卒業論文 2007 年度 (平成 19 年度)

# Life$^2$Guard：私的空間におけるセンサ群を用いた身体異常検知手法の研究


指導教員

慶應義塾大学環境情報学部

徳田　英幸

村井　純

楠本　博之

中村　修

高汐　一紀

重近　範行

湧川　隆次

Van Meter, Rodney D

慶應義塾大学 環境情報学部

生天目 直哉

*namachan@ht.sfc.keio.ac.jp*

# 卒業論文要旨 2007 年度 (平成 19 年度)

# Life$^2$Guard：私的空間におけるセンサ群を用いた身体異常検知手法の研究

本論文では，複数の環境情報センサのデータからコンテクスト抽出を行なうことで，カメラの利用が難しい私的空間で身体異常を検知する手法を提案する．

近年，浴室やトイレなどで倒れてしまうケースが数多く報告されている．浴室やトイレは通常 1 人で利用し，閉鎖されている状態なので，このような危険な状況の発見が遅れてしまい，手遅れになってしまうことも多くある．そのため，このような場所での状況監視の必要性が高まっている．しかし，浴室やトイレはプライバシーを保護しなくてはならないため，現在，監視の手法として広く用いられているカメラは利用できない．

そこで，本研究では，センサのデータから状態抽出を行なうことで，日常生活でのプライバシー (Life) と健康リスク (Life) の両者を守る手法，およびアプリケーション「Life$^2$Guard」を提案し実装を行なう．本研究は，センサが取得する個々の情報が身体異常へと直結しない環境情報センサを複数用いて，利用者の振る舞いを検知し，その振る舞いが異常であるか，正常であるかを判断する．そうすることで，利用者はシステムから何も制約を受けることない生活を送りながら，健康リスクを回避することができる．

本研究では，既存の状況や状態を監視する手法を身体異常状態の検知に適応させるための機能要件を明確にした．それらの機能要件は，環境情報センサによる身体センシング，動作信頼性の確保，実時間推論の実現，誤検知による通報回避の 4 つとした．本論文では，それらを実現する手法を提案し，トイレという私的空間で実験を行なった．

具体的なアプローチとして，環境情報センサによる身体センシングには焦電型の赤外線センサを複数個用いた．焦電型赤外線センサは赤外線を発する生体の行動に反応するため，環境の変化に影響を受けにくく，低コストで利用者の行動を取得することができる．更に，複数個用いることで，パターンを解析することが可能になり，様々な身体異常状態を検知することが出来る．次に動作信頼性の確保は，センサデータの欠落も補完できるベイジアンネットワークを利用することで解決した．実時間推論は近似推論アルゴリズムの利用と，推論モデルの単純化により実現することが出来た．最後に，誤検知が生じた際には利用者に音により状況を問いかけ，ボタンを押してもらうというインタラクションを用いることで，間違った通報を回避した．

本論文で私的空間の具体例であるトイレを対象とし，Life$^2$Guard システムの具体的な実装を行ない，この手法に対する評価行なった結果と考察を述べる．

**慶應義塾大学 環境情報学部**

**生天目 直哉**

# Abstract of Bachelor's Thesis Academic Year 2007

# Life$^2$Guard : A Physical Disorder Detection with Sensors in Privacy Rooms

We propose a method that analyzes the context from environmental sensor information to detect physical disorders in privacy spaces where cameras can not be used.

These days, it is reported that many people are attacked a sudden physical disorder in bathrooms or restrooms caused by a rapid blood pressure change. Since people use such places alone and they are isolated from others, these situations tend to be left undiscovered and the time loss often leads the person to death. Monitoring system for those spaces is extreamly disired. However, rooms such as restrooms are very "privacy-sensitive" spaces. Therefore the monitoring methods using cameras can not be applied to such spaces.

We propose "Life$^2$Guard" System which is an application to protect both users privacy on "Life" and physical risk on "Life" by analyzing context from sensor data. In the research, we use multiple environmental sensors to detect a user's behavior and find out he/she is in physical disorder or not. Unlike a data from vital sensors, a data from environmental sensors do not represent a physical disorders when looking at a single point. However, environmental sensors do not require users to carry with them. This frees carrying sensors all the time with them to receive the service.

To realize a monitoring system for sudden physical disorders, we have found four system requirements namely environmental human sensing, strong system durability, realizeing real-time inference, and false-pistive avoidance. "Life$^2$Guard System" clears all the requirements by following keyword: IR-Sensor, Bayesian Network, Naive Bayes, Approximate Inference, and Confirm Button Interaction.

IRSensor is cost effective and detects only the user behavior. Bayesian Network is an algorithm which compensates the lack of sensor data and prevent the algorithm being collapsing. Naive Bayes Model can save the training cost and reduces miss definition of the context. Approximate Inference can handle a complicated model within real-time. Finally requiring user interaction using Confirm Button in case of potential physical disorder avoids emergency call by misinference.

In this research, we focus on a restroom as a concrete example of privacy space and implement Life$^2$Guard System. We also have a demonstration experiment in actual restroom and evaluate the accuracy of the system.

**Naoya Namatame**

**Faculty of Environmental Information**
**Keio University**

# 目次

**第 1 章　序論** 1
1.1　本研究の背景 . . . 1
1.2　本研究の問題意識 . . . 2
1.3　本研究の目的 . . . 2
1.4　本論文の構成 . . . 3

**第 2 章　状況監視と身体異常** 4
2.1　状況監視 . . . 4
2.1.1　公共空間の状況監視 . . . 4
2.1.2　制約空間での状況監視 . . . 5
2.1.3　私的空間での状況監視 . . . 6
2.2　身体異常 . . . 7
2.2.1　身体異常の定義 . . . 7
2.2.2　身体異常の特徴 . . . 7
2.3　身体異常状態の監視 . . . 7
2.3.1　機能要件 . . . 8
2.3.2　シナリオ . . . 9
2.3.3　本章のまとめ . . . 9

**第 3 章　身体異常状態の抽出手法** 10
3.1　環境情報センサによる身体センシング . . . 10
3.1.1　利用するセンサ . . . 10
3.1.2　センサの配置 . . . 11
3.1.3　センサのサンプリング周波数 . . . 11
3.1.4　身体異常取得イメージ . . . 12
3.2　動作信頼性の確保 . . . 13
3.2.1　ハードウェアの冗長設置 . . . 13
3.2.2　ベイジアンネットワークの利用 . . . 14
3.3　実時間推論の実現 . . . 14
3.3.1　近似推論アルゴリズムの利用 . . . 15
3.3.2　推論モデルの単純化 . . . 15
3.4　誤検知通報の回避 . . . 17

3.4.1 検知漏れの回避 . . . 17
3.4.2 利用者のインタラクション . . . 17
3.4.3 フィードバックによる強化学習 . . . 18
3.5 本章のまとめ . . . 18

**第 4 章 Life²Guard システム設計・実装** 19
4.1 Life²Guard システムの設計 . . . 19
4.2 Life²Guard システムの設計概要 . . . 19
4.2.1 ハードウェア構成 . . . 19
4.2.2 ソフトウェア構成 . . . 19
4.3 Life²Guard システムの動作概要 . . . 20
4.3.1 センサコントロールモジュール動作概要 . . . 20
4.3.2 状態抽出モジュール動作概要 . . . 22
4.3.3 環境の状態による確率変数スイッチング . . . 22
4.3.4 アクチュエーションモジュール動作概要 . . . 24
4.3.5 学習モジュール動作概要 . . . 24
4.4 Life²Guard システムの実装 . . . 25
4.4.1 センサの配置 . . . 25
4.4.2 推論モデルの作成 . . . 26
4.4.3 センサノードの実装 . . . 27
4.4.4 マネージメントセンサの実装 . . . 29
4.4.5 アクチュエーターの実装 . . . 29
4.5 ベイズ推論エンジンの実装 . . . 30
4.5.1 ベイズ推論モデルの定義 . . . 30
4.5.2 ベイズ推論エンジンの多態化 . . . 30
4.5.3 ベイズ推論エンジンの利用方法 . . . 31
4.5.4 ベイズ推論アルゴリズムの実装 . . . 32
4.6 センサデータ操作クラスの実装 . . . 33
4.6.1 センサデータ操作の統一化 . . . 33
4.6.2 センサデータ統一操作クラスの利用方法 . . . 34
4.7 本章のまとめ . . . 35

**第 5 章 評価** 36
5.1 ベイズ近似推論の収束サンプリング回数評価 . . . 36
5.2 身体異常状態の抽出精度評価 . . . 37
5.2.1 実験環境 . . . 37
5.2.2 ソフトウェア設定 . . . 38
5.2.3 検出目標状態の定義 . . . 38
5.2.4 実験手順 . . . 40
5.2.5 実験結果 . . . 40

5.2.6 実験考察 . . . 43
5.3 ユーザビリティ評価 . . . 44
5.3.1 実験結果 . . . 44
5.3.2 実験考察 . . . 44
5.4 本章のまとめ . . . 45

**第 6 章 関連研究** 46
6.1 センサとコンテクストアウェア・アプリケーション . . . 46
6.1.1 ウェアラブルコンピュータによる実現手法の研究 . . . 46
6.1.2 空間に配置された環境情報センサによる実現手法の研究 . . . 48
6.2 類似研究 . . . 49
6.2.1 Sensorized Elderly Care Home の研究 . . . 49
6.2.2 House_n の研究 . . . 49
6.2.3 uEyes の研究 . . . 51
6.3 本章のまとめ . . . 52

**第 7 章 結論** 53
7.1 本研究のまとめ . . . 53
7.2 本研究の貢献 . . . 54
7.3 今後の課題と展望 . . . 54

**参考文献** 56

**付録 A ベイジアンネットワーク** 58
A.1 ベイジアンネットワーク . . . 58
A.1.1 ベイジアンネットワークの構造 . . . 59
A.1.2 ノードと変数 . . . 59
A.1.3 リンク構造 . . . 60
A.1.4 条件付確率テーブル . . . 60
A.2 ベイジアンネットワークにおける推論 . . . 60
A.2.1 推論パターンの分類 . . . 60
A.2.2 結合確率の算出 . . . 61
A.2.3 ベイズの定理 . . . 61
A.2.4 推論アルゴリズムの分類 . . . 62
A.3 ベイジアンネットワークにおける学習 . . . 62
A.3.1 機械学習 . . . 62
A.3.2 学習の分類 . . . 63

**付録 B BayesEngine 抽象クラスの実装** 64

**付録 C LogicSampling クラスの実装** 67

**付録** D　LikelihoodWeighting **クラスの実装**　69

# 図目次


1.1 トイレ内における血圧変動グラフ . . . . . 1
1.2 プライベート空間 1:トイレ . . . . . 3
1.3 プライベート空間 2:風呂場 . . . . . 3

2.1 防犯カメラ . . . . . 4
2.2 ウェブカメラ . . . . . 4
2.3 皮膚温センサ (右) とパルスオキシメータ (左) . . . . . 6
2.4 セコムの防犯センサ . . . . . 6
2.5 防火用の煙感知器 . . . . . 6
2.6 通常状態と異常状態の定義 . . . . . 7
2.7 カーラーの救命曲線 . . . . . 8

3.1 焦電型赤外線センサのパルス例 . . . . . 12
3.2 通常状態と異常コンテクスト . . . . . 12
3.3 多階層推論モデル . . . . . 16
3.4 2 階層推論モデル . . . . . 16
3.5 身体異常状態の推論モデル . . . . . 16
3.6 誤検知回避手法のフローチャート . . . . . 17

