University of California

Los Angeles

Event Predictions for Remote Health Monitoring

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Computer Science

by

Mars Lan

2013

Abstract of the Dissertation

Event Predictions for Remote Health Monitoring

by

Mars Lan

Doctor of Philosophy in Computer Science

University of California, Los Angeles, 2013

Professor Majid Sarrafzadeh, Chair

Recent advances in electronic miniaturization, sensor technology, and wireless communications have opened the possibility of ubiquitous, small, and low-power sensor nodes that gather, process, and transmit various data over a long period of time. One of the key applications for these new technologies is in the area of remote health monitoring. Having the ability to continuously monitor numerous bodily measurements, as opposed to the occasional on-the-spot examination performed at a doctor visit, can potentially revolutionize the health care system. For the first time, physicians can make more informed decisions based the continuous history of a patient's wellness, rather than relying on the incomplete snapshots from the traditional medical records.

More importantly, using advanced data mining and machine learning techniques, it is possible to discover a wealth of patterns, knowledge, and relationships based on the data collected from a large population of different background, ethnicity, age group, and medical history. This thesis focuses specifically on event predictions for remote health monitoring. These events can be either acute clinical episodes, such as falling and epileptic seizure, or chronic conditions, such as congestive heart failure and diabetes. Based on the different requirements, this thesis tackles four key issues of event predictions for remote health monitoring:

(a) Subsequence-based prediction, (b) Sequential pattern mining, (c) Precursor pattern discovery, and (d) Predictions using discrete data. For each issue, one or multiple application areas have been identified, and the proposed algorithms have been validated using data gathered from real patients.

This dissertation is lovingly dedicated to my wife, Margaret, who has supported, encouraged and taken great care of me each step of the way.

Table of Contents

List of Figures

List of Tables

Vita

2003	B.E. (Computer Systems Engineering), University of Auckland, Auckland, New Zealand.
2005	M.E. (Computer Systems Engineering), University of Auckland, Auckland, New Zealand.
2005–2009	Software Engineer, Navico Auckland Limited.
2009–2010	Research Assistant, Computer Science Department, UCLA.
2010	M.S. (Computer Science), UCLA.
2010–2011	Research Assistant, Computer Science Department, UCLA.
2011–present	Teaching Assistant, Computer Science Department, UCLA.

CHAPTER 1

Subsequence-Based Predictions

In this chapter, we present an event prediction algorithm based on subsequence matching. The algorithm, SmartFall, is designed particularly for fall detection and cause identification; however, it can be potentially applied to many other applications with minimal modifications. The system employs subsequence matching techniques, which differ fundamentally from most existing fall detection systems based on multi-stage thresholding. The use of subsequent matching both reduces false positives and provides a unique and powerful fall prediction approach. The system achieves a near 100% sensitivity and specificity when detecting fall and distinguishing falls from normal daily activities. Using the real data collected from our sensing platform, SmartFall is able to achieve a cause identification accuracy of up to 79% using four classes and an improved accuracy of up to 93% using two classes. The accuracy remains relatively constant with a larger sample size involving fall patterns from different subjects.

1.1 Background

Fall-related injury is one of the most serious threats to the well-being of elderly population in recent years [83]. Falls often cause lesions, soft and connective tissue damages, bone fractures, and head injuries that lead to immediate or eventual death [46, 50]. Statistics show that about one-third of Americans aged over 65 fall at least once every year [89], and for those living in nursing homes, the likeliness of falling is almost three times as high [77]. Furthermore, 41% of the patients treated for falls at an emergency room were not able to get up within the first 5 minutes, with 3% ended up lying on the ground for up to 3 hours or more [49]. This highlghts the importance of reliable detection and immediate notification of care givers in the event of falls.

Apart from the health and safety concerns, falls can also incur a significant cost in health care expenditure. On average, a person's first fall can cost up to $20,000 [73]. It is also estimated that fractures sustained from falls cost some 10 billion dollars a year for the elderly population in US [88].

Major causes of falling include gait or balance disorder and lower-extremity weakness [76]. As a result, physicians often prescribe walking sticks or canes to assist the elderly people to overcome these problems [15]. In fact, these assistive devices are so prevalent that there are currently more than 4 million cane users in the US alone. The evident popularity and relative low cost of the cane make it an ideal candidate for wireless health care system. To this end, we have previously developed the SmartCane platform [92], which consists of a traditional-looking cane embedded with several sensors. The cane relays sensor information via a wireless interface to external devices for further processing and analysis. The platform has been subsequently extended to provide actively guided training for proper cane usage [8]. In this paper, we have developed the SmartFall algorithm on this platform and have demonstrated its automatic fall detection and cause identification capability.

SmartFall differs from many other fall detection systems in that it uses subsequence matching of the overall signal envelope instead of predefined thresholds. This helps our system to achieve high sensitivity and specificity even for fall-like activities such as resting the cane horizontally. The same principle can be applied to determine the exact causes of falls.

The remainder of this paper is organized as follows. Section 1.2 presents a review of related studies in automatic fall detection and cause identification as well as the subsequence matching algorithm employed in SmartFall. This is followed by a detailed description on the hardware, software, and algorithmic components of SmartFall in Section 1.3. In Section IV 1.4, a case study is presented as a way to evaluate the effectiveness of SmartFall both in terms of its fall detection rate and cause identification accuracy. Finally, in Section 1.5 we conclude the paper and outline possible improvement and future extension to SmartFall.

1.2 Related Work

1.2.1 Automatic Fall Detection & Cause Identification

Many systems have been developed over the years to automatically detect fall using various sensors. These systems can be generally categorized into wearable devices, ambience devices, and computer vision [66]. In this paper, we limit the scope of discussions to werable devices.

Most worn systems detect falls based on thresholding, which involves continuous comparisons of the raw or transformed sensor data, such as position, orientation, velocity, and force, against pre-defined thresholds. Earlier systems, such as those found in [62, 91], often employ simple single-stage thresholding. The more advanced systems use multi-stage thresholding, which requires a set of thresholds to be exceeded in a particular order over a certain time period for the alarm to be triggered. It has been thoroughly demonstrated that multi-stage thresholding outperforms its single-stage counterparts in terms of low false-positive rates [29]. One example of multi-stage thresholding fall detection system is shown in [24], where a tri-axial accelerometer is embedded into a wrist watch. The falling of the watch-bearer is detected using a three-stage acceleration-impact-inactivity thresholding on the norm of the accelerations. A similar system is presented in [40], where a sensor box is fastened to the chest of the testing subject. Apart from the 3D accelerometer, a tilt sensor and gyroscope have also been incorporated into the sensor box to improve accuracy. There are numerous other wearable fall detection systems that are based on the same principle but attached to different parts of the body, such as trunk [16], head [59], and waist [21, 95]. Some systems even require sensors attached to multiple parts of the body [63, 81]. Moreover, as smartphones become more popular among elderlies, several studies have also implemented fall detection algorithms on these mobile devices with varying degree of success [10, 23, 90].

While most thresholding-based fall detection algorithms claim to have near perfect detection rate in a controlled environment, they often fail to achieve sufficiently low false positive rates for real life usage [67]. One of the possible reasons can be demonstrated using Fig. 1.1. The two solid lines, Fall and Swing, in the graph are X-axis accelerometer signal acquired using the SmartCane during a forward fall and a back-and-forth swinging motion respectively. The two dotted lines, TH1 and TH2, denote the two thresholds for stage 1 and 2 respectively. A typical fall signal consists of a sudden acceleration during the free fall period, which triggers TH1, followed shortly by a sudden deceleration during the impact stage, which is detected by TH2. As can be seen in the graph, in spite of the obvious difference in shape, both signals would be classified as valid falls using multi-stage thresholding. Such a problem can be mitigated when using the subsequence matching algorithm proposed in this paper.

Despite the large amount of research devoted to automatic fall detection, there has been very little work done in analyzing the root cause of the fall. In fact, to our best knowledge, SmartFall is the first system to automatically deduce the possible cause that has led to the fall.

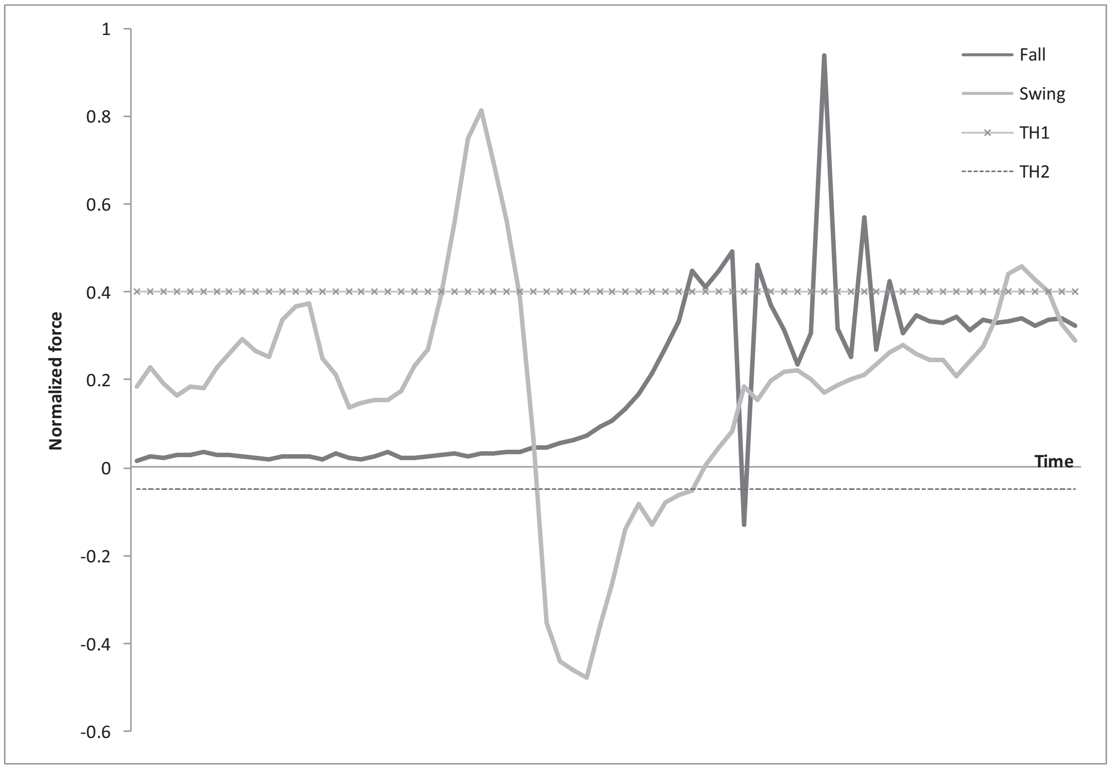


Figure 1.1: X-axis accelerometer signals for fall & swing actions on SmartCane

1.2.2 Subsequence Matching

Subsequence matching is a technique commonly employed in data mining and time series analysis to find exact or closely matched segments of a given subsequence (a query) in a much longer sequence (a candidate) [5]. A query of length k is defined as $Q = Q_1, Q_2, ..., Q_k$ whereas a candidate of length n is defined as $C = C_1, C_2, ..., C_n$.

The matching process involves sliding Q along C in the direction of the time axis and computing a distance metric that is proportional to the dissimilarity between Q and the corresponding segment of C at time t. A commonly used distance metric is Euclidean Distance [47], which is defined as the square of the difference square-rooted,

$$D(Q, C) = \sqrt{\sum_{i=1}^{k} (Q_i - C_{i+1})^2}$$

Note that both the query and the corresponding candidate segment must be normalized to have a mean of zero and a standard deviation of one before computing Euclidean Distance to produce meaningful results [48].

Euclidean Distance is really a special case of another popular metric known as Dynamic Time Warping (DTW) [14]. DTW maps a point in the query to its closest neighbor in the candidate segment to minimize the effect of phase shifting, data misalignment, and speed difference. DTW has been applied to speech recognition [42], bioinformatics [1], and fingerprint verification [51] with much success. Using some clever tricks, the amortized complexity of DTW can be reduced from $O(n^2)$ to essentially $O(n)$ [72]. DTW is defined as

$$DTW(Q, C)_t = \sum_{i=1}^{k} min \left(\sqrt{\sum_{j=i-p}^{i+p} (Q_i - C_{t+j})^2} \right)$$

where p is known as the constrain that limits the amount of time warping a point can undergo. In our experiments, we try to compare the performance of Euclidean and DTW metrics as well as the effect of varying p.

In the field of data mining and time series analysis, there are often many query and candidate sequences of extremely long length to match, which can easily lead to an explosion in computation time. Techniques such as [17, 57, 93] have been proposed to reduce the data dimensionality. However, in the SmartFall system we have only a limited number of candidate sensor signals. Furthermore, the query sequence of interest, i.e. the actual falling motion and the episodes immediately before the fall, is relatively short in duration. As a result, we have decided to keep the full data precision without applying any dimensionality reduction.

1.3 Methodology

1.3.1 The SmartCane System

Fig. 1.2 shows the hardware of the SmartCane system. The system consists of 1) a set of low-cost sensors that output signals related to motion, force, and pressure, 2) an acquisition unit that samples the sensor signal and communicates to external devices via a wireless link, and 3) a personal device that collects and processes the data sent from the acquisition unit.

The onboard sensors of the SmartCane system include a tri-axial accelerometer, three signal-axis gyroscopes, and two pressure sensors. The gyroscopes are placed perpendicularly to each other to measure angular rate in 3D, whereas the accelerometers are mounted near the handle of the cane with a 30° slant from the direction of gravity. The two pressure sensors are fixed at the handle and the tip of the cane, measuring the grip and downward force respectively.

The acquisition unit comprises a MicroLEAP [7] processor and a Bluetooth

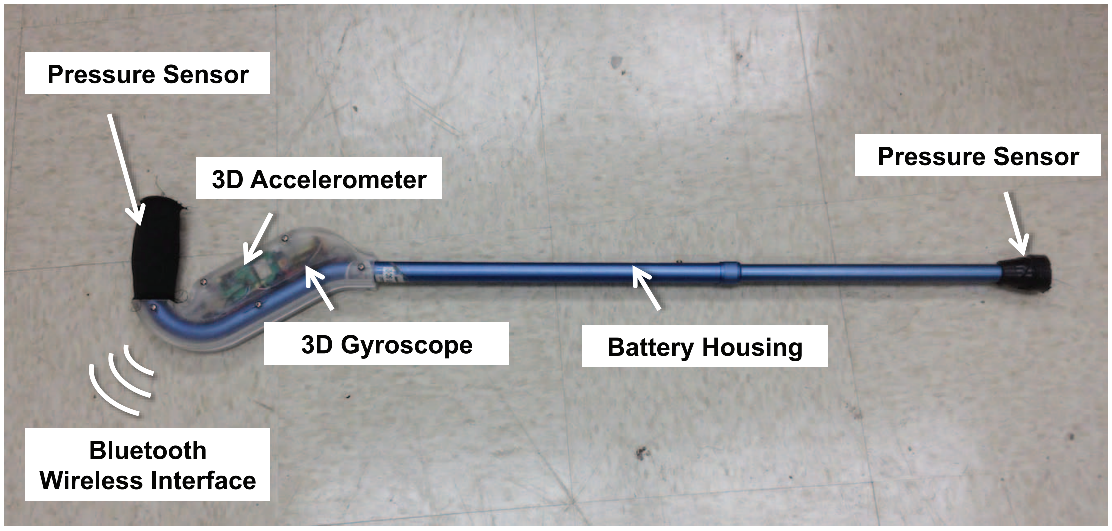


Figure 1.2: The normal acceleration n based on the model.

interface board. Each sensor input channel can be sampled at a rate of up to 300Hz. We have chosen a sampling rate of 26Hz for SmartFall, so that it is high enough to capture the normal human motions while prolonging battery life.

The personal device can be any mobile device that supports Bluetooth. Although we have chosen a tablet PC for ease of programming and data visualization, the algorithm can be easily ported to smartphones. The incoming data is received and saved to a file by a data logging daemon. The SmartFall software then reads directly from the file and performs the detection algorithm in a near real-time fashion.

1.3.2 Fall Modeling

A correct fall model is essential for an effective fall detection and cause identification algorithm. A typical fall for a cane user consists of a three-stage process: 1) collapse, 2) impact, and 3) inactivity. During the collapse stage, the user loses balance and falls towards the ground in an accelerated motion. It is assumed that the cane should experience a similar free-fall process even if the user is not applying force to the cane. We model this free-fall motion as depicted in Fig. 1.3a. In this side view, the cane starts from a near upright orientation, topples under the force of gravity, and, just before hitting the ground, changes to a horizontal orientation. The acceleration perpendicular to the cane, denoted by the vector n in the figure, is the norm of the X- and Y-acceleration of the cane (see the top view in depicted in Fig. 1.3b)). Thus, n can be calculated as $n = g\cos\theta$, where g is the gravitational acceleration, and θ is the angle between the cane and the ground ($0^\circ \leq \theta \leq 90^\circ$). Given the initial height of the accelerometers h, θ can be expressed as a function of time t

$$\theta = \arcsin\left(1 - \frac{g}{2h}t^2\right)$$

Since the accelerometers onboard the SmartCane are tilted by an angle α, the actual norm n' observed is therefore

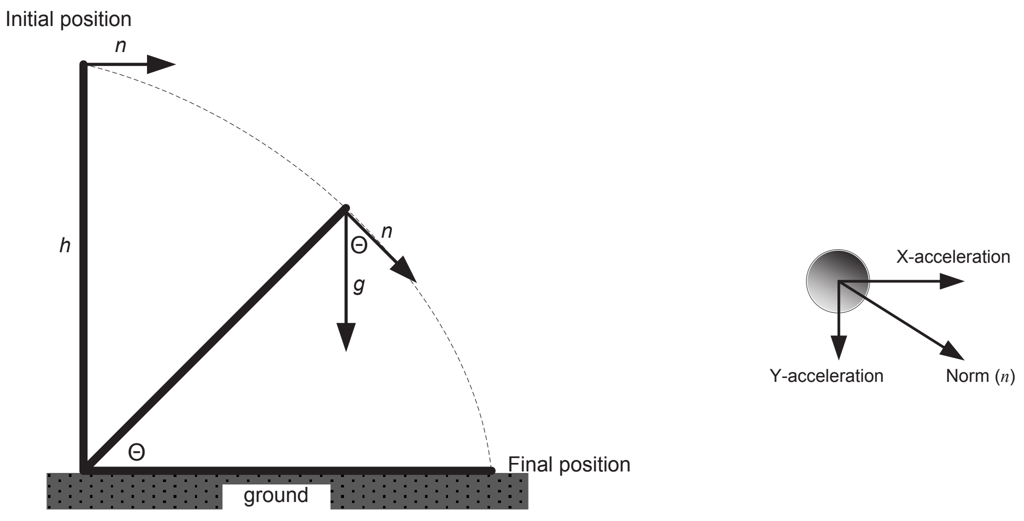
$$n' = g\cos\left(\arcsin\left(1 - \frac{g}{2h}t^2\right)\right)\cos(\alpha)$$

which can be simplified to

$$n' = g\frac{\sqrt{3}}{2}\sqrt{1 - \left(1 - \frac{g}{2h}t^2\right)^2}$$

for $\alpha = 30^\circ$.

The impact stage begins when the cane first makes contact with the ground and finishes when the cane becomes motionless. The impact exerts a counter force on the cane that results in a quick deceleration in the opposite direction of the gravity. Depending on the reaction force, the cane may bounce a few times until the energy completely dissipates. An exact mathematical model for this stage is extremely difficult to derive as the motion depends on many factors, such as the hardness of the ground, material of the cane, shape of the impacting surface, just



(a) side view (b) top view

Figure 1.3: The (a) side view and (b) top view of the typical falling motion of a cane

to name a few. Consequently, we model the impact stage in an empirical way by collecting and averaging several experiment data.

Assuming the impact has caused a serious injury, both the user and the cane should lie still on the ground for a prolonged period. If the ground surface is flat, $n' = g \cdot cos(30°)$ for the inactivity stage. This is obviously the easiest stage to model, and the duration of the stage is the only concern. We choose a period of 1 second, which is comparable to that of the impact stage. This is to ensure that the inactivity stage does not become the dominating factor in the subsequence matching process. Fig. 1.4 shows the resulting signal for n based on the model presented here, with the aforementioned three stages highlighted in the figure. The signal is sampled at 26Hz to match the real sensor signal.

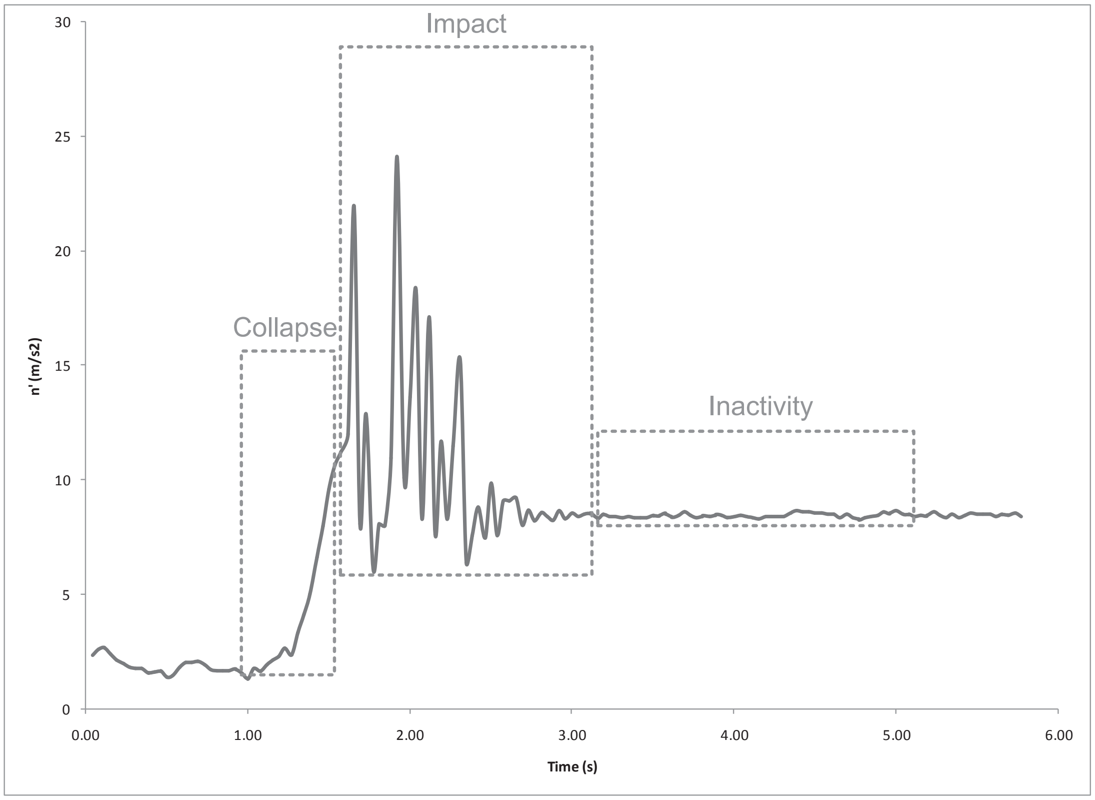


Figure 1.4: The normal acceleration n based on the model.

