the amount of available training examples. Therefore, the purpose of employing a Cluster-Based Under-Sampling (ClusBUS) technique is to get rid of the overlapping class problem and the hypothesis is that achieving success with the overlap problem would also be

helpful in getting rid of the detrimental effects of class imbalance problem to some extent, as the majority class is being under-sampled.

The idea of devising this technique is derived from the use of Tomek links [72] combined with other sampling methods like Condensed Nearest Neighbor [110] and SMOTE [74]. Tomek links are defined as pairs of two minimally distanced opposite class training examples. More formally, given two examples E_i and E_j belonging to different classes, and $d(E_i, E_j)$ being the distance between E_i and E_j, a (E_i, E_j) pair is called a Tomek link if there is not an example E_k such that $d(E_i, E_k) < d(E_i, E_j)$. If two examples form a Tomek link, then either one of these examples is noise or both examples are on or near the class boundary. Tomek links are used both as a data cleansing method and as under-sampling method. In order to perform data cleansing, the Tomek links are removed. On the other hand, under-sampling involves removing the majority class examples from the Tomek links and retaining the minority class examples.

One-sided selection (OSS) [73] is an under-sampling method that applies Tomek links followed by the application of Condensed Nearest Neighbor (CNN). In this method, Tomek links are used to remove noisy and borderline majority class examples. As a small amount of noise can make the borderline examples fall on the wrong side of the decision boundary, borderline examples are considered as unsafe. CNN is used to remove examples from the majority class that are far away from the decision boundary. The rest of the majority and minority class examples are used for learning.

As opposed to using Tomek links in OSS to find minimally distanced nearest neighbor pairs of opposite classes and then remove majority class examples, the proposed ClusBUS algorithm finds *interesting* clusters with a *good mix* of minority and majority class examples. The definition of *good mix* is determined by the degree of minority class dominance in the clusters. The majority class examples from these clusters are then removed.

Table 3.1 summarizes the ClusBUS algorithm. First, clustering is performed on the training data ignoring the class attribute using Euclidean distance as the distance measure. The degree of minority class dominance, denoted by δ, for each of these clusters is calculated as the ratio of number of minority class examples to the size of the cluster. Therefore, $\delta = 0$ indicates that all the examples of the cluster belong to the majority class, and $\delta = 1$ indicate that all the examples belong to the minority class. The clusters whose δ lies between 0 and 1 are of interest in this method as it indicates that the cluster has both minority and majority class examples. For this kind

of cluster, the majority class examples are removed if δ is equal to or greater that an empirically determined threshold value τ. Clearly, if the threshold τ is low more majority class examples would be removed as compared to when τ is high. This method creates a "vacuum" around the minority class examples in each cluster and thus helps the machine learning classifiers learn the decision boundary more efficiently.

Algorithm: Cluster-Based Under-Sampling

1: Let S be the original training set.
2: Use clustering to form clusters on S denoted by C_i where $1 < i < |C|$.
3: Find the degree of minority class dominance for all C_i by:

$$\delta_i = \frac{\text{Number of minority class examples in } C_i}{|C_i|}$$

4: For clusters which satisfy: $0 < \delta_i < 1$ and $\delta \geq \tau$ (where, $\tau = f(\delta)$ is an empirically determined threshold value for δ and is uniform over all the clusters), remove all the majority class examples and retain all the minority class example.

Table 3.1: Algorithm for Cluster-Based Under-Sampling

Figure 3.4 shows an illustration of ClusBUS on a synthetic dataset. The imbalanced and overlapping data is represented in the figure at top-left. Identification of overlapping regions in the data space is performed using clustering as shown in the top-right diagram. The majority class points are removed from the clusters for which $\delta > \tau$. Note that, in the bottom diagram of Figure 3.4, the majority class points have been removed from all the clusters in order to make the visual representation of the step explanatory. In the actual algorithm, removal of majority class points is done only on the basis of $\delta > \tau$.

The clustering approach in order to identify interesting clusters can be any conventional clustering algorithm. Although in our previous study [111] we used K-means clustering, in this experiment partitioning based clustering methods are avoided due to two reasons:

- partitioning-based clustering requires user intervention to specify the number of clusters that need to be constructed on the data
- partitioning-based clustering mostly forms *spherical* clusters only

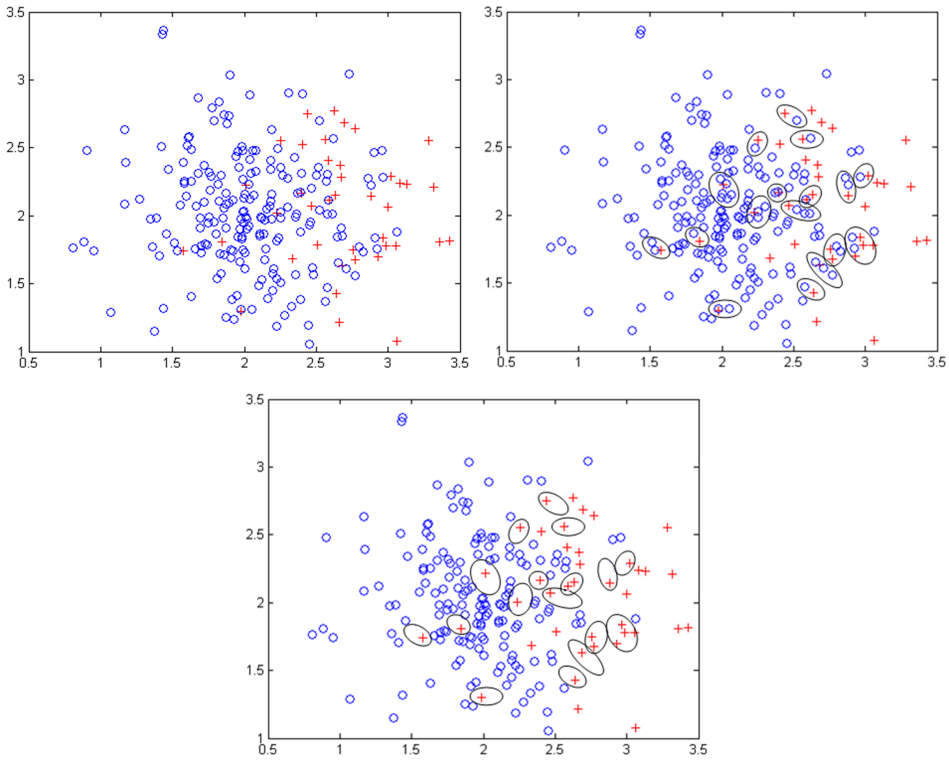

Figure 3.4: Schematic representation of ClusBUS Algorithm

In this study, a density based clustering methods, namely, Density-Based Spatial Clustering of Applications with Noise or DBSCAN is used. The rationale behind using DBSCAN is that there can be arbitrary shapes of clusters in the data that are not necessarily spherical (Gaussian). As there is no prior knowledge of the distribution of the data, the notion of density on which DBSCAN is based, is more meaningful than just specifying the number of clusters and forcing the data to be partitioned accordingly.

DBSCAN [112, 113] is a density based clustering technique that treats clusters as dense regions of objects in the data space that are separated by regions of low density, mostly representing noise. Any object that is not contained in any cluster is considered as noise. In other words, DBSCAN defines a cluster as a maximal set of density-connected points. The neighborhood of an object or data point is defined by a parameter ε. If the ε-neighborhood of a data point contains at least a minimum number of other points denoted by *MinPts*, then the point is called a core point, and the ε-neighboring points

are directly density-reachable from the core point. A point p is density-reachable from point q with respect to ε and *MinPts*, if there is a chain of objects $p_1, \ldots, p_n$, where $p_1 = q$ and $p_n = p$ such that p_{i+1} is directly density-reachable from p_i with respect to ε and *MinPts*. In a similar way, a point p is density-connected to q if there is a point o in the same data space such that both p and q are density-reachable from o with respect to ε and *MinPts*. The DBSCAN algorithm is summarized in Table 3.2.

Algorithm: DBSCAN
1: Search for clusters by checking ε-neighborhood of each point.
2: If ε-neighborhood of a point p contains more than *MinPts*, a new cluster with p as core point is created.
3: Iterative collection of directly density-reachable points from the core points.
4: Terminate when no new point can be added to any cluster.

Table 3.2: Algorithm for DBSCAN

The threshold τ, on the basis of which the majority class points are under-sampled from interesting clusters, is empirically determined by considering statistical properties of the degree of minority class dominance δ. Table 3.3 describes the algorithm to calculate τ.

Algorithm: Find τ
1: Consider all δ such that $0 < \delta < 1$.
2: Find min and max δ.
3: On the basis of an empirical test, select a value of δ in between min and median as τ.

Table 3.3: Algorithm to find τ

It should also be kept in mind that the prompting data has an absolute rarity imbalance problem, that is, the minority class instances are not only relatively less as compared to majority class, but also rare in absolute number. Therefore, this serves as a rationale for not employing sampling methods that involve discarding minority class examples.

3.5 Experiments and Results

For our experiments, learning algorithms are chosen from four widely accepted categories of learners, namely, decision trees, k-nearest neighbors, Bayesian learner and support vector machines. The specific classifiers used in this experiment are WEKA [114] implementations of the aforementioned algorithms namely: C4.5, IBk, Naive Bayes Classifier and SMO, respectively. Observing the results of the classifiers from four different categories would give the readers the opportunity to analyze the improvement that ClusBUS achieves.

In the domain of imbalanced class datasets, most of the experiments are performed by k-fold cross validation. This method of testing is far away from reality for most real-life problems, especially methods which involve preprocessing techniques such as sampling. Most of the times sampling tampers with the original data distribution, and cross validation forces the classifiers to train as well as test on the same tampered data distribution and thus resulting in overly optimistic results which are tangential to reality. In order to avoid this inappropriate evaluation technique, the current experimental setup trains the classifiers on 80% data and considers the rest for testing. Also, the degree of imbalance in the original dataset is maintained in training and testing examples.

As the proposed approach is a preprocessing technique, under-sampling to be more specific, that is performed on the data before it could be fed to the classifiers, the comparison is done with a well known over-sampling technique, known as SMOTE [53]. SMOTE uses a combination of both under and over sampling, but without data replication. Over-sampling is performed by taking each minority class sample and synthesizing a new sample by randomly choosing any or all (depending upon the desired size of the class) of its k minority class nearest neighbors. Generation of the synthetic sample is accomplished by first computing the difference between the feature vector (sample) under consideration and its nearest neighbor. Next, this difference is multiplied by a random number between 0 and 1. Finally, the product is added to the feature vector under consideration. As mentioned earlier, there are a number of downsides if the methods are evaluated with a cross validation technique. In case of SMOTE, as the training and testing were done on the same examples that are synthetically generated, the overfitting of the classifiers caused by an overwhelmed synthesis of artificial minority class examples, would never be detected. For the current experimentation,

SMOTE is used to boost the number of training examples of minority class to be the same as that of majority class examples. Any further boosting to generate more artificial training examples would make is unrealistic in real-life problem domains due to high computation and sometimes monetary cost associated with synthesizing new data points.

As mentioned in the ClusBUS algorithm, the threshold value of the degree of minority class dominance, on the basis of which interesting clusters are identified and the majority class examples are removed, is determined empirically. As mentioned in Table 3.3, τ is calculated by finding a value between min and median. This is done by varying the value of τ from median, 1st tercile, 1st quartile, 1st quintile, and so on, till 1st decile. In other words, the first x% of interesting clusters (where interesting clusters are increasingly ordered based on the r) is considered for under-sampling, where x is 50%, 33%, 25%, 20%, 16.67%, 14.28%, 12.5% 11.11% and 10%. TP Rate, FP Rate and AUC of C4.5 on the data after being pre-processed using ClusBUS are plotted in Figure 3.6. In the x axis, a number k indicates the first part of the data if it is divided into k equal parts.

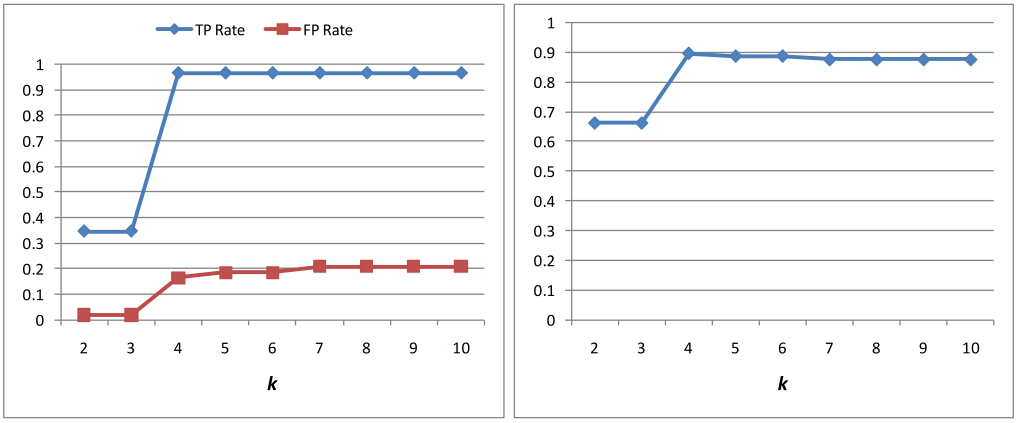

Figure 3.5: Comparison of performance measures: (left) TP and FP Rate, (right) AUC, with respect to different values of k

It can be seen that FP rate from $k = 3$ to $k = 4$. Also, AUC is fairly high at $k = 4$. Therefore, further experimentation with the rest of the classification algorithms is performed considering $k = 4$, that is $\tau = 0.25$.