4.1 ハードウェア構成図 . . . . . 20
4.2 ソフトウェア構成図 . . . . . 20
4.3 環境情報スナップショット作成機構 . . . . . 22
4.4 トイレに設置されたセンサ . . . . . 25
4.5 トイレ外に設置された解析サーバ . . . . . 25
4.6 Life$^2$Guard システム実行画面 . . . . . 25
4.7 焦電型赤外線センサ設置例 (全体) . . . . . 26
4.8 焦電型赤外線センサ設置例 (単体) . . . . . 26
4.9 2 軸加速度設置例 . . . . . 26
4.10 トイレにおける身体異常状態の推論モデル . . . . . 26
4.11 Mica2 Dot . . . . . 27
4.12 焦電型赤外線センサ NaPiOn 狭視野タイプ . . . . . 28
4.13 NaPiOn 視野角 . . . . . 28
4.14 センサモジュールの実装例 . . . . . 28

4.15 センサのアクチュエーション用スイッチインタフェース . . . . . 29
4.16 アクチュエータ . . . . . 29

5.1 サンプリング回数と計算結果 . . . . . 36
5.2 サンプリング回数と計算結果の分散 . . . . . 37
5.3 Vaio TypeU の内臓カメラ . . . . . 38
5.4 Vaio TypeU の内臓カメラを隠した様子 . . . . . 38
5.5 座って気絶している状態の例 . . . . . 39
5.6 倒れて気絶している状態の例 . . . . . 39
5.7 心停止から除細動までの時間と心拍再開率の関係 . . . . . 40

6.1 SmartShoes . . . . . 47
6.2 An apartment with sensor installed . . . . . 48
6.3 uEyes の実行画面 . . . . . 51

A.1 肺がんにおけるベイジアンネットワーク . . . . . 59
A.2 肺がん問題におけるレントゲン結果の条件付確率テーブル . . . . . 60
A.3 予測的推論の例 . . . . . 61
A.4 診断的推論の例 . . . . . 61

# 表目次


1.1 浴槽内での水死者数 (人口動態統計) . . . . . 2

3.1 センサ種類と 3 性質の評価 . . . . . 11
3.2 通常状態でうずくまって座っている時の状況パターンの一例 . . . . . 13
3.3 身体異常でうずくまって座っている時の状況パターンの一例 . . . . . 13

4.1 身体異常状態の推論ノード . . . . . 27
4.2 Mica2 Dot のデータ表 . . . . . 28
4.3 BayesEngine 抽象クラス . . . . . 30
4.4 LikelihoodWeighting クラス . . . . . 31
4.5 SensorData クラス . . . . . 33
4.6 SensorDataConverter クラス . . . . . 34

5.1 推論サーバデータ . . . . . 37
5.2 被験者データ . . . . . 38
5.3 学習データ . . . . . 38
5.4 倒れて気絶している状態の検知制度 (左:60 秒, 右:30 秒) . . . . . 41
5.5 倒れて気絶している状態の検知結果 . . . . . 41
5.6 座って気絶している状態の検知精度 (左:60 秒, 右:30 秒) . . . . . 41
5.7 座って気絶している状態の検知結果 . . . . . 41
5.8 倒れて痙攣している状態の検知精度 (左:60 秒, 右:30 秒) . . . . . 41
5.9 倒れて気絶している状態の検知結果 . . . . . 42
5.10 座って痙攣している状態の検知精度 (左:60 秒, 右:30 秒) . . . . . 42
5.11 座って気絶している状態の検知結果 . . . . . 42
5.12 気絶状態の検知精度 (左:60 秒, 右:30 秒) . . . . . 42
5.13 痙攣状態の検知精度 (左:60 秒, 右:30 秒) . . . . . 42

A.1 肺がん問題におけるノードと変数 . . . . . 59

# 第 1 章

# 序論

## 1.1　本研究の背景

脳卒中や心筋梗塞などの疾患は，突然発作が起こり，さらに発作が起こっている本人が助けを呼ぶことが難しい．しかし，これらの疾患は，一人になってしまうトイレや風呂場において高い確率で発生することが統計的に報告されている．排泄時に伴ういきみが引き起こす血圧の急上昇や，ヒートショックと呼ばれる急な気温差が引き起こす血圧の急変動が主な原因といわれている．図 1.1 に示すグラフは，人がトイレに入って，退出するまでの血圧の変動のグラフである．いきみに伴う血圧の変動は大きいときで 140mmHg から 210mmHg に急変動する [1].

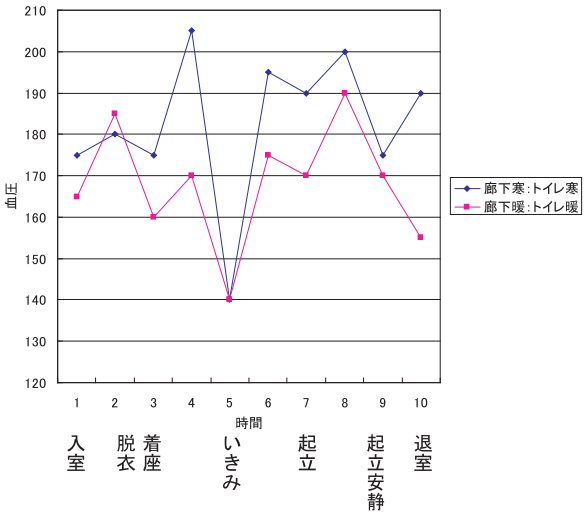


図 1.1　トイレ内における血圧変動グラフ

さらに，2000 年の厚生労働省の統計 [2] では，浴室でなくなった人は 3518 人，また，入浴前後，脱衣所，洗い場，風呂上り直後を含めると，14000 人に上る．交通事故で年間に亡くなってしまう人数が平成 17 年度に全国で 6871 人であることを考えると，この問題はとても重大であることがわかる．このような事故は心肺停止から発見されるまで時間がかかるため，多くの場合が蘇生されることがないという．この事実がこのような数字の増加につながっている要因と見受けられる．さらに，表 1.1 の 89 年から 99 年までのデータを見ると，

浴室内での水死者数における 65 歳以上の高齢者の割合が増加の一途をたどっていることから，今後高齢化社会が進むにつれて，この問題がより重大なものになってくることが予測できる．

表 1.1　浴槽内での水死者数 (人口動態統計)

| 年 | 全体 (人数) | 65 歳以上 (人数) | 65 歳以上の割合 (%) |
|---|---|---|---|
| 89 | 1196 | 781 | 65.3 |
| 90 | 1385 | 951 | 68.7 |
| 91 | 1608 | 1153 | 71.7 |
| ... | ... | ... | ... |
| 96 | 3077 | 2526 | 82.1 |
| 97 | 3008 | 2479 | 82.4 |
| 98 | 3078 | 2563 | 83.3 |
| 99 | 3345 | 2792 | 83.5 |

現在は，暖房などの機器を使って，浴室やトイレとの気温差をなくすことで，ヒートショックを和らげようというソリューションが数多く提案されている．しかし，入浴や力みによる血圧の変動はなかなか抑えられるものではない．トイレで力むことで，血圧が基準値の人でも，一時的に血圧が 200mmHg 前後まで上昇したり，浴室が暖かくても入浴による血圧変動は起こるため，このソリューションでこの問題を完全に解決することは難しい．

## 1.2　本研究の問題意識

一人暮らしで入浴していたり，また，家族が家にいる場合でも，夜中に家族が寝ている間にトイレへ行った際などに，身体異常が起こった場合には，当人は助けを呼ぶことが困難である．これらの病気は発生から発見までの時間がその後の結果に大きく影響するといわれている．そのために，早期発見が重要であり，本人が助けを呼ぶ以外の手段で身体の異常を早急に検知し，通知するシステムの開発が急務である．しかし，図 1.2 や図 1.3 に示されるトイレや風呂場は個人的な空間であり，数ある個人的な空間の中でも他人へ状況を露出することが極めて嫌悪される空間である．そのため，カメラを設置し監視することは難しい．また，人間の目を介さない画像解析システムを利用しても，利用者は映像という形でデータを取得されていることを意識するため，カメラの設置は望まれない．さらに，画像データは不正に入手された場合，悪用される危険性がとても大きい．そのため，トイレや風呂場における身体の異常を検知するシステムはカメラを利用せずに行わなくてはならない．

## 1.3　本研究の目的

本研究の目的は，各種センサを利用し，トイレや風呂場などプライバシーの保護のためカメラを利用することの出来ない私的空間における身体異常を検知することである．そして，私的空間において，利用者のプライバシーを侵害せず，監視を感じさせない監視を実現することで，日常生活の中における脳卒中，心筋梗塞などによる生命リスクを下げることを目標とする．本論文では私的空間のうち，トイレに焦点をおき，その中での身体異常の検知を目的とする．

図 1.2　プライベート空間 1:トイレ

図 1.3　プライベート空間 2:風呂場

## 1.4　本論文の構成

第 1 章において，本研究における背景と問題意識を説明した．本論文では，第 2 章で一般的な状況監視について解説し，その後，身体異常状態にの定義と特徴を明らかにした後，身体異常状態を監視する監視システムが備えるべき機能要件を明確にする．第 3 章では，第二章で明確にした機能要件を満たすための手法について基本的な解説を行う．第 4 章では，議論した身体異常コンテクストを抽出する手法を用いた異常検知通報アプリケーションの設計について説明し，具体的なトイレという空間での実装の説明を行なう．第 5 章において，そのシステムを評価した結果を記述する．第 6 章では，本研究と関連する研究を解説し，それらとの相違点，また参考にして更なる発展に活かせる点を述べる．最後に第 7 章で本研究の貢献，そして今後の発展について述べ，本論文のまとめとする．

# 第 2 章

# 状況監視と身体異常

既存の一般的に用いられているある場所の監視手法は防犯カメラや火災報知機，また病院での脈拍計などが代表的である．本項では，まず既存のさまざまな状況監視手法について紹介する．次に，本研究が監視の対象としている身体異常について定義し，身体異常を監視する上で必要な機能要件を明確にすることを目的とする．

## 2.1　状況監視

人間が不在であったり，目の行き届かない場所で，機械が人間の代わりとなり状況監視を行なっている．技術の発展に伴い，その範囲は様々な場所に及ぶ．本項では状況監視の対象の範囲を，公共空間，制約空間，私的空間の 3 つに分類し，その特徴と手法をまとめる．

### 2.1.1　公共空間の状況監視

公共空間における監視で代表的な手法は，映像を用いた手法である．具体的には防犯カメラなど，カメラを用いた手法が挙げられる．カメラは，不特定多数の人間が出入りする公共空間において，その中の誰かが犯罪行為や迷惑行為を犯しているかどうかを記録したり，判断したりする役目を持っている．カメラは状況を画像，もしくは映像として取得することが可能であるため，個人や，振る舞いなど，特定できる情報量が多いことが特徴的である．そのため，犯罪の犯人特定などの目的にはとても適する手法であるが，家庭内での利用などには取得する情報量が多すぎ，プライバシー情報の漏洩などの問題を引き起こし，不適切であることが分かる．図 2.1 は街中に設置された防犯カメラで，図 2.2 は一般的に販売され，手軽に取り扱うことのできるウェブカメラである．

図 2.1　防犯カメラ

図 2.2　ウェブカメラ

画像情報は人の目を介さずシステムが機械的に処理し，異常を見つけ出すシステムがあったとしても，人間は私的空間にカメラが設置されていることに対して心理的な障壁を覚える．これは，監獄施設「パノプティコン」 [3] によって説明される．パノプティコンとは，1971 年にイギリスの哲学者ベンサムによって考案され，フランスにおける初期の監獄に見ることのできる建築様式である．看守は囚人がどこにいても見渡すことができるが，囚人は看守が見えず，いつ自分が監視されていて，いつ監視されていないのかがわからない仕組みになっている．そのため，囚人はいつでも監視されているといった感覚になり，いつでも監視されているかのごとく振舞うことになる．カメラ設置と，その心理的障壁へは，監獄建築としてのパノプティコンの例が適応できる．私的空間内に設置されたカメラは，たとえセキュリティや人の目に触れないことが保障されていたとしても，電子的に行われる情報交換は目視不可能である．そのため，パノプティコンの心理効果が働き，私的空間を利用する利用者は自分のプライバシーが侵害されている感覚を持ってしまうことが予測される．