1.3.3 Automatic Fall Detection and Logging Algorithms

The flow of the SmartFall fall detection algorithm is depicted in Fig. 1.5. The raw input signals, namely the X- and Y-acceleration from the sensor are first calibrated before the norm is computed. This is then fed through by a signal conditioning module, followed by normalization. The normalized signal is continuously matched against a generated query signal. Based on the calculated distance, the decision module raises the fall alarm if the signals are sufficiently similar.

1.3.3.1 Calibration & Norm Computation

The X- and Y-accelerometer data, shown in Fig. 1.6a, have a constant offset that needs to be calibrated in this module. The offset values are obtained from the accelerometers reading when the SmartCane is in a steady up-right orientation. After the offsets are factored in, the normal acceleration, n (Fig. 1.6b), is calculated using the calibrated X- and Y-accelerations, ax and ay as

$$n' = \sqrt{(a_x)^2 + (a_y)^2}$$

1.3.3.2 Signal Conditioning

The calibrated normal acceleration is then passed through a 6th order biquad IIR low-pass filter with 5Hz cut-off to remove noise and high frequency components. From the filtered signal in Fig. 1.6c, it is clear that the impacting stage, being highly unpredictable and difficult to model as described earlier, has been effectively smoothened into a single peak. The peak follows closely to the envelope of the original transient signal. The filtering process has been shown experimentally to have slightly decreased the distance of genuine matches and greatly increased the distance of non-matches, which essentially improves the overall signal-to-noise ratio.

1.3.3.3 Normalization

To facilitate the subsequence matching process, the normalization module buffers the filtered signals and computes a normalized segment of the same length as the query pattern. This module basically performs the sliding action described in Section 1.2.2.

The standard normalization process involves mean shifting, i.e. subtracting the mean from all values, and autoscaling, i.e. dividing all values by the standard deviation. However, mean shifting is not necessary in this case because the input signal has already been calibrated for the particular SmartCane in the first module. In fact, mean shifting may even have an adverse effect on the end result, since the mean of the short segment buffered by the normalization module is not a true representative of the mean for the overall input signal. Consequently, we decide to only perform autoscaling with a mean value of zero for the normalization process.

1.3.3.4 Query Generation

The query is generated statistically based on the model presented in Section 1.3.2. To facilitate a meaningful comparison with the candidate, the query needs to go through the same filtering and normalization process.

1.3.3.5 Subsequence Matching & Decision

The subsequence matching module continuously outputs the Euclidean Distance between the incoming signal and the matching query. An example of the distance computed from this module is plotted in Fig. 1.6d. Typically the distance dips quickly towards zero when a matching incoming signal and the query start to overlap, reaches a local minimum when the two perfectly lined up, and climbs up

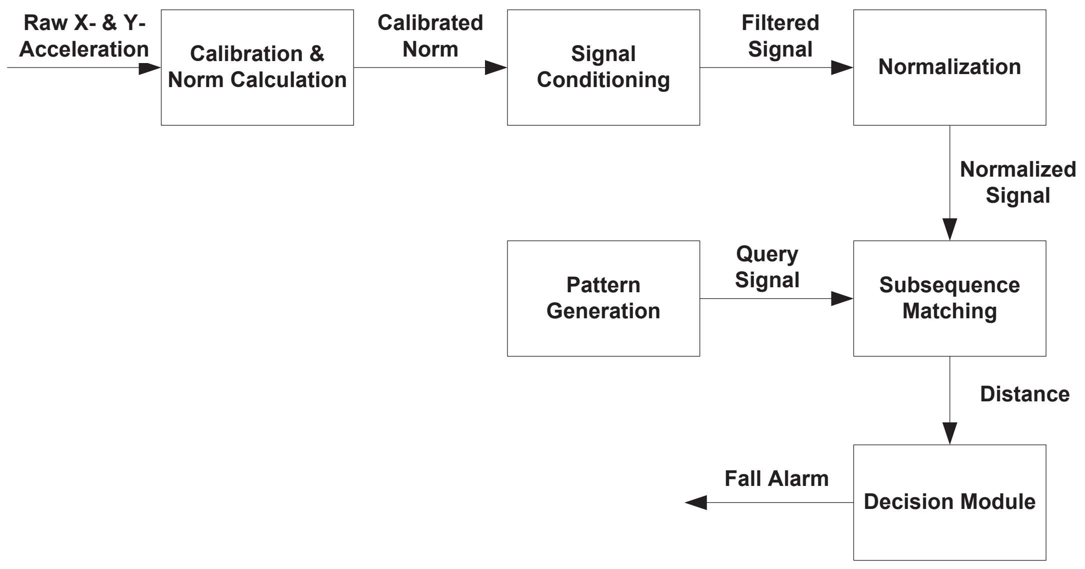


Figure 1.5: The flow of the SmartFall detection algorithm

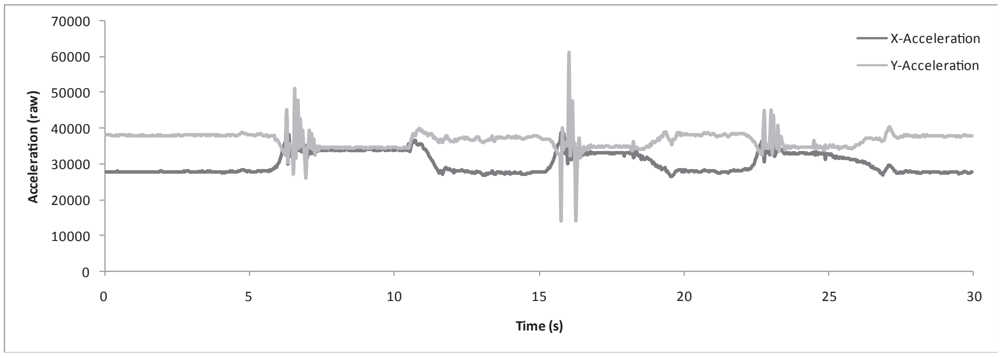
again as the two signals slide pass each other. The resulting trough is often fairly consistent and salient as can be seen from Fig. 1.6d.

The decision module is in charge of raising the fall alarm when distance value computed from the subsequence matching module falls below a certain level. The cut-off level is represented by the light-gray line in Fig. 1.6d

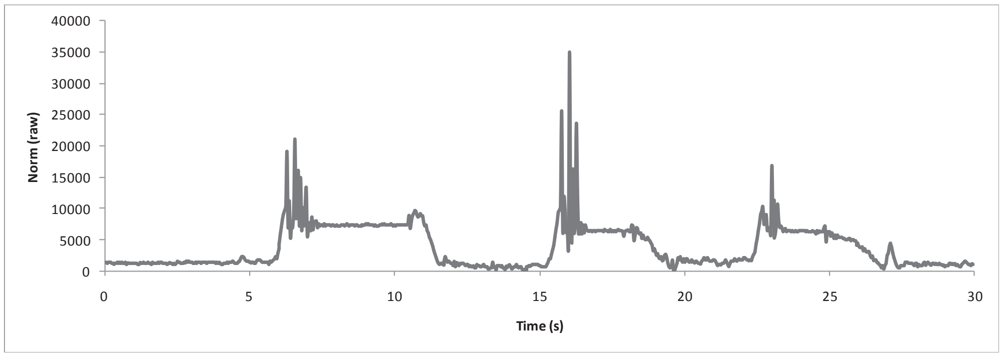
1.3.4 Cause Identification System

To identify the cause of a fall, we assume that there exists, or lacks, distinct events before and after a fall. These events could potentially be responsible for the fall or consequences of the fall. For example, if a person falls after tripping over an obstacle, the SmartCane is likely to experience more violent movements during the process. On the other hand, if a person falls due to sudden dizziness, the SmartCane is likely to move in a more controlled fashion. Fig 1.7 gives an example where such difference can be observed.

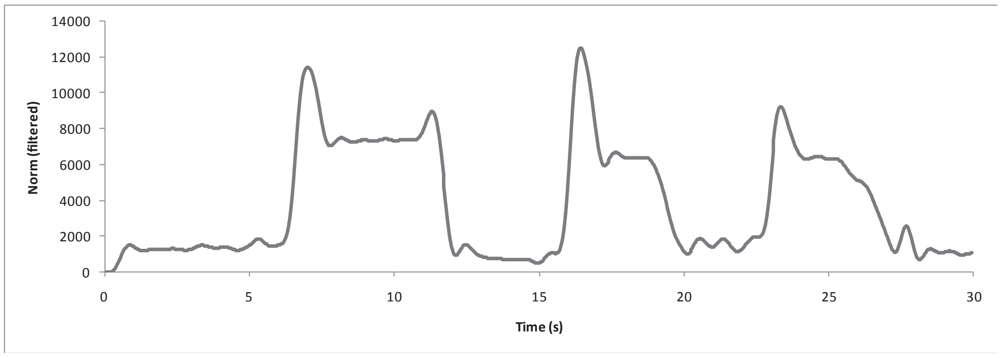
As a result, when a fall is detected using the algorithm described in 1.3.3,



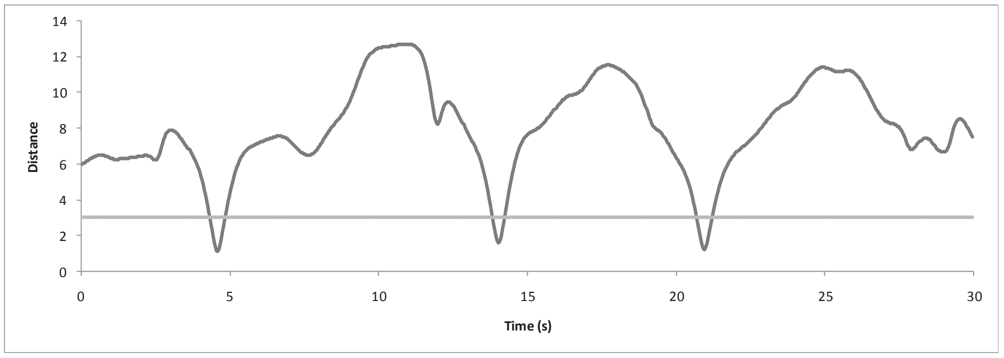
(a)



(b)



(c)



(d)

Figure 1.6: Signal output at various stages: (a) raw X-acceleration & Y-acceleration (b) Calibrated normal acceleration (c) Filtered normal acceleration (d) Computed Euclidian distance and cut-off level.

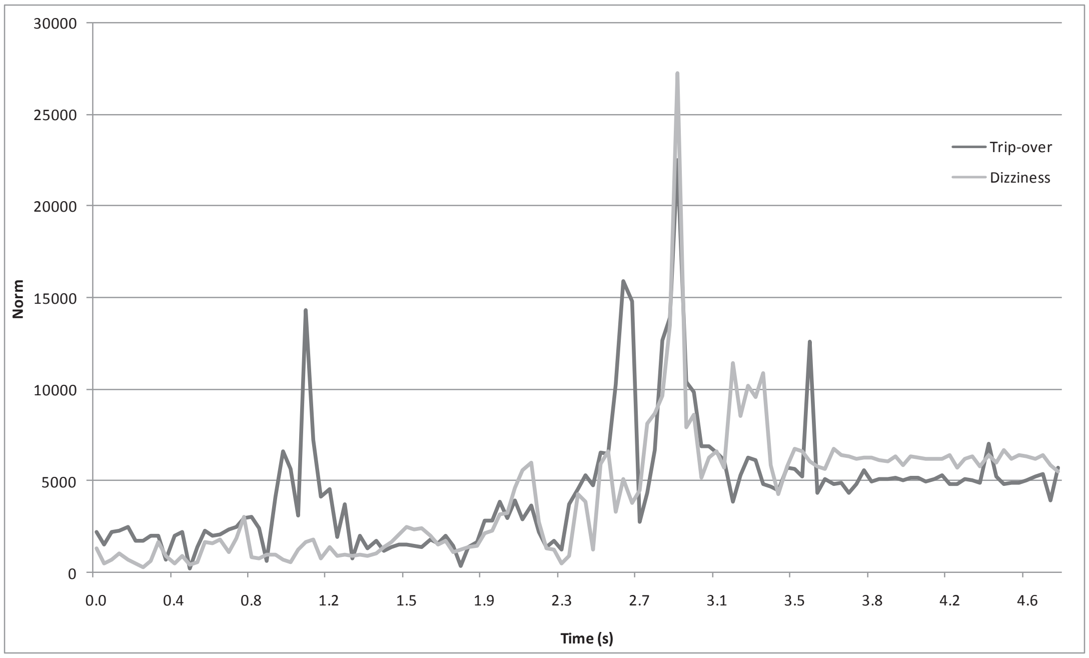


Figure 1.7: Acceleration norms captured using SmartCane for falls due to tripping over and dizziness. The Trip-over signal is more volatile compared to Dizziness before the point of impact at $t = 2.9$.

SmartFall logs and stores 2 seconds of accelerometer signals both before and after the fall. The likely cause of the fall is then determined by comparing the data against an existing database of known patterns.

1.4 Case Study

1.4.1 Experiment Setup

We have selected 10 healthy test subjects, 5 males and 5 females, to perform the experiments. Their ages ranged from 22 to 28, height ranged from 155 to 178cm, and weighed 48 to 72kg. While these subjects probably have low risk of falling due to their age and fitness, conducting similar experiments on elderly people can be potentially hazardous. Furthermore, the subjects were so immersed in the experiments that they often ended up actually falling in an uncontrolled fashion.

A falling platform has been set up for the experiment. The platform is made up of a soft cushion for the subject to fall on, and a hard surface for the cane to hit, simulating what often happens in real life. The sensor data are transmitted via Bluetooth and recorded on a tablet PC. The recoding is not interrupted throughout the process even when the subject completes the instructed action and returns to the starting position.

1.4.2 Fall Detection Cases

Four types of falls have been experimented to gauge the fall detection rate of SmartFall and are listed in Table 1.1. Each type of fall is performed 30 times, and the subjects are allowed to use their hands to brace against the ground when falling. The subjects can also choose to have a firm grip of the cane throughout the falling process or let go the cane at any time. The results include a fair mix of various situations.

Furthermore, it is critical to discriminate fall from ordinary daily activities. Frequent false alarms can seriously undermine the users willingness to adopt the system. The discrimination power of SmartFall is evaluated using 5 different activities, as listed in Table 1.2, made up of commonly performed actions with a cane. Each activity is performed for 30 complete cycles by the test subjects.

1.4.3 Cause Identification Cases

The cause of falls can be categorized into intrinsic and extrinsic [91]. Intrinsic cause of falls includes dizziness, loss of balance, blackouts, and medical episodes. On the other hand, trips, slips, and other environmentally triggered factors are common extrinsic causes.

In our experiments, each subject is asked to simulate falling scenarios that are

Table 1.1: Types of Falls

Activity	Description
Forward	Simulates a faced down fall due to trip over
Backward	Simulates fall on the back or bottom due to a slip
Side	Simulates a sideward fall due to loss of balance
Free Fall	Simulates an unobstructed topple of the cane due to loss of grip

Table 1.2: Types of Daily Activities

Activity	Description
Slow Walk	Walking with the cane at a pace ¡ 1 step/second
Fast Walk	Walking with the cane at a pace 2 steps/second
Sit & Stand	Standing up with the help of the cane from a sitting position
Swing	Swinging the cane back-and-forth at around 1Hz with a angle less than $45^{c}irc$ from the vertical axis
Lay on Lap	Picking up the vertically oriented cane and laying it flat on the lap while seated

Table 1.3: Fall Scenarios

Activity	Description
Trip	Simulate a person getting tripped over by an obstacle. This often results in forward or sideway tumbles.
Slip	Simulate a person slipping on his/her feet and falls backward or sideway.
Dizziness	Simulate a person collapsing due to dizziness or blackout. The dizziness is induced deliberately by spinning the subject around.
Loss of Balance	Simulate a person falling due to loss of balance or sudden weakness in their lower extremity.

caused by four different reasons, two extrinsic (*Trip* and *Slip*) and two intrinsic (*Dizziness* and *Loss of Balance*). Each scenario is repeated 10 times, giving a total of 40 falls per subject. Although subjects are given some general descriptions for each scenario as listed in Table 1.3, they are not required to follow specific instructions to fall. Although this freedom produces results that are closer to real life than in a controlled environment, it also presents significant challenges to the identification algorithm.

Table 1.4: Fall Detection Rate For Different Types Of Falls

Subject	*Forward*	*Backward*	*Side*	*Free Fall*
1	100%	100%	100%	100%
2	100%	100%	96.7%	100%
3	100%	93.3%	100%	100%
4	100%	100%	100%	96.7%
5	100%	100%	93.3%	100%
6	100%	96.7%	100%	100%
7	100%	100%	100%	100%
8	96.7%	93.3%	100%	100%
9	100%	100%	96.7%	100%
10	99.7%	98.3%	98.7%	99.7%
Average	100%	100%	100%	100%

1.5 Results and Discussions

1.5.1 Fall Detection

The fall detection rate and discrimination power of the SmartFall system have been evaluated. The results are shown in Table 1.4 and 1.5 respectively.

The results indicate that SmartFall achieved a near 100% detection rate for all four types of fall performed by the three subjects. The difference in weight and height between subjects appears to have little effect on the end results. As for the false-positive, SmartFall is able to discern daily activities from a genuine fall with a near 0% false alarm in most cases. Even for *Lay on Lap*, the most similar activity to fall, SmartFall is able to attain a low average false positive rate of 2.3%.

Table 1.5: False Positive Rate For Daily Activities

Subject	*Slow Walk*	*Fast Walk*	*Sit & Stand*	*Swing*	*Lay on Lap*
1	0%	0%	0%	0%	10%
2	0%	0%	0%	0%	3.3%
3	0%	0%	0%	0%	0%
4	3.3%	0%	0%	0%	0%
5	0%	0%	0%	0%	3.3%
6	0%	0%	0%	0%	0%
7	0%	0%	0%	0%	0%
8	0%	0%	3.3%	0%	0%
9	0%	0%	0%	0%	0%
10	3.3%	0%	0%	0%	6.6%
Average	0.7%	0%	0.3%	0%	2.3%

1.5.2 Cause Identification

The accuracy of the cause identification algorithm is evaluated using the 3-nearest-neighbor classification and leave-one-out cross-validation. In other words, the class of a pattern is determined by its three closest matches in the remaining patterns in the original sample. After applying the process to all patterns in the sample, the classification accuracy is then computed based on the percentage of correctly classified patterns.

There are several distance metrics that can be used when finding the closest match in the sample. As mentioned previously, we focus on two widely accepted metrics, Euclidean Distance and DTW. We are also interested in how the amount of constrains applied in DTW may affect the classification accuracy. Therefore, each validation is computed three times using 1) Euclidean Distance, 2) DTW with 2% constrain, and 3) DTW with 10% constrain.

As the number of classes increases, the classification accuracy often decreases significantly, especially if the classes are similar. In our case, if we consider the four experiment scenarios as independent classes, many *Dizziness* cases may be wrongly classified as *Loss of Balance* and vice versa due to their similarity in nature. It therefore makes sense to convert the classification into a two-class problem—an Extrinsic class made up of *Tip* and *Slip* versus an Intrinsic class made up of *Dizziness* and *Loss of Balance*. However, for completeness, we present the results for both four- and two-class versions of the classification.

1.5.2.1 Individual Results

The sample is first restricted to the fall patterns generated by individual test subject. This means each fall pattern is compared against the other 39 fall patterns from the same test subject. The average accuracy is calculated by aggregating the validation of all 10 subjects and is plotted in Fig. 1.8a and 1.8b for the four-class and two-class versions respectively.

When classifying into four classes, the accuracy ranges from 62% to 79%, depending on the particular class and metric used. While this may not seem highly accurate, it is already markedly better than a 25% random guess. The major source of classification error is between similar movements, such as *Trip*/*Slip* and *Dizziness*/*Loss of Balance*. By grouping these movements together, the accuracy increases to 83-93% as shown in Fig. 1.8b.

Based on the results it is clear that DTW produces slightly higher accuracy (less than 10%) than Euclidean Distance for the intrinsic type of falls. On the other hand, Euclidean Distance appears to be a more accurate metric for extrinsic type of falls. This is probably due to the fact that extrinsic type of fall often happens unexpectedly and results in a more volatile signal. However, when comparing to a less volatile signal generated from intrinsic type of falls using DTW, the volatility is essentially ignored as the optimally matched point is selected from the warping window.

Our results reaffirm the finding in [72], where the author claimed that the conventional wisdom of 10% constrain for DTW is actually too wide for many real life applications. In our case, a 2% constrain produces the optimal accuracy for intrinsic type of falls and the accuracy degrades gradually with increasing constrain size.

1.5.2.2 Group Results

The sample is increased to cover fall patterns from all subjects. In this case, each fall pattern is matched against 390 patterns generated by other subjects. It is initially assumed that the accuracy is expected to drop with the larger sample and the difference between individual test subjects. However, to our pleasant surprise, the average accuracy remains more or less unchanged for both four- and two-class classification (see Fig. 1.8c and 1.8d). This suggests that there is strong correlation among fall patterns from different individuals.