From Figure 3.6, it can be clearly stated that ClusBUS performs significantly better than SMOTE. ClusBUS does not entail any costly data synthe-

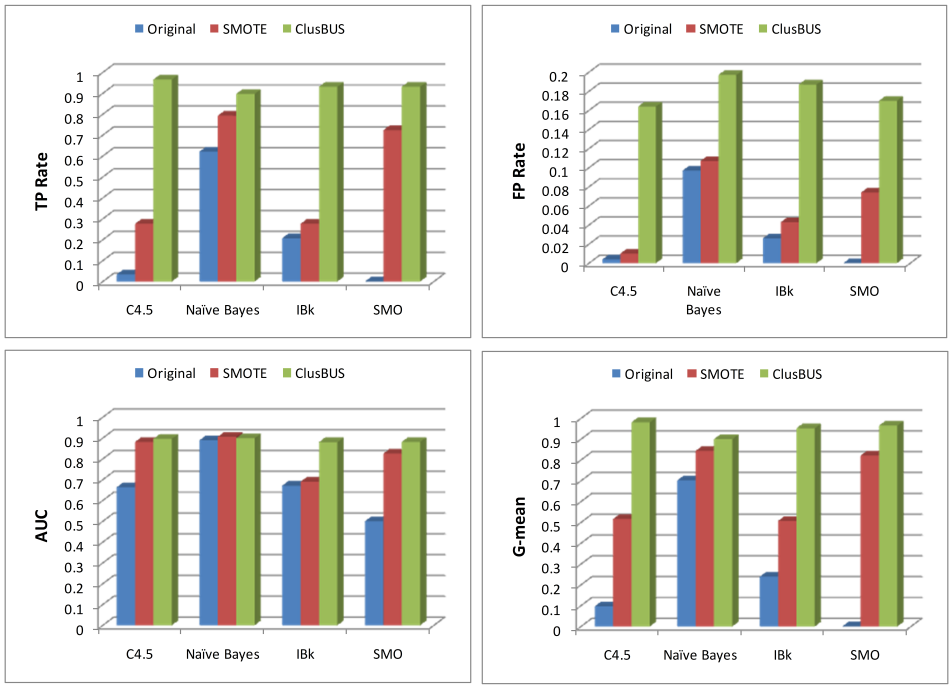

Figure 3.6: Comparison of performance measures: (top-left) TP Rate, (top-right) FP Rate, (bottom-left) AUC, and (bottom-right) G-means

sis technique like SMOTE, but performs under-sampling on majority class training examples that belong to the overlapping region in the data space. TP Rate and G-means has shown significant improvement over SMOTE, although the same trend is not followed for AUC. For C4.5 and Naive Bayes classifiers AUC is marginally better than that achieved by SMOTE. However, ClusBUS has caused increase in FP Rate as compared to the performance of the classifiers on original data and after applying SMOTE. But, this increase is not significant enough to demean the advantages and improvement of ClusBUS on other performance metrics.

3.6 Summary

This chapter proposes a novel preprocessing technique, ClusBUS, to deal with the class overlap problem in the presence of imbalanced class distribution in the dataset. A density-based clustering technique is used to find

interesting clusters that would be chosen for under-sampling. This problem occurs in many real-life problem domains like automated prompting task in smart environments, credit card fraud detection, network intrusion detection, etc. The effectiveness of the approach is validated by selecting smart environments as the target domain. The data of automated prompting task is therefore considered for experimentation. Experiments show that it is possible to achieve significant improvement over other well-known sampling techniques like SMOTE.

Chapter 4

Unsupervised Learning of Prompt Situations on Streaming Data

4.1 Introduction

Supervised approaches of classifying activity steps as *prompt* and *no-prompt* class labels as mentioned in the last two chapters perform quite well from a classification accuracy perspective. However, an automated prompting system should ideally work on streaming data. The supervised learning approach makes a strong assumption that the streaming data is labeled with activity and predefined sub-activity steps information in real time. This assumption has the following drawbacks which are difficult to overcome:

- It takes several hundred man-hours to annotate streaming sensor data with activity and sub-activity level information.
- Annotating sensor data stream is a tedious task and there is high probability for the annotations to be less accurate.
- Testing streaming data in real time requires some form of recognition model to recognize sub-activity steps, which is a hard problem.

Although, in the current work we do not completely get rid of the above issues, we do factor the issues down by using real-time recognition of activities only and not sub-activity steps. Given the start of an activity recognized

in real-time, we then take an unsupervised approach on streaming data to discover potential prompt situations. It should be noted that unlike the supervised approaches mention in Chapters 2 and 3, in the current approach we do not make any effort tp "recognize" sub-activity steps. Rather, every sensor event is tested with the likelihood of being a prompt situation.

Prompting at the level of daily activities has been addressed in the last decade to some extent. However, the problem gets tougher when prompts need to be issued at the level of sub-activities in real time. There are very few research work on addressing the problem of sub-activity level prompting. Most of the work either use video analysis [36, 115] or sensors modalities that can track every possible object in the environment [116, 117](for example, by using RFID tags). The current infrastructure that is being used for this study does not have either of these sensing platforms. We rely completely on motion, door and selected object sensors for our data collection. The proposed approach uses daily activity data performed by real human participants and generates statistical property that are capable of capturing potential prompt situations. It is assumed that activity information is available (more discussion in later sections) from a real-time activity recognition model [118]. Please note that this work does not address the problem of predicting the content of prompts along with prompt situations.

4.2 Related Work

A related work on prompting systems has already been given in Chapter 1. Therefore, this section focuses of reviewing prompting systems that aim at providing assistance with activity steps in real-time.

Hoey et al. [115] proposed a real-time vision-based system to assist dementia patients with hand washing task. This system combines a Bayesian sequential estimation framework for tracking hands and towel and a decision theoretic framework for computing policies of actions. The decision policies which dictate system actions are computed using Partially Observable Markov Decision Process or POMDP using a point-based approximate solution technique. The tracking and decision making systems are coupled using a heuristic method for temporally segmenting the input video stream based on the continuity of the belief state. This methodology performs a very good job with prompting at appropriate activity steps, however, it can cause major privacy concerns to the participants due to the use of video input. Moreover,

due to reliance of this technique on video analysis, it is not a very generalizable framework because major modifications need to be made to use similar strategies with other activities of daily living. Although privacy concern is a philosophical question, our group's focus is on environmental sensors with no input of audio or video. Also, in the current approach we intend to have the fundamental statistical measures uniform for all 6 ADLs under consideration.

On the other hand, other proposed approaches [116, 117, 119] use sensor platforms that can provide deep and precise insight into sub-activity steps information. For example, usage of RFID tags with every possible object in the environment is a very common strategy to gather one-to-one mapping between sensors and activity steps. Some of these approaches use complex plan recognition models for predicting the probability of a certain action for a state when activity is modeled as a stochastic process such as a Markov chain.

However, due to infrastructural limitations we are motivated to propose more efficient or at least equally effective solution to automated sub-activity level prompting problem. Therefore, if we have to structure the project goal in one line, it would be: *given the start of a specific activity in real-time, we propose an unsupervised approach to identify potential sub-activity level prompt situation in on streaming sensor data.* To the best of our knowledge, this problem with the given infrastructural limitations has not been addressed in the literature before.

4.3 Data Collection

As the problem that is currently being address in this chapter has a strong real-life implementation feasibility concern, we first start by describing how the experiments with human participants were conducted to collect sensor data in the smart home test bed.

We conducted experiments with 33 younger adults who committed scripted errors for 6 activities of daily living (ADLs). The experiments were conducted in the same smart home testbed as mentioned in Chapter 1. The participants performed each of the following ADLs four times, once with no errors (henceforth referred as *normal activity sequence*), twice with two different errors and once with both of the previous error (henceforth referred as *erroneous activity sequence*).

- Sweeping

- Medication
- Cooking
- Watering plants
- Hand washing
- Cleaning kitchen countertops

The ADLs have predefined sub-activity level steps identified by the psychologists. In a normal run of an ADL, the participant is supposed to perform the activities by roughly following the ordering of the predefined steps. On the other hand, while doing the rest of three erroneous runs for the activity, the participant is supposed to commit a scripted error (e.g. skipping an activity step or using an object that is not part of the specific activity) and continue with the rest of the activity until an audio prompt is issued by the experimenter who monitor the participant via cameras from a control room situated in the apartment. The ADLs and corresponding errors with which experiments were conducted are summarized in Tables 4.1, 4.2, 4.3. After performing activity labeling on the data, segmented activity data is extracted out of the database. Therefore, we have 33 normal sequences (from 33 participants) and 33×3 (3 types of errors) erroneous sequences for all 6 activities. Table 4.4 shows a sample normal sequence for *Cooking* activity. The normal activity sequences are treated as training data and erroneous sequences as test data in the proposed model.

Activity 1: Sweeping

Steps	**Error**
1. Participant retrieves broom from supply closet	Failed to retrieve and use dustpan and brush.
2. Participant retrieves duster from supply closet	
3. Participant retrieves dust pan and brush from closet	
4. Participant sweeps kitchen floor	
5. Participant uses dust pan and brush	
6. Participant dusts living room	
7. Participant dusts dining room	Failed to dust dining room.
8. Participant returns broom to supply closet	
9. Participant returns duster to supply closet	
10. Participants returns dust pan and brush to supply closet	

Activity 2: Medication

Steps	**Error**
1. Participant retrieves materials from cupboard "A"	Pick up wrong medicine
2. Participant reads instructions	
3. Participant fills dispenser with medication	
	Put medicine back in wrong cabinet

Table 4.1: Sub-activity steps and corresponding errors for *Sweeping* and *Medication*

Activity 3: Cooking (Noodles)

Steps	**Error**
1. Participant retrieves materials from cupboard "A"	
2. Participant fills measuring cup with water	
3. Participant boils water in microwave	Did not boil water in microwave
4. Participant pours water into cup of noodles	
5. Participant retrieves pitcher of water from refrigerator	Failed to retrieve pitcher and pour glass of water
6. Participant pours glass of water	
7. Participant returns pitcher of water	
8. Participant waits for water to simmer in cup of noodles	
9. Participant brings all items to dining room table	

Activity 4: Watering Plants

Steps	**Error**
1. Participant retrieves watering can from supply closet	Failed to retrieve or fill watering can
2. Participant fills watering can	
3. Participant waters plants (windowsill)	Did not water plants on windowsill
4. Participant waters plants (coffee table)	
5. Participant empties extra water into sink	
6. Participant returns watering can to supply closet	

Table 4.2: Sub-activity steps and corresponding errors for *Cooking* and *Watering Plants*

Activity 5: Hand Washing

Steps	**Error**
1. Participant moves to the kitchen	
2. Participant turns water faucet on	
3. Participant uses hand soap	Choose incorrect soap
4. Participant washes hands	
5. Participant turns water faucet off	Leave water running
6. Participant dries hands	

Activity 6: Cleaning Kitchen Countertops

Steps	**Error**
1. Participant enters kitchen and goes to the sink	
2. Participant pulls sink faucet handle up in the "on" position	
3. Participant wets sponge and puts dish soap on the sponge	Not using soap
4. Participant washes counter with soapy sponge	Washing another area instead of countertop
5. Participant rinses off dirty sponge with water from the sink faucet	
6. Participant turns off the faucet by pressing the handle down	
7. Participant puts sponge back on drying rack	
8. Participant dries wet countertops with a dishtowel	

Table 4.3: Sub-activity steps and corresponding errors for *Hand Washing* and *Kitchen Countertop Cleaning*

4.4 Failure Model

We discuss the failure model of the problem first as this gives a better understanding of the data and approach that would not work.

The initial idea while proposing a solution to the current problem was to take a combination of pattern mining and probabilistic graphical model. During training, the idea was to discover common pattern (called, *activitons*) for an activity across the *normal* sequences of the activity data collected from 33 participants. A classifier model is built on these activitons for the purpose of real-time labeling of sensor events on *erroneous* sequences during test. Also, activitons are treated as states in a probabilistic graphical model. While performing the test, when sensor data streams in real-time, classification of activitons would be performed on every sensor event. On estimation of the current state and thus its transition from the previous one, a likelihood value is calculated for the prompt. However, this approach assumes that the common patterns discovered on the training data are useful and that the real-time classification on the activitons are also perfect. The activity discovery approach proposed by Cook et al. [120] was used to discover activitons and the real-time activity recognition algorithm proposed by Krishnan et al. [118] was used to label sensor events with activiton labels in real time. The very first experiment performed on this model was done on the normal instances of *Sweeping* activity. There were over 450 patterns (activitons) that were discovered out of 33 sequences for sweeping activity. The real-time recognition accuracy on the discovered patterns was as low as 16%. This is because we were trying to use pattern mining techniques which are used mainly for data compression to mine common patterns across activity sequences. Moreover, each of the activity sequences are not sufficient in length to perform any kind of pattern mining on them. Therefore, it was concluded that pattern mining and hence real-time pattern recognition cannot be used with the current dataset.

4.5 Proposed Approach

We start the description of the proposed model by explaining the real-time activity recognition algorithm that is assumed to correctly predict the start of an activity while testing an erroneous sequence. Once the start of an activity is confirmed, the proposed model could be used to identify prompt

situations.

Activity Recognition on Streaming Sensor Data Krishnan et al. [118] propose a sliding window-based approach to perform activity recognition on streaming sensor data as and when they are recorded in a smart home. As different activities can be best characterized by different window lengths of sensor events, time decay and mutual information-based weighting of sensor events within a window are considered. Also, additional conceptual information in the form of previous activity and activity of the previous window are considered as features of the sensor window. The proposed approach uses SVM as the classification algorithm and reports to achieve recognition accuracy as high as 78% for some datasets when fixed length sensor windows with time-based weighting of sensor events is performed.

Modeling Activity Errors The six ADLs that have been considered to perform experiments for data collection has two different errors for each activity. These scripted errors mentioned in Tables 4.1, 4.2, 4.3 can be broadly classified into two categories:

- **Abnormal Occurrence:** Triggering of sensors that were not seen in the training data for a specific activity.
- **Delayed Occurrence:** Delay or infinite delay of certain sensor triggers that are present in the normal sequences of a specific activity.

Examples of abnormal occurrence category are: *picking up wrong medicine bottle* for the *Medication* activity and *choosing incorrect soap* for *Handwashing* activity. Examples of delayed occurrence error category are: *failing to dust dining room* for the *Sweeping* activity and *not boiling water in microwave* for the *Cooking* activity. Ideally, a prompt situation is when an activity sequence encounters either of these errors. Therefore, the two errors are modeled differently and use different measures during test to predict prompt situations.