既存の研究で，Yutaro Aoki，Motonori Doi らは，をカメラを用いて一人暮らしの高齢者を見守るシステム [4] を開発しているが，カメラを用いるため，このシステムはトイレや風呂場といった私的空間で利用することは出来ない．その他にも，老人ホームなどの施設の居住スペースにカメラを設置し，ケアステーションの役員が常に見張ることが出来るシステムもある．このシステムの場合はケアステーションというケアが前提である特別な場所であるため許されるが，日常生活で部屋の中にカメラを設置し，24 時間監視されることは誰も許容しないであろう．

### 2.1.2 制約空間での状況監視

状況監視の分類として制約空間での状況監視というものがある．これは入院中などで，身体に異変が起こる確率が極めて高い場合や，健康状態を常に計測する強い必要性がある状況での状況監視のことを指す．このような状況では，利用者に特別な制約を課すことが許容されるだけの必要性があるため，利用者はその制約を受け入れ，サービスを受ける．制約空間での状況監視に利用される具体的なデバイスの例として，生体情報センサが挙げられる．

生体情報センサは，心拍数や，血圧などをはじめとする人間の生体情報を取得するセンサの分類である．血圧や心拍数は，それぞれに身体状況に対しては絶対的な指標があるため，身体状況を監視する上ではとても強力なデバイスである．しかし，生体情報を取得するためには，常に利用者の肌に密着させ，携帯する必要があるため，利用者がサービスを受けるための負担は大きくなる．近年の技術発展により，これまで大きかった生体情報センサも小型化が進んでいる．その代表的なものに，パルスオキシメータという針などを体内に差し込むことなく，皮膚の上から心拍数や酸素飽和度を調べることができる医療用センサがある．このセンサは心肺機能が正常かどうかを判断する予備的な方法として利用されることもある．また，皮膚温センサも代表的な生体情報センサの一つである．皮膚温センサは，皮膚内を流れる血液の量の変動を確認することができるため，ストレスや疲労などのバロメーターとして用いられることがある．図 2.3 はそれぞれを装着した際の図である．

生体情報センサを用いた状況抽出における関連研究に Rocalind Picard and Jennifer Healey らの研究 [5] がある．彼らは，呼吸，皮膚電気反射，筋電位，心拍数などの情報を取得でき生体情報センサを搭載したウェアラブルデバイス “Affective Wearables” を開発し，そのデバイスを用いて，利用者の感情の取得に挑戦している．感情を把握して動作するアプリケーションの優位性は示されているが，それらのサービスを受けるためには，そのたびに Affective Wearables を装着する必要がある点で，日常で常用するアプリケーションには適していないことが言える．そのため，生体情報センサは，常に肌に密着させるという負担が許容されるだけの

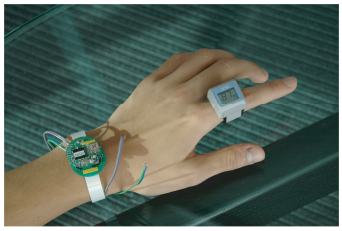

図 2.3　皮膚温センサ (右) とパルスオキシメータ (左)

強力な制約状況の下での監視に適応される．

### 2.1.3　私的空間での状況監視

状況監視の最後の分類に私的空間の監視がある．私的空間の例は寝室や，浴室内，トイレなどが対象となる．家庭内での監視などでは，セコムの防犯センサや防火用の煙感知器など，主に，環境側から実空間情報を取得する環境情報センサを用いた手法が一般的である．このようなセンサは、カメラとは違い検知対象となる情報のみを検知することが出来るため，そこに生活する利用者のプライバシーを侵害することがない．セコムの防犯センサは人間の出入りのみを検知し、煙感知器は煙のみを検知する．そのため，煙感知器などは利用者の抵抗なくトイレなどにも設置できる．図 2.4,2.5 にそれぞれのセンサの例を挙げる．

図 2.4　セコムの防犯センサ

図 2.5　防火用の煙感知器

環境情報センサを利用した状況・状態解析の研究に，Ryan Aipperspach らが提案する，家の中に設置された簡易センサを利用して，利用者の次の行動を予測する手法 [6] がある．また，Emmanuel Munguia Tapia らは，家庭内における環境情報センサを用いて排泄，入浴，身支度などの行動抽出 [7] を行っている．また Stephen S. Intille らは，いくつもの環境情報センサをちりばめた居住空間を設計し，実際に被験者に長期間生活を行ってもらうことで，センサが設置された居住空間デザインの研究 [8] を行っている．

これらの研究は，共通して利用者の状況や行動の抽出を，行動を阻まずに実現する手法の開発を目指している．環境情報センサを用いることで，利用者は普段どおりの生活を続けられ，さらに，サービスを受けることができる．そのため，環境情報センサは日常生活における私的空間の監視に適し，広く利用されている．

## 2.2 身体異常

人間の身体には様々な異常が起こりうる．本項では，その中でも本研究が取り扱う身体異常について明確な定義を与え，どのような特徴があるものであるかを説明する．

### 2.2.1 身体異常の定義

本研究が対象としている身体異常状態とは，突発的に人間が通常の活動を持続できない身体状態に陥り，そのままにしておくと生命に危険が及ぶ状態である．その例として，心臓発作や，脳梗塞などで苦しんでいる状態や，昏倒してしまっている状態が挙げられる．

このような身体異常状態は，ごく稀にしか発生しないため，そのような状態をシステムに学習させることは極めて難しい．一方で，通常な状態は普段の生活の中で行っている行動を基に容易に学習することが可能である．そのため，本研究では通常な状態の補集合を身体に異常のある状態として定義する．そうすることで，通常から逸した場合を異常として検知することができる．図 2.6 では身体異常状況をベン図を用いてを表している．

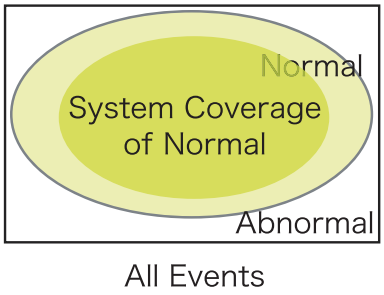


図 2.6　通常状態と異常状態の定義

### 2.2.2 身体異常の特徴

身体異常が持つ特有の重大な特徴は，発生から時間がたつと生命に危険が生じる危険があることである．これは突発的な身体異常を抱えた利用者は自分で助けを呼ぶことができず，誰からも発見されなかった場合は死に至ってしまう可能性さえもある，という特徴である．身体異常が発生してから，何もしない場合，図 2.7 カーラーの救命曲線 [9](図は「救命の連鎖」の確保と実現に向けて，より引用) に従うと，呼吸停止の場合，10 分後には 50% の確率で死に至り，20 分後には確実に死に至ってしまう．また，心臓停止に至っては，後 3 分で 50% の確率でしに至り，6 分後にはほぼ 100% 死に至ってしまう．このデータからも分かるとおり，身体異常は早期発見が重大な鍵となる．

## 2.3 身体異常状態の監視

本研究の目指す私的空間での身体異常の検知は，通常に既存のシステムが私的空間を監視するのと同じように，環境情報センサを用いて行なうものとする．これは，プライバシー情報を含まず，必要な情報のみを取得

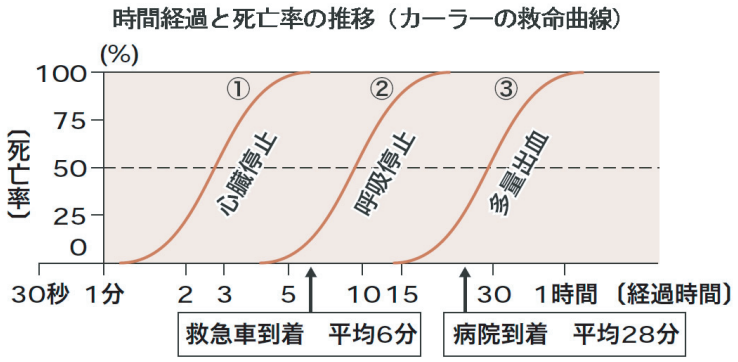


図 2.7　カーラーの救命曲線

することができ，さらに，利用者の負担がないためである．しかし，既存の私的空間の監視手法をそのまま適応しても，身体異常を監視し，検知するには不十分である．既存の私的空間の監視を身体異常検知に適応させるためには，いくつかの機能要件を満たさなければならない．本項では，その機能要件を上げ，詳しく説明する．

### 2.3.1　機能要件

既存の私的空間の監視手法を，身体異常の状況監視に適応させるには，以下の 4 つの機能要件を兼ね備える必要がある．上記の 4 つの機能要件を満たして，初めて私的空間での身体異常検知が実現できる．

1. **環境情報センサによる身体のセンシング**
   身体異常の検知における精度の向上を目指すのであれば，制約空間での状況監視に広く利用されている生体情報センサを利用するべきである．生体情報センサは，センサデータと身体異常との関連性が高い上に，生活におけるプライバシーを著しく侵害するものではなく，プライバシーの議論では問題がないためである．しかし，生体情報センサを利用して私的空間の異常検知を行なった場合，利用者はその空間を利用するたびにセンサを肌に着用しなくてはならず，負担になってしまう上に，風呂場などの私的空間では何も着用していない場合も考えられ，常に利用者を監視することが出来なくなってしまう．結果，センサ不着用のため異常を見落として，発見が遅れてしまう可能性も考えられる．このことから，私的空間での身体異常の検知には生体情報センサを利用せず，環境情報センサを利用しなくてはならない．
2. **動作信頼性の確保**
   センサの故障や電池切れ、無線衝突におけるパケットロスなどで，到達することを前提としているセンサデータがシステムに到達しない場合がある．そのような場合，全てのセンサデータがないと，状況を解析できないシステムであれば，システムの停止を招いてしまう．一時的なパケットロスであれば，1 フレーム分システムが動作しないという問題で済むが，電池切れなどにより恒常的にデータの一部がかけてしまうような状況では，システムは継続して動作せず，その間，利用者の異常を見逃してしまう恐れがある．そのため，システムはそのようなセンサデータの到達状況に左右されず，知っているデータのみから状態を推論できなくてはならない．

3. **実時間推論の実現**
   実時間推論の実現は，送られてくるセンサデータの解析を，次のデータが来る前に終了させる必要があるということである．身体異常の特徴でも述べられていた通り，身体異常は発見までの経過時間が重要である．そこで，センサデータから身体異常であるか否かを解析するのに数分かかってしまった場合，監視の意味はなくなってしまう．そのために実時間内での推論完了はとても重要な機能要件である．
4. **誤検知通報の回避**
   誤検知通報の回避は，正常状態を身体異常として検知し，システムが間違えた通報を行なわないようなメカニズムが必要であるということを示している．センサデータの解析ミスにより身体異常を見逃してしまうことを防ぐため，システム内で身体異常であるかどうか判断に迷うようなデータが求められた場合には，身体異常として扱うことが妥当である．そのため，必然的に誤検知の数が増えることがわかる．誤検知通報の回避の仕組みは，そのため一層重要度を増すことになる．

### 2.3.2 シナリオ

以下では，これらの機能要件を満たした身体異常監視システムを利用したときの有益な例を，具体的なシナリオを用いて説明する．

**シナリオ１:利用者の入浴時**

80 歳の家族と一緒に暮らす男性 (太郎) がいつものように入浴のために浴室に向かう．太郎は通常、小型で携帯できる血圧や脈拍を計測するセンサを健康管理のために装着して生活しているが，入浴の際は外さなければならない．そのため，脱衣所で衣服と，その小型センサを外し，浴室に入っていった．しかし，いったん浴槽で温まり，そのあと体を洗っているときに，急に気分が悪くなり，意識が朦朧とし床に座り込んでぐったりとしてしまった．すると，システムがすぐに太郎の身体異常を検知し，家族に通報を行なってくれた．結果，すぐに家族が助けに入り，事なきをえることが出来た．

**シナリオ２:センサ電池切れ時**

30 歳の男性 (健太) は現在，実家から離れて 1 人で暮らしている．いつものようにトイレに行き，用を済ませた後，しばらくボーっと考え事をしていら，普段の疲れから，うとうととしてしまい，寝てしまった．すると，システムが睡眠状態で動きがなくなったため気絶などの身体異常と勘違いし，身体異常状況を検知したとの反応を出してきた．しかし，起きた健太はシステムの誤検知通報回避のメカニズムを利用して，それが誤検知であることをシステムに知らせ，健太が身体異常にあるとを間違えて病院に通報されずに済んだ．さらに，この時には，利用しているセンサの電池が 1 つ切れていたが，それでもシステムは推論を実行し続けることができていた．

### 2.3.3 本章のまとめ

本章ではまず，現在様々な場所で行なわれている機械による状況監視を，公共空間，制約空間，私的空間での監視という 3 種類に分類し，それぞれの特徴や，利用するデバイスを説明した．次に，本研究が検出対象としている身体異常について定義し，その特徴を説明した．最後に，私的空間での監視技術を身体異常検知に適応させるために必要な機能要件を列挙し，具体的なシナリオを用いながら，それぞれの解説を行なった．

# 第 3 章

# 身体異常状態の抽出手法

本章では，前章で述べた私的空間で身体異常を抽出する上で必要な機能要件を満たすための，身体異常状態抽出手法を解説する..