The similar problem of DTW underperforming Euclidean Distance for extrinsic type of falls observed for individualized results also occurs here, and the same explanation can be given as well. Albeit, the effect appears to be less pronounced with a larger sample size. Once again, DTW with 2% constrain works well with intrinsic type of falls while larger constrain reduces the accuracy.

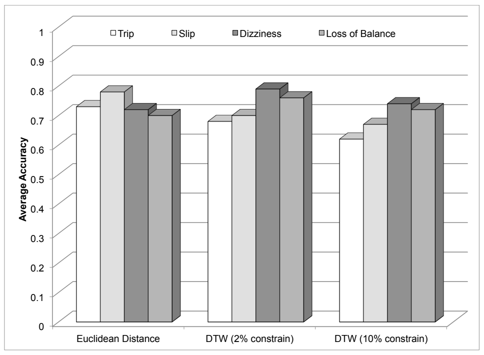
1.6 Summary

In this chapter, we present SmartFall, the first automatic fall detection and cause identification system based on subsequence matching. SmartFall uses data from the accelerometers embedded closely to the handle of the SmartCane to make inferences of current status. The detection algorithm differs fundamentally from most existing thresholding-based fall detection solutions in that the overall shape of the sensor signal is considered. When the shape matches the signature of a typical fall pattern, the alarm is raised and the signal immediately before and after the fall is recorded for further cause analysis.

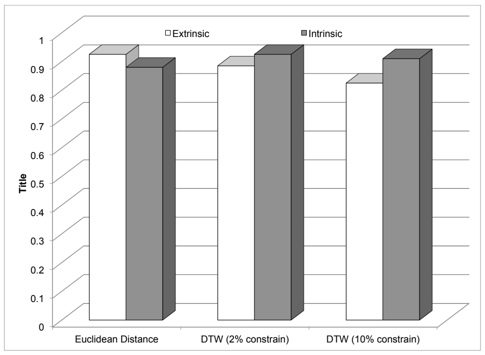
Several experiments simulating various types of fall and other common daily activities have been conducted to evaluate the fall detection performance of SmartFall. The results have indicated that SmartFall is able to detect nearly all cases of falling in the experiment while achieving extremely low false-positive rates for most non-falling activities.

Similar experiments, which encompass four scenarios, trip, slip, dizziness, and loss of balance, have also been conducted to measure the accuracy of SmartFall's cause identification algorithm. A classification accuracy of 62-79% has been achieved when four scenarios are considered as independent classes and the sample is restricted to individual test subjects. The accuracy is improved to 83-93% when scenarios are grouped into intrinsic and extrinsic classes. We have also shown that DTW is more accurate than Euclidean Distance for intrinsic type of falls but less suited for extrinsic ones. Furthermore, applying a 2% constrain to DTW, instead of the commonly used 10% constrain, gives a noticeable improvement in accuracy for this particular application. Experimental results also suggest that the fall patterns generated by different test subjects are strongly correlated. This allows us to reliability classification the cause of a persons fall based on existing fall patterns generated from other subjects.

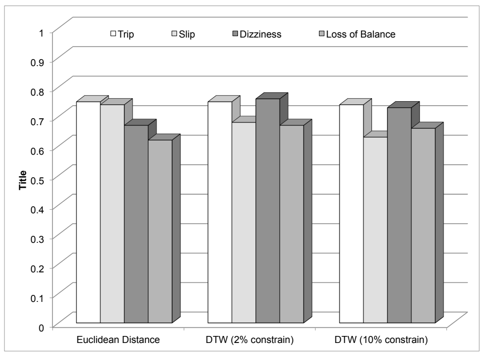
As a future extension, it may be desirable to detect the cases where the patients are not actually falling, but feel like falling or simply not feeling well. These symptoms may introduce enough of changes to the normal cane usage pattern such that the motion can be picked up as abnormality. When these abnormalities are detected, the caregivers can be alerted so immediate actions can be taken.



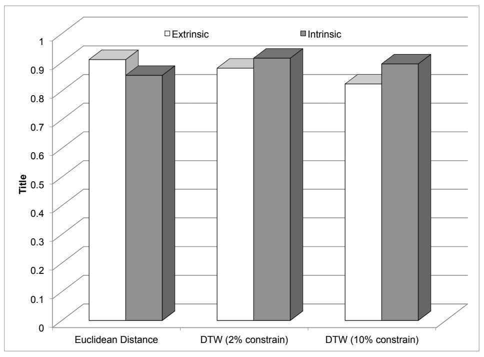
(a)



(b)



(c)



(d)

Figure 1.8: Average cause identification accuracy for (a) individual results using four classes (b) individual results using two classes (c) group results using four classes (d) group results using two classes.

CHAPTER 2

Sequential Patterns Mining

Remote health monitoring often produces long sequence of data. Events can often be predicted by discovering rules that separate one class of sequences from another. Contrasting patterns, such as Minimal Distinguishing Subsequence, have been shown to be a succinct way to differentiate datasets and often outperform other types of statistical classifiers. In this chapter, we introduce a new algorithm called Fast MDS Minter that efficiently mines MDS from large datasets. FMM employs a support counting mechanism that is scalable and memory efficient. It also explores the search space in a breadth-first fashion, resulting in a more powerful pruning strategy and a simpler on-the-fly minimization process.

Experimental results based on four large, publicly available datasets have clearly indicated that FMM outperforms the state of the art MDS mining algorithm, ConSGapMiner, in terms of both speed and memory usage. Overall, FMM is about 1.9 to 7.7 times faster than ConSGapMiner under a variety of settings. This is largely owing to the FMM's fast support counting mechanism that scales exceptionally well with gap size and the number of sequences, as well as the max-suffix based pruning strategy that removes up to 82% of the unpromising candidates generated by ConSGapMiner. Results also confirm that the peak memory usage of FMM is insensitive to the number of sequences in the dataset, enabling it to mine much larger datasets.

2.1 Background

Sequence serves as a fundamental abstraction for more complicated problems in many scientific fields. For example, protein and DNA are often represented as long string [33], whereas retail transactions and network activity logs are frequently reduced into sequence of items or events [3, 82]. There are also many sequence representation developed for high dimensional data, such as time series and video [58, 69].

Once the data have been converted into sets of sequences, a typical machine learning and data mining task is to infer rules that separate one class of sequences from another. As these rules can be discovered by means of contrasting, many data mining algorithms have been developed based on this principle (see [68] for a comprehensive review). It has also been shown that classifiers built on these contrasting patterns often achieve higher accuracy and are less susceptible to overfitting than other statistical methods, such as C4.5 and CBA [27]. In this paper, we are particularly interested in mining the Minimal Distinguishing Subsequence (MDS) patterns, which are essentially patterns that occur frequently in one group of sequences (known as *positive examples*), but occur rarely in another (known as *negative examples*).

The problem of MDS mining is first introduced by Ji et. al. in [45]. This type of problem is proven to be NP-complete [38] and can thus only be realistically solved using heuristics. Ji et. al. propose an algorithm called ConSGapMiner that consists of three stages: a) candidate generation, b) support and gap calculation, and c) minimization. The algorithm can handle user-specified maximum gap size when verifying the subsequence-supersequence relationship. They have also demonstrated the performance of ConSGapMiner using protein datasets and text from the Bible books.

Allowing gaps in MDS mining is a desirable feature for analyzing purchase history, website logs, and biosequences [18]. However, doing so makes the problem much more challenging at the same time. It has been pointed out by Zaki that the maximum gap constrain is not class-preserving [97]. This means that the support of k-item sequence cannot be directly inferred from $(k-1)$-item sequences. Consequently, the well known Apriori property is no longer applicable in this case [3], ruling out a large number of existing algorithms.

ConSGapMiner appears to be the only promising algorithm that mines MDS efficiently to date. Indeed, to our best knowledge there has been no major improvement or better alternative proposed since 2005. Nevertheless, ConSGapMiner is still limited to relatively small datasets in practice due to its poor scalability. Firstly, the bitset-based subsequence testing procedure employed by ConSGapMiner requires space that grows linearly with the size of the dataset. At the same time, performance of subsequence testing degrades dramatically at larger gap sizes. Secondly, ConSGapMiner prunes the candidate search space in a suboptimal way, causing it to execute a large number of costly but unnecessary support countings. Finally, the minimization process of ConSGapMiner relies on a special prefix-tree data structure, which uses additional memory and introduces extraneous complexity to the algorithm.

2.2 Related Work

Since the introduction of sequential pattern mining in [4], many algorithms have been developed to efficiently mine frequent sequential patterns (see [37] for a detail review). While MDS mining can be seen as a special application of sequential pattern mining, the problem is significantly challenging as the Apriori property is no longer valid when mining MDS.

Similar to MDS mining, emerging patterns [28] and contrast-sets [13] minings try to identify salient differences between two or multiple datasets. It has been shown that such difference can often be used to construct highly accurate classification models [54]. However, the fundamental difference between itemset and sequence prevents the various techniques developed for emerging patterns and contrast set, e.g. [11, 55, 99], to be applied to MDS mining.

Chan et. al broaden the definition of emerging patterns and introduce the emerging substrings mining problem in [18], along with a suffix tree-based mining framework and three pruning strategies. Since substring is just a special case of subsequence, FMM can be easily tailored to mine emerging strings by setting the maximum gap size to zero. However, Chan's suffix tree-based algorithm cannot be easily modified to mine subsequences.

Zaıane et. al. introduce the concept of Emerging Sequence (ES) , which can also be used to contrast groups of sequences [96]. They propose an algorithm to discover top-n ES based on heuristics. Different from emerging patterns and distinguishing sequence, ES uses distance comparison instead of frequency count as the contrasting basis. Strictly speaking, even if a subsequence occurs rarely in the positive group, as long as it is sufficiently different from its nearest neighbor in the negative group, it is considered to be an ES. The definition has been adopted by [56] and applied to time series and multimedia data.

In a series of papers, Takeda et. al. present several algorithms to find the best subsequence [38], episode [78], Variable Length Don't Care (VLDC) [41], and Fixed/Variable Length Don't Care (FVLDC) [87] patterns that separate two given set of strings. Although these problems are closely related to MDS mining, finding a single best pattern up to a certain length is arguably an easier problem than finding all patterns that satisfies some fitness measurement without any limitation on length. Furthermore, the proposed algorithms search for subsequence in the most general sense—effectively allow a gap of any size. This makes them unsuitable for mining MDS with specific gap constrain.

2.3 Preliminaries

Let I be a finite set of distinct items. I is called the alphabet, and its cardinality is denoted as $|I|$. A sequence S over I is an ordered list of items, $s_1, s_2, ..., s_n$, where $s_i \in I$ for $1 \leq i \leq n$. The length of S is denoted as $|S|$, and the i-th element of S, namely s_i, is denoted as $S_{[i]}$. Although it is possible to have $S_{[i]}$ containing multiple items, we consider only the case where $S_{[i]}$ consists of exactly one item.

A dataset, d, is a collection of one or more sequences of arbitrary length. Duplicated sequences are allowed to coexist in the same dataset. $|d|$ is the size of the dataset, which is the number of sequences in d.

A sequence T is a subsequence of S, written as $T \subseteq S$, if there exists a series $1 \leq i_1 < i_2 < ... < i_m \leq n$ such that $T = S_{[i_1]}, S_{[i_2]}, ..., S_{[i_m]}$. S is called a supersequence of T in this case. The maximum gap size for T is defined as $\max(i_{j+1} - i_j)\ \forall 1 \leq j < m, j \in \mathbb{R}$ and the minimum gap size is defined as $\min(i_{j+1} - i_j)\ \forall 1 \leq j < m, j \in \mathbb{R}$. If both the maximum and minimum gap sizes are zero, T is said to be a substring of S.

Definition 1 *A gap constrain $\{g_{min}, g_{max}\}$ is a limit such that T is a subsequence of S only if the minimum gap size $\geq g_{min}$ and maximum gap size $\leq g_{max}$.*

For example, with a gap constrain $\{1, 2\}$, sequence AB is a subsequence of ACB but is not a subsequence of $ACDEB$.

Definition 2 *The support of a sequence S in dataset d, denoted as $supp_d$, is the ratio of the number of sequences in d that are supersequences of S to $|d|$.*

In other words, a support of 1 means that S is a subsequence of every sequences in d, whereas a support of 0 means that none of the sequences in d is a supersequence of S. Note that the notion of subsequence can be subjected to certain gap constrains $\{g_{min}, g_{max}\}$.

Definition 3 *Given two threshold values, δ and α, two datasets q and r, and a gap constrain $\{g_{min}, g_{max}\}$, a sequence S is called a distinguishing subsequence, or Semi-Minimal Distinguishing Subsequence (SMDS) based on the definition in [44], if S has $supp_q \geq \delta$ and $supp_r \leq \alpha$.*

Definition 4 *S is a Minimal Distinguishing Subsequence (MDS) if S is a SMDS, and there exists no other SMDS T such that $T \subseteq S$.*

With these preliminaries, the problem of MDS mining can therefore be defined as follows.

Definition 5 *Given two datasets, pos (positive examples) and neg (negative examples), and some user-specified parameters δ, α, g_{min}, and g_{max}, find the complete set of MDS that satisfies these constrains.*

In addition, we introduce several terms to facilitate the discussion in the reminder of the paper.

Definition 6 *The max-prefix of a sequence S of length n is $S_{[1]}, S_{[2]}, ..., S_{[n-1]}$, whereas the max-suffix of S is $S_{[2]}, S_{[2]}, ..., S_{[n]}$.*

Definition 7 *Sequence T is called the younger twin of S if $T = S_{[1]}, S_{[2]}, ..., S_{[n-1]}, x, S_{[n]}$, where $x \in I$. S is called the elder twin of T in this case.*

2.3.1 Example

We use a simple example to demonstrate the various aspects of MDS mining. Given the alphabet $\{A, B, C\}$ and two sets of sequences

$$pos : (ABAC, BAAC, BCA, BC, ABC)$$
$$neg : (AB, ABB)$$

and the parameters are set to $\delta = 0.8$, $\alpha = 0$, $g_{min} = 0$, $g_{max} = 1$, a sequence is distinguishing if it is a subsequence of at least 4 sequences from *pos* and at most zero sequence from *neg*. Based on inspection, both C and BC have a $supp_{pos}$ of 1.0 and $supp_{neg}$ of 0 and are therefore distinguishing. In other words, C and BC are SMDS. However, only C is considered a MDS as BC is simply a supersequence of C. Note that C is a max-suffix of BC.

If g_{max} were to be increased to 2, BC is no longer a subsequence of $BAAC$, which decreases its $supp_{pos}$ to 0.8. By the same token, if g_{min} were set to 1, none of the sequences in *pos*, except $BAAC$, is a supersequence of BC any more. This means that BC is no longer distinguishing. C is nonetheless unaffected in either case.

From the *pos* set, $ABAC$ is a younger twin of ABC as the two are identical if the former has its second to last item (A) removed. Conversely, ABC is called an elder twin of $ABAC$. From the *neg* set, AB is a max-prefix of ABB because the latter is simply an one-item extension of the former.

2.4 Fast MDS Miner

Our MDS mining algorithm is made up of three components: support counting, candidate generation, and minimization. While the names are similar to those used in ConSGapMiner, each component differs substantially and is designed to be faster and more memory efficient. The following sections describe each component in details.

2.4.1 Support Counting

Support is counted twice, one for *pos* and one for *neg*, for each candidate sequence generated. This can easily become the most time-consuming part of the mining process as the number of sequences in *pos* and *neg* increases. In fact, even with a relatively small number of sequences, support counting quickly takes up the majority of the time as shown in Figure 2.1. As a result, it is paramount to optimize support counting to the fullest extent in order to make the mining process scalable. This essentially comes down to speeding up the subsequence testing procedure.

A naïve subsequence testing procedure loops through the *target sequence* one item at a time and see if a supersequence of the *test sequence* can be formed starting from the current item until the end of the sequence. Such an algorithm exhibits a worst case complexity of $O(n^2)$, where n is the length of the *target sequence*. Let m be the length of the *target sequence*, adding the gap constrain $\{g_{min}, g_{max}\}$ can reduce the complexity to $O((m + g_{max}) \times n)$ assuming $(m-1) \times g_{max} \ll n$.

A bitset-based approach is proposed in [45] in order to decrease the time spent in subsequence testing by maintaining multiple bitset structure for each *target sequence*. It relies on the fact that most computers can handle boolean operations, such as logical OR and bit-shifting, between long string of bits efficiently. One of the apparent downsides of such an approach is the amount of space required to store these bitsets, which is $O(l \times |I| \times L \times N)$ in the worst case, where l is length of the longest *test sequence*; I is the alphabet; L is the average length of *target sequence*; and N is the combined number of sequences in *pos* and *neg*. Moreover, with every new candidate generated, each bitset needs to be shifted right $g_{max} + 1$ times before OR-ing with the intermediate bitset. This operation becomes prohibitively expensive with large g_{max} and therefore limits the scalability of the algorithm.

Another possible approach for fast subsequence testing is to make use of a pre-constructed automaton, or the so-called Direct Acyclic Subsequence Graph (DASG) in [9], for each *target sequences*. This allows a $O(m \log |I|)$ subsequence query time, where m is the length of the *test sequence* and I is the alphabet, at the expense of $O(n^2)$ space complexity. However, since the gap constrain is not class-preserving, DASG needs to be modified to support multiple start states. This will essentially increases the query time to $O(mn \log |l|)$.

In order to balance the performance and space usage, we introduce a new subsequence testing procedure called FastTest. The implementation for FastTest is detailed in Algorithm 1. Similar to the naïve algorithm, there is no need to pre-compute space-consuming data structure for each *targte sequence*. However, FastTest performs multiple subsequence testings simultaneously as it iterates through each item in the *target sequence* (line 9-17). This results in a worst cast time complexity of $O(nk)$ where n is the length of the *target sequence*, and k is the maximum number of prefixes being compared ($|C|$ in Algorithm 1). Note that k is bounded by $min(m, g_{max})$, where m is the length of the *test sequence*. Since g_{max} is relatively small in most cases, the time complexity is approximately $O(n)$. Finally, based on the FastTest procedure, the support counting component becomes a simple loop as shown in Algorithm 2.

2.4.2 Candidate Generation & Pruning

Different from ConSGapMiner, FMM generates candidate sequences in a breadth-first search (BFS) fashion. In other words, we start from a single-item sequence in lexicographical order, for example $\{A\}, \{B\}, \{C\}$, etc. Each sequence is then appended with one item chosen from the alphabet in lexicographical order until all two-item sequences have been exhausted. The process then repeats for all three-item sequences, four-item sequences, and so on and so forth. This forms a tree-shaped search space similar to the one shown in Figure 2.2. However, the combinatorial nature quickly makes the testing of every possible sequences intractable even with a relatively short length. Consequently, we need to introduce some effective pruning strategies to limit the growth of the search space.

Algorithm 1 FastTest: Test if s is a subsequence of u under the gap constrain $\{g_{min}, g_{max}\}$

```
Input  $s, u, g_{max}, g_{min}$
Outpu  true if $s$ is a subsequence of $u$, false otherwise
 1: if $|s| > |u|$ then
 2:     return false
 3: $C = \phi$  // A set of tuple $(p, g)$ where $p$ is the current comparison position
    and $g$ is the current gap size
 4: for $i = 1$ to $|u|$ step 1 do
 5:     if $u_{[i]} = s_{[1]}$ then
 6:         // New starting position with zero gap size
 7:         $C = C \cup (0, 0)$
 8:     $T = \phi$  // Temporary set holding qualified tuples
 9:     for all $(p, g) \in C$ do
10:         if $u_{[i]} = s_{[p]}$ then
11:             // Move to next position on match
12:             $g = 0, p = p + 1$
13:             if $p = |s|$ then
14:                 return true
15:         else
16:             // Increase gap size on mismatch
17:             $g = g + 1$
18:             if $g < g_{min}$ or $g > g_{max}$ then
19:                 // Drop unqualified tuple
20:                 continue
21:         $T = T \cup (p, g)$
22:     $C = T$
23: return false
```

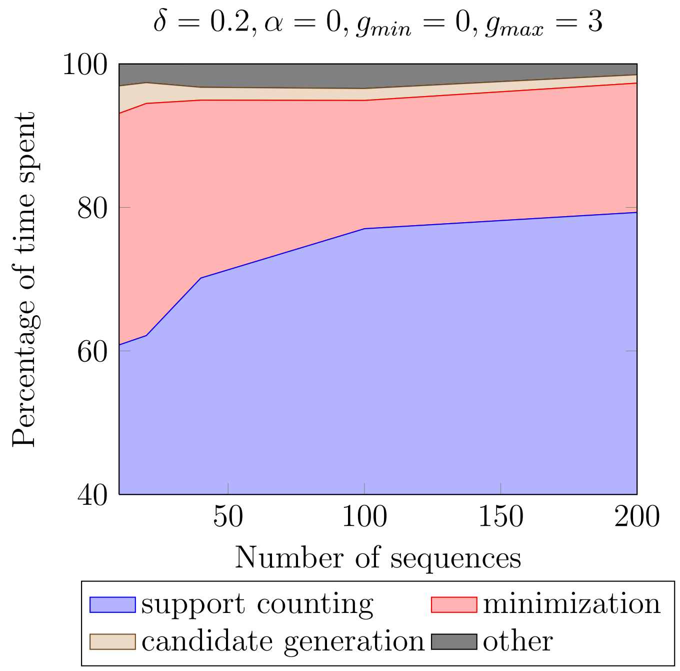


Figure 2.1: Percentage of time spent on each part of the MDS mining process for different numbers of sequences from pfam2 dataset.