The first model to address abnormal occurrence of sensor triggers uses the likelihood of occurrence of every sensor across the whole population and for a specific participant. While the probability of occurrence of a sensor across the population gives the idea of how often participants trigger it, the probability of occurrence for a particular participant gives the idea of how the sensor was

triggered for that specific participant. This is because, it is understood that all participants would not have the exact same *sensor triggering signature*. Thus, when training the model on normal activity sequences the following two parameters are evaluated from the training data:

- ***Support:*** Support of a sensor s_i is the probability of the sensor being present in the population. It is given as follows:

$$Support(s_i) = \frac{\#\text{ participants triggering sensor } s_i}{\text{Total }\#\text{ participants}} \tag{4.1}$$

- ***Membership:*** Membership probability or simply membership is the probability of sensor s_i being triggered for participant p_j. It is given by the following equation:

$$Membership(s_i, p_j) = \frac{\#\text{ times sensor } s_i \text{ is triggered by participant } p_j}{\text{Total }\#\text{ sensor triggering for participant } p_j} \tag{4.2}$$

The delayed occurrence error category is modeled by forming a Gaussian distribution of the time elapsed from the beginning of the activity to the n^{th} occurrence of every sensor with respect to the current sensor event. In the training data, for every n^{th} occurrence of a sensor s_i the time spent from the beginning of the activity to the current sensor event is stored as samples that are together modeled as a Gaussian distribution denoted by $\mathcal{N}_{n,s_i}(\mu, \sigma^2)$. The means ($\mu_{n,s_i}$) and variances ($\sigma^2_{n,s_i}$) for n-th occurrences of s_i are calculated from the training data and stored to be used during tests.

It should be noted that only the *on* events of the sensors are used to calculate the statistical measures mentioned above.

Predicting Prompt Situation While performing tests on erroneous sequences of activities, at every valid *on* sensor event the statistical measures calculated during training are used to derive parameters which are indicative of a potential prompt situation. The parameter corresponding to abnormal occurrence errors is a combination of *support* and *membership*, denoted by ρ, is expressed as follows:

$$\rho = (1 - Support_{s_i}) + \left(1 - Membership_{s_i,p_j}\right) \tag{4.3}$$

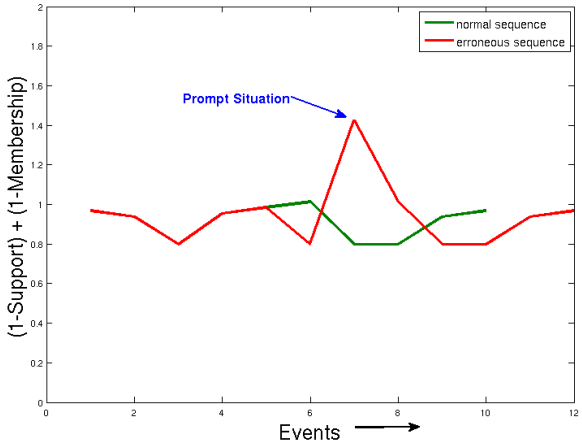

Figure 4.1: Comparison of *normal* and *erroneous* sequences for *Medication*

ρ is directly proportional to the likelihood of a prompt situation. It can have minimum and maximum values of 0 and 2 respectively, as Support and Membership have 0 and 1 as the minimum and maximum values respectively. In Figure 4.1, we perform the comparison between a normal and an erroneous sequence for Medication activity performed by a participant. For this example, the error was to *pick up the wrong medicine* which was tracked by a specific object sensor. The figure clearly indicates that ρ is definitely a valid indicator of potential prompt situation.

For delayed occurrence error category, the probability distribution function of Gaussian distribution (given in the following equation) is used to calculate the probability of current elapsed time for the current n^{th} occurrence of sensor s_i.

$$f_{n,s_i}\left(x;\mu_{n,s_i},\sigma^2_{n,s_i}\right) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu_{n,s_i}}{\sigma_{n,s_i}}\right)^2} \tag{4.4}$$

f_{n,s_i} is inversely proportional to the likelihood of a prompt situation. Although, initial analysis on the values of f_{n,s_i} indicate that it is a potentially good measure for the current problem, we need to perform further analysis of this measure to validate our claim.

4.6 Evaluation Criteria

Once we come up with suitable statistical measures to identify potential prompt situations, we would use the erroneous activity sequences to validate the performance of the proposed model. However, there is an underlying problem with how the erroneous activity data was collected. According tot the experimentation protocol, the participant commits the scripted error and continues with the rest of the activity until the experimenter issues an audio prompt. However, as the experimenter had a prior knowledge of issuing a prompt, she issued it right after the error was committed. Hence, the participant rectified the error immediately. From the perspective of model validation, this kind of erroneous activity sequence is difficult to validate because the data does not "allow" the f_{n,s_i} values of sensors to fall beyond a certain level where a derivation in probability can actually be observed.

However, this kind of situation would not occur in a realistic setting. That is, if an older adult commits an error, she would be in the state of error until she herself realized or a caregiver tells her that an error was committed.

Assuming that we either simulate erroneous sequences for the purpose of validation or collect new data that would be suitable for our purpose, we can evaluate the effectiveness of the model on the basis of the following criteria:

- How many sensor events before or after the actual prompt event was the prediction made?
- How many prompt situations were predicted when the ground truth says that only one prompt was actually issued?

4.7 Plan for Further Work

Further work to solve the prediction of prompt situation problem involves contribution in terms of methodology as well as collecting suitable data. We plan to have a formalized method of combining f_{n,s_i} values for all the sensors together which would be indicative of a prompt situation. In addition to modeling elapsed times of all sensors from the beginning of the activity, we also want to model the sensor triggering frequencies for every n-th occurrence of each sensor.

From the perspective of using validation data, we would start with creating simulation data from the current erroneous sequences by removing the

activity corrections made by the participants that are present in the data currently. If possible, we would also conduct additional experiment to collect tailored data for our purpose.

2011-03-15 14:58:26.545253	M016	ON
2011-03-15 14:58:27.144553	M002	OFF
2011-03-15 14:58:27.852217	M017	ON
2011-03-15 14:58:27.889078	D007	OPEN
2011-03-15 14:58:27.925423	M009	OFF
2011-03-15 14:58:28.690177	M008	OFF
2011-03-15 14:58:28.720142	M016	OFF
2011-03-15 14:58:29.069909	M007	OFF
2011-03-15 14:58:29.912595	M015	OFF
2011-03-15 14:58:30.548572	I001	ABSENT
2011-03-15 14:58:31.581525	I002	ABSENT
2011-03-15 14:58:31.965936	M017	OFF
2011-03-15 14:58:35.841897	SS007	MOVED
2011-03-15 14:58:37.008892	D007	CLOSE
2011-03-15 14:58:39.822391	M052	ON
2011-03-15 14:58:41.950923	M017	ON
2011-03-15 14:58:45.040906	M052	OFF
2011-03-15 14:58:45.52218	A002	0.0586337
2011-03-15 14:58:45.613679	SS007	STILL
2011-03-15 14:58:46.348079	M052	ON
2011-03-15 14:58:47.790778	D010	OPEN
2011-03-15 14:58:48.635277	M052	OFF
2011-03-15 14:58:50.05478	M017	OFF
2011-03-15 14:58:50.48101	A002	0.0279692
2011-03-15 14:58:50.903867	D010	CLOSE
2011-03-15 14:58:59.344789	M017	ON
2011-03-15 14:59:00.586945	M017	OFF
2011-03-15 14:59:00.72869	SS008	STILL
2011-03-15 14:59:01.893765	P001	2932
2011-03-15 14:59:01.935531	P001	3244
2011-03-15 14:59:01.949772	M016	ON
2011-03-15 14:59:03.053178	M017	ON
2011-03-15 14:59:04.119287	M016	OFF
2011-03-15 14:59:07.886206	P001	2339
2011-03-15 14:59:09.847279	M017	OFF
2011-03-15 14:59:13.889363	P001	1972
2011-03-15 14:59:14.997723	M052	ON
2011-03-15 14:59:20.24564	M017	ON
2011-03-15 14:59:22.361049	M017	OFF
2011-03-15 14:59:25.798872	M052	OFF
2011-03-15 14:59:26.174757	M017	ON
2011-03-15 14:59:26.554584	M052	ON
2011-03-15 14:59:27.933041	M017	OFF
2011-03-15 14:59:28.750661	M016	ON
2011-03-15 14:59:29.128173	M017	ON
2011-03-15 14:59:30.330534	D010	OPEN
2011-03-15 14:59:31.164393	M016	OFF
2011-03-15 14:59:31.549982	M017	OFF
2011-03-15 14:59:31.97821	P001	1997
2011-03-15 14:59:32.522972	M052	OFF
2011-03-15 14:59:33.911047	SS008	MOVED
2011-03-15 14:59:37.898572	P001	1969
2011-03-15 14:59:38.08094	D010	CLOSE
2011-03-15 14:59:39.306538	M017	ON
2011-03-15 14:59:39.924028	A001	2.86332
2011-03-15 14:59:42.385977	A001	2.81981
2011-03-15 14:59:43.354757	M052	ON
2011-03-15 14:59:45.299796	M017	OFF
2011-03-15 14:59:46.453738	SS007	MOVED
2011-03-15 14:59:49.385819	M052	OFF
2011-03-15 14:59:53.194241	M017	ON
2011-03-15 14:59:53.209204	M016	ON
2011-03-15 14:59:53.811462	M015	ON
2011-03-15 14:59:54.19512	M014	ON
2011-03-15 14:59:54.715708	A001	2.84536
2011-03-15 14:59:55.589932	M017	OFF
2011-03-15 14:59:55.609773	M013	ON
2011-03-15 14:59:56.196043	M016	OFF

Table 4.4: Sample *normal* activity sequence for *Cooking*

Chapter 5

Ambient Intelligence on Mobile Device

5.1 Introduction

Human activity recognition is an important area of machine learning research because of its real-world applications. Automated activity recognition reduces the necessity for humans to oversee difficulties individuals (especially older adults) might have performing activities, such as falling, when they try to get out of bed. Activity recognition can also be used in conjunction with pattern recognition to determine changes in a subject's routine. For these reasons the technology has many potential uses in healthcare and eldercare.

Two methods of collecting data for performing activity recognition have been extensively researched. The first method relies upon environmental sensors to track features such as motion, location, and object interaction. Alternatively, the second method uses a network of sensors attached to the human body to track the acceleration of specific limbs as well as the body as a whole. Both of these methods have demonstrated impressive results in constrained laboratory settings.

A major hurdle in implementing these systems outside of trials is how unnatural the sensors are. Environmental sensors are generally bulky and costly. They also must be wired or have their batteries maintained. Both cases involve a large investment into setting up and maintaining the system. Body sensors require daily effort from the user to wear and maintain them or else they are useless for collecting data. Additionally they are bulky and

require batteries, two factors that reduce the likelihood of constant use by the user.

This paper describes the effectiveness of using the accelerometer and gyroscope of a smart phone as a more natural alternative to a combination of body sensors. While many older adults are currently not likely to carry smart phones, younger people are increasingly likely to carry a mobile phone on them. These people represent the next generations of elders. So while a phone may still be an unnatural accessory for older people it is becoming increasingly less so. Using a phone as the primary device for data collection increases the likelihood of data coverage and represents a minimal cost and maintenance commitment to the user.

Current generation smart phones are equipped with a variety of sensors such as GPS sensors, microphones, image sensors (camera), light sensors, proximity sensors, inertial sensors (accelerometers and gyroscopes), and direction sensors (compass). The small form factor of the smart phones coupled with its ubiquity and the substantial computing power makes them an effective tool for understanding the current state of the smart phone user. In this paper, we explore the use of inertial sensors (accelerometers and gyroscopes) in the smart phone to identify the activity that a user is performing by mining the sensor data.

We have chosen an Android smart phone as the platform for the project for multiple reasons - the Android operating system is open-source and easily programmable and more importantly, its dominance in the smart phone market. A Samsung Captivate smart phone is used as the device running Android 2.1.

Activity recognition from inertial sensor data is not a new problem. There have been many approaches proposed in the literature as discussed in Section 5.2. Our approach is unique from the existing methods in many ways. First and foremost, no additional body sensor is used on the subject. Secondly, the subject's body location where the phone should be placed and the orientation of the phone are not predetermined. In addition, we perform activity recognition on complex activities such as, cooking, cleaning, etc. Unlike other works, where environmental sensors are used to recognize complex activities, we make an endeavor to use accelerometer and gyroscope data to achieve the same goal. To our knowledge, there has been no previous work in this direction.

5.2 Related Work

Advances in ubiquitous and pervasive computing have resulted in the development of a number of sensing technologies for capturing information related to human physical activities. The different approaches to activity recognition can be categorized based on the underlying sensing mechanism - network of environmental sensors of body area networks.

Environmental sensor-based activity recognition has received significant focus in the recent years. This is a promising approach for recognizing activities that are not easily distinguishable by body movement alone. Motion sensors, door contact sensors, object sensors RFID tags and video cameras are some of the most commonly used environmental sensors for gathering activity related information [121].

Wearable accelerometers have proved to be effective sensors for human activity recognition. Some of the earliest work on wearable sensor based activity recognition used multiple accelerometers placed on different parts of the body. Bao and Intille [122] used a series of five biaxial accelerometers placed on the left bicep, right wrist, left quadriceps, right ankle, and right hip for recognizing twenty different activities ranging from walking to folding laundry to strength training. Ravi et. al. [123] use a single accelerometer mounted onto the pelvic region of subjects to collect data on eight activities: Standing, walking, running, climbing up stairs, climbing down stairs, sit-ups, vacuuming, and brushing teeth. The peak accuracy was achieved using plurality voting which selects the most common prediction from five classifiers: decision tables, decision trees, k-nearest neighbors, SVM, and naÃ¯ve Bayes. Tapia et. al. [124] collected data from five accelerometers placed on various body locations for implementing a real-time system to recognize thirty gymnasium activities. Krishnan et. al. [125] collected data using two accelerometers for recognizing locomotion activities in real-time using adaptive boosting on decision stumps.