## 3.1 環境情報センサによる身体センシング

本研究では環境情報センサを用いて身体異常状態の抽出を実現する．そのため，身体異常状態のパターンを正確に取得ことのできる環境情報センサを選定しなければならない．本項では，本研究で利用するセンサの選定とその理由について述べる．

### 3.1.1 利用するセンサ

生体情報を用いず，利用者の身体異常を検知しようとすると，利用者の振る舞いが唯一の手がかりとなる．利用者の振る舞いを検知しようとする際に，照度センサ，温度センサ，赤外線センサ，床圧力センサや，レーザセンサ，超音波センサなど様々な環境情報センサの利用が検討できる．このような環境情報センサで本研究に適しているセンサを選ぶ上で，考慮すべき指標が 3 つある．それらは 1) 即時伝達性，2) 対環境ノイズ性，そして 3) コストパフォーマンス性である．即時伝達性は，環境に変化が起こってから，それがセンサデータとして現れるまで時間がかからないという性質である．対環境ノイズ性は，利用者の行動以外の環境変化がセンサに影響を与えないという性質である．そして，コストパフォーマンス性はセンサが価格的に一般利用可能であるという性質である．上記で検討されているセンサそれぞれの性質に対して評価を与えると表 3.1 のようになる．

照度，温度，マイクセンサはコストパフォーマンスはすばらしいが，自然界や日常生活に多く存在し，常に変化する可能性のある自然光，温度，音を検知対象としていることから，環境の変化と利用者の振る舞いを見分けることが出来ない．結果，対環境ノイズ性を満たすことができない．さらに，温度センサに関しては，ある空間上の位置で温度が変化してから，それをセンサが検知するまでしばらく時間がかかってしまうことから，即時伝達性にも反する．一方で，自然界や日常生活に多くは存在しない超音波や，レーザー光線などを検知するセンサの類は環境の変化には反応せず，利用者の振る舞いのみを検知できるため，対環境ノイズ性は満たしているが，その分コストパフォーマンスに欠け，実用性が見込まれない．圧力は利用者が持ってきた荷物などに反応してしまうが，荷物の加重は変化するものではないため，無視できる範囲ではある．しかし，コストパフォーマンスに欠けるところがある．残った赤外線センサは，若干の環境ノイズを受けるが，コストパ

表 3.1　センサ種類と 3 性質の評価

| センサ | 1) 即時伝達 | 2) 対環境ノイズ | 3) コスト |
|---|---|---|---|
| 照度センサ | ◎ | × | ◎ |
| 温度センサ | × | × | ◎ |
| マイクセンサ | ◎ | × | ◎ |
| 赤外線センサ | ◎ | ○ | ◎ |
| 床圧力センサ | ◎ | △ | × |
| レーザセンサ | ◎ | ◎ | × |
| 超音波センサ | ◎ | ◎ | × |

フォーマンスもよく，即時伝達性も良好である．

そのため，本研究では赤外線センサを利用することとする．また，その中でもピンポイントで狭角視野のセンサを利用し，それを複数利用することで，利用者の振る舞いのみを高い粒度で検知することができる．1 つの広域を見るタイプのセンサを利用すると，気絶して動かなくなったという身体異常は検知できるが，その他、うずくまって苦しんでいる、転倒して痙攣しているなどといった無数に存在する身体異常状態を検知することができないためである．さらに，カメラによる画像のように個人を特定しうる情報やプライバシーを侵害するような情報をハードウェア的に取得できないため，利用者に安心感を持って使ってもらうことができる．現に焦電型赤外線センサは一般的に広くトイレで利用されている．身体異常検知に必要十分な機能を補うことができ，私的空間における利用も認められるため，本研究では複数の赤外線人感センサを利用する．

### 3.1.2　センサの配置

利用する焦電型赤外線センサのサイズが十分小さく，安価なものであれば，壁の中にグリッド上に設置し，細かい粒度で人間の行動を把握することができる．しかし，現状では，サイズ，コスト共にそれを行うには不十分であり，センサは最少の個数で必要な情報を取得できる最適な配置にしなければならない．そこで，本研究では，壁上の頭，胴体，手，足の位置に相当する箇所に焦電型赤外線センサを配置し，人間の体の各部位の動きをとらえる．

### 3.1.3　センサのサンプリング周波数

焦電型赤外線センサは人間の動きに反応し，Hi または Lo の状態を出力するセンサである．この Hi と Lo の矩形パルスが一定の間隔で周期的に入れ替わる場合であれば，サンプリング定理に基づき，周波数の 2 倍以上の早い周波数でサンプリングを行なうと，取りこぼし無く正確に状態を取得することができる．しかし，このセンサでは人間の動きをセンスするため，Hi と Lo の状態は不規則に入れ替わり，サンプリング定理をそのまま利用することはできない．そのため，これらの状態の変化を取りこぼしなく取得するには，限りない速さのサンプリング周波数を必要としてしまう．しかし，本研究では，焦電型赤外線センサを，そのセンサが指す場所に動きがあるのかを確認する目的のために設置しているため，単発的な Lo 状態を取りこぼしても，Hi 状態が確実に取得できれば十分であることがわかる．

この場合，動いた際にはその動きを即座に検知し Hi の状態を出力するが，動きを止めた際，もしくはそこ

から人が遠のいた際，状態が Lo に戻るまでに若干の時間がかかる焦電型赤外線センサの性質を利用することができる．センサは一瞬でも動きを検知すると Hi 状態を出力するが，その後一定時間幅 Hi 状態をとり続ける．状態変化を表した様子を図 3.1 に示す．

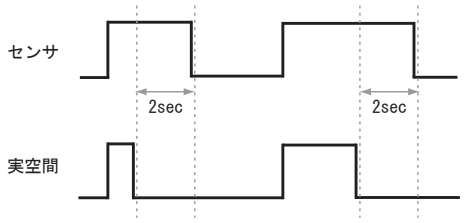


図 3.1　焦電型赤外線センサのパルス例

センサによって，その時間幅は異なるが，それが約 3 秒であった場合は，サンプリングレートをその 2 倍である 1.5 秒以下にすることで確実にその場所の動きを取りこぼし無く検知することができる．本研究では，焦電型赤外線センサの Hi 状態が動きを検知しなくなってから Lo 状態へ下がるまでの時間 p の 2 倍の周波数をサンプリング周波数とする．サンプリング周波数を算出する式を数式 3.1 に示す．

$$SamplingRate = \frac{Time\ To\ Become\ Lo}{2} \tag{3.1}$$

### 3.1.4　身体異常取得イメージ

利用者の通常状態の行動，異常状態の行動にはそれぞれパターンが存在する．そのパターンをセンサを利用して解析することで，一見類似している 2 つの状態も，システムは区別することが出来る．その例として，利用者がトイレにおいて，うずくまって座っている時，身体異常のもとにうずくまって気絶している場合と，通常の状態のもとでうずくまって座っている場合がある．図 3.2 では，その両方を図でイメージで表してる．その違いを本研究でセンサデータを用いて判断する大まかな流れの例を以下に示す．ここで紹介する流れはプロセスのイメージであ．

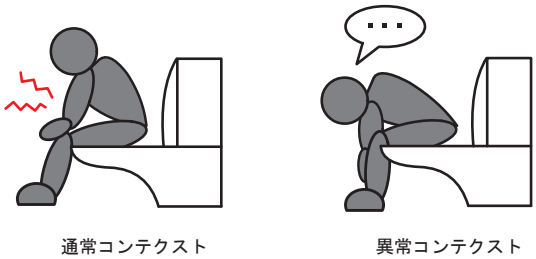


図 3.2　通常状態と異常コンテクスト

**通常時コンテクスト**

トイレにおいて，「うずくまって座っている」ことは，身体に異常がなくても起こりえる．そのような環境の状況の組み合わせを表す例を表 3.2 に示す．表 3.2 に表される状況の組み合わせを見ると，この場合，うず

くまっている為に，利用者の頭部の動きは頭部に設置してあるセンサからでは検出することが出来なかった．しかし，胴体や足の動きは通常どおり検知しているため，ここではこの組み合わせを通常な状態であると考えられる．

表 3.2　通常状態でうずくまって座っている時の状況パターンの一例

| 観察事象 | 状態 (継続時間) |
|---|---|
| 頭部の動き | 不検出 (5 秒) |
| 胴体の動き | 検出 (5 秒) |
| 足の動き | 検出 (7 秒) |

**身体異常状態**

一方，同じうずくまって座っている状態でも，同時に表 3.3 のような環境の状況の組み合わせが判断された場合，システムはこの行動が身体異常状態のもとで取られたと判断する．頭はセンサの検知領域から外れているため，上の例と同じく検出されていない．しかし，動きが検出されていない状態の継続時間も長く，さらに，胴体，足に関してもある程度の時間検出がされていない状況が続いているため，システムは利用者が身体異常状態の下でうずくまって座っていると推測できる．

表 3.3　身体異常でうずくまって座っている時の状況パターンの一例

| 観察事象 | 状態 |
|---|---|
| 頭部の動き | 不検出 (11 秒) |
| 胴体の動き | 不検出 (10 秒) |
| 足の動き | 不検出 (15 秒) |

## 3.2　動作信頼性の確保

センサネットワークの研究は多くの場合，電池駆動の無線センサノードを用いてる．そのため，無線衝突によるパケットロス，電池切れによるセンサノードの停止，センサノードの故障などの様々な原因により，全データの到達が保障されていない．しかし，本システムの停止は利用者にとって致命的な問題になってしまうため，そのようなデータの欠落によりシステムを停止することは出来ない．そのため，本システムはこのようなデータの不到達が発生しても，動作の継続させなくてはならない．本項では，データの欠落が発生しても，システムの動作を持続させるために採用した手法について述べる．

### 3.2.1　ハードウェアの冗長設置

データの欠落をなくす対策として，冗長的にセンサデバイスを設置する方法が考えられる．同じ位置につき，2 つ以上のセンサを設置し，それらのセンサからのデータのうち 1 つを証拠データとして推論に採用する．そうすることで，センサのうち 1 つが故障したり，電波が届かなかった場合にも，もう一方のセンサのデータで補うことができる．このようにハードウェアの故障に対しては，センサの冗長設置である程度対応はでき