Algorithm 2 CountSupport: Compute the ratio of strings in U that support s under the gap constrain $\{g_{min}, g_{max}\}$

Input s, U, g_{max}, g_{min}

Outpu support for s

```
1: $c = 0$
2: for all $u \in U$ do
3:     if FastTest$(s, u)$ then
4:         $c = c + 1$
5: return $c/|U|$
```

Two pruning strategies, Non-Minimal Distinguishing Pruning and Max-Prefix Infrequency Pruning have been proposed in [45]. The former is based on the observation that if a sequence is distinguishing, any of its supersequences formed by appending items cannot be minimal. This can also be applied to the *younger twins* of any distinguishing subsequences to further reduce the search space. Max-Prefix Infrequency Pruning takes advantage that when the $supp_{pos}$ of a candidate is less than δ, not only is the sequence itself infrequent, but any direct children of the sequence are also infrequent by definition. As a result no further exploration stemming from the sequence is necessary. It is clear that when both pruning strategies are applied, the only thing remaining in the search space at any given moment are candidates that have $supp_{pos} \geq \delta$ and $supp_{neg} \geq \alpha$. We call these sequences Potential Prefixes (PP).

FMM also incorporates both pruning strategies. However, thanks to the BFS style of candidate generation, we are able to expand the Non-Minimal Distinguishing Pruning strategy to make it even more powerful. This results in the algorithm listed in Algorithm 3. When a new candidate c, where $|c| > 1$, is generated, its max-suffix m, or any direct parents of m, must have been previously visited. Therefore, if m is not a PP, m or any direct parents of m must be either distinguishing or infrequent, which is a sufficient criterion to preclude c from the MDS set. In Section 2.5.2, we will demonstrate the effectiveness of this strategy at pruning the search space and reducing the number of support counting performed.

However, in order to quickly test if m is a PP, all PP discovered thus far needs to be stored in a hash-table. This can potentially consume a considerable amount of space as we explore deeper into the search space. Fortunately, in reality we only need PP of length n when generating $(n + 1)$-item candidates. Hence, the size of the hash table can be roughly cut in half by maintaining only the PP that is one item shorter than the current candidate.

Algorithm 3 MineSMDS: Find a set of SMDS over alphabet I from positive dataset *pos* (with maximal support δ) and negative dataset *neg* (with minimal support α) under the gap constrain $\{g_{min}, g_{max}\}$

Input $I, pos, neg, \delta, \alpha, g_{min}, g_{max}$

Outpu a set of SMDS

```
 1: $Q$: Empty queue
 2: $V = \phi$   // set of PP
 3: $C = \phi$   // set of SMDS
 4: while $Q$.size() $> 0$ do
 5:     $p = Q$.dequeue()
 6:     for all $i \in I$ do
 7:         $s = p + i$
 8:         if $|c| = 1$ or (max-suffix($s$) $\in V$ and elder-twin($s$) $\notin C$) then
 9:             $supp_{pos}$ = CountSupport($s, pos, g_{max}, g_{min}$)
10:             $supp_{neg}$ = CountSupport($s, neg, g_{max}, g_{min}$)
11:             if $supp_{pos} \geq \delta$ then
12:                 if $supp_{neg} \leq \alpha$ then
13:                     $C = C \cup s$
14:                 else
15:                     $Q$.enqueue($c$)
16:                     $V = V \cup c$
17: return $C$
```

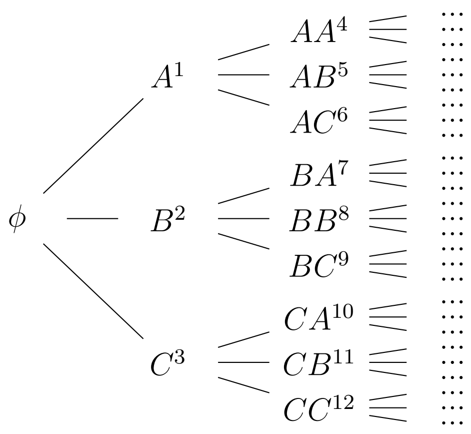


Figure 2.2: An example candidate search space over alphabet $\{A, B, C\}$. The number in the superscript denotes the sequence generation order in a BFS fashion.

2.4.3 On-The-Fly Minimization

The final stage of ConSGapMiner is minimization. During this time, all the SMDS patterns discovered in the candidate generation stage are compared against each other to remove any non-minimal sequences. Ji etl. al. have identified the redundancies in the naïve pair-wise comparison algorithm and proposed a prefix-tree based minimization algorithm over the set of SMDS presorted in increasing length.

Because FMM generates candidates in a breadth-first fashion, the SMDS patterns are discovered in the order of increasing length by definition. As a result, the minimization can be fully integrated into the candidate generation process. We call this on-the-fly minimization. Specifically, when a new candidate is generated and meets the support constrains on line 13 of Algorithm 3, it is first tested for minimality using Algorithm 4 with C and s as input arguments. s is then added to C if it is indeed minimal, or else discarded.

Note that on-the-fly minimization is not a mere reimplementation of the batch minimization used in ConSGapMiner without pre-sorting. For one, the new algorithm no longer requires the prefix tree data structure for subsequence testing.

Furthermore, we change the way exhaustive comparison is performed. Instead of testing all possible subsequences of s against the existing MDS set, s is tested to see if it is a supersequence of any sequence in the current MDS set. At first this may seem counter intuitive as exact sequence matching should be significantly faster than subsequence-supersequence relationship testing. However, as the number of all possible subsequences grows exponentially with the sequence's length, it out-grows the number of MDS by several order of magnitudes as s gets longer. Consequently, our comparison process becomes faster on average.

Algorithm 4 IsMinimal: Given a set of existing MDS M, return **true** if distinguishing subsequence s is minimal, else return **false**

```
Input $M, s$
Outpu true if $s$ is minimal, false otherwise
1: for all $m \in M$ do
2:     if FastTest($m, s, 0, \infty$) is true then
3:         return false
4: return true
```

2.5 Evaluation

In order to evaluate the performance of FMM, we have chosen four datasets from two fundamentally different sources. They are listed in Table 2.1 along with the number of sequences and their average length in the *pos* and *neg* sets.

pfam1 and pfam2 contain sequences extracted from the Pfam Protein Family Database [12]. These sequences use the 20 amino acids as the alphabet and are relatively long in length. The protein families chosen for *pos* and *neg* are also shown in Table 2.1.

stock1 and stock2 are datasets derived from the historic price of S&P 500 companies. The companies are first grouped by their respective industry and their

Table 2.1: Dataset used in the experiments.

Name	*pos* (# seq, avg. len)	*neg* (# seq, avg. len)
pfam1	SrfB $(12, 941.4)$	Spheroidin $(12, 758.2)$
pfam2	B5 $(85, 69.9)$	B3 $(124, 98.3)$
stock1	Technology $(72, 60)$	Utilities $(35, 60)$
stock2	Health Care $(50, 60)$	Financials $(81, 60)$

monthly adjusted closing price from the past five years (2006 to 2010) are analyzed. We compute the monthly price variations and convert them into sequences over the alphabet $\{U, S, D\}$, where U denotes an increase of more than 5%; D represents a decrease of more than 5%; and S corresponds to a fluctuation within 5%. In summary, stock1 compares technology companies to large utilities whereas stock2 compares companies in the health care industry to those in the financial sector.

All experiments have been conducted on a 2.7GHz Intel machine with 8GB of RAM. We compare our results with ConSGapMiner as it is the only published MDS mining algorithm to our best knowledge. For fair comparison, both FMM and ConSGapMiner have been written in Python without employing any hardware-specific optimization. Moreover, all time-related experiments, such as those discussed in Section 2.5.1 and 2.5.3, the reported results are the average of ten repeated experiments.

2.5.1 Support Counting Performance

The first set of experiments aim to study the performance of support counting, which is one of the determining factors of the effectiveness of a MDS mining algorithm. We exam this in terms of the average time spent for each support counting process invocation. This is calculated by dividing the number of candidate sequences by the total time spent in support counting. While the time may vary for each procedure invocation depending on the length of the candidate sequence, by average over a large number of sequences, the resulting value served as a fair indicator of the performance.

Figure 2.3 shows the effect of g_{max} on the support counting performance of ConSGapMiner and FMM. g_{min} is fixed at zero since it has minimal impact on the performance. As g_{max} becomes larger, it takes both algorithms longer time to count support for each candidate sequence. However, the increase is notably more dramatic for ConSGapMiner than for FMM. At $g_{max} = 16$, it takes ConSGapMiner 1.79 ms to count the support for each sequence on average, which is more than 10 times longer when $g_{max} = 0$. On the contrary, each support counting process in FMM only takes around 25% longer, from 205 µs to 255 µs, when g_{max} is increased from 0 to 16.

As a matter of fact, the performance of FMM is so insensitive to g_{max}, its data almost appears as a flat line in the figure when compared to ConSGapMiner. This is in line with our expectation since the bitset shifting operations in ConSGapMiner becomes more expensive with larger g_{max}.

We then fix g_{max} at 2 and subject both algorithms to different numbers of sequences. As can be seen in Figure 2.4, FMM once again scales considerably better than ConSGapMiner. When the number of sequences is increased from 10 to 160, the support counting performance of ConSGapMiner suffers a 15.3 fold drop, taking more than 1.5 ms to complete each counting. In the case of FMM, while the support counting process is slightly slower than that of ConSGapMiner when there are less than 20 sequences, it becomes marked faster with more sequences. With 160 sequences, FMM counts support at 4.2 times faster than ConSGapMiner. It is also noteworthy that despite a 16-time increase in the number of sequences, it takes only 2.1 times longer for FMM to perform each support counting on average.

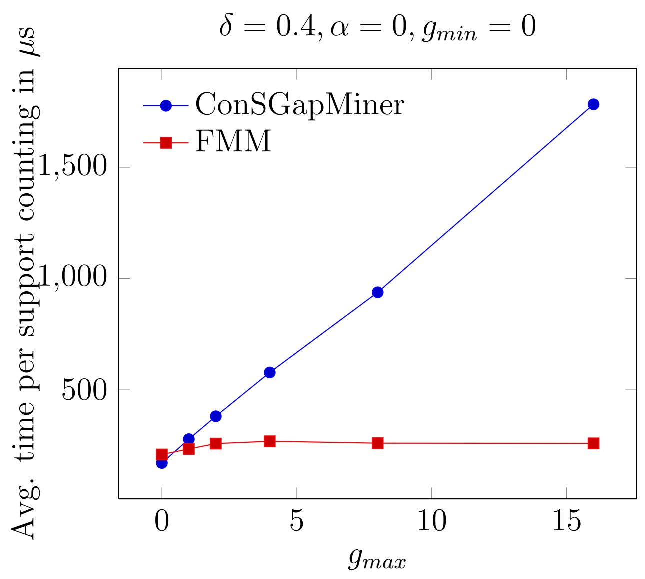


Figure 2.3: Average time per support counting vs. g_{max} for pfam1 dataset.

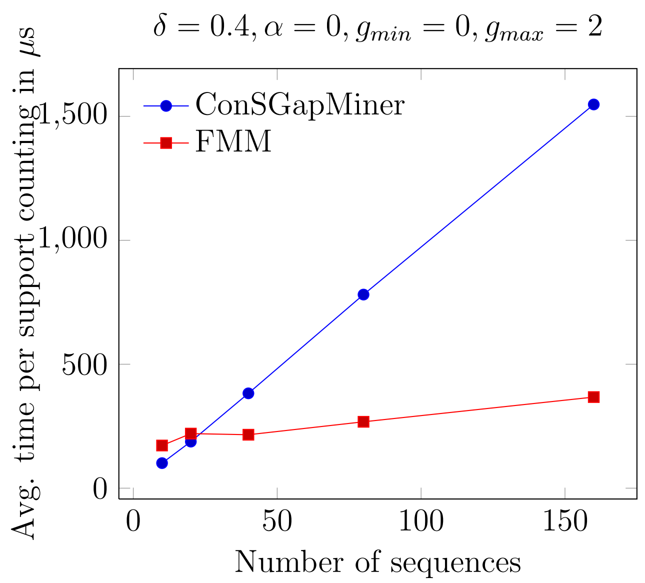


Figure 2.4: Average time per support counting vs. number of sequences for pfam2 dataset.

2.5.2 Pruning Performance

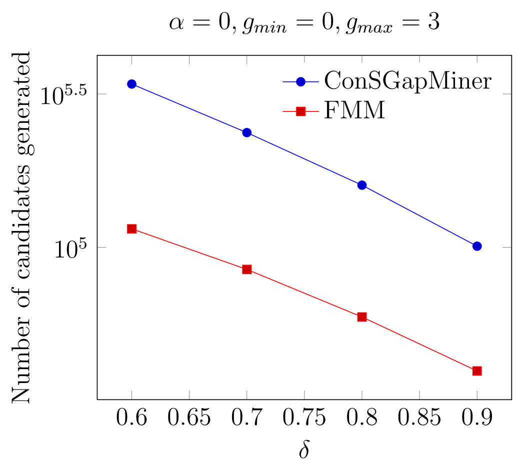
Another way to measure the effectiveness of an MDS ming algorithm is by analyzing its pruning power. By pruning more non-MDS candidates, an algorithm in turn performs fewer support countings, which ultimately improves the performance. In this section we gauge the pruning power of FMM and ConSGapMiner by using two metrics, numbers of candidates generated and redundancy.

A candidate means a un-pruned sequence generated in the search space, and its support must be counted to determine if it is a PP, SMDS, or neither. As a result, the number of candidates essentially dictates how many times the support counting process is called. Figure 2.5 shows the number of candidates generated at different δ levels for ConSGapMiner and FMM. Note that because the number of candidates grow in a qusai-exponential fashion as δ decreases, we choose a logarithmic scale for the y-axis to better demonstrate the difference between the two algorithms. Based on the figure, it is obvious that FMM consistently generates fewer candidates than ConSGapMiner for all four datasets. The most significant difference occurs in pfam2 dataset at $\delta = 0.2$, where FMM generates 5.8 times fewer candidates than ConSGapMiner. This equals to a reduction of 82%. Since the search space shrinks as δ increases, the difference between the two algorithms become less dramatic. For stock2 dataset at $\delta = 0.4$, ConSGapMiner generates about 38% more candidates than FMM does.

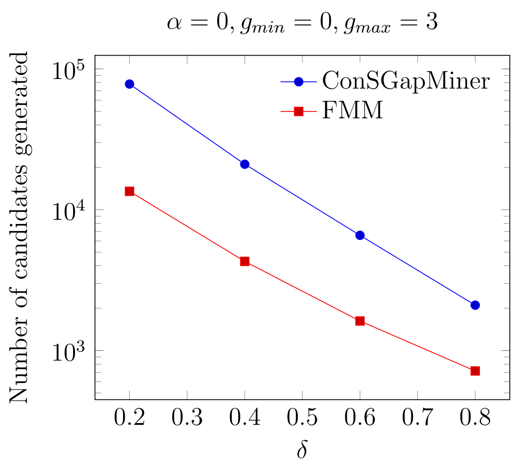
For the redundancy experiments, we compute the metric as follows,

$$\text{redundancy} = \frac{|\text{SMDS}| - |\text{MDS}|}{|\text{SMDS}|}$$

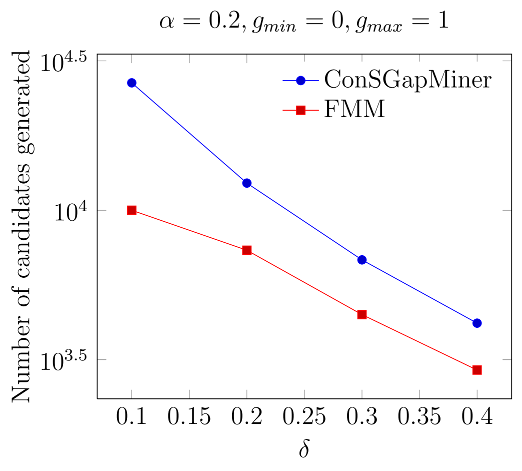
In other words, redundancy is the ratio of distinguishing subsequences that are eventually discarded during the minimization process. An ideal algorithm would employ a pruning strategy that avoids as many non-minimal distinguishing subsequences as possible and thus attains a near zero redundancy. Based on the results demonstrated in Figure 2.6, FMM is much closer to the ideal algorithm than ConSGapMiner. In the case of stock1 at $\delta = 0.1$, 93% of the distinguishing sequences discovered by ConSGapMiner are redundant, whereas 79% of those discovered by FMM are redundant. Therefore, not only does ConSGapMiner waste more time on mining useless sequences, but it also ends up spending more time in the minimization process.



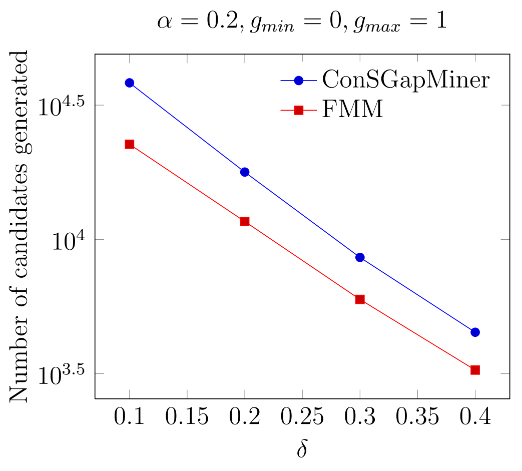
(a) pfam1



(b) pfam2

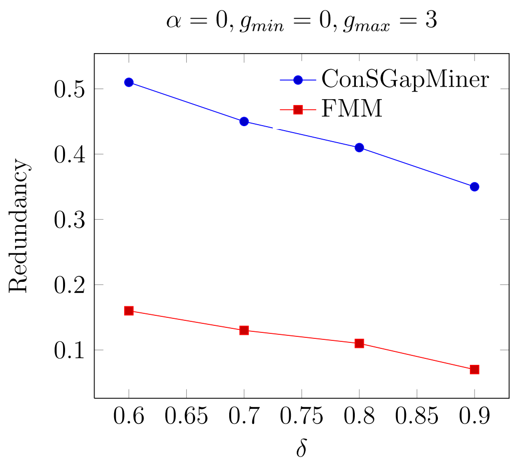


(c) stock1

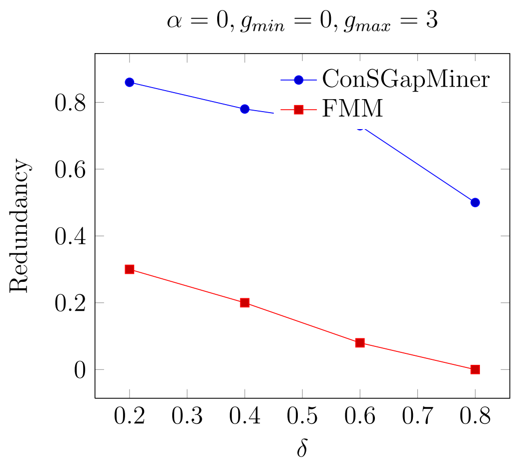


(d) stock2

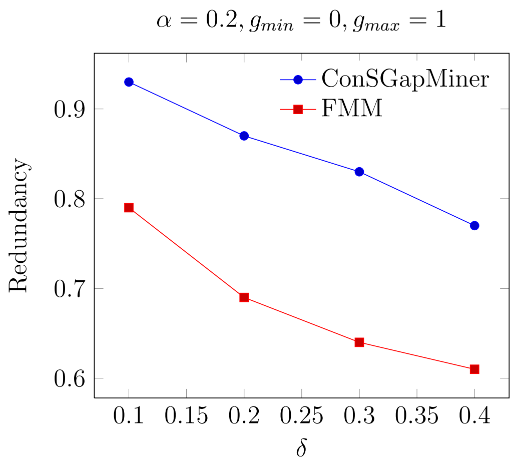
Figure 2.5: Number of candidates generated vs. δ. All y-axes are in logarithmic scales.



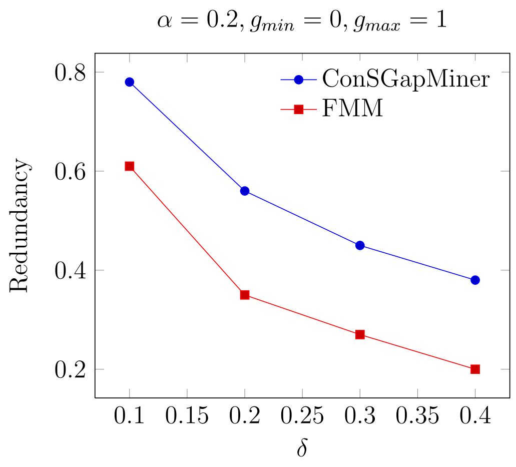
(a) pfam1



(b) pfam2



(c) stock1



(d) stock2

Figure 2.6: Redundancy vs. δ.

2.5.3 Overall Performance

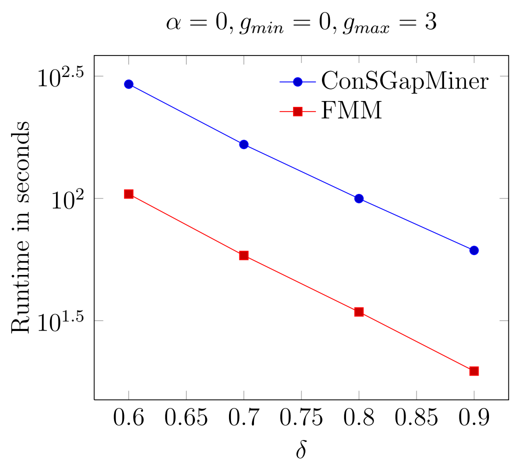
After comparing the support counting performance and pruning power individually, we look at the overall performance of FMM in comparison to ConSGapMiner. The performance is measured based on the runtime of the algorithm over the four datasets. The experimental results can be found in Figure 2.7. Once again, we opt to use logarithmic scale for the y-axis so the difference is clearly visible at smaller values. The specific parameters used for each dataset can also be found on top of its corresponding figure.

Based on the results from previous sections, it should not be surprising to see that FMM consistently outperforms ConSGapMiner at different δ levels. The difference is most notable with smaller δ, where FMM is 7.7 and 6.1 times faster than ConSGapMiner for pfam2 ($\delta = 0.2$) and stock2 ($\delta = 0.1$), respectively. This is mainly due to the fact that the candidate search space tends to grow significantly larger at smaller δ, which enables FMM to prune more effectively than ConSGapMiner. However, even at very high delta level, e.g. pfam1 ($\delta = 0.9$), FMM still manages to complete in less than one-third of the time used by ConSGapMiner. The two algorithm differs the least on when δ is set to 0.3 for stock2 dataset. It takes ConSGapMiner 14.55 seconds to mine all the MDS, whereas it takes 7.71 seconds for FMM to do the same.