Other research has explored the use of multiple kinds of on-body sensors for activity recognition. Maurer et. al. [1] use accelerometers, temperature sensors and microphones for recognizing human locomotion. Their research also analyzed multiple time domain feature sets for activity recognition. Lee and Mase [126] proposed a system for recognizing activities using information about the user's location and inertial sensors such as accelerometers and gyroscopes. Subramanya et. al. [127] also proposed a similar approach involving accelerometers, microphones, light sensors, thermometers, baro-

metric pressure sensors and GPS sensors.

There have been a few studies similar to the one proposed in the paper that use commercially available mobile devices to collect data for activity recognition. Kwapisz et. al. [128] use an Android-based smart phone for recognizing very simple activities such as walk, jog, climb up and down the stairs, sit and stand. Yang VI developed an activity recognition system using the Nokia N95 cell phone for distinguishing between different locomotion. Brezmes et. al. [129] proposes a subject dependent real time activity recognition system again using the Nokia N95 smart phone. Hache et. al. VI use an accelerometer integrated with a blackberry Bold 9000 platform for detecting changes in the state of the subject caused by starting/stopping and postural changes in activities. Khan et. al. [130] use kernel discriminant analysis for recognizing very simple activities such as walking, up and down the stairs, running and resting on data collected from Samsung Omnia. Zhang et. al. [131] use an HTC smart phone for recognizing again simple activities using a support vector machines.

Mobile devices offer a number of advantages including unobtrusiveness of the system and not requiring any additional equipment for data collection or computing that make them an attractive platform for activity recognition. We build on these approaches and extend them to test the ability of mobile sensing and computing platforms for recognizing simple and complex activities such as watering plants and sweeping.

Simple activities, such as walking, can be represented as a single repeated action: taking a step forward; whereas complex activities, such as taking medication, involve multiple actions: opening a cupboard, taking out pills, swallowing, and returning the remaining pills. Other complex activities may involve simultaneous or overlapping actions. These traits decrease the ability to reduce these actions down to discrete features.

5.3 Design

In this section we describe the design of our experiments for performing activity recognition from smart phone. We begin this with a description of the data collection process and the set of activities considered for this work. We then describe the features that were extracted from the sensor data and finally discuss the machine learning algorithms that were used for performing the recognition task.

5.3.1 Data Collection

The data collection was done by performing experiments on ten undergraduate students who participated in National Science Foundation's REU program at Washington State University. Subjects wore an Android 2.1 operating system-based Samsung Captivate smart phone that contained a tri-axial accelerometer and gyroscope. The location and orientation of the phone was not standardized and was left to the convenience of the subject. However, orientation information was taken into consideration while making comparative study in Section 5.4. Each subject carried the phone while performing the different activities. We created an application that stored the sensor data while the subject performed the activities. The application permitted us to control the sensor type from which the data was collected along with the sampling rate of the sensor.

Subjects controlled the data that was being collected through the application (Figure 5.1) we created that executed on the phone. The application allows the subjects to input the activity that they are about to perform along with the ability to start and stop recording the sensor data. The application also allows the subjects to input the location they were wearing the phone. Though, we collected this information, we did not use it the current study.

Figure 5.1: Application used for activity data collection

5.3.2 Activities

Activities were divided into two categories: simple and complex. Simple activities consist of a single repeated action whereas complex activities are the compilation of a series of multiple actions. Subjects performed simple activities in variable amounts and in various environments; the action, location, and length of performance was not controlled. The simple activities included *biking*, *climbing stairs*, *driving*, *lying*, *running*, *sitting*, *standing* and *walking*. In addition, we also wanted to test the possibility of detecting the smart phone not being worn by the subject. We call this activity as *phone not on person*. While some of the simple activities also feature on the list of activities of other researchers, our list contains additional activities such as *driving*, that not has been studied before.

Every subject also performed a set of complex activities wearing the smart phone. The subjects repeated execution of these complex activities four times. The complex activities had definite starting and finishing points and lasted until the subject completed them. The complex activities consisted of:

- ***Cleaning*** Subject wiped down the kitchen counter top and sink.
- ***Cooking*** Subject simulated cooking by heating a bowl of water in the microwave and pouring a glass of water from a pitcher in the fridge.
- ***Medication*** Subject retrieved pills from the cupboard and sorted out a week's worth of doses.
- ***Sweeping*** Subject swept the kitchen area.
- ***Handwashing*** Subject washed hands using the soap at the kitchen sink.
- ***Watering Plants*** Subject filled a watering can and watered three plants in two rooms.

5.3.3 Feature Extraction

Raw data is collected as a series of instances containing a timestamp, three values corresponding to acceleration along the x-axis, y-axis, and z-axis, and a second set of three orientation values representing azimuth, pitch, and

roll. Rather than a set sampling rate, the accelerometer in this Android phone triggers an event whenever the accelerometer values change. The rate of events can be set to one of four thresholds: fastest, game, normal, and UI, with fastest being the fastest sampling rate and UI being the slowest. The phone used for this experiment was set to fastest. The sampling rate varies because of this but can reach a maximum of 80 Hz. The three axes of acceleration are dependent upon the orientation of the phone. The x-axis runs parallel to the width of the phone, the y-axis runs the length of the phone, and the z-axis runs perpendicular to the face of the phone, as shown in Figure 5.2.

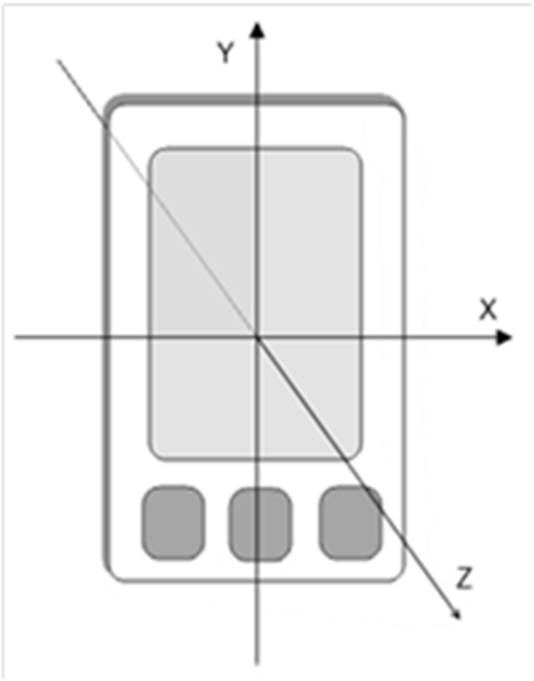

Figure 5.2: Axes of acceleration relative to the phone

Raw data is processed to normalize the acceleration axes so that the x-axis, y-axis, and z-axis run north and south, east and west, and up and down respectively. Samples of raw tri-axial accelerometer data for simple and complex activities can be found in AppendixC respectively. The orientation data is left as is. From the raw accelerometer data for simple activities, it could be clearly seen that every simple activity has a distinct pattern of acceleration on three different axes. However, intuitively such distinctive pattern is

difficult to be extracted from the raw data of the complex activities. This poses a difficulty in complex activity recognition which would be explained in Section 5.4.

Standard classifiers do not work well on the raw sensor data. It is essential to transform the raw data into a representation that captures the salient characteristics of the raw data. This is typically performed by breaking the continuous data into windows of certain duration. In this work we experimented with one, two, four, eight, twelve and sixteen seconds time windows. Windows always overlapped by one half of the window length, e.g., a four second window slides over two seconds at a time. Thus, each window is a single instance, but any given data point contributes to two instances. This method has been shown to be effective in earlier work using accelerometer data [122]. We then extracted a number of features (as listed in Table 5.1) to encode each window. Thus each window was represented as a feature vector of length 30. These features were then used to train the classifiers.

Feature	**Accelerometer**	**Orientation**	**Description**
Mean	X, Y, Z	Azimuth, Pitch, Roll	Average acceleration values along the three axes and average orientation along the three directions
Min	X, Y, Z	Azimuth, Pitch, Roll	The minimum acceleration and orientation value within the window along three axis of acceleration and directions of orientation
Max	X, Y, Z	Azimuth, Pitch, Roll	The maximum acceleration and orientation value within the window along the three axis of acceleration and directions of orientation
Standard Deviation	X, Y, Z	Azimuth, Pitch, Roll	The standard deviation in the acceleration and orientation values within the window
Zero-Cross	X, Y, Z		The number of zero crossings for the three axis of acceleration
Correlation	X/Y, X/Z, Y/Z		The pairwise correlation between the three axes of acceleration

Table 5.1: Description of the features extracted from the raw data

5.3.4 Classification

The WEKA machine learning toolkit [114] was used to test classifiers using the features extracted from the raw data set. Six different classifiers were tested: Multi-layer Perceptron, Naive Bayes, Bayesian network, Decision Table, Best-First Tree, and K-star. The accuracy of the classifiers was tested using ten-fold cross-validation with. Data collected from all the subjects was pooled together. The train and test data for each fold was randomly drawn from this pool, ensuring no overlap between the train and test sets of each fold. We used this approach as it is a good estimator for the generalized performance of the different classifiers. We used the default parameters associated with each of the classifier.

5.4 Results

We categorized the activities into three groups: simple, complex and combined to study the ability of the different classifiers to recognize them separately. The classification accuracies obtained from the different classifiers are summarized in Figure 5.3. It can be observed that the classification accuracies for simple activities remain consistently above 90% except for Naive Bayes. However adding complex activities to the set reduced the performance of all the classifiers uniformly. The best accuracy noted for complex activities was 50% (guessing the class by chance stands at 17%). The best accuracy was obtained with the Multi-Layer Perceptron.

The confusion matrix for the combined set of activities shows how heavily the misclassification leans towards complex activities. This is not surprising alone but does show the resilience of simple activities against being misclassified when more activities are added. To gain further insights into the performance on these complex activities, we computed additional measures such as precision, recall and F score that is summarized in Table 5.2. When looking only at complex activities in Table 5.2, cooking, medication, and sweeping were all (correctly and incorrectly) classified far more often than cleaning, washing hands, or watering plants. The former activities also involved little movement in the experiment whiles the latter (with the exception of washing hands) contained much more movement from one area to another.

The complex activity set was also tested without using a sliding window for feature extraction. Instead, an entire activity acted as a single instance.

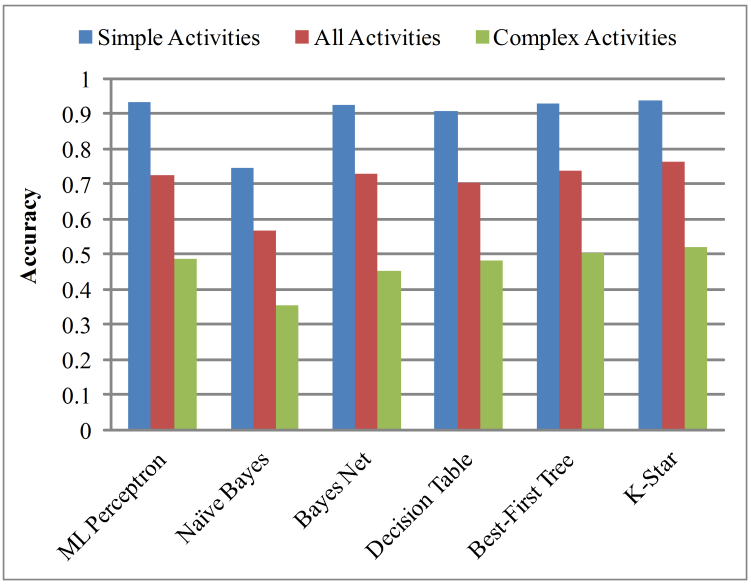

Figure 5.3: Performance of Different Classifiers

Class	TP Rate	FP Rate	Precision	Recall	F-Measure	AUC
Cleaning Kitchen	0.182	0.030	0.418	0.182	0.254	0.737
Cooking	0.536	0.189	0.453	0.536	0.491	0.766
Medication	0.673	0.136	0.614	0.673	0.642	0.828
Sweeping	0.600	0.235	0.489	0.600	0.539	0.761
Washing Hands	0.153	0.016	0.390	0.153	0.219	0.693
Watering Plants	0.318	0.036	0.474	0.318	0.381	0.735
Weighted Average	0.506	0.147	0.496	0.506	0.489	0.769

Table 5.2: Activity accuracies for complex activities (Multilayer Perceptron, 1-second window

This greatly improved performance from 52% to 78% accuracy. It is worth noting that the amount of data used to train and test the classifier was drastically reduced when the data is treated this way to forty instances per activity.

Our next experiment studied the effect of different window lengths on the ability of the classifiers to recognize the activities. We used K-star as the classifier as it resulted in the best performance in the previous experiment. The results of this experiment are summarized Figure 5.5. It can be noted that the overall classification accuracy for simple activities remains above 90% for each of the different window lengths. This trend did not continue

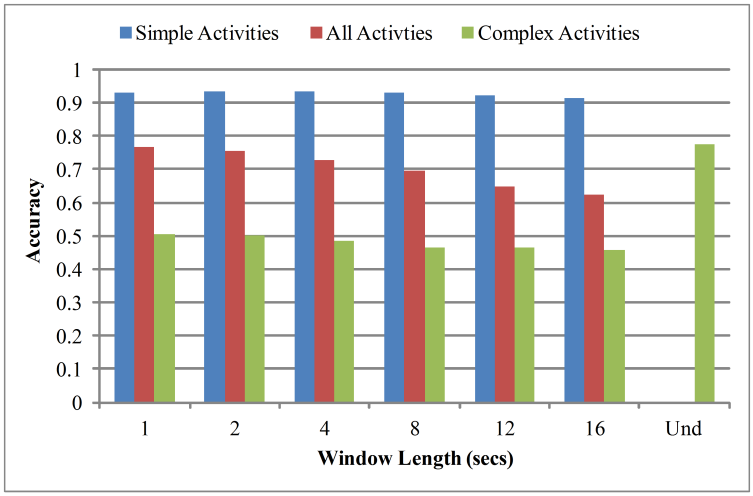

Figure 5.4: Classification accuracies for K-Star with different window lengths. Und corresponds to the scenario where train models for recognizing complex activities without using the sliding window protocol

when complex activities were added to the set. Shorter window frames performed significantly better than longer ones. This result is surprising as a shorter window is less likely to account for more than a single one of the actions involved in a complex activity.