る．しかし，この方法は，センサデータの欠落を少なくする手立てであり，センサデータの欠落時にもシステムの動作信頼性を確保するという本質的な解決策ではない．システムが全てのデータが届くことを前提として設計されていると，センサを冗長的に設置しても，何らかの理由でいづれかのデータに欠落が起こってしまったときに，システムは対応できない．システムの動作信頼性の確保は，システムの推論処理において対策をとる必要がある．

### 3.2.2 ベイジアンネットワークの利用

ハードウェアの冗長設置以外の方法で動作信頼性を保障するためには，推論エンジンを，データの欠落に対応できるようなアルゴリズムを用いて実装する必要がある．状態をセンサデータから抽出する際に，利用されているアルゴリズムは多々ある．代表的なものにはファジー理論を用いた述語論理 [10]，有限状態オートマトン [11] に始まり，ニューラルネットワーク [12]，隠れマルコフモデル [13][6]，ベイジアンネットワーク [7] などが上げられる．これらのアルゴリズムの中で，本研究の対象としている身体異常の検知に適し，かつ，データの欠落が起こった際にも動作を継続できるアルゴリズムを選択しなくてはならない．

まず，有限状態オートマトンや隠れマルコフモデルなどのアルゴリズムは状態の遷移を考慮するため，利用者の行動モデルを元にした状態抽出や人間の行動予測などに適している．一方，ファジー理論を用いた述語論理，ニューラルネットワーク，そしてベイジアンネットワークはセンサデータを元にして，人間の行動や場所の状況などの抽出に適し，この分野の研究に用いられている．本研究はセンサデータを元としているため，本研究では後者のグループに属するアルゴリズムが適していることがわかる．後者のグループであるファジー理論を用いた述語論理，ニューラルネットワーク，そしてベイジアンネットワークの 3 つに関して考えてみると，入力データに完全性を要求せず，データに欠落が起こった場合にも推論を続けられるのはベイジアンネットワークのみである．

そのため，データの到達が保障されていないセンサネットワークの推論を継続的に行なうために，本研究ではベイジアンネットワークのアルゴリズムを，センサデータからの身体異常状況解析に利用する．ベイジアンネットワークは統計的にデータを処理することで，3 種類の不確実性を取り扱うことが出来ることで知られている．その不確実性とは、無知による不確実性，無造作による不確実性，そして曖昧さによる不確実性，である．センサの電池切れや、無線衝突によるデータの欠落は，無知による不確実性として扱うことができる．

具体的にはベイジアンネットワークは潜在変数を利用することでこの問題に対処している．潜在変数とは，定められた条件付確率に基づいたランダムな状態変数である．この潜在変数を用いて，システムは欠落データを補うことが出来るため，システムは全てのデータが届くことを前提としなくても，与えられたデータのみから身体異常を推論することができる．本研究では，このアルゴリズムの特性を利用し，観測されたデータだけを証拠としてベイジアンネットワークに与え，不確実なデータは潜在変数として確率的に推測された状態変数を補うことで，継続動作を確保している．

## 3.3 実時間推論の実現

身体異常を推論する際に，ある程度の時間を要すると，身体異常の発見が遅れてしまい，利用者の生命に関わる事態になりかねない．また，次のフレームのデータ集合が到達するまでに，そのフレームのデータ集合の解析が終了しなければ，システムはタスクを溜め込み，推論の速度も低下し，何分も前の推論結果を出力してしまう，といった状況が起こってしまう．そのため，システムは 1 フレーム分のデータ集合を，次のフレーム

のデータ集合が到達するまでの間に解析し終えなければならない．

しかし，ベイジアンネットワークのモデルを厳密推論アルゴリズムを用いて計算すると，モデルの複雑度によっては計算に時間がかかり，センサがデータを送信してくるサンプリングレート内での推論が間に合わない可能性がある．そこで，本研究では，「近似推論アルゴリズムの利用」と「推論モデルの単純化」という 2 つのアプローチを取り，推論時間の短縮化を図り，実時間推論の実現を図った．

### 3.3.1 近似推論アルゴリズムの利用

厳密推論とは，ベイズの定理の数式に基づき，厳密に確率を計算し，解を求める手法である．厳密推論では理論的に正確な解が求められるが，推論するネットワークの構造の複雑化に伴い，必要な計算量が大幅に増え，長い計算時間を要する．一方で近似推論アルゴリズムは，コンピュータシミュレーションによるアプローチを取り，近似的な解を求める．シミュレーションを十分な回数繰り返すことで，数値は収束し，論理的な解に近づくことが出来る．近似推論は，妥当な時間で，妥当な解を求めることが出来るアルゴリズムである．そこで，本研究では，近似推論アルゴリズムを用いて実時間推論を実現する．

近似推論の代表的な例には Logic Sampling や Likelihood Weighting がある．これらの 2 つの推論アルゴリズムは共に親ノードの条件付確率分布に基づきランダムに値を生成し，生成されたパターンを元に確率を推定する手法である．しかし LogicSampling はパターンの中でも証拠として与えられた条件を満たしているパターン以外は破棄してしまう．そのため，ある証拠として与えられた事象が P(e) の確率で出現するとき，システムは生成するサンプル数のうち 1-P(e) 分のサンプルを破棄してしまう．本研究のようにセンサによって観測され，証拠として与えられるノードの数が大いモデルにおいては，さらに破棄されてしまうサンプル数が大きくなる．例えば，センサによって観測され，証拠として与えられるノード数が i 個ある場合，全サンプル数の $1 - P(e_1 \times e_2 \times \cdots \times e_i)$ 分のデータが破棄されてしまう．このように，Logic Sampling は，結果として，(指定したサンプリング回数)+(破棄する回数) 分のサンプルを生成しなければならず，推論効率が悪いことがわかる．一方，Likelihood Weighting は，証拠として与えられたものはランダムにサンプルせず，あらかじめ決定されたものとして，サンプリングによって求められたそれらの親ノードの値の尤度を利用することで確率を推論する．そのため，いかなるサンプルも破棄する必要がなく，計算効率が良い．そのため，本研究では近似アルゴリズムに Likelihood Weighting を利用し，あらゆるモデルにおける推論の実行の実時間性を実現する．

### 3.3.2 推論モデルの単純化

本研究では，もう一つのアプローチとして，推論モデル自体を単純な構造にすることで実時間推論を実現している．身体異常状況をセンサデータから推論する際，考えられる推論の階層モデルは「多階層推論モデル」と「2 階層推論モデル」の 2 つが存在する．

- **多階層推論モデル**
  図 3.3 のように表すことのできる多階層推論モデルでは，取得したセンサのデータを人間の行動という単位に抽象化し，その抽象化された人間の行動のパターンによって身体異常状態を推論するモデルである．
- **2 階層推論モデル**
  図 3.4 にあるような 2 階層推論モデルは、取得したセンサのデータが人間の行動へと抽象化されずに，

直接身体異常状態の推論に利用されるモデルである．

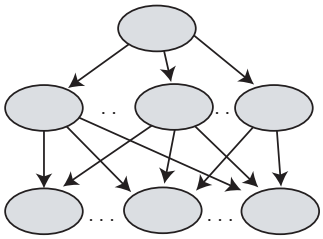

図 3.3　多階層推論モデル

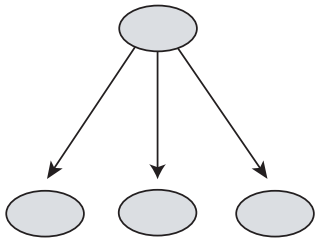

図 3.4　2 階層推論モデル

本研究では，より単純なモデルである身体異常状態の推論に 2 階層推論モデルを採用する．モデルを単純化することで，子ノードの条件付確率の組み合わせは減り，計算量は少なくなるため効率的に処理ができる．更に，2 階層推論モデルでは，通常な状態をシステムに学習させる際，センサデータ，状況は共にシステムに答えが与えられている為，利用者がトイレを利用するだけでシステムが自動的に通常状態の学習を行うことができる．一方，多階層推論モデルにおいて，人間の行動はシステムが確実に把握できない事象なため，学習を行う際，利用者が逐一システムに状況を入力する必要がある．また，本研究が対象としている身体異常は様々な種類があるため，人間の行動という単位に抽象化を行い，推論モデルを作成する 3 階層推論モデルでは構造が複雑化し，定義しきれない身体異常状態が出現する．推論精度と利用者のシステム利用コストの両面からも，本研究では 2 階層推論モデルを利用し身体異常状態を推論する．この 2 階層モデルは一般に NaiveBayes と呼ばれる形態である．NaiveBayes の推論精度のよさは Emmanuel Manguia Tapia ら [8] の研究でも証明されている．

ナイーブベイズとして有名な 2 階層推論モデルと，現在のセンサデータ，そして過去のセンサデータ履歴からから身体異常状態の推論モデルを作成すると図 3.5 のグラフ構造になる．

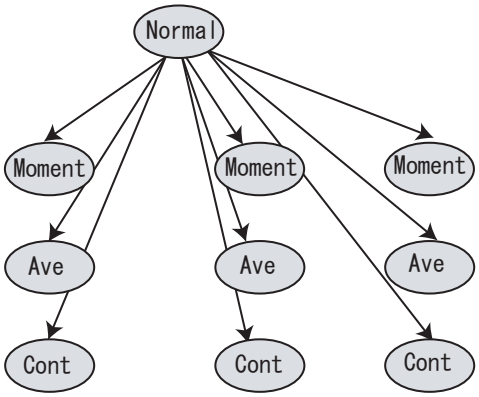


図 3.5　身体異常状態の推論モデル

モデルに利用する推論ノードは，それぞれのセンサから得られる時刻 t のデータ (Movement)，一定フレーム分の平均値 (Ave)，そしてそれぞれのデータが継続されているフレーム回数 (Cont) である．これらのノードすべてに向かって通常状態を表すノードから有効リンクが張られている．そのため，身体異常状態の推論モデルは通常状態を表すノードと，その他センサにより証拠が与えられるノードすべてによる単純なナイーブベイズのモデルとなる．

## 3.4 誤検知通報の回避

本研究は身体異常を取得することを第一の目的とするため，検知漏れを極力下げなくてはならないため，検知漏れ減少を優先させるアプローチを取る．このように身体異常を検知するシステムでは，必然的に誤検知率が多くなってしまう．通常状態にも関わらず，誤検知による通報を行なってしまわないように，システムは利用者に負担の少ない誤検知による通報の回避手法を持たなければならない．

### 3.4.1 検知漏れの回避

身体異常の誤動作には，通常状態を身体異常としてしまう false-positive 型の誤検知と，身体異常を通常状態としてしまう false-negative 型の検知漏れが存在する．誤検知と検知漏れによるの誤動作の出現頻度は反比例の関係をとる．あるセンサデータの集合が観測されたとき，システムはその状況が通常状態である確率を算出する．システムは一定の閾値を持ち，通常状態である確率がその閾値以上であるとき，その状況を通常とし，それを下回る確率であったときにはその状態を身体異常状態とする．ある状態を通常状態と認識する確率の閾値を下げることで，検知漏れを減少させることができるが，誤検知は増加する．逆に閾値を上げることで，誤検知は減少するが，検知漏れが増加してしまう．

### 3.4.2 利用者のインタラクション

誤検知に対して，本研究では利用者に対して状況の確認を取るというアプローチを用いて対策を取る．利用者の手の届く範囲にパネル，もしくはボタンを設置し，身体異常をシステムが検知した際には音声により利用者に身体異常があるかを問いかける．結果，利用者から返答を得られる場合には，システムは利用者は通常な状態であることが確認でき，得られない場合には，利用者に何らかの身体異常が起こっていることとすることができる．図 3.6 に誤検知回避手法のフローチャートを記す．