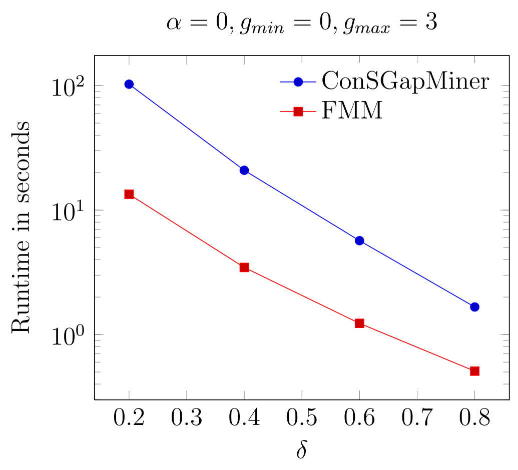
2.5.4 Memory Usage

In this section we examine the memory usage of ConSGapMiner and FMM when mining MDS from different numbers of sequences. In order to demonstrate the effect of large numbers of sequences, we generate extra sequences from the pfam2 dataset by randomly modifying sequences in the same group. The peak memory usage of both algorithms are illustrated in Figure 2.8.

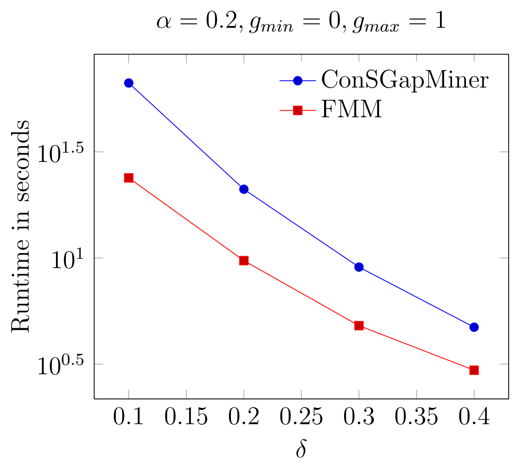
As noted previously in Section 2.4.1, ConSGapMiner maintains multiple bitsets for each sequence in the dataset to speed up the support counting process. Consequently, the space requirement is expected to scale linearly as the number of sequences increases, which is clearly visible in the figure. In fact, the memory footprint of ConSGapMiner more than doubles as the number of sequences is increased from 100 to 1600. On the other hand, the peak memory usage of FMM remains fairly constant regardless of the number of sequences. This is largely owing to its memory efficient FastTest subsequence algorithm. The results are relatively uniform for the other datasets and are therefore not included here.



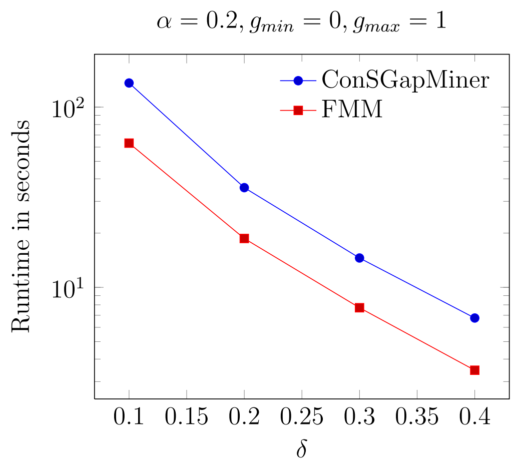
(a) pfam1



(b) pfam2



(c) stock1



(d) stock2

Figure 2.7: Runtime vs. δ. All y-axes are in logarithmic scales.

2.6 Summary

Minimal Distinguishing Subsequence is a salient and succinct way to contrast different classes of sequences, and has many applications in the field of bioinformatics, time series analysis, and machine learning. However, the problem of MDS mining is also extremely challenging with a complexity that grows exponentially with the size of the dataset. This paper presents a new algorithm, called FMM, that makes mining of MDS from large datasets practical.

FMM employs a fast and memory-efficient subsequence testing procedure, a

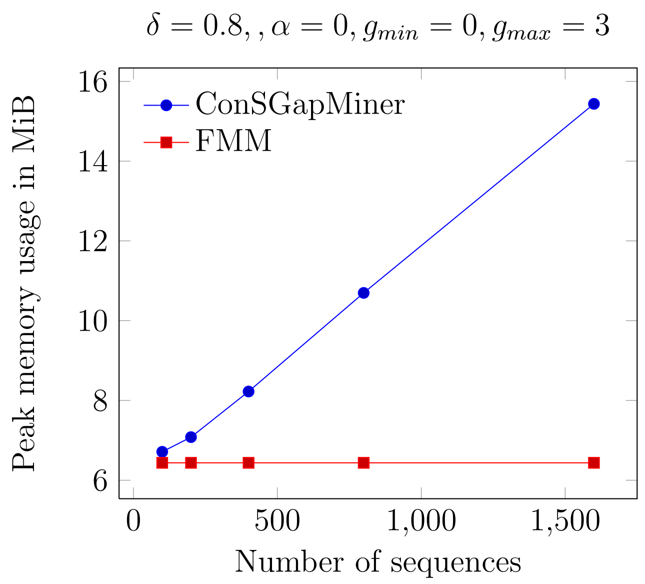


Figure 2.8: Peak memory usage vs. number of sequences for pfam2 dataset.

BFS-based candidate generation scheme with powerful pruning strategies, and a simple on-the-fly minimization process. Together these changes make FMM a much more scalable algorithm than the state of the art MDS mining algorithm, ConSGapMiner.

We use four real-world datasets to demonstrate the performance advantage of FMM over ConSGapMiner. The experimental results show that FMM consistently outperforms ConSGapMiner by a factor of 1.9 to 7.7 times over a wide range of δ. This is mainly due to the facts that a) FMM's support counting process scales significantly better with g_{max} and the number of sequences than the bitset-based approach used by ConSGapMiner; b) The max-suffix pruning strategy employed by FMM proves to remove the number of generated candidate sequences by 27% to 82% over a wide range of δ. The strategy also helps reduce the number of non-minimal distinguishing subsequences, which in turn speeds up the minimization process. In terms of space usage, thanks to the memory-efficient subsequence testing algorithm and on-the-fly minimization, the peak memory usage of FMM remains constant despite a 16-fold increase in the number of sequences. At the same time, ConSGapMiner needs to more than double its memory footprint to accommodate the extra sequences.

CHAPTER 3

Precursor Pattern Discovery

In this chapter we present a generalized algorithm that discovers potential precursor patterns without prior knowledge or domain expertise. The algorithm makes use of wavelet transform and information theory to extract generic features, and it is also classifier agnostic. Based on experiment results using three distinct datasets collected from real-world patients, our algorithm has attained performance comparable to those obtained from previous studies that rely heavily on domain-expert knowledge. Furthermore, the algorithm also discovers non-trivial knowledge in the process.

3.1 Background

Recent advancements in sensor and wireless communication technologies have opened up many opportunities to acquire biomedical signals at a very low cost. Many sensors, such as the ones described in [71], are compact enough to be worn by the subjects, and can continuously gather data for a prolonged period. These technologies have quickly found their natural applications in health care in the form of eHealth and telemedical systems.

Most of the early eHealth systems focus on remote monitoring and abnormality detection. For example, [31] presents a system that monitors patients with Type I diabetes using a glucometer connected to a mobile phone. A more sophisticated system is presented in [86] where health care professionals are alerted when the reading from one of the many biosensors falls outside the normal range. An extensive list of other similar systems can be found in [70]. While providing a low-cost and convenient way for health care personnel to monitor the well being of patients, the majority of these systems are essentially infrastructures for data collection and storage.

One of the main goals of the next generation eHealth system is to mine and analyze the sensor data for the so-called precursor patterns. These patterns are highly correlated to an ensuing medical condition or clinical episode that they served as good prognoses. Thus far there have been several studies dedicated to identifying precursor patterns with a varying degree of success. For example, in [60], an automatic prognosis system is presented to predict the mortality of ICU patients based on heart rate variability and vital signs using support vector machine (SVM). An accuracy between 60% and 80% is reported depending on the parameters used. On the other hand, Yien et. al discover that the low-frequency components of spectrum of arterial pressure and heart rate are highly correlated to the survival of ICU patients and hence can be used as a reliable predictor of the outcome [94].

Another area where extensive research has been carried out on searching precursor patterns is epileptic seizure prediction. Different features of electroencephalography (EEG) signals, including Lyapunov exponents [35], correlation dimension [52], and accumulated energy [30], have been utilized to construct predictive models (see [64] for a comprehensive list). Other precursor pattern discovery algorithms have been developed for various clinical episodes such as sleep apnea [74], arrhythmia [2], and acute hypotension [6].

However, one common problem with these studies is the requirement of domain-specific knowledge to develop their discovery algorithms. Also most of the algorithms require prior knowledge of the duration of the precursor patterns, as well as the time the patterns are likely to occur relative to the clinical episode. Con-

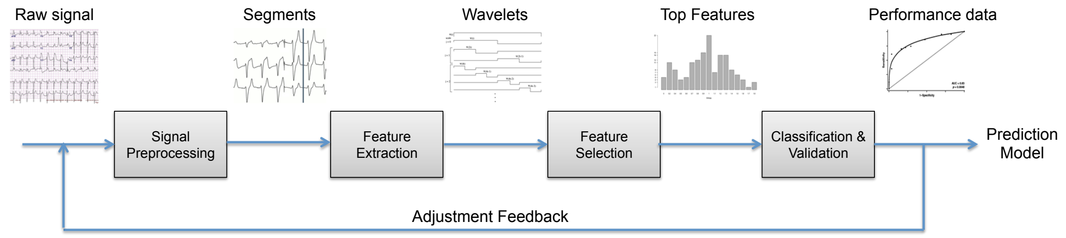


Figure 3.1: The precursor pattern discovery process.

sequently, it is often impossible to apply these algorithms to a different medical condition without significant modification or degradation in performance.

In the remaining sections we present a new generalized precursor pattern discovery algorithm that works with a wide range of biomedical signals and applications. The algorithm does not require domain-specific knowledge, hence it is also possible to discover patterns unknown to experts.

3.2 Precursor Pattern Discovery

The precursor pattern discovery process consists of four stages: a) Signal Preprocessing, b) Feature Extraction, c) Feature Selection, and d) Classification and Verification. At the end of the process, a group of precursor patterns are identified along with a statistical model that can be further used to predict future clinical episodes. The process is depicted in Figure 3.1.

3.2.1 Signal Preprocessing

The first stage of precursor pattern discovery is signal preprocessing. The raw signals received from the sensors often require calibration and filtering. However, this step is often extremely difficult to be generalized due to many application-specific factors, such as the characteristics of the sensor, the transmission noise and error rate, and sensor manufacturing process variation. As a result, we assume that the signals have first been properly calibrated and filtered based on the application of interest.

After the preprocessing, we need to extract the segments, known as the *positive segments*, where the precursor patterns may occur. A positive segment is a fixed portion of the signals t seconds before the clinical episode. t is called the *prediction horizon* and typically ranges from a few seconds to several minutes depending on the application. We also need to extract equal-length segments that are known to contain no precursor patterns. This is normally achieved by using signals from healthy subjects or from the portions of the signals that are distant from any clinical episodes. These segments are known as *negative segments* and are used in conjunction with the positive segments in later stages.

3.2.2 Feature Extraction

After the signals have been properly calibrated and segmented, we extract features from both the positive and the negative segments. Given that we are only interested in features that do not require domain-specific expertise, these features must be generic and easily extractable for most applications. Furthermore, even though the precursor pattern should occur before the particular clinical episode, there is no prior knowledge about its precise time and duration. This makes it even harder to extract the correct features.

In order to overcome these difficulties, we choose to extract features that encompass both spectral and temporal information of the signal. The spectral information is generic and widely applicable, whereas the temporal information helps us identify the time and length of the precursor pattern. An ideal candidate for capturing both kind of information is wavelet transform, where the signal is represented using orthonormal function basis called mother wavelet. There are a number of different types of wavelet transforms differentiated mainly on the mother wavelet used. In our system we have chosen the Haar wavelet transform, also known as Daubenchies-2, because of its low $O(n)$ complexity and because it is shown to work well with time series data [84]. Using the mother wavelet

$$\psi(t) = \begin{cases} 1 & 0 \leq t < 0.5, \\ -1 & 0.5 \leq t < 1, \\ 0 & \text{otherwise.} \end{cases}$$

and the scaling function

$$\phi(t) = \begin{cases} 1 & 0 \leq t < 1, \\ 0 & \text{otherwise.} \end{cases}$$

we can compute the wavelet coefficients at multiple resolutions, each at half of the scale of the previous one. These coefficients are then used as features of the signal segment.

3.2.3 Feature Selection

The number of wavelet coefficients grows as the length of the prediction horizon lengthens. For example, a one minute prediction horizon on a 128Hz signal generates a total of 7680 coefficients. Building a model using every coefficients available is not only slow, but also likely to result in overfitting. In reality, only a small portion of the coefficients actually correlate to the clinical episode, and thus constitute the precursor pattern. In this stage we try to identify the most promising wavelet coefficients by means of feature selection.

Generally speaking, there are three types of feature selection algorithms: Wrapper, Embedded and Hybrid. While Embedded and Hybrid feature selection algorithms tend to select stronger features, they only work with a specific model and classification methods. This precludes them from being used in a generalized environment. On the contrary, Wrapper algorithms select features purely based on their natures, such as correlation, relevance and redundancy, and are therefore model-agonistic. In our system we use Information Gain as basis for feature selection. The Information Gain (IG) for a feature f_i given a set of training samples S_X is

$$IG(S_X, f_i) = H(S_X) - H(S_X|f_i)$$

$H(S)$ is the information entropy of S

$$H(S) = -\sum_{i=1}^{n} p(S_i) \log p(S_i)$$

The conditional entropy $H(S_X|f_i)$ is therefore

$$H(S_X|f_i) = \sum_{x \in S_X} p(x, f_i) \log \frac{p(f_i)}{p(x, f_i)}$$

In other words, $IG(S_X, f_i)$ is the change in entropy if f_i is known in advance. A feature with small IG is considered less relevant and can thus be discarded without weakening the classification model.

3.2.4 Classification and Verification

The last stage of the discovery process involves constructing a statistical model for the selected features and verify the performance of the precursor pattern. Note that during this stage, it is important to separate the training data from the testing data to prevent model overfitting and overly optimistic results. However, if the number of positive segments is limited due to the rare nature of the clinical episode, it is also possible to perform n-fold validation using the same number of positive and negative segments.

Since our algorithm is designed to be classifier-agonistic, theoretically any type of classifier can be used to validate the results. Nevertheless, we recommend validating the results using multiple types of classifiers that differ substantially in terms of the underlaying statistical models. Doing so ensures that the discovered precursor patterns is generic and robust.

Furthermore, if the results indicate that the precursor pattern does not provide satisfactory classifying power, the process should repeat itself with different parameters for preprocessing, feature extraction and feature selection. For example, one may discover that limiting the number of features from the top 100 to top 50 helps to prevent the classifier from overfitting its model, and thus improves the final result.

3.3 Experiment Results

3.3.1 Dataset and Setup

We use the following three large, real-world, and publicly available datasets from PhysioNet [34] to evaluate our work:

1. *chbmit*: This dataset is collected at the Children's Hospital Boston. It consists of EEG recordings of 22 pediatric subjects with epileptic seizure. The EGG signals are sampled at 256Hz with 16-bit resolution. During the 800 hours of recordings, there are 129 instances of annotated seizure attacks.

2. *apnea-ecg*: This dataset comprises 70 records of a continuous Electrocardiography (ECG) signal, sampled at 100Hz with 16-bit resolution, and a set of apnea annotation derived by human experts at 1-minute interval. The total recording lasts about 500 hours.

3. *MIMIC II*: The dataset is made up of 4448 records from ICU patients. The records include ECG, blood pressure, respiration, and vital signs. There are also alerts annotated automatically by ICU monitor.

For each dataset, we perform the process described in Section 3.2 to identify the precursor patterns and use seven well-known classifiers to validate their performance. The classifiers used are Naïve Bayes, Bayes Network, Logistic Regression, C4.5 Decision Tree, SVM, Voting Feature Interval (VFI), and Artificial Neural Network (ANN).

3.3.2 Prediction Accuracy

The first metric used to evaluate the performance of our precursor discovery algorithm is prediction accuracy. Given the limited number of positive segments in the dataset, we choose to conduct the experiment using 10-fold validation. A 10-minute prediction horizon is used throughout the experiment and the same number of positive and negative segments are used in each case to prevent screwing.

From the results listed in Table 3.1, it is clear that the prediction accuracy is fairly consistent for all three datasets regardless of the type of classifier used. The highest accuracy of 84.7% is achieved using SVM based on the precursor patterns from *apena-ecg*, whereas Logistic Regression is only able to predict 75.4% of the alerts in *MIMIC II*. Overall, C4.5, SVM and ANN perform slightly better than other classifiers in terms of prediction accuracy. Note that while the results here are comparable to many of those reported by studies listed in Section 3.1, our algorithm does not require any medical domain expertise to attain this level of accuracy.

3.3.3 False-Positive Rate

The second part of the experiment investigates the false-positive rate, measured in number of false-positive per hour. A balanced algorithm should not give up false-positive rate in favor of unrealistically high prediction accuracy [64]. To

Table 3.1: Prediction Accuracy

	chbmit	*apnea-ecg*	*MIMIC II*
Naïve	77.8%	81.5%	78.7%
Bayes Network	79.8%	80.6%	79.3%
Logistic Regression	76.7%	79.3%	75.4%
C4.5	81.3%	82.1%	80.9%
SVM	80.2%	84.7%	82.1%
VFI	79.4%	79.7%	78.8%
ANN	79.8%	81.5%	81.3%

measure false-positive rate, the precursor patterns and models are used to classify all unseen negative segments in the dataset. Ideally none of these segments should trigger a positive prediction, thus resulting in 0 false-positive rate.

Table 3.2 shows the false-positive rate of all three datasets using different types of classifier. *chbmit* and *apnea-ecg* produce similar false-positive rate ranging from 0.29/hr to 0.91/hr, which is considerably higher than that of *MIMIC II*. One possible explanation for the difference is misalignment. The annotation in *chbmit* and *apnea-ecg* are both done manually, whereas *MIMIC II* contains automatic annotations generated by machines. The false-positive rate should reduce if the annotations are properly aligned.

In terms of differences between classifiers, C4.5 once again produces the best overall results, followed closely by SVM and ANN. While Logistic Regression achieves the lowest false-positive rate of 0.04/hr for *MIMIC II*, it performs poorly for *chbmit*. The inconsistent suggests that Logistic Regression may not be a suitable classifier for our generalized algorithm.

Table 3.2: False-positive Rate

	chbmit	*apnea-ecg*	*MIMIC II*
Naive Bayes	0.52/hr	0.41/hr	0.17/hr
Bayes Network	0.49/hr	0.39/hr	0.11/hr
Logistic Regression	0.91/hr	0.33/hr	0.04/hr
C4.5	0.33/hr	0.29/hr	0.05/hr
SVM	0.37/hr	0.27/hr	0.13/hr
VFI	0.45/hr	0.38/hr	0.20/hr
ANN	0.51/hr	0.44/hr	0.06/hr

3.3.4 Precursor Pattern Interpretation

One of the strengths of our generalized algorithm is the ability to discover non-trivial patterns. For example, when selecting the most prominent features in the *chbmit* dataset, we discovered that the top-100 features consist entirely of signals acquired from only two EEG channels, $F_4 - C_4$ and $F_{p1} - F_3$, which are highlighted in Figure 3.2. In other words, these are the only two channels that are relevant when it comes to seizure prediction.

Figure 3.3 demonstrates another interesting characteristic uncovered by our algorithm. The figure shows the temporal-spectral distribution of the top-100 most relevant features for the *MIMIC II* dataset. The majority of features concentrate at the lower left corner with frequency less than 50Hz and within 5 minutes prior to the alerts. This suggests that a reliable prediction can still be made even if the signals were sampled at a lower sampling frequency with a shorter prediction horizon.

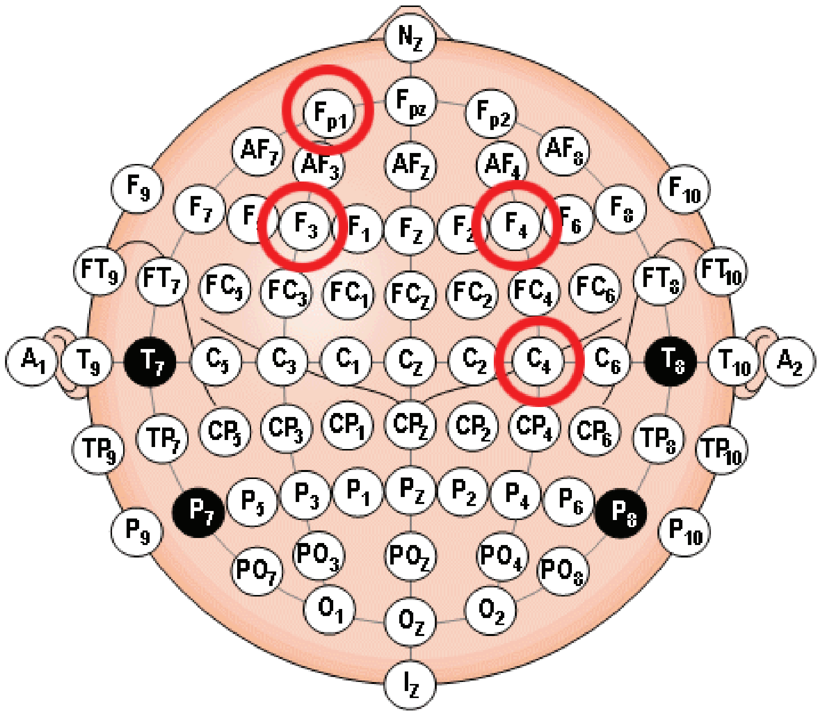


Figure 3.2: International 10-20 EEG electrode placement map with the channels most relevant to seizure prediction highlighted.