Figure 5.6 shows the results of classifying each activity set with and without orientation features extracted from gyroscope data. Previous studies have generally relied solely on acceleration data and did not take the orientation of the sensors into account after beginning an activity. In this study, orientation data represented an average of a 10-12% increase in accuracy over pure acceleration data. Given the format of this experiment it is likely that this is in part due to the fact that the starting position and orientation of the phone was not standardized. Features extracted from orientation helped overcome this deficiency.

This approach for recognizing activities does have some limitations. Clearly, the methodology is feasible for recognizing simple activities. However, it fares poorly in the context of complex activities. The process was made possible because the subjects input the start and stop times of the activities they performed. A fully automated recognition system would need some way of determining the start and end of an activity rather than relying on the user.

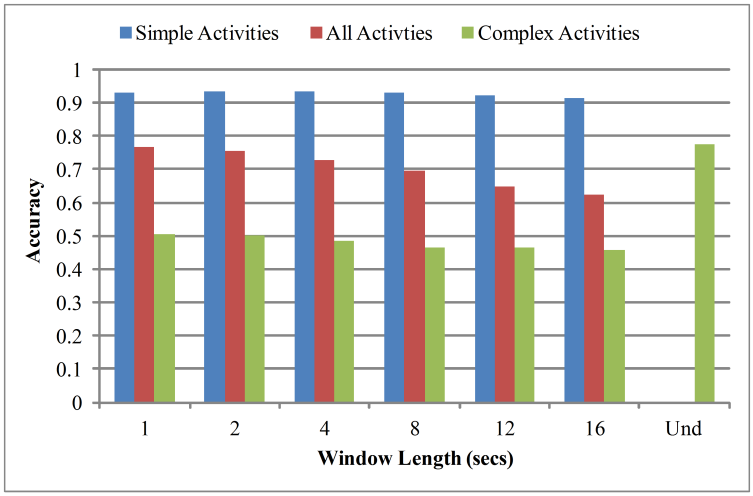

Figure 5.5: Classification accuracies for K-Star with different window lengths. Und corresponds to the scenario where train models for recognizing complex activities without using the sliding window protocol

Additionally, unlike a sliding window which can do recognition in pseudo-real-time, this method requires an activity to be completed before it can be recognized. The results do however; represent the potential for improving the recognition system.

5.5 Summary

Simple activities can be recognized with very high accuracy using only a single smart phone carried naturally. Performance was over 93% using a Multi-layer Perceptron and a two second time window. The length of the window had very little effect on results for simple activities which implies that it can be reduced for recognizing short activities or extended as needed. Activity sets that included complex activities did not perform as well but still achieved over 50% accuracy. Simple activities retained their high classification accuracy even when paired with complex activities.

The results for simple activities are at par with previous work on body sensors [132]. This shows a lot of promise for using mobile phones as an alternative to dedicated accelerometers. The recognition of complex activities was also similar to that of the less recognizable activities in [133] While 50%

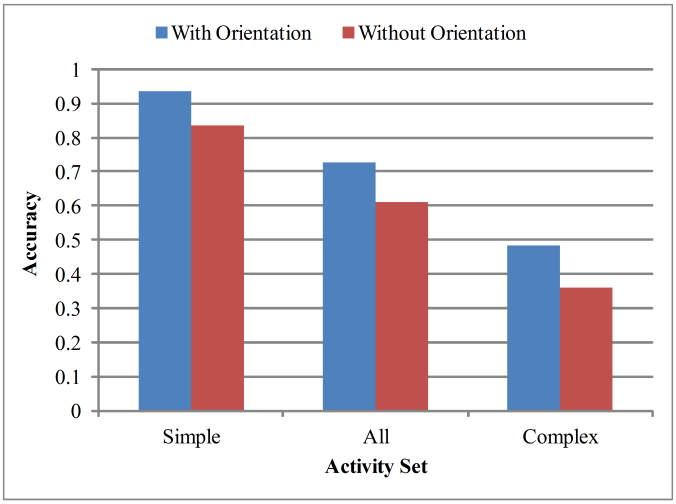

Figure 5.6: Accuracy of K-Star with and without using orientation information from gyroscope data

accuracy is not high enough for many real-world uses of activity recognition, it does show that a phone could be effective as part of a data collection system for recognizing complex activities even if it cannot function as a standalone system.

Appendix A

Publications

Book Chapters

- B. Das, N. C. Krishnan, D. J. Cook, "Handling Imbalanced and Overlapping Classes in Smart Environments Prompting Dataset", Springer Book on Data Mining for Services, 2012. (To appear)
- B. Das, N. C. Krishnan, D. J. Cook, "Automated Activity Interventions to Assist with Activities of Daily Living", IOS Press Book on Agent-Based Approaches to Ambient Intelligence, 2012.

Journal Articles

- B. Das, N. C. Krishnan, D. J. Cook, "RACOG and wRACOG: Two Gibbs Sampling-Based Oversampling Techniques", Journal of Machine Learning Research, 2012. (Under review)
- A. M. Seelye, M. Schmitter-Edgecombe, B. Das, D. J. Cook, "Application of Cognitive Rehabilitation Theory to the Development of Smart Prompting Technologies", IEEE Reviews on Biomedical Engineering, 2012.
- B. Das, D. J. Cook, M. Schmitter-Edgecombe, A. M. Seelye, "PUCK: An Automated Prompting System for Smart Environments", Journal of Personal and Ubiquitous Computing, 2012.

Conferences

- S. Dernbach, B. Das, N. C. Krishnan, B. L. Thomas, D. J. Cook, "Simple and Complex Acitivity Recognition Through Smart Phones", International Conference on Intelligent Environments (IE), 2012.

- B. Das, C. Chen, A. M. Seelye, D. J. Cook, "An Automated Prompting System for Smart Environments", International Conference on Smart Homes and Health Telematics (ICOST), 2011.

- E. Nazerfard, B. Das, D. J. Cook, L. B. Holder, "Conditional Random Fields for Activity Recognition in Smart Environments", International Symposium on Human Informatics (SIGHIT), 2010.

- C. Chen, B. Das, D. J. Cook, "A Data Mining Framework for Activity Recognition in Smart Environment", International Conference on Intelligent Environments (IE), 2010.

Workshops and Demos

- B. Das, B. L. Thomas, A.M. Seelye, D. J. Cook, L. B. Holder, M. Schmitter-Edgecombe, "Context-Aware Prompting From Your Smart Phone", Consumer Communication and Networking Conference Demonstration (CCNC), 2012.

- B. Das, A. M. Seelye, B. L. Thomas, D.J. Cook, L. B. Holder, M. Schmitter-Edgecombe, "Using Smart Phones for Context-Aware Prompting in Smart Environments", CCNC Workshop on Consumer eHealth Platforms, Services and Applications (CeHPSA), 2012.

- B. Das, D. J. Cook, "Data Mining Challenges in Automated Prompting Systems", IUI Workshop on Interaction with Smart Objects Workshop (InterSO), 2011.

- B. Das, C. Chen, N. Dasgupta, D. J. Cook, "Automated Prompting in a Smart Home Environment", ICDM Workshop on Data Mining for Service, 2010.

- C. Chen, B. Das, D. J. Cook, "Energy Prediction Using Resident's Activity", KDD Workshop on Knowledge Discovery from Sensor Data (SensorKDD), 2010.

- C. Chen, B. Das, D. J. Cook, "Energy Prediction in Smart Environments", IE Workshop on Artificial Intelligence Techniques for Ambient Intelligence (AITAmI), 2010.

Appendix B

Additional Results of Chapter 2

Sensitivity

Dataset	Baseline	SMOTE	SMOTEBoost	RUSBoost	RACOG	wRACOG
prompting	0.0000 ± 0.0000	0.4000 ± 0.1312	0.1600 ± 0.0796	0.0000 ± 0.0	**0.7467 ± 0.1193**	**0.7933 ± 0.0863**
abalone	0.0000 ± 0.0000	0.8615 ± 0.0417	0.8231 ± 0.0832	0.0000 ± 0.0	0.8846 ± 0.0360	**0.9577 ± 0.0285**
car	0.0000 ± 0.0000	0.6286 ± 0.1549	0.8429 ± 0.0931	0.0000 ± 0.0000	1.0000 ± 0.0000	1.0000 ± 0.0000
nursery	0.6697 ± 0.0291	0.7333 ± 0.0968	0.9667 ± 0.0377	0.6030 ± 0.0918	1.0000 ± 0.0000	1.0000 ± 0.0000
letter	0.6355 ± 0.0240	0.7395 ± 0.0590	0.8513 ± 0.0321	0.6487 ± 0.0577	0.8171 ± 0.0296	**0.9846 ± 0.0100**
connect-4	0.1280 ± 0.0415	0.3433 ± 0.1021	0.3440 ± 0.0288	0.1220 ± 0.0217	**0.6240 ± 0.0546**	**0.9767 ± 0.0115**

G-mean

Dataset	Baseline	SMOTE	SMOTEBoost	RUSBoost	RACOG	wRACOG
prompting	0.0000 ± 0.0000	0.6136 ± 0.0978	0.3868 ± 0.1016	0.0000 ± 0.0000	**0.8224 ± 0.0674**	**0.8352 ± 0.0380**
abalone	0.0000 ± 0.0000	0.8247 ± 0.0187	0.8120 ± 0.0301	0.0000 ± 0.0000	0.8304 ± 0.0194	0.7767 ± 0.0085
car	0.0000 ± 0.0000	0.7801 ± 0.0960	0.9136 ± 0.0506	0.0000 ± 0.0000	**0.9842 ± 0.0026**	**0.9867 ± 0.0042**
nursery	0.8161 ± 0.0176	0.8518 ± 0.0546	0.9827 ± 0.0192	0.7735 ± 0.0588	0.9938 ± 0.0005	0.9944 ± 0.0006
letter	0.7958 ± 0.0148	0.8546 ± 0.0345	0.9204 ± 0.0177	0.8034 ± 0.0353	0.9340 ± 0.0139	**0.9501 ± 0.0052**
connect-4	0.3522 ± 0.0579	0.5574 ± 0.0803	0.5756 ± 0.0237	0.3461 ± 0.0308	**0.7024 ± 0.0299**	0.5584 ± 0.0433

AUC-ROC

Dataset	Baseline	SMOTE	SMOTEBoost	RUSBoost	RACOG	wRACOG
prompting	0.5000 ± 0.0000	0.8681 ± 0.0359	0.8546 ± 0.0334	0.5000 ± 0.0000	0.8851 ± 0.0207	0.8717 ± 0.0404
abalone	0.5000 ± 0.0000	0.8693 ± 0.0272	0.8718 ± 0.0262	0.5000 ± 0.0000	0.8646 ± 0.0233	0.8727 ± 0.0208
car	0.5000 ± 0.0000	0.9595 ± 0.0320	0.9957 ± 0.0025	0.5000 ± 0.0000	0.9879 ± 0.0027	0.9904 ± 0.0067
nursery	0.9849 ± 0.0077	0.9515 ± 0.0219	0.9999 ± 0.0001	0.7999 ± 0.0456	0.9968 ± 0.0005	0.9967 ± 0.0005
letter	0.8745 ± 0.0056	0.9340 ± 0.0153	0.9865 ± 0.0080	0.8226 ± 0.0286	0.9659 ± 0.0102	0.9707 ± 0.0026
connect-4	0.6215 ± 0.0310	0.6907 ± 0.0335	0.8317 ± 0.0140	0.5552 ± 0.0092	0.7333 ± 0.0181	0.7115 ± 0.0282

Table B.1: **Results for C4.5 Decision Tree**

Sensitivity

Dataset	Baseline	SMOTE	SMOTEBoost	RUSBoost	RACOG	wRACOG
prompting	0.0000 ± 0.0000	0.6667 ± 0.0667	0.2600 ± 0.0641	0.0000 ± 0.0000	**0.5600 ± 0.1011**	**0.5400 ± 0.0723**
abalone	0.0000 ± 0.0000	0.9462 ± 0.0251	0.8462 ± 0.0769	0.0000 ± 0.0000	**0.9385 ± 0.0161**	**0.9846 ± 0.0161**
car	0.0000 ± 0.0000	1.0000 ± 0.0000	0.9000 ± 0.1195	0.0000 ± 0.0000	1.0000 ± 0.0000	1.0000 ± 0.0000
nursery	0.0000 ± 0.0000	0.9939 ± 0.0136	0.9848 ± 0.0186	0.7091 ± 0.0529	1.0000 ± 0.0000	1.0000 ± 0.0000
letter	0.4645 ± 0.0430	0.9026 ± 0.0409	0.9092 ± 0.0388	0.6487 ± 0.0396	0.9289 ± 0.0220	0.9364 ± 0.0166
connect-4	0.0000 ± 0.0000	0.6233 ± 0.0473	0.5680 ± 0.0383	0.0000 ± 0.0000	**0.7880 ± 0.0390**	**0.9867 ± 0.0153**

G-mean

Dataset	Baseline	SMOTE	SMOTEBoost	RUSBoost	RACOG	wRACOG
prompting	0.0000 ± 0.0000	0.7653 ± 0.0419	0.5008 ± 0.0587	0.0000 ± 0.0000	**0.7184 ± 0.0612**	**0.7053 ± 0.0430**
abalone	0.0000 ± 0.0000	0.8049 ± 0.0079	0.8087 ± 0.0324	0.0000 ± 0.0000	0.8020 ± 0.0096	0.7850 ± 0.0140
car	0.0000 ± 0.0000	0.9629 ± 0.0028	0.9278 ± 0.0633	0.0000 ± 0.0000	0.9632 ± 0.0076	0.9660 ± 0.0075
nursery	0.0000 ± 0.0000	0.9742 ± 0.0065	0.9817 ± 0.0095	0.8403 ± 0.0320	0.9660 ± 0.0020	0.9760 ± 0.0042
letter	0.6804 ± 0.0316	0.9341 ± 0.0223	0.9484 ± 0.0195	0.8034 ± 0.0244	0.9351 ± 0.0115	0.9395 ± 0.0046
connect-4	0.0000 ± 0.0000	0.7306 ± 0.0282	0.7135 ± 0.0221	0.0000 ± 0.0000	**0.8007 ± 0.0210**	**0.8269 ± 0.0351**