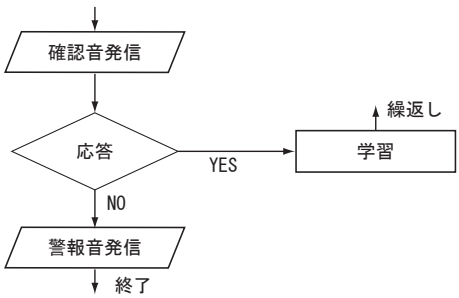


図 3.6　誤検知回避手法のフローチャート

### 3.4.3 フィードバックによる強化学習

この利用者に対する問いかけを利用した誤検知回避手法は，センサデータの学習にも効果を発揮する．誤検知を無視するための入力は，同時にその瞬間のセンサデータのパターンが通常な状態であることの教師データともなる．そのため，このフィードバックを利用することで，システムは，利用者がシステムを利用している最中にも学習データを取得することができ，より精度の高い推論を行なうことができる．

## 3.5 本章のまとめ

本章では，前章で明確にした身体異常を監視するシステムが備えるべき 4 つの機能要件，「環境情報センサによる身体センシング」，「動作信頼性の確保」，「実時間推論の実現」そして「誤検知通報の回避」を解決する具体的なアプローチを説明した．環境情報センサによる身体センシングには，ピンポイント視野の赤外線センサを複数利用することとした．次に，動作信頼性の確保ではベイジアンネットワークのアルゴリズムを用いて，データが何らかの理由で欠落しても推論を持続可能な推論エンジンを実装することで解決した．そして実時間推論は，近似推論のアルゴリズムを利用することと，推論モデルを単純化することで実現を試みるとした．そして，最後に誤検知通報の回避はシステムが利用者に状態を伺い，インタラクションすることを通じて回避する手法を提案した．

# 第 4 章

# Life$^2$Guard システム設計・実装

本章では，身体異常状態抽出手法を用いて，私的空間における身体異常の検知，通報を行なうシステムを設計し，トイレという具体的な空間を対象に実装したシステムの実装について述べる．

## 4.1 Life$^2$Guard システムの設計

前章で説明をした身体異常状態の抽出手法を用いて，本研究では身体異常状態検知，通報を行なうシステム「Life$^2$Guard」を設計する．本システムは，まず利用者の通常状態での私的空間の利用を学習する．一定量のサンプルデータを用いて，通常状態を学習した後に，利用者がトイレを利用すると，システムは常時リアルタイムで利用者の現在の状態が通常であるかどうかを確率で求める．結果，得られた確率が一定閾値以下であったとき，システムは利用者に対し，異常を検知したことを伝え，誤検知であるかどうかの応答を求める．システムに応答があれば，その状態が通常であることを学習し，応答がなければ，異常であると判断し，警報を発信する．

## 4.2 Life$^2$Guard システムの設計概要

本項では Life$^2$Guard システムのハードウェア構成，及びソフトウェア構成の概要を説明する．

### 4.2.1 ハードウェア構成

ハードウェアは，環境側に設置されたセンサ群，それらのセンサにコマンド，データを送信するためのベースノード，ベースノードからデータを収集して，解析を行うサーバ，そして環境に働きかけるアクチュエーターの 5 種類で構成されている．ハードウェア構成図を図 4.1 に示す．

### 4.2.2 ソフトウェア構成

ソフトウェアは，サーバに実装されている，センサとコンピュータとの通信を行う「センサコントロールモジュール」，センサデータから状態を抽出する「状態抽出モジュール」，そして，インタフェースパネル内に実装され，身体状態を抽出したときに環境に働きかける「アクチュエーションモジュール」の 4 モジュールから構成されている．図 4.2 にソフトウェア構成図とデータの流れを示す．

センサコントロールモジュールが取得したセンサデータを状態抽出モジュールが受け取り，解析して，状態

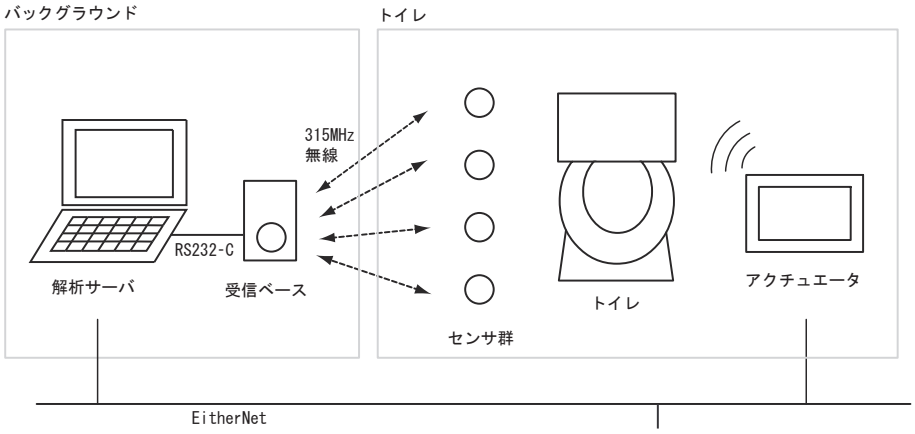


図 4.1　ハードウェア構成図

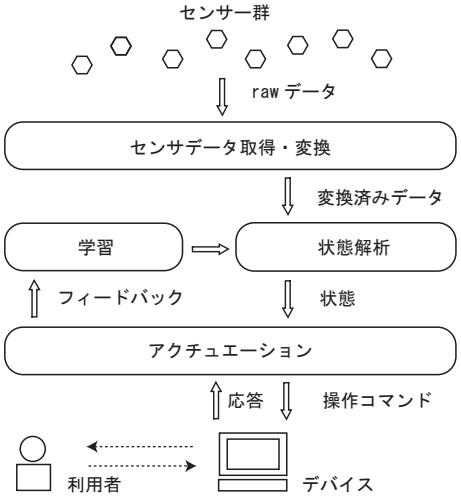


図 4.2　ソフトウェア構成図

を抽出する．抽出された状態に応じて，アクチュエーションモジュールが環境に設置されているデバイスを動作させ，利用者に安否を確認したり，外部に助けを求める連絡を行う．また，状態抽出モジュールが抽出した状態が正当なものかどうかを，アクチュエーションモジュールが受け付ける利用者の入力を元に，学習モジュールが学習を行う．

## 4.3　Life$^2$Guard システムの動作概要

本システムを構成する 4 つのソフトウェアモジュールの動作概要を詳しく説明する．

### 4.3.1　センサコントロールモジュール動作概要

センサコントロールモジュールは，推論が必要なときにセンサを起動し，一定時間毎にデータの取得コマンドをシリアル通信でベースステーションに対して送信し，その瞬間の私的空間に設置されたセンサ群から環境

情報を取得するモジュールである．

**センサのアクチュエーション**

私的空間内のセンサの電池は有限であり，毎秒 1 回環境情報を取得し，無線送信していると，人がいない間も送信し続けるため電力消費の効率がとても悪い．そのため，センサは人間が私的空間内にいる場合にのみ起動され，環境情報の取得，送信を行うことで大幅な電力の削減につながる．そのため，センサコントロールモジュールには必要なときにのみセンサをアクチュエートする機能を実装している．

プライベート空間内に 1 つ，室内の赤外線反応を監視する「マネージメントセンサ」を設置し，そのセンサが人の入退室を検知する役割を果たす．マネージメントセンサは毎 1 秒おきに環境情報を取得し，赤外線反応に変化があったときだけ，センサコントロールモジュールにそのことを無線送信する．マネージメントセンサはセンサノード上で消費電力の大部分を占める無線送信の回数を必要最低限にしているため，マネージメントセンサの電池寿命は 1 秒毎に環境情報を取得していても，長期間の連続駆動が可能である．

マネージメントセンサが入室を検知している間，センサコントロールモジュールはすべてのセンサから環境情報の取得を行うためのコマンドを送信し，マネージメントセンサが退室を検知した場合，室内のセンサへのコマンド送信を中断する．結果，センサデータを常に取得し，無線送信し続けている場合に比べて大幅に駆動時間を長くすることができる．

**環境情報スナップショットの取得**

サンプリングレートが等しくても，それぞれのセンサノードが自分のタイミングをもってデータを取得・送信する形では，ある瞬間の環境情報のスナップショットを取ることができない．センサがそれぞれのタイミングでデータを送信してくると，システムはある瞬間の空間における状況を抽出することができない．そのため，本モジュールでは，ある瞬間の環境情報のスナップショットを取得する機能を実装している．

例えば，センサ A とセンサ B があり，どちらものセンサも 3 秒毎にデータを取得し，送信してくるが，そのタイミングがずれているとする．すると，推論のタイミングとしては，センサ A とセンサ B の両方を受信できたタイミング，もしくは毎回センサデータを受信したタイミングがあげられる．それぞれについて，問題点を具体的に以下で説明する．

最初に前者，センサ A とセンサ B の両方を受信できたタイミングで推論した場合の問題を説明する．センサ A が時刻 t にデータを受信してから時刻 t+1 秒に環境に変化が起こったとする．そして時刻 t+2 秒にセンサ B が環境情報を取得，送信を行い，システムは推論を実行するとする．この場合，システムの推論結果は，どの時刻における空間の状況にも属さず，間違った結果を導き出してしまう．

次に，後者の毎回センサデータを受信したタイミングで推論を実行した際の問題点を説明する．まず，センサ A が時刻 t にデータを受信して，システムは推論を実行する．その後，前例と同様，時刻 t+1 秒に環境に変化が起こったとする．変化の後の値を時刻 t+2 にセンサ B が取得，送信し，そのタイミングで再びシステムは推論を実行する．この場合も，時刻 t+2 において実行された推論の結果はどの時刻にも属さない間違った結果であるということがわかる．

このように，センサデータの送信タイミングにずれがある程度のずれが生じると，どのような場合にも，ある瞬間の空間における正しい状況を抽出することができないことがわかる．センサのデータ取得タイムラグをなくすために，本システムでは，センサの情報を取得する方式に，ベースノードが一定時間毎にコマンドを送り，それに応答する形で環境に設置されているセンサがデータを取得し送信してくるという方式を採用し，実装を行った．

具体的には，空間内に利用者の存在を確認すると，センサコントロールモジュールがベースノードに「get パケット」をシリアル通信で送信を行う．ベースノードが「get パケット」を受信すると，「sense パケット」を無線で空間内にブロードキャストする．そして，空間内に設置されたセンサはそのパケットを受信して環境情報を取得，送信を行う．結果，環境情報センサの取得時間にずれはなく，正しい推論結果を得ることができる．図 4.3 はその動作概要を表したものである．

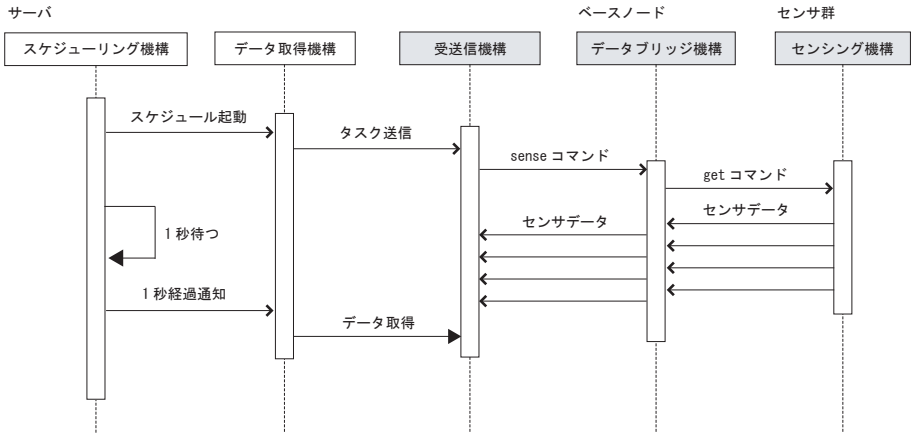


図 4.3　環境情報スナップショット作成機構

この場合，get パケットを送信する送信周期がセンサのサンプリング周波数である．そこで，第 3 章のサンプリング周波数を決定する際の数式 5.1 に基づき，本システムの実装におけるサンプリング周波数を求める．今回利用したセンサの実装では Hi 状態から動きを検出しなくなってから Lo になるまで約 2 秒かかることから，サンプリング周波数はその 1/2 の 1 秒となる．