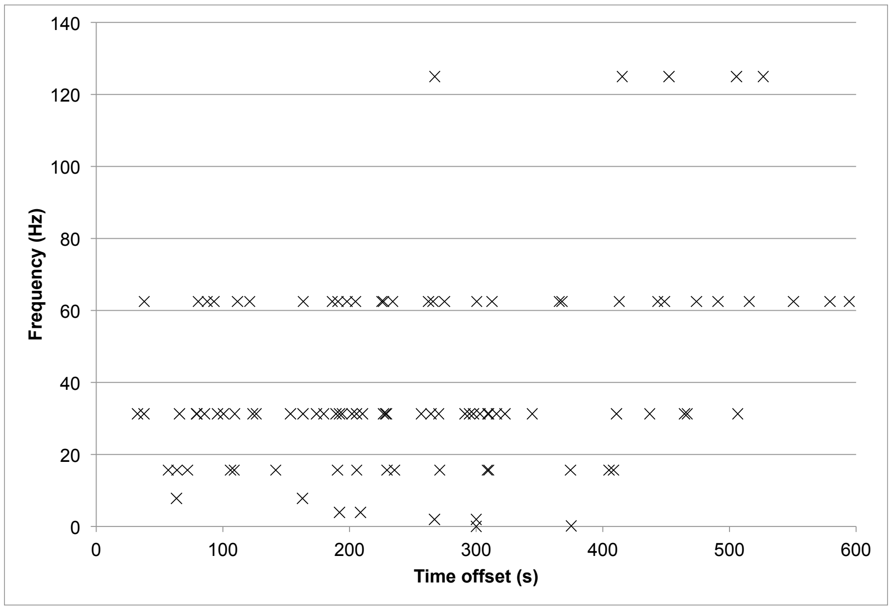


Figure 3.3: Distribution of top 100 features for *MIMIC II*.

3.4 Summary

As continuous remote monitoring becomes more prevalent, the demand grows stronger for discovering precursor patterns in biomedical signals which predict medical conditions and clinical episodes. In this chapter, we present a generalized algorithm that is able to discover such patterns without domain-specific knowledge and expertise. The algorithm is classifier agonistic and is applicable to a wide range of medical conditions. Experiments using three real-world datasets show that the algorithm can achieve a prediction accuracy as high as 84.7% without producing high false-positive rate. Furthermore, using the precursor patterns we are able to infer non-trivial knowledge such as the most relevant EEG channels to predict epileptic seizure and the minimal sampling rate required for predicting ICU alerts.

CHAPTER 4

Predictions Using Discrete Data

Not every remote monitoring system is capable of capturing and storing continuous data stream. In fact, many current systems are still largely limited to collection data from discrete type of sensors, such as weight scale, blood pressure monitor, glucometer, etc. This chapter introduces the design and implementation of WANDA, an end-to-end remote health monitoring and analytics system designed specifically for heart failure patients. WANDA supports the collection of data from a wide range of discrete sensory devices. The collected data are stored in an Internet-scale data storage and search system. WANDA also provides a backend analytics engine for diagnostic and prognostic purposes.

Tthe practicality and efficacy of WANDA through several clinical trials, including an on-going heart failure and readmission trial that involves up to 1500 HF patients. Using the data gathered in the clinical trials and machine learning techniques, it has been demonstrated that WANDA is capable of predicting the worsening of patients' heart failure symptoms with up to 74% accuracy as well as achieving at least a 45% improvement in sensitivity compared to the commonly used thresholding algorithm based on daily weight change. Moreover, the accuracy is only 9% lower than the theoretical upper bound.

4.1 Background

Heart failure (HF) is one of the leading causes of death in the US and around the world for adults over 65 years of age [61]. Nearly one-forth of the patients treated for HF are re-hospitalized within 30 days, and almost half of them are readmitted within 6 months [75]. These unplanned readmissions are estimated to cost the American healthcare system more than $17 billion annually, or 15% to 20% of the total Medicare expenditure in acute hospital care [43].

To reduce the morbidity, mortality, and economic cost associated with HF, remote health monitoring appears to be a promising solution that can work at scale. However, a major challenge to the realization of a large-scale remote monitoring system is the ability to collet, store, and process the large amount of data gathered from the sensors in an effective, robust, and automated fashion. Furthermore, in order for remote health monitoring to be truly successful, it should also be able to perform intelligent analysis on collected data. This means the system should provide a wealth of analytical algorithms that can infer useful information or discover predictive patterns from the data. Armed with the information, early interventions can be made to prevent the medical event from happening. Unfortunately, most current remote monitoring systems lack such advanced analytics capabilities.

In the remaining sections, we present the design and implementation of WANDA, a remote health monitoring system for HF patients. The system is an end-to-end solution that covers all key aspects of remote health monitoring, from data collection, data storage and access, to data analytics. It also features several auxiliary tools such as an administrative portal, a questionnaire system, and a social network component. The practicality and efficacy of WANDA has been validated by three clinical trials over the past 4 years.

We also showcase the analytics engine of WANDA by processing real clinical data and generating statistical models that significantly improve the accuracy of predicting the worsening of HF patients' symptoms when compared to the traditional thresholding algorithm based on daily weight changes.

4.2 Related Work

The use of technology to monitor patients at a distance is increasingly gaining attention as a strategy to improve the care of patients with chronic diseases. Remote patient monitoring systems provide the opportunity for a better follow-up, early detection of signs of clinical deterioration, and early intervention to prevent hospitalization or even death. Several studies targeting the remote monitoring of heart failure patients have been conducted in the past 5 years.

Chaudhry's telemonitoring study [19] required participants to make daily phone calls to an automated telemonitoring system (provided by Pharos Innovations [5]) for a period of 6 months. Each call played a prerecorded voice message that consisted of a series of questions about symptoms and weight for which the participants had to provide answers using the keypad on the phone. The responses were then downloaded from the telemonitoring system to an Internet website for daily review by clinicians. However, this study proved to be unsuccessful in reducing the risk of readmission or death in heart failure patients, compared to standard care.

Another heart failure study conducted by Soran [79,80] included an electronic scale and an individualized symptom response system (Alere DayLink monitor) connected to a computer database via a standard phone line. Patients were instructed to weigh themselves and answer a series of heart failure questions daily. Nurses reviewed the transmitted data on a daily basis and immediately contacted patients whenever the data fell out of a healthy range. After contacting the patient, the nurses immediately notified the patient's primary physician of any symptom changes by means of a fax report. Again, this study showed no improvement in clinical outcomes when compared to standard care of heart failure.

The Home or Hospital in Heart failure (HHH) study is another study that was conducted to evaluate a home telemonitoring system to supervise heart failure patients outside the hospital setting [65]. In this study, patients took weekly measurements of their weight, heart rate and blood pressure. Measurements were then entered via a phone keyboard, in reply to questions from a computerized interactive voice response system. Electrodes were attached to the patients' bodies to allow for a 24-hour cardiorespiratory monitoring. Since the raw ECG signal could not easily be transferred through the standard telephone lines, only the RR time series was actually transmitted to a database and then to an analysis center, together with the other measurements. Despite high patient compliance (81% of vital signs transmissions and 92% of cardiorespiratory recordings were completed) results showed no significant effect of remote health monitoring in reducing bed-days occupancy, cardiac death or hospitalization when compared to usual care of HF.

According to Desai [26], an effective home monitoring system must contain the necessary elements that together complete the circle of heart failure management. Some of the important circle elements are the reliable measurement of physiological variables that can help in the early detection of adverse events, efficient transmission of data to enable a timely response, the direct reception of data by personnel qualified to recommend an effective intervention, and patient adherence.

The lack of significant differences between the telemonitoring solutions used in Chaudhry, Soran's and the HHH studies and routine care with regard to readmission rates or death can then be explained by several breaks in this circle. In Chaudhry and Sorans' studies, the lack of patient adherence as well as the requirement for decision making by midlevel providers to be supervised by physicians may stall timely responses to detected early signs, especially in large-scale studies.

More importantly, however, weight and symptoms alone do not suffice as accurate and responsive measures for the early prediction of heart failure. Although rapid weight gain is a relatively specific predictor of heart failure decompensation, it is not a very sensitive marker because weight can vary with changes in caloric intake and the quantity of weight gain before hospitalization is typically fairly modest, with fewer than 50% of patients gaining more than 2 lb (0.9 kg). Therefore, using weight as the only measure is inadequate to recognize impending decompensation in sufficient time to intervene to prevent hospitalization [25]. These reasons, among other things, disrupt Desai's circle of HF management, preventing even an effective intervention from improving outcomes in practice. Although the HHH study measures three different markers (weight, heart rate and blood pressure) as well as continuous cardiovascular signals, it severely violates the efficient data transmission and timely response element of the circle. Since the measurements are only taken and transmitted once every week and the data is transmitted over a slow telephone line (as is the case in [20] and [79]) the transmission of rich raw ECG data is hindered and the granularity of intervention is very coarse for effective outcomes to be observed.

It is therefore necessary to design a remote monitoring system that can effectively close or at least tighten the circle of management in practice by encouraging patient compliance, offering a reliable measurement of accurate physiological variables that can enable early detection and efficiently transmitting data to permit a timely response. WANDA is an automated real-time scalable remote health monitoring system developed with these results and findings in mind and designed to address all of these requirements.

While many systems exist today that offer remote monitoring for heart failure patients, they are either invasive, requiring sensors to be attached to a patient's body or implanted inside of it, cumbersome and complicated that specially trained technicians are required just to set them up, or they are reactive in nature [14]

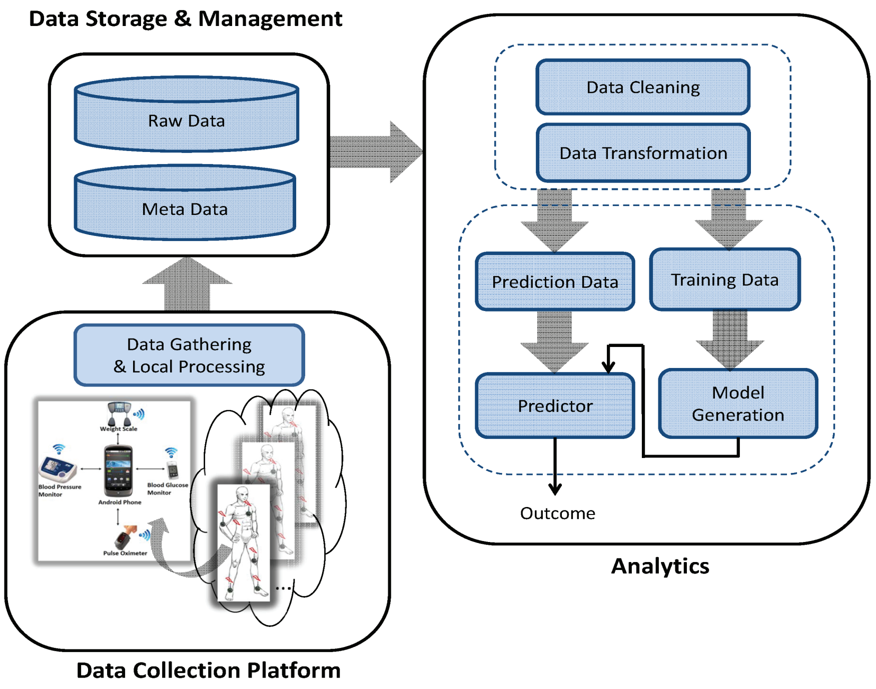


Figure 4.1: Overall architecture for WANDA

where patients are contacted only when their symptoms are already exacerbated or they are not feeling well. WANDA, in contrast, is a noninvasive, easy-to-use, proactive remote monitoring system for heart failure patients that can detect early vital signs and predict adverse events using a set of powerful algorithms.

WANDA is a three-tier, end-to-end remote monitoring system with extensive hardware and software components designed to cover the broad spectrum of the telehealth and remote monitoring paradigm. The overall architecture is summarized in Figure 4.1.

The first tier of the architecture consists of a data collection framework, which is formed from a heterogeneous set of sensing devices that measure various bodily statistics such as weight, body fat, body water, blood pressure, heart rate, blood glucose, blood oxygen saturation and body movements. The data from these sensors are collected, processed, and transmitted via a smartphone-based gateway to the cloud—the second tier of the WANDA architecture. The large amount of data are stored and indexed using a scalable database and can be accessed easily using a RESTful style web interface [32]. The last tier of the WANDA architecture is a backend analytics engine capable of continuously generating statistical models and predicting outcomes using various machine learning and data mining algorithms. In the following sections, we describe each component of the system briefly; however, the main focus of this section is on the analytics engine and the new features that it provides for health care management.

4.2.1 Data Collection

Due to the increasingly ubiquitous nature of smartphones and their portability and connectivity, we decided to utilize an Android smartphone as a central hub for receiving patient measurements and communicating them to the WANDA server. All the devices we utilized are Bluetooth enabled. Users can readily collect data on their WANDA smartphone application (WANDA App). It provides a user-friendly data management tool for the patient, while transmitting data to a central WANDA server for storage and backend data analytics.

The WANDA App was designed to run on any device running Android OS version 2.0 or above and is compatible with multiple sensing devices. Before the phone can establish a connection with any device, a pairing process is required. When two Bluetooth nodes are paired, they are aware of each other's existence, have a shared-link key that can be used for authentication, and are capable of establishing an encrypted connection. Once two devices are paired, the basic information about the device such as device name, class, MAC address is saved and no more pairing is needed in the future. For security reasons, we require users to specify the sensors that they intend to use and the phone only accepts the connection request from these devices. Depending on the device, the phone can act as a slave or master. The device data are received by the Bluetooth service module, before they are processed by a measurement module. The measurement module then stores the information through a record module. The history module provides a data management tool for users to analyze their own data, and the services module performs activity recognition and other sensor services in a background thread. All the data are then displayed to the user through a graphical user-interface (GUI) module that interacts with the user.

4.2.2 Storage and Access

WANDA takes advantage of Amazon's highly scalable cloud data storage to handle its growing storage needs. Amazon S3 provides data security through multiple access control mechanisms and encryption for both secure transit and secure storage on disk. Data durability and reliability are guaranteed by Amazon's redundant storage and regular verification and repair of corrupted stored data. Security and reliability are by far the most important considerations for a data storage and management system used for storing confidential medical patient information.

We also use Amazon's SimpleDB to store structured data such as patient and sensor information, data indices, adverse events, and data annotations. SimpleDB is a NoSQL database management system as opposed to the traditional relational database. It offers superior scalability and an extremely flexible database schema. The latter is particularly important as it allows data stored in WANDA to be both backward and forward compatible without undergoing painful schema migrations.

However, the NoSQL nature of SimpleDB presents several technical challenges when it comes to data access. For example, SimpleDB lacks many features commonly found in a relational database such as the JOIN operation and full ACID guarantee. This makes it complicated to develop software that interacts with the database directly. As a result, we have developed an intermediate layer that provides a web-based RESTful interface for the developers. Not only does this interface free the developers from learning the specifics of SimpleDB, it also gives a pseudo object-oriented structure to the database. Table I gives a few examples of RESTful resources with available methods and expected results.

The WANDA database also features a Publisher-Subscriber (PubSub) interface for applications that need to be notified of changes to the database in real-time. For example, a simple safety system may wish to check every blood pressure reading and alert the medical staff when the reading exceeds or falls below a certain value. Instead of polling the database at a regular interval, the system can subscribe to the blood pressure topic published by the database. It then receives a notification via HTTP requests or email whenever a new reading is added to the database. The database publishes a wide range of topics on addition, editing, and removal of objects at both fine- or coarse-grained levels, e.g. for a particular patient or for all patients from a particular study.

4.2.3 Data Analytics

One of the strengths of WANDA over other remote health monitoring systems is its analytics engine. Based on the data and annotations collected, the analytics engine can generate multiple statistical models using various machine learning and data mining algorithms, including classification, clustering, association rule mining, etc. These models can then be used for both diagnostic and prognostic purposes. For example, in the case of HF patients, it is highly desirable to be able to predict the worsening of symptoms before the patient is actually hospitalized.

The analytics process normally consists of two stages. Firstly, the data are downloaded and analyzed offline based on various hypotheses. Once a strong model has been generated and validated, it can then be uploaded to the server to perform online prediction. One of the challenges here is to optimize the algorithm so that it can be executed in a real-time fashion. This often requires using the

PubSub data access interface, in conjunction with a temporal storage for pre-computed historic data. Finally, when the algorithm detects a pattern that is strongly associated with an undesirable outcome, an alarm is triggered and the personnel in charge are contacted immediately.

The analytics engine integrates nicely with the Weka framework [36]. This allows the engine to support a wealth of commonly used machine learning and data mining algorithms. Furthermore, an API is also provided for adding new algorithms to the engine.

4.2.4 Supporting Tools

We have also developed a number of tools that work closely with the core WANDA system to support the need of various clinical trials and studies. These tools include an administrative portal, a questionnaire system, and social network applications.

4.2.4.1 Administrative Portal

A Web-based tool serving as a study and information management platform was developed in-house. The tool is a web application that facilitates labeling, annotating and entering data into the database. Medical clinicians and nurses can easily enter information including patient demographics, like gender, age and ethnicity, as well as notes of events like hospital discharge, hospitalization or death. The data is stored in the second tier of the system described in Section 3.2. In addition to data entry and data annotation, the tool also offers a rich and customizable interface that provides advanced search functionality for stored data. The search results can also be visualized in the form of graphs and charts to display information like patient measurements over time. Good data visualization can enhance fast diagnostic decision making based on simple trends seen in the displayed data, like a constantly increasing weight, for example. Data annotation also serves our analytics engine by providing our algorithms with more training data. The web application was implemented in JAVA using Google Web Toolkit (GWT), a development toolkit that creates highly-optimized Web applications. The tool provides easy navigation through different panels to deliver an intuitive user experience.

4.2.4.2 Questionnaire System

Another supporting tool for WANDA is the questionnaire system. Questions about symptoms are entered by clinicians into the administrative portal described in Section 3.4.1 and are made available on the patient's Android phone on at a predefined frequency. The patients can answer the heart failure symptom questions using the phone's keyboard or touch screen. The user responses are then uploaded to the WANDA storage platform described in Section 3.2. Medical staff can then access these responses using the administrative portal.

4.2.4.3 Social Network Applications

WANDA Social is WANDA's social component which can be a powerful tool for encouraging healthy behavior as well as improving patient compliance. A Facebook application was developed in-house to offer two motivating factors: cooperation and competition. By connecting with people with similar conditions and health concerns, users can work together and encourage each other to achieve a common goal like losing weight, for instance. The application also provides competitive events such as online tournaments, where users can challenge each other to accomplish certain health-related tasks like lowering their blood pressure to a healthy level within a time frame. A leaderboard and nominal prizes can motivate users to achieve the greatest improvement in a given health indicator. The WANDA

Social platform was fully integrated with Facebook to take advantage of its large user base. To respect the user's privacy, the application will allow authenticated and registered participants of the study to post their information, e.g. sensor readings and achievements, using their real identity or as anonymous users. By tapping into cooperative and competitive relationships through WANDA Social, we can motivate patients to adhere to a treatment regimen or adopt a healthier lifestyle.

4.2.5 Clinical Trials

Since November 2009, the WANDA system has been used for health data collection on the intervention arm including 26 different congestive heart failure patients [85]. The population of this first clinical trial is approximately 68% male; 40% White, 13% Black, 32% Latino, and 15% Asian/Pacific Islander, with a mean age of approximately 68.7 12.1. The gender distribution and anticipated age of participants are representative of the incidence and natural history of congestive heart failure. Study participants were all provided with Bluetooth weight scales, blood pressure monitors, and personal activity monitor devices.

The second clinical trial of WANDA started on February 2011 with 18 low literacy Latinos with heart failure. This population is disproportionately affected with HF, is more likely to be hospitalized with HF, and is at greatest risk for re-hospitalization, and dying from HF. The population of the participants in this study is approximately 89% male with a mean age of approximately 54. Study participants were provided with Bluetooth weight scales, blood pressure monitors, landline or Ethernet gateways, and Android activity monitoring applications.

Finally, the system is currently deployed in a much larger study that targets the remote monitoring of 1500 patients of 50 years or older with heart failure problems. This on-going project is conducted in collaboration with the UCLA Department of

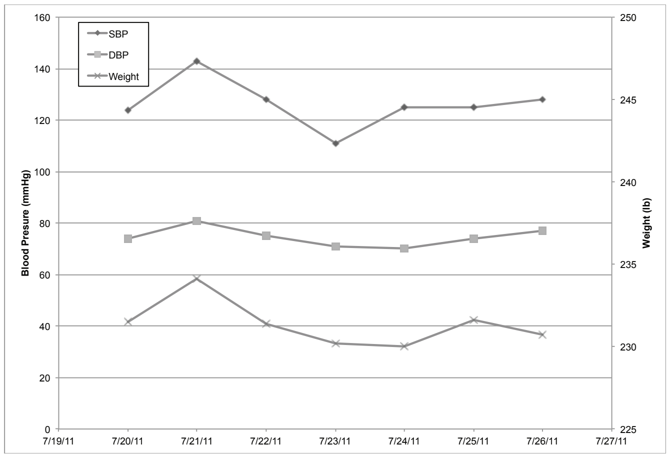
Medicine, UC-Davis, UCSF, UCI, UCSD, and Cedar Sinai Hospital. Heart failure patients who were hospitalized at any of the six participating medical centers are being considered to be recruited in either the control or intervention arm through a randomized trial process. Data collection for this study has started in November 2011 and is scheduled to end in April 2013. Patients measure their weights and blood pressures and reply to questionnaires on a daily basis, and the collected data are transmitted to the database via a phone-line, an Internet connection at home, or through cellular networks. We have currently enrolled more than 400 patients in the study.