AUC-ROC

Dataset	Baseline	SMOTE	SMOTEBoost	RUSBoost	RACOG	wRACOG
prompting	0.5000 ± 0.0000	0.7734 ± 0.0384	0.8620 ± 0.0170	0.5000 ± 0.0000	0.7437 ± 0.0490	0.7332 ± 0.0316
abalone	0.5000 ± 0.0000	0.8156 ± 0.0100	0.8792 ± 0.0339	0.5000 ± 0.0000	0.8121 ± 0.0074	0.8054 ± 0.0102
car	0.5000 ± 0.0000	0.9636 ± 0.0027	0.9772 ± 0.0132	0.5000 ± 0.0000	0.9639 ± 0.0073	0.9666 ± 0.0073
nursery	0.5000 ± 0.0000	0.9744 ± 0.0065	0.9958 ± 0.0006	0.8530 ± 0.0264	0.9665 ± 0.0020	0.9763 ± 0.0041
letter	0.7315 ± 0.0213	0.9348 ± 0.0214	0.9915 ± 0.0050	0.8222 ± 0.0194	0.9351 ± 0.0114	0.9396 ± 0.0045
connect-4	0.5000 ± 0.0000	0.7402 ± 0.0238	0.8503 ± 0.0081	0.5000 ± 0.0000	0.8010 ± 0.0204	0.8210 ± 0.0203

Table B.2: **Results for SVM**

Sensitivity

Dataset	Baseline	SMOTE	SMOTEBoost	RUSBoost	RACOG	wRACOG
prompting	0.0533 ± 0.0298	0.6333 ± 0.0471	0.4667 ± 0.2055	0.0600 ± 0.0435	**0.7533 ± 0.0767**	**0.6600 ± 0.0760**
abalone	0.0000 ± 0.0000	0.8885 ± 0.0498	0.8308 ± 0.0896	0.0000 ± 0.0000	0.8846 ± 0.0360	**0.9423 ± 0.0136**
car	0.0429 ± 0.0639	0.7571 ± 0.1644	0.9286 ± 0.0505	0.0286 ± 0.0391	1.0000 ± 0.0000	1.0000 ± 0.0000
nursery	0.3121 ± 0.0409	0.9364 ± 0.0561	0.9727 ± 0.0166	0.3303 ± 0.0046	1.0000 ± 0.0000	1.0000 ± 0.0000
letter	0.7566 ± 0.0372	0.9118 ± 0.0374	0.8987 ± 0.0253	0.7250 ± 0.0225	**0.9211 ± 0.0186**	0.9189 ± 0.0166
connect-4	0.0360 ± 0.0134	0.7133 ± 0.0586	0.4940 ± 0.0654	0.0380 ± 0.0110	**0.6100 ± 0.0436**	**0.8833 ± 0.0651**

G-mean

Dataset	Baseline	SMOTE	SMOTEBoost	RUSBoost	RACOG	wRACOG
prompting	0.2243 ± 0.0607	0.7547 ± 0.0268	0.6449 ± 0.1423	0.2144 ± 0.1318	**0.8272 ± 0.0418**	**0.7798 ± 0.0405**
abalone	0.0000 ± 0.0000	0.8301 ± 0.0198	0.8177 ± 0.0367	0.0000 ± 0.0000	0.8311 ± 0.0192	0.7841 ± 0.0127
car	0.1290 ± 0.1810	0.8093 ± 0.0729	0.9193 ± 0.0244	0.1069 ± 0.1464	0.8916 ± 0.0187	**0.9471 ± 0.0071**
nursery	0.5578 ± 0.0358	0.9262 ± 0.0206	0.9670 ± 0.0105	0.5741 ± 0.0309	0.9171 ± 0.0040	**0.9832 ± 0.0025**
letter	0.8683 ± 0.0213	0.9361 ± 0.0198	0.9366 ± 0.0122	0.8503 ± 0.0130	0.9350 ± 0.0102	0.9431 ± 0.0099
connect-4	0.1869 ± 0.0358	0.7097 ± 0.0295	0.6596 ± 0.0445	0.1932 ± 0.0276	**0.7153 ± 0.0250**	0.6911 ± 0.0184

AUC-ROC

Dataset	Baseline	SMOTE	SMOTEBoost	RUSBoost	RACOG	wRACOG
prompting	0.8707 ± 0.0225	0.8670 ± 0.0228	0.8168 ± 0.0197	0.5290 ± 0.0214	**0.8854 ± 0.0307**	**0.8905 ± 0.0168**
abalone	0.8819 ± 0.0179	0.8760 ± 0.0338	0.8783 ± 0.0243	0.5000 ± 0.0000	0.8812 ± 0.0210	0.8725 ± 0.0089
car	0.9930 ± 0.0035	0.9543 ± 0.0129	0.9724 ± 0.0112	0.5143 ± 0.0196	**0.9932 ± 0.0035**	**0.9941 ± 0.0043**
nursery	0.9999 ± 0.0003	0.9875 ± 0.0073	0.9949 ± 0.0016	0.6652 ± 0.0173	**0.9994 ± 0.0002**	**0.9990 ± 0.0004**
letter	0.9872 ± 0.0051	0.9818 ± 0.0074	0.9808 ± 0.0077	0.8613 ± 0.0110	0.9860 ± 0.0062	**0.9887 ± 0.0018**
connect-4	0.7744 ± 0.0426	0.7732 ± 0.0185	0.7844 ± 0.0235	0.5183 ± 0.0054	**0.8262 ± 0.0241**	0.7824 ± 0.0349

Table B.3: **Results for k-Nearest Neighbor**

Sensitivity

Dataset	Baseline	SMOTE	SMOTEBoost	RUSBoost	RACOG	wRACOG
prompting	0.0867 ± 0.0447	0.5667 ± 0.0850	0.2933 ± 0.1657	0.1533 ± 0.0380	$\mathbf{0.4667 \pm 0.1179}$	$\mathbf{0.4867 \pm 0.1043}$
abalone	0.0000 ± 0.0000	0.9154 ± 0.0258	0.8962 ± 0.0322	0.0000 ± 0.0000	0.8923 ± 0.0421	0.9308 ± 0.0586
car	0.3429 ± 0.1174	0.9571 ± 0.0639	0.8714 ± 0.0782	0.3286 ± 0.1481	1.0000 ± 0.0	1.0000 ± 0.0000
nursery	0.7394 ± 0.0127	0.9424 ± 0.0628	0.9515 ± 0.0392	0.7576 ± 0.0557	1.0000 ± 0.0000	1.0000 ± 0.000
letter	0.6171 ± 0.0482	0.9000 ± 0.0471	0.9197 ± 0.0086	0.6566 ± 0.0205	0.9184 ± 0.0231	$\mathbf{0.9518 \pm 0.0249}$
connect-4	0.2920 ± 0.0785	0.6300 ± 0.0361	0.5600 ± 0.0543	0.3040 ± 0.0439	$\mathbf{0.8120 \pm 0.0327}$	$\mathbf{1.0000 \pm 0.0000}$

G-mean

Dataset	Baseline	SMOTE	SMOTEBoost	RUSBoost	RACOG	wRACOG
prompting	0.2845 ± 0.0782	0.7186 ± 0.0503	0.5141 ± 0.1537	0.3861 ± 0.0497	$\mathbf{0.6596 \pm 0.0784}$	$\mathbf{0.6722 \pm 0.0778}$
abalone	0.0000 ± 0.0000	0.8441 ± 0.0184	0.8264 ± 0.0245	0.0000 ± 0.0000	0.8290 ± 0.0197	0.7971 ± 0.0496
car	0.5746 ± 0.0979	0.9543 ± 0.0322	0.9092 ± 0.0410	0.5556 ± 0.1362	$\mathbf{0.9747 \pm 0.0050}$	$\mathbf{0.9716 \pm 0.0040}$
nursery	0.8573 ± 0.0066	0.9600 ± 0.0309	0.9655 ± 0.0189	0.8679 ± 0.0319	$\mathbf{0.9860 \pm 0.0017}$	$\mathbf{0.9887 \pm 0.0013}$
letter	0.7826 ± 0.0305	0.9243 ± 0.0252	0.9329 ± 0.0076	0.8082 ± 0.0122	0.9270 ± 0.0114	0.9286 ± 0.0100
connect-4	0.5293 ± 0.0778	0.7383 ± 0.0234	0.7051 ± 0.0317	0.5438 ± 0.0409	$\mathbf{0.8046 \pm 0.0164}$	$\mathbf{0.8142 \pm 0.0451}$

AUC-ROC

Dataset	Baseline	SMOTE	SMOTEBoost	RUSBoost	RACOG	wRACOG
prompting	0.8714 ± 0.0173	0.8643 ± 0.0371	0.8404 ± 0.0246	0.5691 ± 0.0192	0.8269 ± 0.0482	0.8222 ± 0.0124
abalone	0.8914 ± 0.0082	0.8900 ± 0.0167	0.8756 ± 0.0191	0.4999 ± 0.2852	0.8914 ± 0.0149	0.8832 ± 0.0304
car	0.9778 ± 0.0031	0.9758 ± 0.0039	0.9652 ± 0.0054	0.6562 ± 0.0754	$\mathbf{0.9731 \pm 0.0080}$	$\mathbf{0.9741 \pm 0.0065}$
nursery	0.9954 ± 0.0007	0.9948 ± 0.0010	0.9923 ± 0.0009	0.8764 ± 0.0279	$\mathbf{0.9956 \pm 0.0007}$	$\mathbf{0.9948 \pm 0.0010}$
letter	0.9817 ± 0.0025	0.9782 ± 0.0055	0.9756 ± 0.0066	0.8258 ± 0.0096	0.9810 ± 0.0048	0.9787 ± 0.0065
connect-4	0.8769 ± 0.0086	0.8538 ± 0.0088	0.8320 ± 0.0157	0.6403 ± 0.0238	$\mathbf{0.8742 \pm 0.0276}$	$\mathbf{0.8840 \pm 0.0066}$