**無線衝突回避**

ベースノードが「sense パケット」を無線でブロードキャストした際，環境に設置されたすべての環境情報センサが一度に無線でデータを送信するため，無線の衝突によるパケットロスが頻繁に生じる．そのため，センサノードには「sense パケット」への返答の無線送信タイミングをある一定範囲の中でランダムに決定し，衝突を回避する機能を実装している．また，センサコントロールモジュールはベースノードに「get パケット」を送信した後，タイムアウトを設定し，タイムアウトまでに受信したデータを有効なデータとして採用する．

### 4.3.2　状態抽出モジュール動作概要

状態抽出モジュールは受信したセンサデータを解析しやすい形に処理し，解析を行い身体異常状態を検知する，本システムの中核となるモジュールである．

### 4.3.3　環境の状態による確率変数スイッチング

利用者における通常の状態は，利用者のそのときの行動により大きく異なる．例えば，便座に座っている状態か，立っている状態であるかといったトイレの利用モデルによって，利用者の異常である状態は大きく異なる．そのため，推論モデルはそれぞれの環境の状態に対して一つずつ必要になり、環境の状態に応じてそれら

を使い分けていかなくてはならない．本研究では，推論に対する補助データとして，利用者の行動を大きく分類する状態を，私的空間に存在する日用品の利用状態をもって把握し，その状態に応じた推論モデルを推論に利用することで推論の向上を目指す．

**データと推論ノードの対応付け**

それぞれのセンサデータは，そのデータが意味する推論モデルのノードに対応付けられる．センサデータと推論モデルのノードの対応には独自に定義した XML データを用いる.XML では，センサ ID$< id >$，センサの種類 $< type >$，そして対応する推論ノード $< node >$ の 3 つの要素によって構成されているセンサデータ $< SensorData >$ を定義し，センサデータと推論ノードの対応付けを行っている．XML の例を以下に示す．

センサデータとノード対応 XML

```
<SensorVariable>
     <SensorData>
          <id>1</id>
          <type>redRay</type>
          <node>PointRay1</node>
     </SensorData>
</SensorVariable>
```

**センサデータの変換**

センサから送られてくるセンサデータをシステムに適した形に変換する手続きを行う機能が状態抽出モジュールには実装されている．本システムが実装している変換には，生データを工業単位に変換する単位変換と，データの差分や分散などを求めるデータ処理がある．単位変換はセンサが取得した電圧の値を，そのままのをセンサデータとして送ってくるアナログセンサの生データを，工業単位に変換する機能である．一方データ処理は，複数のデータを処理することで，新しいデータの形に変換する機能である．具体的には加速度の分散を求めることで，加速度を「揺れ」という単位に変換することができる．

**センサデータの量子化**

ベイジアンネットワークの推論ノードの変数に，送られてきたセンサデータ，もしくは変換されたセンサデータの整数値一つ一つをそのまま利用すると，推論ノードが取ることのできる状態が多くあることから，サンプルデータの結果が分散してしまい，うまく推論ノードの条件付確率に反映できない．そのため，条件付確率の学習を収束させるまでに必要なサンプルデータ数が非常に多く必要になってくる．

効率よくサンプルデータを利用して推論ノードの条件付確率を学習させるために，推論ノードにはセンサデータを量子化し，一定範囲のセンサデータを含む変数を持たせる．そのため，送られてきたセンサデータ，もしくは変換されたセンサデータを元に推論を行うためには，それらの値を量子化することが必要である．状態抽出モジュールには，このセンサデータの量子化を行う機能を実装している．

**ベイズ推論の計算**

ベイジアンネットワークの各モデルには証拠変数を与え，サンプリング回数を指定し，実行する．その際，受信したセンサデータのみを証拠変数として与えることが可能で，何らかの理由で受信できなかったセンサ

データがある場合は，そのデータに対応する推論ノードは潜在変数として扱われ，不確実のまま推論を実行することができる．実行した結果は，確率で返却される．

### 4.3.4 アクチュエーションモジュール動作概要

状態抽出モジュールにおいて，現在，利用者が身体異常状態下にある確率が一定以上であった場合，アクチュエーションモジュールは利用者，またその周囲に対して働きかけを行う. アクチュエーションモジュールは，ユーザインタフェースパネル内に実装され，環境とシステムとのコミュニケーションの役割を果たす.

#### 異常確認メッセージ発信

状態抽出モジュールが異常状態を検知した場合，アクチュエーションモジュールは，推論の結果検知したその異常が実際に異常であるのか，それとも誤検知によるものであるのかを利用者にアラート音を鳴らして問いかける. その問いかけに対して，利用者がインタフェースパネル上の「無視」ボタンを押した場合，アクチュエーションモジュールは，検知した異常状態を誤認識と判断し，周囲に助けを発信せず，継続して推論を実行し続ける.

#### 異常通知メッセージ発信

異常確認メッセージに対して，利用者が一定時間以内に「無視」ボタンを押して反応してこない場合，アクチュエーションモジュールは，検知した異常状態は正しいものと認識し，利用者の指定した連絡先に e-mail でメッセージの発信を行い，屋内に警報を鳴らす.

### 4.3.5 学習モジュール動作概要

学習モジュールは，事前に通常状態の示すセンサデータパターンを記録し，それぞれのセンサデータを表すノードに対する条件付確率を更新する．システムの利用開始後も，ユーザインタフェースパネルの誤検知を無視するボタンが利用者に押されるたびに，フィードバックを得て，通常状態を記録し，学習を継続して行うことができる.

## 4.4 Life$^2$Guard システムの実装

本研究では，私的空間の例としてトイレを設定とし，前項で述べたシステム設計を基に Life$^2$Guard システムの具体的な実装を行なった．本項では，トイレにおける Life$^2$Guard システムの具体的な実装を説明する．図 4.4，4.5 は Life$^2$Guard システムのインスタレーションである．トイレの壁にセンサが設置されて利用者の振る舞いを取得している．トイレの外の無線電波が届く範囲に状態解析サーバを設置し，サービスの存在感を抑えている．図 4.6 はシステムの実行画面である．上段のテーブルにはそれぞれのセンサの種類，名前とセンサデータを表示している．下段のテーブルには抽出目標としている状態がどれくらいの確率で現在起こっているのかを表す確率が表示されている．

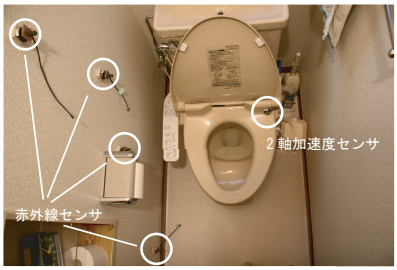


図 4.4　トイレに設置されたセンサ

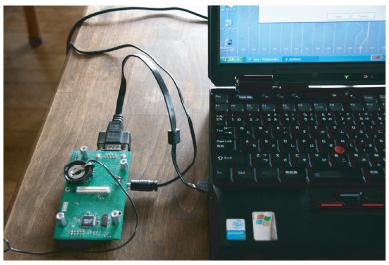

図 4.5　トイレ外に設置された解析サーバ

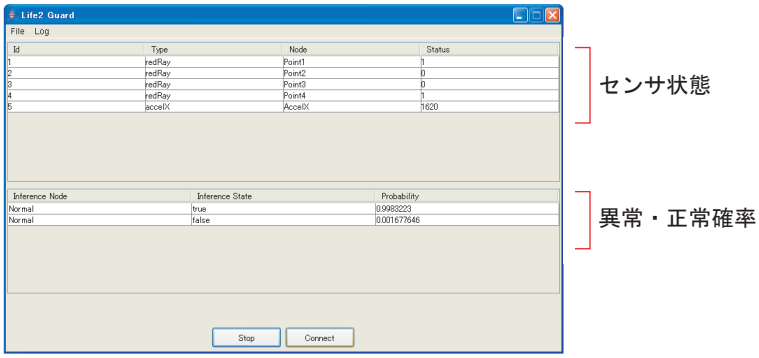


図 4.6　Life$^2$Guard システム実行画面

### 4.4.1 センサの配置

トイレを対象とした本実装では，利用者の体勢を取得するための焦電型赤外線センサに加え，補助的に便座の状況を把握するための 2 軸加速度センサは便座に設置している．焦電型赤外線センサは，4 つ，それぞれ壁上の頭，手，腰，足首の位置に設置した．センサ群を設置した際の全体図を 4.7 に，それぞれのセンサを装着した際の例を図 4.8 と図 4.9 に示す．

図 4.7　焦電型赤外線センサ設置例 (全体)

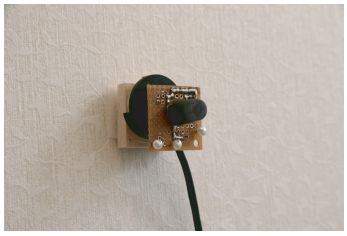

図 4.8　焦電型赤外線センサ設置例 (単体)

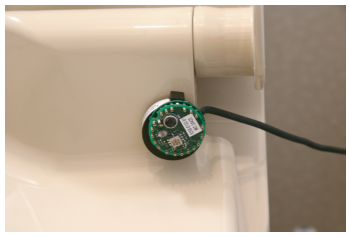

図 4.9　2 軸加速度設置例

### 4.4.2　推論モデルの作成

トイレ内における身体異常を推論モデルとして定義すると図 4.10 のグラフ構造になる．

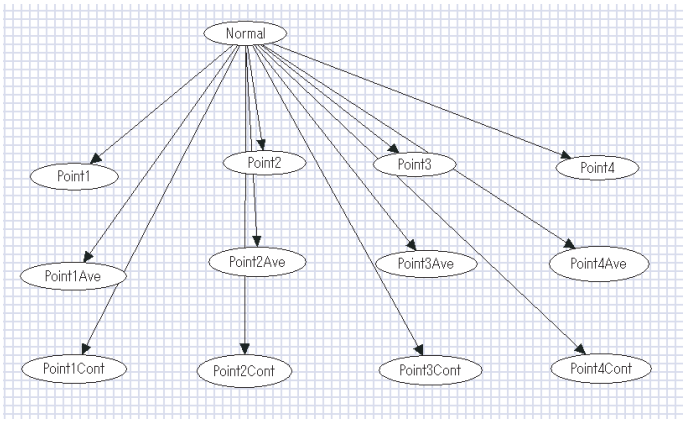


図 4.10　トイレにおける身体異常状態の推論モデル

モデルは第 4 章で述べた 2 階層モデル (NaiveBayes) のモデルを基にして作成している．それぞれの推論ノードは頭，手，腰，足首の位置に設置した 4 つの赤外線センサの現在の値，平均の値，動きなしの継続のフレーム数を表している．表 4.1 にノード名と変数リストをまとめた．

表 4.1　身体異常状態の推論ノード

| ノード | 変数 |
|---|---|
| 通常状態 | true, false |
| 赤外線センサ 1(頭部) | {1, 0} |
| 赤外線センサ 2(胴体) | {1, 0} |
| 赤外線センサ 3(手) | {1, 0} |
| 赤外線センサ 4(足元) | {1, 0} |
| 赤外線センサ 1 平均 (頭部) | {0.2, 0.4, 0.6, 0.8, 1.0} |
| 赤外線センサ 2 平均 (胴体) | {0.2, 0.4, 0.6, 0.8, 1.0} |
| 赤外線センサ 3 平均 (手) | {0.2, 0.4, 0.6, 0.8, 1.0} |
| 赤外線センサ 4 平均 (足元) | {0.2, 0.4, 0.6, 0.8, 1.0} |
| 赤外線センサ 1 継続 (頭部) | {2, 4, 6, 8, 10} |
| 赤外線センサ 2 継続 (胴体) | {2, 4, 6, 8, 10} |
| 赤外線センサ 3 継続 (手) | {2, 4, 3, 6, 10} |
| 赤外線センサ 4 継続 (足元) | {2, 4, 3, 8, 10} |
| 加速度センサ (便座) | {up, down} |