4.3 Symptom Prediction

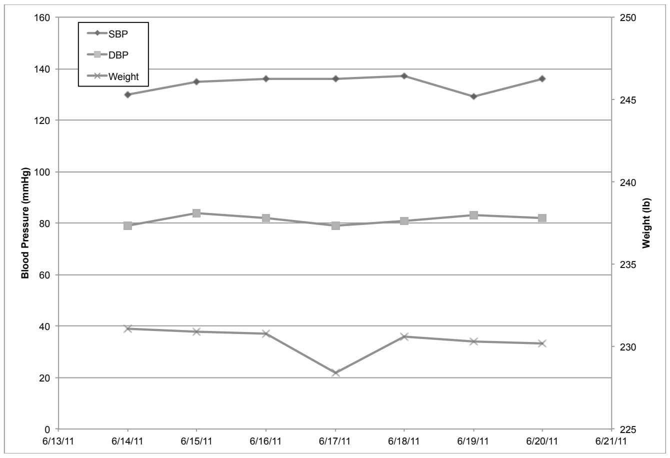
In order to demonstrate the capability of the WANDA analytics engine, we use the dataset gathered from one of the clinical trials to predict the worsening of HF symptoms using advanced machine learning algorithms.

So far the most widely used metric to predict the worsening of HF symptoms is Daily Weight Change (DWC). It is even recommended by American College of Cardiology/American Health Association as a potential indicator for water retention, which leads to swollen ankles and other HF symptoms [39]. However, several studies have shown that DWC has relatively weak correlation with the worsening of HF symptoms [20, 53, 98]. We have also experienced the poor predictive power of DWC first -hand during our clinical trials. When the medical staffs follow up when a DWC of more than 2lb was observed, the patient almost always attributes the change to food or normal weight fluctuation and denies any worsening of HF symptoms.

The case of using DWC as a predictor is further weakened from the example depicted in Figure 4.2. The figure shows systolic and diastolic blood pressure (SBP & DBP) and weight readings from the same patient over two separate 7-day



(a)



(b)

Figure 4.2: 7-day blood pressure and weight readings of a patient when (a) worsening of HF symptom is reported, and (b) no change in HF symptom is reported

periods. By the 7th day in Figure 4.2a, the patient is known to have reported worsening of HF symptoms, whereas the patient reported no change in symptom during the 7-day period in Figure 4.2b. Although the weight has fluctuated in both cases, there is very limited change in weight (< 1lb) measured on the 7th day. If DWC were used to predict the symptom, we would likely end up having either a large number of false positives, or miss the true positives altogether. On the other hand, it is clear from the figure that the systolic and diastolic blood pressures are less stable before the worsening of HF symptoms, suggesting that they may serve as better predictors in this case.

4.3.1 Experiment Setup

WANDA's second HF clinical trial is used as the data source for the experiments. We were particularly interested in patients' daily weight, systolic and diastolic blood pressure, and heart rate measurements as they are relatively free of missing data for most patients over the three-month period.

Patients also answer a series of 9 questions, as listed in Table 4.1, on a daily

Table 4.1: Daily questionnaire

#	Question
1	Are you coughing more than usual?
2	Are you more tired than usual?
3	Did you use an extra pillow last night?
4	Did you eat a low-salt diet?
5	Did you take your heart medications?
6	Did you take your water pill this morning?
7	Did you walk or exercise today?
8	Have you noticed increased difficulty in breathing?
9	Have you noticed increase in swelling of feet or ankles?

basis. We interpret a positive response to question 8 and 9 as a worsening of a patient's HF symptoms, whereas a negative or a lack of response is interpreted as a stabilized HF condition. The answers are also crosschecked with nurses' call logs to remove any accidental false reporting. Based on these criteria there were a total of 34 instances of worsening of HF symptoms self-reported by 9 patients during the clinical trials.

Once instances of worsening HF symptoms have been identified, they are labeled as positive. All other instances that are at least 3 days away from the positive instances are considered negative. Instances occurred within 3 days of positive instances are discarded. A total of 9 features, listed in Table 4.2, are extracted at each instance and used by the prediction algorithms.

Six different machine learning algorithms from the WANDA analytics engine have been evaluated in the experiments. They are listed in Table 4.3 with a short description of the algorithm and the corresponding parameters used. These algorithms are chosen as they utilize a wide range of fundamentally different statistical principles. This ensures that the results are based on the intrinsic discriminatory

Table 4.2: Classification Features

Feature	Description
dcs	Daily change in systolic blood pressure
dcd	Daily change in diastolic blood pressure
dcw	Daily change in weight
$sds.3d$	Standard deviation of systolic blood pressure over the past 3 days
$sdd.3d$	Standard deviation of diastolic blood pressure over the past 3 days
$sdw.3d$	Standard deviation of weight over the past 3 days
$sds.7d$	Standard deviation of systolic blood pressure over the past 7 days
$sdd.7d$	Standard deviation of diastolic blood pressure over the past 7 days
$sdw.7d$	Standard deviation of weight over the past 7 days

power of the extracted features rather than a particular algorithm overfitting the model. The same set of experiments is also conducted using simple thresholding based on DWC of 2 pounds.

Several measures have been taken to ensure the fairness of the experiments. Firstly, equal numbers of positive and negative instances are used in each experiment to prevent biasing toward a particular class. Secondly, the results are computed using ten-fold cross validation to guard against overfitting given the relatively small number of instances. Finally, each experiment is conducted ten times and the average results are reported, thus reducing the effect of outliers.

4.3.2 Results and Discussion

The algorithms are firstly evaluated in terms of their accuracy, sensitivity, and specificity. They are computed based on the number of True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN) in the prediction results

Table 4.3: Classification Algorithms Used in Experiment

Algorithm	Description	Parameters
Naïve Bayes Classifier (NBC)	NBC uses Bayes' theorem and assumes each feature is conditionally independent. The posterior probability of a class C given a set of features $F_1, F_2, ...F_n$ is therefore $$p(C\|F_1, F_2, ...F_n) \propto p(C) \prod_{i=1}^{n} p(F_i\|C)$$ The prior and likelihood can be calculated from the training set directly and applies to the test set.	
Nearest Neighbor (kNN)	An unlabeled instance is classified based on its nearest k neighbors from the training set based on majority vote. Euclidean distance of the features is often used to determine the closeness of two instances $$d = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$ where x_i and y_i are corresponding features from the two instances.	$k = 5$
Logistic Regression (LR)	Logistic regression is a type of regression that predicts the outcome of a binary depend variable (class) based on a set of independent variables (features) X_i using the logistic function $$p(x) = \frac{1}{1+e^{-f(x)}}, \quad f(x) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + +\beta_i X_i$$ where $p(x)$ is the probability of x being class 1, given regression coefficients $\beta_1 ... \beta_i$ and intercept β_0.	
Voting Feature Interval (VFI)	VFI classifies builds upper and lower bounds around each class for each feature. Classification is based on majority voting, where a vote for class C based from feature a is computed as $$v(a, C) = \frac{\text{interval\_class\_count}(a, i, C)}{\text{class\_count}(C)} \left(\frac{H(C\|a)}{\text{max uncertainty}} \right)$$	bias = 0.6
Ripple-Down Rule Learner (RIDOR)	RIDOR is a version of Ripple Down Rule using the Indcut algorithm where the default rules are first generated based on least error rate. A set of exception rules are generated to predict classes other than those covered by the default rules.	
C4.5 Decision Tree (C4.5)	C4.5 constructs a decision tree based on information entropy. Each feature in the training set is evaluated for its information gain $$IG(T, a) = H(T) - H(T\|a) \quad H(T) = -\sum_{i=1}^{n} p(x_i) \ln p(x_i)$$ Feature with the largest IG and have yet been used is chosen to split the decision tree at each node until the confidence falls below certain threshold.	confidence = 0.25

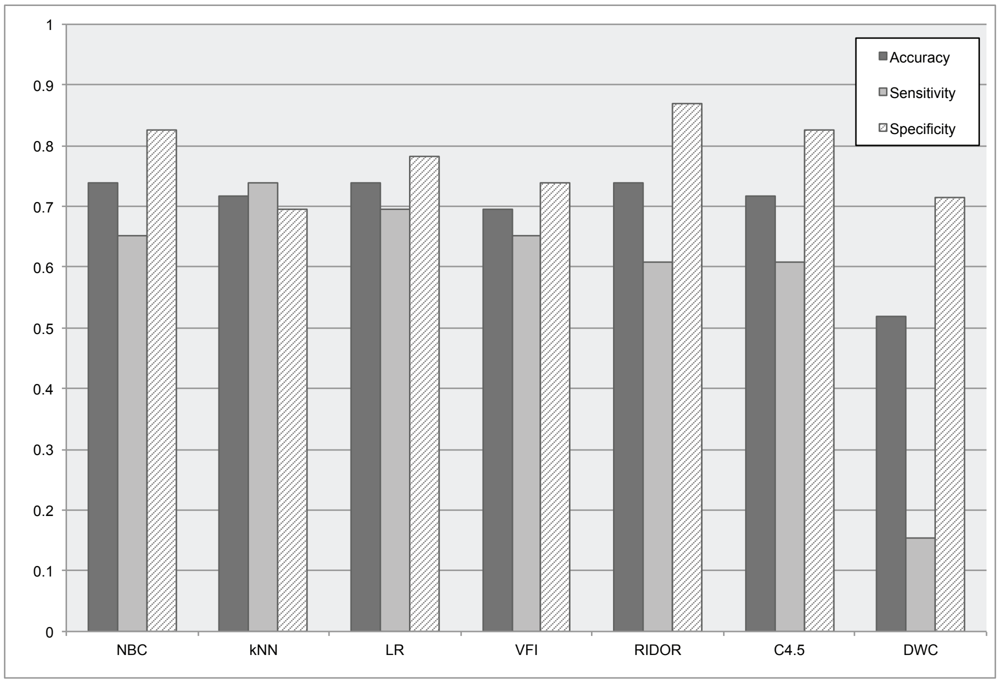


Figure 4.3: Accuracy, sensitivity, and specificity of NBC, 5NN, LR, VFI, RIDO, and C4.5 vs. DWC

$$\text{Accuracy} = \frac{\text{TF} + \text{TN}}{\text{TF} + \text{FP} + \text{TN} + \text{FN}}$$

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \quad \text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

The results for each algorithm are plotted in Figure 4.3. Since the number of positive and negative instances is balanced, the minimal accuracy of any better-than-chance algorithm should be greater than 0.5. The figure shows that predicting the worsening of HF symptoms using DWC seems to be only marginally more accurate than guessing, with an accuracy of 0.519. This is not surprising considering that only 15.9% of the positive instances have been classified as such, whereas a substantial amount of negative instances (28.6%) turn out to have a DWC of more than 2lb.

On the other hand, the algorithms from the WANDA analytics engine attain much higher accuracy. Three algorithms, NBC, LR, and RIDOR, are able to correctly predict the worsening or stabilization of HF symptoms 74% of the time. Even though VFI has the lowest accuracy of 0.696, it is still nearly 20% more accurate than DWC. Furthermore, all six algorithms have sensitivity values that are at least 45% greater than that of DWC. This suggests that the features listed in Table III have a much stronger correlation to the worsening of HF symptoms than simple daily weight change, which can be further confirmed by the high specificity ranged from 0.696 (for kNN) to 0.87 (for RIDOR).

However, not all the features are equally predictive. In fact, we already know that dcw has very limited predictive power based on the results for DWC. One way to rank the features is based on their relationships with the class labels, such as information gain, relevance, correlation etc. Many machine learning algorithms, including the one used in our experiment, make use of these relationships directly when constructing their statistical model by weighting the features differently. Out of the 9 features, we discover that the algorithms consistently identify sds.3d, sdd.3d, sds.7d, and sdd.7d as the most predictive one. Actually C4.5, RIDOR, and LR even ignore the other features entirely when constructing the models.

Figure 4.4 shows the Receiver Operating Characteristic (ROC) curves for the WANDA analytics engine algorithms and DWC. These curves demonstrate the effect of varying the classification algorithm's discrimination threshold has on true positive rate (TPR) vs. true negative rate (TNR), which are calculated as

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}, \quad \mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{TN} + \mathrm{FP}}$$

A perfect algorithm will produce a point at the top left corner of the figure where TPR = 1 and FPR = 0. On the other hand, an algorithm that guesses randomly will produce a curve that follows the dotted 45-degree line in the figure where TPR = FPR. Therefore, the area under the curve (AUC) is often used as

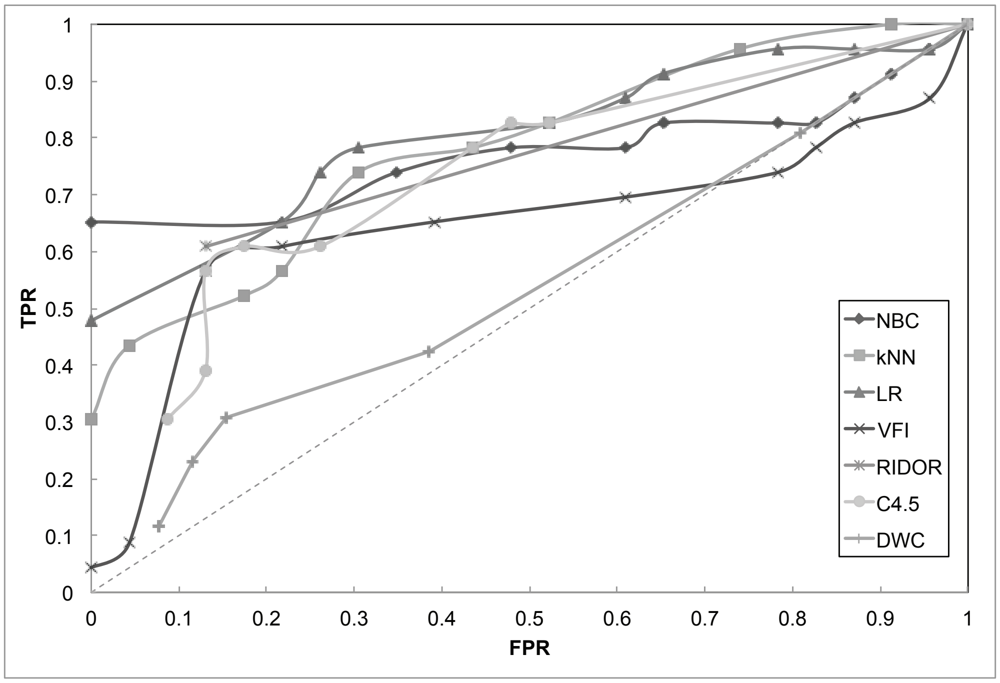


Figure 4.4: ROC curve for NBC, kNN, LR, VFI, RIDOR, and C4.5 vs. DWC

a metric to gauge the performance of a classification algorithm.

It is clear from the figure that DWC is not significantly better than random guessing. In fact, the AUC for DWC is estimated to be 0.573, which is less than 8% higher than that of a completely random guess. In comparison, the other algorithms from the WANDA analytics engine perform significantly better. The AUC ranged from 0.817 for LR to 0.621 for VFI. It is worth noting that NBC can be tuned to achieve a decent TPR of 0.652 while minimizing FPR to 0 at the same time. This makes NBC the most suitable algorithm to use when the consequence of a false alarm is more costly than that of a missed prediction.

4.3.3 Error Bound Analysis

It is often important to find out the lower bound of the classification error in order to put the accuracy of an algorithm into perspective. After all, a prediction algorithm is only as good as the features given to it. The lowest error rate achievable by any binary classifier is bounded by the Bayes Error Rate,

$$p(error) = \int_{R_2} p(x|w_1)p(w_1)\mathrm{d}x + \int_{R_1} p(x|w_2)p(w_2)\mathrm{d}x$$

where R_1 and R_2 are the regions where x is misclassified given class distributions $w1$ and $w2$. Unfortunately, the distributions are unknown in reality and can therefore only be estimated. Using the proof in [22], we can estimate the Bayes Error Rate as half of the error rate of 1-NN, which can be obtained empirically from the data.

As a result, the Bayes Error Rate is estimated to be 0.174. This implies a theoretical optimal accuracy of 0.826, which is only about 9% higher than the best accuracy achieved by the WANDA analytics engine. While this suggests that the algorithms can be better tuned, the improvement may come at a cost of reduced sensitivity or specificity. It is thus more fruitful to find features that have stronger predictive power.

The other issue is whether such an error rate is acceptable in a real clinical setting or not. This can be viewed as an optimization problem assuming the resulting cost of missing a possible intervention and the cost of waste resource due to false alarm can be estimated. By minimizing the overall cost, one can argue the if a certain error rate is acceptable. However, such kinds of optimization often neglect factors that are difficult to quantify, such as the well being of the patients. Consequently, it is important to have input from clinical experts when evaluating the error rate.

4.4 Summary

This chapter has introduced the design and implementation of a remote health monitoring system called WANDA. The system is an end-to-end solution including a smartphone-based gateway that wirelessly collects data from various sensors, a scalable cloud-based database with a RESTful web interface and PubSub system, and an analytics engine capable of prognostic prediction using machine learning and data mining algorithms. The system has also been successfully field validated from three clinical trials involving heart failure patients.

Using real data collected from the clinical trial, we have demonstrated the strength of WANDA's analytics engine by accurately predicting the worsening of HF symptoms in patients. Compared to the commonly accepted predictor of daily weight changes, the algorithms of WANDA analytics engine have identified the 3-day and 7-day fluctuation of blood pressure readings to be highly correlated to the worsening of HF symptoms. As a result, the WANDA analytics engine is able to build prediction models that are up to 73% accurate, which is more than 20% higher than using daily weight change alone. Furthermore, the sensitivity is also improved by at least 45% when using these models. Finally, we have shown that the accuracy of the WANDA analytics engine is only 9% lower than the estimated upper bound.

CHAPTER 5

Conclusion

Remote health motioning is the future of an affordable and salable health care system. Not only can it provide physicians and care givers direct and instantaneous access to a patient's complete history, the data collected from remote health monitoring can also offer the opportunity to take preventative and prognostic actions based on reliable event prediction.

In this thesis, we presented the various challenges faced in event prediction for remote health monitoring. In particular, we addressed four key issues in this area, (a) Subsequence-based prediction, (b) Sequential pattern mining, (c) precursor pattern discovery, and (d) predictions using discrete data. For each issue, we have proposed generalized algorithms to process, extract key features, and make event predictions based on the data collected from real patients or subjects.

For subsequence-based prediction, we present SmartFall, the first automatic fall detection and cause identification system based on subsequence matching. SmartFall uses data from the accelerometers embedded closely to the handle of the SmartCane to make inferences of current status. The detection algorithm differs fundamentally from most existing thresholding-based fall detection solutions in that the overall shape of the sensor signal is considered. When the shape matches the signature of a typical fall pattern, the alarm is raised and the signal immediately before and after the fall is recorded for further cause analysis. Several experiments simulating various types of fall and other common daily activities have been conducted to evaluate the fall detection performance of SmartFall. The results have indicated that SmartFall is able to detect nearly all cases of falling in the experiment while achieving extremely low false-positive rates for most non-falling activities. Similar experiments, which encompass four scenarios, trip, slip, dizziness, and loss of balance, have also been conducted to measure the accuracy of SmartFall's cause identification algorithm. A classification accuracy of 62-79% has been achieved when four scenarios are considered as independent classes and the sample is restricted to individual test subjects. The accuracy is improved to 83-93% when scenarios are grouped into intrinsic and extrinsic classes. We have also shown that DTW is more accurate than Euclidean Distance for intrinsic type of falls but less suited for extrinsic ones. Furthermore, applying a 2% constrain to DTW, instead of the commonly used 10% constrain, gives a noticeable improvement in accuracy for this particular application. Experimental results also suggest that the fall patterns generated by different test subjects are strongly correlated. This allows us to reliability classification the cause of a persons fall based on existing fall patterns generated from other subjects.

For sequential pattern mining, we have presented an algorithm called FMM to mine the Minimal Distinguishing Subsequence. FMM employs a fast and memory-efficient subsequence testing procedure, a BFS-based candidate generation scheme with powerful pruning strategies, and a simple on-the-fly minimization process. Together these changes make FMM a much more scalable algorithm than the state of the art MDS mining algorithm, ConSGapMiner. We use four real-world datasets to demonstrate the performance advantage of FMM over ConSGapMiner. The experimental results show that FMM consistently outperforms ConSGapMiner by a factor of 1.9 to 7.7 times over a wide range of δ. This is mainly due to the facts that a) FMM's support counting process scales significantly better with g_{max} and the number of sequences than the bitset-based approach used by ConSGapMiner; b) The max-suffix pruning strategy employed by FMM proves to remove the number of generated candidate sequences by 27% to 82% over a wide range of δ. The strategy also helps reduce the number of non-minimal distinguishing subsequences, which in turn speeds up the minimization process. In terms of space usage, thanks to the memory-efficient subsequence testing algorithm and on-the-fly minimization, the peak memory usage of FMM remains constant despite a 16-fold increase in the number of sequences. At the same time, ConSGapMiner needs to more than double its memory footprint to accommodate the extra sequences.

For precursor pattern discovery, we have presented a generalized algorithm that is able to discover such patterns without domain-specific knowledge and expertise. The algorithm is classifier agonistic and is applicable to a wide range of medical conditions. Experiments using three real-world datasets show that the algorithm can achieve a prediction accuracy as high as 84.7% without producing high false-positive rate. Furthermore, using the precursor patterns we are able to infer non-trivial knowledge such as the most relevant EEG channels to predict epileptic seizure and the minimal sampling rate required for predicting ICU alerts.

Finally, for predictions using discrete data, we have introduced the design and implementation of an complete remote health monitoring system called WANDA. The system is an end-to-end solution including a smartphone-based gateway that wirelessly collects data from various sensors, a scalable cloud-based database with a RESTful web interface and PubSub system, and an analytics engine capable of prognostic prediction using machine learning and data mining algorithms. The system has also been successfully field validated from three clinical trials involving heart failure patients. Using real data collected from the clinical trial, we have demonstrated the strength of WANDA's analytics engine by accurately predicting the worsening of HF symptoms in patients. Compared to the commonly accepted predictor of daily weight changes, the algorithms of WANDA analytics engine have identified the 3-day and 7-day fluctuation of blood pressure readings to be highly correlated to the worsening of HF symptoms. As a result, the WANDA analytics engine is able to build prediction models that are up to 73% accurate, which is more than 20% higher than using daily weight change alone. Furthermore, the sensitivity is also improved by at least 45% when using these models. Finally, we have shown that the accuracy of the WANDA analytics engine is only 9% lower than the estimated upper bound.