Table B.4: **Results for Logistic Regression**

Appendix C

Raw Accelerometer Data

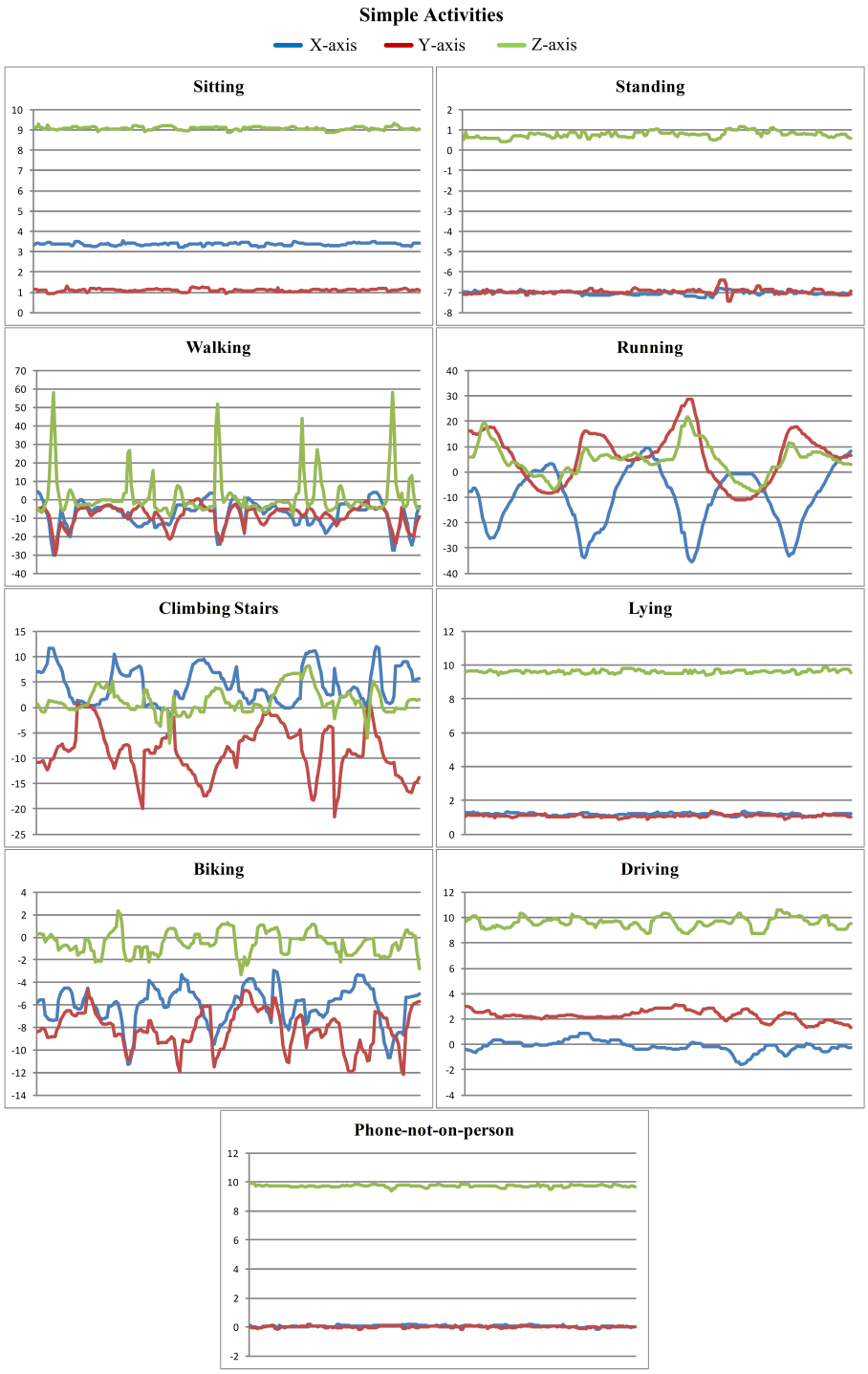

Figure C.1: Raw data for simple activities

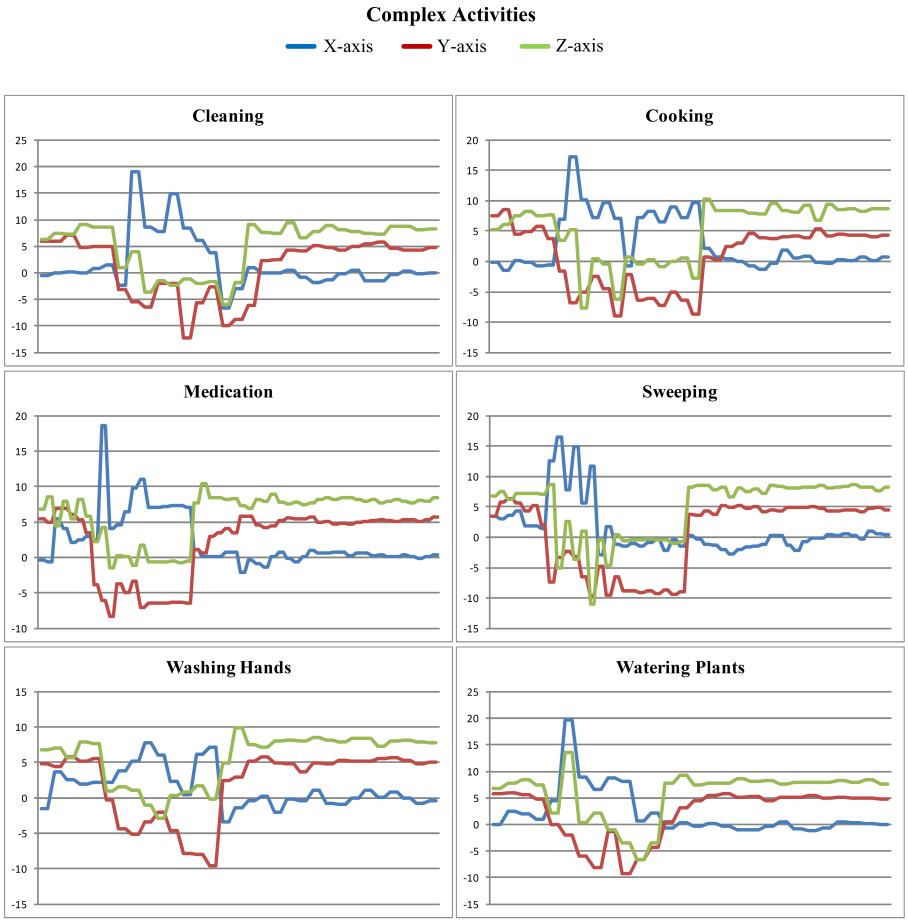

Figure C.2: Raw data for complex activities

Bibliography

[1] U. Maurer, A. Smailagic, D. Siewiorek, and M. Deisher, "Activity recognition and monitoring using multiple sensors on different body positions," in *Wearable and Implantable Body Sensor Networks, 2006. BSN 2006. International Workshop on.* IEEE, 2006, pp. 4–116.

[2] G. Singla, D. Cook, and M. Schmitter-Edgecombe, "Tracking activities in complex settings using smart environment technologies," *International journal of biosciences, psychiatry, and technology (IJBSPT)*, vol. 1, no. 1, p. 25, 2009.

[3] U. C. Bureau, "International database, Table 094," 2010. [Online]. Available: http://www.census.gov/population/www/projections/natdet-D1A.html

[4] J. Bates, J. Boote, and C. Beverley, "Psychosocial interventions for people with a milder dementing illness: a systematic review," *Journal of Advanced Nursing*, vol. 45, no. 6, pp. 644–658, 2004.

[5] V. Wadley, O. Okonkwo, M. Crowe, and L. Ross-Meadows, "Mild cognitive impairment and everyday function: evidence of reduced speed in performing instrumental activities of daily living," *American Journal of Geriatric Psych*, vol. 16, no. 5, p. 416, 2008.

[6] Alzheimer's and Dementia, Eds., *Alzheimer Association Report: Alzheimer's disease facts and figures*, vol. 6, 2010.

[7] B. Das and D. Cook, "Data mining challenges in automated prompting systems," in *Proceedings of 2011 Internatuional Conference on Intelligent User Interfaces Workshop on Interaction with Smart Objects*, 2011.

[8] P. Kaushik, S. Intille, and K. Larson, "Observations from a case study on user adaptive reminders for medication adherence," in *Pervasive Computing Technologies for Healthcare, 2008. PervasiveHealth 2008. Second International Conference on.* IEEE, 2008, pp. 250–253.

[9] C. Sjogreen, "How we built google calendar," in *Carson Workshop*, 2006.

[10] N. Marmasse and C. Schmandt, "Location-aware information delivery with commotion," in *Handheld and Ubiquitous Computing.* Springer, 2000, pp. 361–370.

[11] C. Kingdon, B. Zadeh, M. Roel-Ng, and S. Hayes, "System and method for authorization of location services," Oct. 24 2000, uS Patent 6,138,003.

[12] S. Zellner, M. Enzmann, and R. Moton Jr, "Anonymous location service for wireless networks," May 18 2004, uS Patent 6,738,808.

[13] T. Sohn, K. Li, G. Lee, I. Smith, J. Scott, and W. Griswold, "Place-its: A study of location-based reminders on mobile phones," *UbiComp 2005: Ubiquitous Computing*, pp. 232–250, 2005.

[14] D. Cook and S. Das, *Smart environments: technologies, protocols, and applications.* Wiley-Interscience, 2005, vol. 43.

[15] ——, "How smart are our environments? an updated look at the state of the art," *Pervasive and Mobile Computing*, vol. 3, no. 2, pp. 53–73, 2007.

[16] P. Rashidi and D. Cook, "Keeping the resident in the loop: Adapting the smart home to the user," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 39, no. 5, pp. 949–959, 2009.

[17] B. Das, C. Chen, A. Seelye, and D. Cook, "An automated prompting system for smart environments," *Toward Useful Services for Elderly and People with Disabilities*, pp. 9–16, 2011.

[18] H. Kautz, L. Arnstein, G. Borriello, O. Etzioni, and D. Fox, "An overview of the assisted cognition project," in *AAAI-2002 Workshop on*

Automation as Caregiver: The Role of Intelligent Technology in Elder Care, 2002, pp. 60–65.

[19] H. Kautz, O. Etzioni, D. Fox, D. Weld, and L. Shastri, "Foundations of assisted cognition systems," *University of Washington, Computer Science Department, Technical Report, Tech. Rep*, 2003.

[20] M. Lamming and M. Flynn, "Forget-me-not: Intimate computing in support of human memory," in *Proceedings of FRIEND21*, vol. 94. Citeseer, 1994, pp. 2–4.

[21] A. Dey and G. Abowd, "Cybreminder: A context-aware system for supporting reminders," in *Handheld and Ubiquitous Computing*. Springer, 2000, pp. 201–207.

[22] A. Dey, G. Abowd, and D. Salber, "A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications," *Human–Computer Interaction*, vol. 16, no. 2, pp. 97–166, 2001.

[23] A. Hristova, A. Bernardos, and J. Casar, "Context-aware services for ambient assisted living: A case-study," in *Applied Sciences on Biomedical and Communication Technologies, 2008. ISABEL'08. First International Symposium on*. IEEE, 2008, pp. 1–5.

[24] A. Mihailidis and G. Fernie, "Context-aware assistive devices for older adults with dementia," *Gerontechnology*, vol. 2, no. 2, pp. 173–189, 2002.

[25] S. Helal, C. Giraldo, Y. Kaddoura, C. Lee, H. El Zabadani, and W. Mann, "Smart phone based cognitive assistant," in *UbiHealth 2003: The 2nd International Workshop on Ubiquitous Computing for Pervasive Healthcare Applications*, 2003.

[26] S. Vurgun, M. Philipose, and M. Pavel, "A statistical reasoning system for medication prompting," *UbiComp 2007: Ubiquitous Computing*, pp. 1–18, 2007.

[27] K. Du, D. Zhang, X. Zhou, M. Mokhtari, M. Hariz, and W. Qin, "Hycare: A hybrid context-aware reminding framework for elders with mild dementia," *Smart Homes and Health Telematics*, pp. 9–17, 2008.

[28] Y. Chang, W. Chang, and T. Wang, "Context-aware prompting to transition autonomously through vocational tasks for individuals with cognitive impairments," in *Proceedings of the 11th international ACM SIGACCESS conference on Computers and accessibility*. ACM, 2009, pp. 19–26.

[29] V. Osmani, D. Zhang, and S. Balasubramaniam, "Human activity recognition supporting context-appropriate reminders for elderly," in *Pervasive Computing Technologies for Healthcare, 2009. Pervasive-Health 2009. 3rd International Conference on*. IEEE, pp. 1–4.

[30] L. Tang, X. Zhou, Z. Yu, D. Zhang, and H. Ni, "Adaptive prompting based on petri net in a smart medication system," in *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference on*. IEEE, 2010, pp. 328–333.

[31] F. Marquardt and A. Uhrmacher, "Evaluating AI planning for service composition in smart environments," in *Proceedings of the 7th International Conference on Mobile and Ubiquitous Multimedia*. ACM, 2008, pp. 48–55.

[32] ——, "Creating AI planning domains for smart environments using pddl," *Intelligent Interactive Assistance and Mobile Multimedia Computing*, pp. 263–274, 2009.

[33] R. Levinson, "The planning and execution assistant and training system," *Journal of Head Trauma Rehabilitation*, vol. 12(2), pp. 769–775, 1997.

[34] M. Pollack, L. Brown, D. Colbry, C. McCarthy, C. Orosz, B. Peintner, S. Ramakrishnan, and I. Tsamardinos, "Autominder: An intelligent cognitive orthotic system for people with memory impairment," *Robotics and Autonomous Systems*, vol. 44, no. 3-4, pp. 273–282, 2003.

[35] J. Pineau, M. Montemerlo, M. Pollack, N. Roy, and S. Thrun, "Towards robotic assistants in nursing homes: Challenges and results," *Robotics and Autonomous Systems*, vol. 42, no. 3-4, pp. 271–281, 2003.

[36] J. Boger, J. Hoey, P. Poupart, C. Boutilier, G. Fernie, and A. Mihailidis, "A planning system based on markov decision processes to guide

people with dementia through activities of daily living," *Information Technology in Biomedicine, IEEE Transactions on*, vol. 10, no. 2, pp. 323–333, 2006.

[37] A. Mihailidis, J. Boger, T. Craig, and J. Hoey, "The coach prompting system to assist older adults with dementia through handwashing: An efficacy study," *BMC geriatrics*, vol. 8, no. 1, p. 28, 2008.

[38] M. Rudary, S. Singh, and M. Pollack, "Adaptive cognitive orthotics: combining reinforcement learning and constraint-based temporal reasoning," in *Proceedings of the twenty-first international conference on Machine learning*. ACM, 2004, p. 91.

[39] K. Haigh, L. Kiff, and G. Ho, "The independent lifestyle assistant: Lessons learned," *Assistive Technology*, vol. 18, no. 1, pp. 87–106, 2006.

[40] B. Das, C. Chen, N. Dasgupta, D. Cook, and A. Seelye, "Automated prompting in a smart home environment," in *Data Mining Workshops (ICDMW), 2010 IEEE International Conference on*. IEEE, 2010, pp. 1045–1052.

[41] K. Woods, C. Doss, K. Bowyer, J. Solka, C. Priebe, and K. W., "Comparative evaluation of pattern recognition techniques for detection of microcalcifications in mammography," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 7, no. 6, pp. 1417–1436, 1993.

[42] M. Kubat, R. Holte, and S. Matwin, "Machine learning for the detection of oil spills in satellite radar images," *Machine learning*, vol. 30, no. 2, pp. 195–215, 1998.

[43] C. Phua, D. Alahakoon, and V. Lee, "Minority report in fraud detection: classification of skewed data," *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 50–59, 2004.

[44] P. Turney *et al.*, "Learning algorithms for keyphrase extraction," *Information Retrieval*, vol. 2, no. 4, pp. 303–336, 2000.

[45] D. Lewis and W. Gale, "A sequential algorithm for training text classifiers," in *Proceedings of the 17th Annual International ACM SIGIR*

Conference on Research and Development in Information Retrieval. Springer-Verlag New York, Inc., 1994, pp. 3–12.

[46] A. Liu, J. Ghosh, and C. Martin, "Generative oversampling for mining imbalanced datasets," in *Proceedings of the 2007 International Conference on Data Mining, DMIN*, 2007, pp. 25–28.

[47] B. Zadrozny, J. Langford, and N. Abe, "Cost-sensitive learning by cost-proportionate example weighting," in *Third IEEE International Conference on Data Mining, 2003. ICDM 2003.* IEEE, 2003, pp. 435–442.

[48] C. Elkan, "The foundations of cost-sensitive learning," in *International Joint Conference on Artificial Intelligence*, vol. 17, no. 1. Lawrence Erlbaum Associates Ltd., 2001, pp. 973–978.

[49] K. McCarthy, B. Zabar, and G. Weiss, "Does cost-sensitive learning beat sampling for classifying rare classes?" in *Proceedings of the 1st international workshop on Utility-based data mining.* ACM, 2005, pp. 69–77.

[50] M. Maloof, "Learning when data sets are imbalanced and when costs are unequal and unknown," in *ICML-2003 Workshop on Learning from Imbalanced Data Sets II*, 2003.

[51] G. Weiss, "Mining with rarity: a unifying framework," *SigKDD Explorations*, vol. 6, no. 1, pp. 7–19, 2004.

[52] C. Seiffert, T. Khoshgoftaar, J. Van Hulse, and A. Napolitano, "RUSBoost: A hybrid approach to alleviating class imbalance," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 40, no. 1, pp. 185–197, 2010.

[53] N. Chawla, K. Bowyer, L. Hall, and W. Kegelmeyer, "SMOTE: synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, June 2002.

[54] N. Chawla, A. Lazarevic, L. Hall, and K. Bowyer, "SMOTEBoost: Improving prediction of the minority class in boosting," *Knowledge Discovery in Databases: PKDD 2003*, pp. 107–119, 2003.

[55] N. Chawla, N. Japkowicz, and A. Kotcz, "Editorial: special issue on learning from imbalanced data sets," *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 1–6, 2004.

[56] H. He and E. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[57] Y. Sun, A. Wong, and M. Kamel, "Classification of imbalanced data: A review," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 23, no. 4, p. 687, 2009.