References

[1] J. Aach and G. Church. Aligning gene expression time series with time warping algorithms. *Bioinformatics*, 17:495–508, —2001—.

[2] R. Abbas, W. Aziz, and M. Arif. Prediction of ventricular tachyarrhythmia in electrocardiograph signal using neuro-wavelet approach. In *National Conference on Emerging Technologies*, page 82. Citeseer, 2004.

[3] R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. In *ACM SIGMOD Record*, volume 22, pages 207–216. ACM, 1993.

[4] R. Agrawal and R. Srikant. Mining sequential patterns. In *icde*, page 3. Published by the IEEE Computer Society, 1995.

[5] Rakesh Agrawal, Christos Faloutsos, and Arun Swami. Efficient similarity search in sequence databases. In *proceedings of the 4th International Conference on Foundations of Data Organization and Algorithms*, pages 69–84, Chicago, IL, —1993—.

[6] A. Arasteh, A. Janghorbani, and MH Moradi. Application of empirical mode decomposition in prediction of acute hypotension episodes. In *Biomedical Engineering (ICBME), 2010 17th Iranian Conference of*, pages 1–4. IEEE, 2010.

[7] L. K. Au, W. H. Wu, M. A. Batalin, D. H. McLntire, and W. J. Kaiser. Microleap: energy-aware wireless sensor platform for biomedical sensing applications. In *Biomedical Circuits and Systems Conference, 2007. BIOCAS 2007. IEEE*, pages 158–162, —2007—.

[8] Lawrence K. Au, Winston H. Wu, Maxim A. Batalin, and William J. Kaiser. Active guidance towards proper cane usage. In *BSN 2008*, Hong Kong, —2008—.

[9] R.A. Baeza-Yates. Searching subsequences. *Theoretical Computer Science*, 78(2):363–376, 1991.

[10] Y-W Bai, S-C Wu, and C-L Tsai. Design and implementation of a fall monitor system by using a 3-axis accelerometer in a smart phone. *Consumer Electronics, IEEE Transactions on*, 58(4):1269–1275, 2012.

[11] J. Bailey, T. Manoukian, and K. Ramamohanarao. Fast algorithms for mining emerging patterns. *Principles of Data Mining and Knowledge Discovery*, pages 187–208, 2002.

[12] A. Bateman, L. Coin, R. Durbin, R.D. Finn, V. Hollich, S. Griffiths-Jones, A. Khanna, M. Marshall, S. Moxon, E.L.L. Sonnhammer, et al. The pfam protein families database. *Nucleic acids research*, 32(suppl 1):D138, 2004.

[13] S.D. Bay and M.J. Pazzani. Detecting change in categorical data: Mining contrast sets. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 302–306. ACM, 1999.

[14] D. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In *AAAI94 workshop on knowledge discovery in databases*, pages 359–370, 1994.

[15] W. Blount. Don't throw away the cane. *Journal of Bone & Joint Surgery*, 38:695–708, —1956—.

[16] A.K. Bourke and G.M. Lyons. A threshold-based fall-detection algorithm using a bi-axial gyroscope sensor. *Medical Engineering & Physics*, 30:84–90, —2006—.

[17] K. Chan and Fu. A. W. Efficient time series matching by wavelets. In *proceedings of the 15th IEEE International Confernce on Data Engineering*, pages 126–133, Sydney, Australia, —1999—.

[18] S. Chan, B. Kao, C.L. Yip, and M. Tang. Mining emerging substrings. In *Database Systems for Advanced Applications, 2003.(DASFAA 2003). Proceedings. Eighth International Conference on*, pages 119–126. IEEE.

[19] S.I. Chaudhry, B. Barton, J. Mattera, J. Spertus, and H.M. Krumholz. Randomized trial of telemonitoring to improve heart failure outcomes (tele-hf): study design. *Journal of cardiac failure*, 13(9):709–714, 2007.

[20] S.I. Chaudhry, Y. Wang, J. Concato, T.M. Gill, and H.M. Krumholz. Patterns of weight change preceding hospitalization for heart failure. *Circulation*, 116(14):1549–1554, 2007.

[21] J. Chen, K. Kwong, D. Chang, J. Luk, and R. Bajcsy. Wearable sensors for reliable fall detection. In *Engineering in Medicine and Biology Society, 2005. IEEE-EMBS 2005. 27th Annual International Conference of the*, pages 3551–3554, —2005—.

[22] T. Cover and P. Hart. Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, 13(1):21–27, 1967.

[23] Jiangpeng Dai, Xiaole Bai, Zhimin Yang, Zhaohui Shen, and Dong Xuan. Mobile phone-based pervasive fall detection. *Personal and ubiquitous computing*, 14(7):633–643, 2010.

[24] T. Degen, H. Jaeckel, M. Rufer, and S. Wyss. Speedy: a fall detector in a wrist watch. In *Proceedings of the 7th IEEE international Symposium on Wearable Computers7th IEEE international Symposium on Wearable Computers*, page 184, Washington, D. C., —2003—.

[25] Akshay S. Desai. Home monitoring heart failure care does not improve patient outcomes: Looking beyond telephone-based disease management. *Circulation*, 125:828–836, 2012.

[26] A.S. Desai and L.W. Stevenson. Connecting the circle from home to heart-failure disease management. *New England Journal of Medicine*, 363(24):2364–2367, 2010.

[27] G. Dong, X. Zhang, L. Wong, and J. Li. Caep: Classification by aggregating emerging patterns. In *Discovery Science*, pages 737–737. Springer, 1999.

[28] Guozhu Dong and Jinyan Li. Efficient mining of emerging patterns: discovering trends and differences. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '99, pages 43–52, New York, NY, USA, 1999. ACM.

[29] K. Doughty, R. Lewis, and A. McIntosh. The design of a pratical and reliable fall detector for community and institutional telecare. *Journal of Telemedicine and Telecare*, 6:S1:150–S1:154, —2000—.

[30] R. Esteller, J. Echauz, M. D'Alessandro, G. Worrell, S. Cranstoun, G. Vachtsevanos, and B. Litt. Continuous energy variation during the seizure cycle: towards an on-line accumulated energy. *Clinical neurophysiology*, 116(3):517–526, 2005.

[31] A. Farmer, O. Gibson, P. Hayton, K. Bryden, C. Dudley, A. Neil, and L. Tarassenko. A real-time, mobile phone-based telemedicine system to support young adults with type 1 diabetes. *Informatics in primary care*, 13(3):171–178, 2005.

[32] R. Fielding. Representational state transfer (rest). *Architectural Styles and the Design of Network-based Software Architectures. University of California, Irvine*, page 120, 2000.

[33] E. Frank, M. Hall, L. Trigg, G. Holmes, and I.H. Witten. Data mining in bioinformatics using weka. *Bioinformatics*, 20(15):2479, 2004.

[34] A.L. Goldberger, L.A.N. Amaral, L. Glass, J.M. Hausdorff, P.C. Ivanov, R.G. Mark, J.E. Mietus, G.B. Moody, C.K. Peng, and H.E. Stanley. Physiobank, physiotoolkit, and physionet: Components of a new research resource for complex physiologic signals. *Circulation*, 101(23):e215–e220, 2000.

[35] N.F. Güler, E.D. Übeyli, and İ. Güler. Recurrent neural networks employing lyapunov exponents for eeg signals classification. *Expert Systems with Applications*, 29(3):506–514, 2005.

[36] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten. The weka data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

[37] J. Han, H. Cheng, D. Xin, and X. Yan. Frequent pattern mining: current status and future directions. *Data Mining and Knowledge Discovery*, 15(1):55–86, 2007.

[38] M. Hirao, H. Hoshino, A. Shinohara, M. Takeda, and S. Arikawa. A practical algorithm to find the best subsequence patterns. In *Discovery Science*, pages 141–154. Springer, 2000.

[39] S.A. Hunt, D.W. Baker, M.H. Chin, M.P. Cinquegrani, A.M. Feldman, G.S. Francis, T.G. Ganiats, S. Goldstein, G. Gregoratos, M.L. Jessup, et al. Acc/aha guidelines for the evaluation and management of chronic heart failure in the adult: executive summary: a report of the american college of cardiology/american heart association task force on practice guidelines (committee to revise the 1995 guidelines for the evaluation and management of heart failure) developed in collaboration with the international society for heart and lung transplantation endorsed by the heart failure society of america. *Journal of the American College of Cardiology*, 38(7):2101, 2001.

[40] JY Hwang, JM Kang, YW Jang, and HC Kim. Development of novel algorithm and real-time monitoring ambulatory system using bluetooth module for fall detection in the elderly. In *Engineering in Medicine and Biology Society, 2004. IEMBS'04. 26th Annual International Conference of the IEEE*, volume 1, pages 2204–2207. IEEE, 2004.

[41] S. Inenaga, H. Bannai, A. Shinohara, M. Takeda, and S. Arikawa. Discovering best variable-length-dont-care patterns. In *Discovery Science*, pages 169–216. Springer, 2002.

[42] Frederick Jelinek. *Statistical methods for speech recognition.* The MIT Press, Cambridge, MA, —1998—.

[43] S.F. Jencks, M.V. Williams, and E.A. Coleman. Rehospitalizations among patients in the medicare fee-for-service program. *New England Journal of Medicine*, 360(14):1418–1428, 2009.

[44] X. Ji, J. Bailey, and G. Dong. Mining minimal distinguishing subsequence patterns with gap constraints. *Knowledge and Information Systems*, 11(3):259–286, 2007.

[45] Xiaonan Ji, James Bailey, and Guozhu Dong. Mining minimal distinguishing subsequence patterns with gap constraints. *Data Mining, IEEE International Conference on*, 0:194–201, 2005.

[46] Pekka Kannus, Jari Parkkari, Seppo Niemi, and Mika Palvanen. Fall-induced deaths among elderly people. *American Journal of Public Health*, 95(3):422–434, —2005—.

[47] E. Keogh, K. Chakrabati, M. Pazzani, and S. Mehrota. Dimensionality reduction for fast similarity search in large time series database. *KAIS Journal*, 3:263–286, —2001—.

[48] E. Keogh and S. Kasetty. On the need for time series data mining benchmarks: a survey and empirical demonstration. *Data Mining and Knowledge Discovery*, 7(4):349–371, 2003.

[49] Mary B. King and Mary E. Tinetti. Falls in community-dwelling older persons. *Journal of the American Geriatrics Society*, 44(8):1010, 1996.

[50] K Koski, H Luukinen, P Laippala, and SL Kivela. Physiological factors and medications as predictors of injurious falls by elderly people: a prospective population-based study. *Age and Ageing*, 25:29–38, 1996.

[51] Z.M. Kovacs-Vajna. A fingerprint verification system based on triangular matching and dynamic time warping. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1266–1276, —2000—.

[52] K. Lehnertz and C.E. Elger. Can epileptic seizures be predicted? evidence from nonlinear time series analysis of brain electrical activity. *Physical Review Letters*, 80(22):5019–5022, 1998.

[53] J. Lewin, M. Ledwidge, C. O'Loughlin, C. McNally, and K. McDonald. Clinical deterioration in established heart failure: What is the value of bnp and weight gain in aiding diagnosis? *European journal of heart failure*, 7(6):953–957, 2005.

[54] J. Li, G. Dong, and K. Ramamohanarao. Making use of the most expressive jumping emerging patterns for classification. *Knowledge and Information systems*, 3(2):131–145, 2001.

[55] J. Li, K. Ramamohanarao, and G. Dong. The space of jumping emerging patterns and its incremental maintenance algorithms. In *Proceedings of the Seventeenth International Conference on Machine Learning, Stanford, CA, USA*, pages 551–558. Citeseer, 2000.

[56] J. Lin and E. Keogh. Group sax: Extending the notion of contrast sets to time series and multimedia data. *Knowledge Discovery in Databases: PKDD 2006*, pages 284–296, 2006.

[57] J. Lin, E. Keogh, S. Lonardi, and B. Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 2–11. ACM, 2003.

[58] J. Lin, E. Keogh, S. Lonardi, and B. Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, page 11. ACM, 2003.

[59] U. Lindemann, A. Hock, M. Stuber, W. Keck, and C. Becker. Evaluation of a fall detector based on accelerometers: A pilot study. *Medical and Biological Engineering and Computing*, 43(5):548–551, 2005.

[60] N. Liu, Z. Lin, Z. Koh, G.B. Huang, W. Ser, and M.E.H. Ong. Patient outcome prediction with heart rate variability and vital signs. *Journal of Signal Processing Systems*, 64(2):265–278, 2011.

[61] D. Lloyd-Jones, R.J. Adams, T.M. Brown, M. Carnethon, S. Dai, G. De Simone, T.B. Ferguson, E. Ford, K. Furie, C. Gillespie, et al. Executive summary: heart disease and stroke statistics–2010 update: a report from the american heart association. *Circulation*, 121(7):948, 2010.

[62] C. J. Lord and D. P. Colvin. Falls in the elderly: Detection and assessment. In *Engineering in Medicine and Biology Society, 1991. Vol.13: 1991., Proceedings of the Annual International Conference of the IEEE*, pages 1938–1939.

[63] DNP Marisa Ferrari, Barbara Harrison, Osamah Rawashdeh, Muawea Rawashdeh, Robert Hammond, and Michael Maddens. A pilot study testing a fall prevention intervention for older adults: determining the feasibility of a five-sensor motion detection system. *Journal of Gerontological Nursing*, 38(1):13–16, 2012.

[64] F. Mormann, R.G. Andrzejak, C.E. Elger, and K. Lehnertz. Seizure prediction: the long and winding road. *Brain*, 130(2):314, 2007.

[65] A. Mortara, G.D. Pinna, P. Johnson, R. Maestri, S. Capomolla, M.T. La Rovere, P. Ponikowski, L. Tavazzi, and P. Sleight. Home telemonitoring in heart failure patients: the hhh study (home or hospital in heart failure). *European journal of heart failure*, 11(3):312–318, 2009.

[66] Muhammad Mubashir, Ling Shao, and Luke Seed. A survey on fall detection: Principles and approaches. *Neurocomputing*, 2012.

[67] N. Noury, A. Fleury, P. Rumeau, A.K. Bourke, G.O. Laighin, V. Rialle, and JE Lundy. Fall detection-principles and methods. In *Engineering in*

Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE, pages 1663–1666. IEEE, 2007.

[68] P.K. Novak, N. Lavrač, and G.I. Webb. Supervised descriptive rule discovery: A unifying survey of contrast set, emerging pattern and subgroup mining. *The Journal of Machine Learning Research*, 10:377–403, 2009.

[69] S. Nowozin, G. BakIr, and K. Tsuda. Discriminative subsequence mining for action classification. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

[70] C. Orwat, A. Graefe, and T. Faulwasser. Towards pervasive computing in health care–a literature review. *BMC Medical Informatics and Decision Making*, 8(1):26, 2008.

[71] R. Paradiso, G. Loriga, and N. Taccini. A wearable health care system based on knitted integrated sensors. *Information Technology in Biomedicine, IEEE Transactions on*, 9(3):337–344, 2005.

[72] C.A. Ratanamahatana and E. Keogh. Everything you know about dynamic time warping is wrong. In *Third Workshop on Mining Temporal and Sequential Data*, pages 22–25, 2004.

[73] John A. Rizzo, Rebecca Friedkin, Christianna S. Williams, Janett Nabors, Denise Acampora, and Mary E. Tinetti. Health care utilization and costs in a medicare population by fall status. *Medical Care*, 36(8):1174–1188, —1998—.

[74] HJ Robertson, JJ Soraghan, C. Idzikowski, and BA Conway. Emd and pca for the prediction of sleep apnoea: a comparative study. In *Signal Processing and Information Technology, 2007 IEEE International Symposium on*, pages 419–424. IEEE, 2007.

[75] J.S. Ross, J. Chen, Z. Lin, H. Bueno, J.P. Curtis, P.S. Keenan, S.L.T. Normand, G. Schreiner, J.A. Spertus, M.T. Vidán, et al. Recent national trends in readmission rates after heart failure hospitalization. *Circulation: Heart Failure*, 3(1):97–103, 2010.

[76] L.Z. Rubenstein. Falls in older people: epidemiology, risk factors and strategies for prevention. *Age and Ageing*, 35:ii37–ii41, —2006—.

[77] L.Z. Rubenstein, K.R. Josephson, and A.S. Robbins. Falls in the nursing home. *Annals of Internal Medicine*, 121:442–451, —1994—.

[78] A. Shinohara, M. Takeda, S. Arikawa, M. Hirao, H. Hoshino, and S. Inenaga. Finding best patterns practically. *Progress in Discovery Science*, pages 20–23, 2002.

[79] O.Z. Soran, A.M. Feldman, I.L. Piña, G.A. Lamas, S.F. Kelsey, F. Selzer, J. Pilotte, and J.R. Lave. Cost of medical services in older patients with heart failure: those receiving enhanced monitoring using a computer-based telephonic monitoring system compared with those in usual care: the heart failure home care trial. *Journal of cardiac failure*, 16(11):859–866, 2010.

[80] O.Z. Soran, I.L. Piña, G.A. Lamas, S.F. Kelsey, F. Selzer, J. Pilotte, J.R. Lave, and A.M. Feldman. A randomized clinical trial of the clinical effects of enhanced heart failure monitoring using a computer-based telephonic monitoring system in older minorities and women. *Journal of cardiac failure*, 14(9):711–717, 2008.

[81] Aino Sorvala, Esko Alasaarela, Hannu Sorvoja, and R Myllyla. A two-threshold fall detection algorithm for reducing false alarms. In *Medical Information and Communication Technology (ISMICT), 2012 6th International Symposium on*, pages 1–4. IEEE, 2012.

[82] M. Spiliopoulou. Web usage mining for web site evaluation. *Communications of the ACM*, 43(8):127–134, 2000.

[83] Judy A. Stevens and Sarah Olson. Reducing falls and resulting hip fractures among older women. *Home Care Provider*, 5(4):131–141, 2000.

[84] Z. Struzik and A. Siebes. The haar wavelet transform in the time series similarity paradigm. *Principles of Data Mining and Knowledge Discovery*, pages 12–22, 1999.

[85] M. Suh, C.A. Chen, J. Woodbridge, M.K. Tu, J.I. Kim, A. Nahapetian, L.S. Evangelista, and M. Sarrafzadeh. A remote patient monitoring system for congestive heart failure. *Journal of medical systems*, pages 1–15, 2011.

[86] M. Suh, L.S. Evangelista, V. Chen, W.S. Hong, J. Macbeth, A. Nahapetian, F.J. Figueras, and M. Sarrafzadeh. Wanda b.: Weight and activity with blood pressure monitoring system for heart failure patients. In *World of Wireless Mobile and Multimedia Networks (WoWMoM), 2010 IEEE International Symposium on a*, pages 1–6. IEEE, 2010.

[87] M. Takeda, S. Inenaga, H. Bannai, A. Shinohara, and S. Arikawa. Discovering most classificatory patterns for very expressive pattern classes. In *Discovery Science*, pages 486–493. Springer, 2003.

[88] M.E. Tinetti, D.I. Baker, G. McAvay, E.B. Claus, P. Garrett, and M. Gottschalk. A multifactorial intervention to reduce the risk of falling among elderly people living in the community. *New England Journal of Medicine*, 311:821–827, 1994.

[89] M.E. Tinetti and M. Speechley. Prevention of falls among the elderly. *New England Journal of Medicine*, 320:1055–1059, —1989—.

[90] Vo Quang Viet, Gueesang Lee, and Deokjai Choi. Fall detection based on movement and smart phone technology. In *Computing and Communication Technologies, Research, Innovation, and Vision for the Future (RIVF), 2012 IEEE RIVF International Conference on*, pages 1–4. IEEE, 2012.

[91] G. Williams, K. Doughty, K. Cameron, and D. A. Bradley. A smart fall and activity monitor for telecare applications. In *Engineering in Medicine and Biology Society, 1998. Proceedings of the 20th Annual International Conference of the IEEE*, volume 3, pages 1151–1154 vol.3.

[92] W. Wu, L. Au, B. Jordan, T. Stathopoulos, M. Batalin, W. Kaiser, A. Vahdatpour, M. Sarrafzadeh, M. Fang, and J. Chodosh. The smartcane system: an assistive device for geriatrics. In *Proceedings of the ICST 3rd international conference on Body area networks*, page 2. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.

[93] B. Yi and Christos Faloutsos. Fast time sequence indexing for arbitrary lp norms. In *proceedings of the 26th International Conference on Very Large Databases*, pages 385–394, Cairo, Egypt, —2000—.

[94] H.W. Yien, S.S. Hseu, L.C. Lee, T.B.J. Kuo, T.Y. Lee, and S.H.H. Chan. Spectral analysis of systemic arterial pressure and heart rate signals as a prognostic tool for the prediction of patient outcome in the intensive care unit. *Critical care medicine*, 25(2):258, 1997.

[95] Mitchell Yuwono, Bruce D Moulton, Steven W Su, Branko G Celler, Hung T Nguyen, et al. Unsupervised machine-learning method for improving the performance of ambulatory fall-detection systems. *Biomedical engineering online*, 11(1):1–11, 2012.

[96] O.R. Zaıane, K. Yacef, and J. Kay. Finding top-n emerging sequences to contrast sequence sets. Technical report, Citeseer, 2007.

[97] M.J. Zaki. Sequence mining in categorical domains: incorporating constraints. In *Proceedings of the ninth international conference on Information and knowledge management*, pages 422–429. ACM, 2000.

[98] J. Zhang, K.M. Goode, P.E. Cuddihy, and J.G.F. Cleland. Predicting hospitalization due to worsening heart failure using daily weight measurement: analysis of the trans-european network-home-care management system (tenhms) study. *European journal of heart failure*, 11(4):420–427, 2009.

[99] X. Zhang, G. Dong, and R. Kotagiri. Exploring constraints to efficiently mine emerging patterns from large high-dimensional datasets. In *Proceedings*

of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 310–314. ACM, 2000.