[58] N. Chawla, "Data mining for imbalanced datasets: An overview," *Data Mining and Knowledge Discovery Handbook*, pp. 875–886, 2010.

[59] X. Liu and Z. Zhou, "The influence of class imbalance on cost-sensitive learning: an empirical study," in *Data Mining, 2006. ICDM'06. Sixth International Conference on.* IEEE, 2006, pp. 970–974.

[60] F. Provost, T. Fawcett, and R. Kohavi, "The case against accuracy estimation for comparing induction algorithms," in *Proceedings of the Fifteenth International Conference on Machine Learning*, vol. 445, 1998.

[61] C. Drummond and R. Holte, "Exploiting the cost (in) sensitivity of decision tree splitting criteria," in *Machine Learning-International Workshop then Conference*, 2000, pp. 239–246.

[62] M. Kukar and I. Kononenko, "Cost-sensitive learning with neural networks," in *Proceedings of the 13th European Conference on Artificial Intelligence (ECAI-98).* Citeseer, 1998, pp. 445–449.

[63] G. Weiss, K. McCarthy, and B. Zabar, "Cost-sensitive learning vs. sampling: Which is best for handling unbalanced classes with unequal error costs," in *International Conference on Data Mining*, 2007, pp. 35–41.

[64] X. Hong, S. Chen, and C. Harris, "A kernel-based two-class classifier for imbalanced data sets," *IEEE Transactions on Neural Networks*, vol. 18, no. 1, pp. 28–41, 2007.

[65] G. Wu and E. Chang, "KBA: Kernel boundary alignment considering imbalanced data distribution," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 6, pp. 786–795, 2005.

[66] G. Fung and O. Mangasarian, "Multicategory proximal support vector machine classifiers," *Machine Learning*, vol. 59, no. 1, pp. 77–97, 2005.

[67] H. Han, W. Wang, and B. Mao, "Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning," *Advances in Intelligent Computing*, pp. 878–887, 2005.

[68] H. He, Y. Bai, E. Garcia, and S. Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," in *IEEE International Joint Conference on Neural Networks, 2008 (IEEE World Congress on Computational Intelligence)*. IEEE, 2008, pp. 1322–1328.

[69] T. Jo and N. Japkowicz, "Class imbalances versus small disjuncts," *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 40–49, 2004.

[70] X. Liu, J. Wu, and Z. Zhou, "Exploratory undersampling for class-imbalance learning," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 39, no. 2, pp. 539–550, 2009.

[71] V. García, R. Mollineda, and J. Sánchez, "On the k-nn performance in a challenging scenario of imbalance and overlapping," *Pattern Analysis & Applications*, vol. 11, no. 3, pp. 269–280, 2008.

[72] I. Tomek, "Two modifications of CNN," *IEEE Transaction on Systems, Man and Cybernetics*, vol. 6, pp. 769–772, 1976.

[73] M. Kubat and S. Matwin, "Addressing the curse of imbalanced training sets: one-sided selection," in *Machine Learning-International Workshop then Conference*. Morgan Kaufmann Publishers, Inc., 1997, pp. 179–186.

[74] G. Batista, R. Prati, and M. Monard, "A study of the behavior of several methods for balancing machine learning training data," *ACM SIGKDD Explorations Newsletter*, vol. 6, no. 1, pp. 20–29, 2004.

[75] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, and F. Herrera, "A review on ensembles for the class imbalance problem: Bagging-,

boosting-, and hybrid-based approaches," *IEEE Transactions on Systems, Man and Cybernetics-Part : Applications and Reviews*, vol. 42, no. 4, pp. 463–484, July 2012.

[76] Y. Freund and R. Schapire, "A desicion-theoretic generalization of online learning and an application to boosting," in *Computational Learning Theory*. Springer, 1995, pp. 23–37.

[77] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[78] J. Błaszczyński, M. Deckert, J. Stefanowski, and S. Wilk, "Integrating selective pre-processing of imbalanced data with ivotes ensemble," in *Rough Sets and Current Trends in Computing*. Springer, 2010, pp. 148–157.

[79] S. Wang and X. Yao, "Diversity analysis on imbalanced data sets by using ensemble models," in *Computational Intelligence and Data Mining, 2009. CIDM'09. IEEE Symposium on*. IEEE, 2009, pp. 324–331.

[80] N. Meteopolis and S. Ulam, "The monte carlo method," *Journal of the American statistical association*, vol. 44, no. 247, pp. 335–341, 1949.

[81] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The Journal of Chemical Physics*, vol. 21, p. 1087, 1953.

[82] E. Chen, Y. Lin, H. Xiong, Q. Luo, and H. Ma, "Exploiting probabilistic topic models to improve text categorization under class imbalance," *Information Processing & Management*, vol. 47, no. 2, pp. 202–214, 2011.

[83] C. Geyer, "Practical markov chain monte carlo," *Statistical Science*, vol. 7, no. 4, pp. 473–483, 1992.

[84] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian network classifiers," *Machine learning*, vol. 29, no. 2, pp. 131–163, 1997.

[85] G. Weiss and F. Provost, "Learning when training data are costly: The effect of class distribution on tree induction," *Journal of Artificial Intelligence Research (JAIR)*, vol. 19, pp. 315–354, 2003.

[86] W. Cohen and Y. Singer, "A simple, fast, and effective rule learner," in *Proceedings of the National Conference on Artificial Intelligence*. John Wiley & Sons Ltd., 1999, pp. 335–342.

[87] T. Khoshgoftaar, C. Seiffert, J. Van Hulse, A. Napolitano, and A. Folleco, "Learning with limited minority class data," in *Machine Learning and Applications, 2007. ICMLA 2007. Sixth International Conference on*. IEEE, 2007, pp. 348–353.

[88] A. Raftery and S. Lewis, "How many iterations in the gibbs sampler," *Bayesian statistics*, vol. 4, no. 2, pp. 763–773, 1992.

[89] P. Chan, W. Fan, A. Prodromidis, and S. Stolfo, "Distributed data mining in credit card fraud detection," *Intelligent Systems and their Applications, IEEE*, vol. 14, no. 6, pp. 67–74, 1999.

[90] T. Fawcett, "ROC graphs: Notes and practical considerations for researchers," *Machine Learning*, vol. 31, pp. 1–38, 2004.

[91] ——, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006.

[92] J. Huang and C. Ling, "Using auc and accuracy in evaluating learning algorithms," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 3, pp. 299–310, 2005.

[93] J. LeSage and R. Pace, *Introduction to spatial econometrics*. Chapman & Hall/CRC, 2009, vol. 196.

[94] M. Denil, "The effects of overlap and imbalance on svm classification," Ph.D. dissertation, Dalhousie University, 2010.

[95] I. Jolliffe, *Principal component analysis*. Wiley Online Library, 2005.

[96] H. Xiong, J. Wu, and L. Liu, "Classification with class overlapping: A systematic study."

[97] T. Trappenberg and A. Back, "A classification scheme for applications with ambiguous data," in *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*, vol. 6. IEEE, 2000, pp. 296–301.

[98] S. Hashemi and T. Trappenberg, "Using SVM for classification in datasets with ambiguous data," *SCI 2002*, 2002.

[99] T. Yaohua and G. Jinghuai, "Improved classification for problem involving overlapping patterns," *IEICE Transactions on Information and Systems*, vol. 90, no. 11, pp. 1787–1795, 2007.

[100] Y. Lin, X. Wang, W. Ng, Q. Chang, D. Yeung, and X. Wang, "Sphere classification for ambiguous data," in *2006 International Conference on Machine Learning and Cybernetics*. IEEE, 2006, pp. 2571–2574.

[101] C. Liu, "Partial discriminative training for classification of overlapping classes in document analysis," *International Journal on Document Analysis and Recognition*, vol. 11, no. 2, pp. 53–65, 2008.

[102] S. Andrews, *Learning from ambiguous examples*, 2007, vol. 68, no. 07.

[103] R. Prati, G. Batista, and M. Monard, "Class imbalances versus class overlapping: an analysis of a learning system behavior," *MICAI 2004: Advances in Artificial Intelligence*, pp. 312–321, 2004.

[104] G. Batista, R. Prati, and M. Monard, "Balancing strategies and class overlapping," *Advances in Intelligent Data Analysis VI*, pp. 741–741, 2005.

[105] V. García, R. Alejo, J. Sánchez, J. Sotoca, and R. Mollineda, "Combined effects of class imbalance and class overlap on instance-based classification," *Intelligent Data Engineering and Automated Learning–IDEAL 2006*, pp. 371–378, 2006.

[106] V. García, R. Mollineda, J. Sánchez, R. Alejo, and J. Sotoca, "When overlapping unexpectedly alters the class imbalance effects," *Pattern Recognition and Image Analysis*, pp. 499–506, 2007.

[107] S. Visa and A. Ralescu, "Learning imbalanced and overlapping classes using fuzzy sets," in *Proceedings of the ICML*, vol. 3, 2003.

[108] G. Batista, A. Bazan, and M. Monard, "Balancing training data for automated annotation of keywords: a case study," in *Proceedings of the Second Brazilian Workshop on Bioinformatics*, 2003, pp. 35–43.

[109] M. Denil and T. Trappenberg, "Overlap versus imbalance," *Advances in Artificial Intelligence*, pp. 220–231, 2010.

[110] P. Hart, "The condensed nearest neighbor rule," *IEEE Transaction on Information Theory*, vol. 3, pp. 515–516, 1968.

[111] B. Das, N. Krisgnan, and J. Cook, D, "Automated activity interventions to assist with activities of daily living," *Agents and Ambient Intelligence: Achievements and Challenges in the Intersection of Agent Technology and Ambient Intelligence*, vol. 12, pp. 137–158, 2012. [Online]. Available: http://www.booksonline.iospress.nl/Content/View.aspx?piid=30550

[112] M. Ester, H. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the 2nd International Conference on Knowledge Discovery and Data mining*, vol. 1996. AAAI Press, 1996, pp. 226–231.

[113] J. Han and M. Kamber, *Data mining: concepts and techniques*. Morgan Kaufmann, 2006.

[114] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. Witten, "The weka data mining software: an update," *ACM SIGKDD Explorations Newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[115] J. Hoey, P. Poupart, A. Bertoldi, T. Craig, C. Boutilier, and A. Mihailidis, "Automated handwashing assistance for persons with dementia using video and a partially observable markov decision process," *Computer Vision and Image Understanding*, vol. 114, no. 5, pp. 503–519, 2010.

[116] S. Zhang, S. McClean, B. Scotney, X. Hong, C. Nugent, and M. Mulvenna, "Decision support for alzheimer's patients in smart homes," in *Computer-Based Medical Systems, 2008. CBMS'08. 21st IEEE International Symposium on.* IEEE, 2008, pp. 236–241.

[117] M. Feki, J. Biswas, and A. Tolstikov, "Model and algorithmic framework for detection and correction of cognitive errors," *Technology and Health Care*, vol. 17, no. 3, pp. 203–219, 2009.

[118] C. Krishnan and D. J. Cook, "Activity recognition on streaming sensor data," *Journal of Pervasive and Mobile Computing*, 2012, to appear.

[119] Y. Chu, Y. Song, H. Kautz, and R. Levinson, "When did you start doing that thing that you do? interactive activity recognition and prompting," in *AAAI 2011 Workshop on Artificial Intelligence and Smarter Living: The Conquest of Complexity*, 2011, p. 7.

[120] D. J. Cook, N. C. Krishnan, and P. Rashidi, "Activity discovery and activity recognition: A new partnership," *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 2012, to appear.

[121] E. Tapia, S. Intille, and K. Larson, "Activity recognition in the home using simple and ubiquitous sensors," *Pervasive Computing*, pp. 158–175, 2004.

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

[123] N. Ravi, N. Dandekar, P. Mysore, and M. Littman, "Activity recognition from accelerometer data," in *Proceedings of the national conference on artificial intelligence*, vol. 20, no. 3. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2005, p. 1541.

[124] E. Tapia, S. Intille, W. Haskell, K. Larson, J. Wright, A. King, and R. Friedman, "Real-time recognition of physical activities and their intensities using wireless accelerometers and a heart rate monitor," in *Wearable Computers, 2007 11th IEEE International Symposium on*. IEEE, 2007, pp. 37–40.

[125] N. Krishnan, D. Colbry, C. Juillard, and S. Panchanathan, "Real time human activity recognition using tri-axial accelerometers," in *Sensors, signals and information processing workshop*, 2008.

[126] S. Lee and K. Mase, "Activity and location recognition using wearable sensors," *Pervasive Computing, IEEE*, vol. 1, no. 3, pp. 24–32, 2002.

[127] A. Subramanya, A. Raj, J. Bilmes, and D. Fox, "Recognizing activities and spatial context using wearable sensors," in *Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence*, 2006.

[128] J. Kwapisz, G. Weiss, and S. Moore, "Activity recognition using cell phone accelerometers," *ACM SIGKDD Explorations Newsletter*, vol. 12, no. 2, pp. 74–82, 2011.

[129] T. Brezmes, J. Gorricho, and J. Cotrina, "Activity recognition from accelerometer data on a mobile phone," *Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living*, pp. 796–799, 2009.

[130] A. Khan, Y. Lee, S. Lee, and T. Kim, "Human activity recognition via an accelerometer-enabled-smartphone using kernel discriminant analysis," in *Future Information Technology (FutureTech), 2010 5th International Conference on*. IEEE, 2010, pp. 1–6.

[131] S. Zhang, P. McCullagh, C. Nugent, and H. Zheng, "Activity monitoring using a smart phone's accelerometer with hierarchical classification," in *Intelligent Environments (IE), 2010 Sixth International Conference on*. IEEE, 2010, pp. 158–163.

[132] D. J. Cook, "Learning setting-generalized activity models for smart spaces," *IEEE Intelligent Systems*, vol. 27, no. 1, pp. 32–38, 2012.

[133] M. Philipose, K. Fishkin, M. Perkowitz, D. Patterson, D. Fox, H. Kautz, and D. Hahnel, "Inferring activities from interactions with objects," *Pervasive Computing, IEEE*, vol. 3, no. 4, pp. 50–57, 2004.