### 4.4.3　センサノードの実装

本システムで利用するセンサを実装した基盤センサには CrossBow 社が開発している Mica2 Dot[14] を利用し，実装を行った．Mica2 Dot は低消費電力の無線センサネットワークを実現するための，直径 25mm の円形で小型のセンサノードである．無線通信には 868/916 MHz，433MHz，315MHz のマルチチャネルが利用でき，OS には TinyOS が搭載されている．TinyOS とは，限られた計算資源下で効率的な処理を行えるように特化された OS である．Mica2 Dot の写真を図 4.11 に，そのデータを表 4.2 に示す．

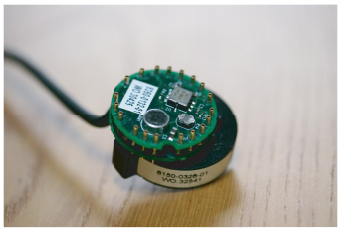

図 4.11　Mica2 Dot

本研究では，利用者の状態を把握するために，人感センサとして一般的に利用されている焦電型赤外線センサをこの基盤センサに装着する．Mica2 Dot の基本センサ基盤には，加速度，温度，照度，音量センサが搭載されているが，赤外線センサは搭載されていない．そのため，本研究では，Mica2 Dot を基盤センサノードとして，焦電型赤外線センサを取り付け動作させる．

表 4.2　Mica2 Dot のデータ表

| 性能 | Mica2 Dot |
|---|---|
| サイズ | 25mm * 6mm |
| プロセッサ | ATmega 128L 8bit processer at 8MHz |
| メモリ | 128KB program memory(flash) |
| メモリ | 512KB additional data flash memory |
| 無線周波数 | 868/916MHz，433MHz，315MHz |
| 電圧 | 3V |

焦電型センサには図 4.12 にある Panasonic 社が出している NaPiOn[15] を利用した．NaPiOn には，標準タイプ，狭視野タイプ，そして，微動検知タイプがあり，本研究ではピンポイントで利用者の体の部分部分の状態を把握することを目的としていることから狭視野タイプのセンサを利用する．図 4.13 は狭視野タイプの視野角の説明図で，napion のデータシートから引用したものである．

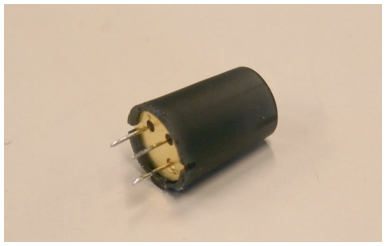

図 4.12　焦電型赤外線センサ NaPiOn 狭視野タイプ

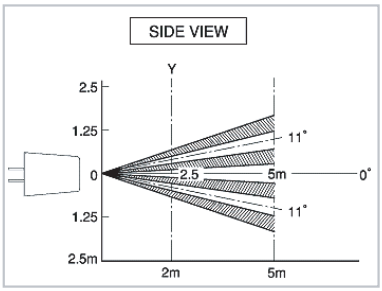


図 4.13　NaPiOn 視野角

紹介した焦電型赤外線センサを，基盤ノード Mica2Dot へ取り付ける際の実装に関して説明する．NaPiOn にはアンプ内臓のセンサモジュールで，Vcc, GND, GIPO 出力の 3 つの pin がある．Vcc の動作電圧は最小 3.0V DC，最大 6.0V DC である．一方，基盤ノードである Mica2Dot はインタフェースとして 19 の pin があり，それぞれが，電源コントロールピン，ADC チャンネル，電源，グラウンド，汎用デジタル IO 及びシリアルプログラミングボードのいずれかに割り当てられている．電源の電圧は 3.0V である．そのため，モジュールの装着は，NaPiOn の Vcc を Mica2Dot の 6pin へ,Gnd を 1pin へ，GPIO 出力を 18pin へ繋ぐことで完了する．実際の実装は図 4.14 の写真である．

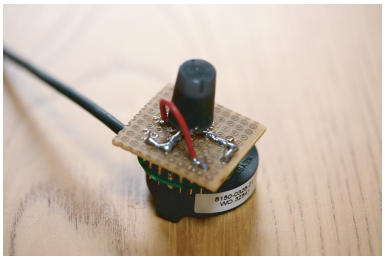

図 4.14　センサモジュールの実装例

### 4.4.4　マネージメントセンサの実装

センサのアクチュエーションを行なうマネージメントセンサは，今回の実装では，センサを用いていない．今回はトイレの電気のスイッチと統合できるということを前提に図 4.15 に示すトグルスイッチをインタフェースとして用いた．

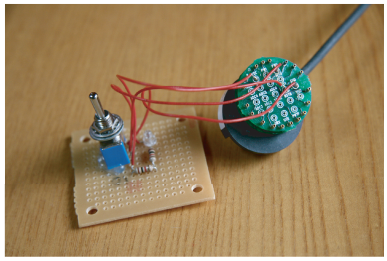

図 4.15　センサのアクチュエーション用スイッチインタフェース

### 4.4.5　アクチュエーターの実装

アクチュエーターには Sony 社製の Vaio Type U[16] を利用して実装を行なう．Vaio Type U にはタッチスクリーン機能が搭載されており，利用者は専用のペンなしに，指でそのまま押す事ができる．OS には Windows が搭載されており，java で実装されたアクチュエーションモジュールが動作している．図 4.16 はアクチュエーションモジュールを起動させ，トイレに設置されているアクチュエータである．

図 4.16　アクチュエータ

システムが身体異常を検知すると，「トントントン」というノック音で利用者に身体状態を確認する．その際，利用者が通常な状態であれば，真ん中に表示されている緑色のボタンを押すことで，利用者は誤通報をしないように警報を無視することができる．

## 4.5 ベイズ推論エンジンの実装

本項では，システムの中核となるベイズ推論エンジン実装部分について述べる．

### 4.5.1 ベイズ推論モデルの定義

ベイズ推論エンジンにおいて推論処理を行なうベイジアンネットワークのモデルを定義するためのフォーマットには門田昌哉氏が開発した Jenga[17] を用いて作成した XML フォーマットの JengaFormat を利用する．JengaFormat では，それぞれのノードを定義している $< RandomVariable >$，ノード同士の連結を保持し，グラフ構造を定義している $< GraphStructure >$，そして各ノードの条件付確率を定義している $< ConditionalProbabilityTable >$ によってベイジアンネットワークのモデルが定義されている．

### 4.5.2 ベイズ推論エンジンの多態化

ベイジアンネットワークを用いて事象の出現確率を求めるアルゴリズムには様々なものがある．しかし，アルゴリズムの部分はそれぞれ実装は違うが，構造解析や，証拠変数のセットなどに関しては，アルゴリズムに依存せず，共通の処理が行なえる．そのため，様々なアルゴリズムを共通のインタフェースを用いて実装できるように，本システムでは，アルゴリズム部分を除いて実装された抽象クラスである BayesEngine クラスをつくり，それぞれのアルゴリズムをその抽象クラスを継承したクラスに実装を行なった．そうする事で，アルゴリズムの変更後にもシステムのコード修正にコストがかからず，すぐに行なえる利点がある．

#### BayesEngine 抽象クラス

BayesEngine 抽象クラスには，主にベイジアンネットワークの構造解析や，証拠変数のセットなど，ベイズ推論で汎用的に行なわれている関数を実装し，推論処理のコア部分であるアルゴリズム部分に関してはインタフェースのみを与えている．public なメソッドを一覧すると setUpNodes，explore，setEvidence，そして clearAllEvidence，そして infer がある．infer に関してはアルゴリズム部分であるため，BayesEngine 抽象クラスでは実装されていない．メンバ変数と関数リストを表 4.3 に示す．

表 4.3　BayesEngine 抽象クラス

| アクセス | メソッド |
|---|---|
| private | NodeArrayList nodeList |
| private | int nodeCount |
| private | int topHierarchyLevel |
| public | void setUpNodes(String filePath) |
| public | void explore(Node child) |
| public | setEvidence(String nodeName, String state) |
| public | void clearAllEvidence() |
| public | void infer(int samplingRate) |

LikelihoodWeighting **クラス**

LikelihoodWeighting クラスは BayesEngine 抽象クラスを継承して，infer メソッドを実装して作られたベイズ推論アルゴリズムのクラスの一例である．メソッドは infer のみであり，その他の必要なクラスは親クラスである BayesEngine 抽象クラスに全て実装されている．メンバ変数と関数リストを表 4.4 に示す．

表 4.4　LikelihoodWeighting クラス

| アクセス | メソッド |
|---|---|
| private | float score[] |
| public | void infer(int samplingRate) |

### 4.5.3　ベイズ推論エンジンの利用方法

ベイズ推論エンジンは，推論処理が必要なアプリケーションで簡単に利用できるよう，単純で明解なインタフェースを提供している．そのため，1. エンジンの作成，2. モデルのセットアップ，3. 証拠のセットアップ，4. サンプリング，5. 結果取得の全 5 ステップで結果取得までを行なうことができる．本ベイズ推論エンジンを利用した，推論結果を取得するまでのコード例を以下に示す．

ベイズ推論エンジンの利用例

```
/*エンジンの作成*/
LikelihoodWeighting engine = new LikelihoodWeighting();

/*モデルをパースしてセットアップ*/
engine.setUpNodes("./ModelSample.jng");

/*証拠変数を与える*/
engine.setEvidence("Node1","true");
engine.setEvidence("Node2","true");

/*10000 回のサンプリング*/
engine.infer(10000);

/*結果の取得*/
System.out.println(engine.getLikelihood("Node3","true"));
System.out.println(engine.getLikelihood("Node4","true"));
```

本システムでは，2 ステップ目のモデルのセットアップまでをプログラムの起動と同時に行い，その後の 3 ステップ目から 5 ステップ目までは，センサがデータを取得する毎に処理を行なっている．

### 4.5.4 ベイズ推論アルゴリズムの実装

本システムでは，証拠としてノードの集合 $E = \{E_1 = e_1 \cdots, E_n = e_n\}$ が与えられたときのノード X の状態を求める際のベイジアンネットワークを計算するアルゴリズムとして Likelihood Weighting を実装した．Likelihood Weighting は対象の事象の尤度をスコアとして，その総和を正規化して，確率を出すアルゴリズムである．実装したロジックを以下に記述する．下のアルゴリズムは “Bayesian Artificial Intelligence”[18] より抜粋した．

Likelihood Weighting のロジック

**Initialization**

- **For each value $x_i$ for node X**
  **Create a count variable Count($x_i$,e)**
- **Create a count variable Count(e)**
- **Initialize all count variables to 0**

**For each round of simulation**

1. **For all root nodes**
   **If a root node is an evidence node, $E_j$**
   **choose the evidence value, $e_j$**
   **likelihood($E_j = e_j$) $\leftarrow$ P($E_j = e_j$)**
   **Else**
   **Choose a value for the root node, weighting the choice by the priors.**
2. **Loop**
   **If a child is an evidence node, $E_j$**
   **choose the evidence value, $e_j$**
   **likelihood($E_j = e_j$) = P($E_j = e_j$ | chosen present values)**
   **Else**
   **Choose values randomly for children, using the conditional probabilities given the known values of the parents.**
   **Untill all the leaves are reached**
3. **Update run counts**
   **If the case includes E = e**
   **Count(e) $\leftarrow$ Count(e) + $\prod_j$likelihood($E_j = e_j$)**
   **If this case includes both X = $x_i$ and E = e**
   **Count($x_i$,e) $\leftarrow$ Count($x_i$,e) + $\prod_j$likelihood($E_j = e_j$)**

**Current estimate for the posterior probability**

$$P(X = x_i | E = e) = \frac{Count(x_i, E = e)}{Count(E = e)}$$

## 4.6 センサデータ操作クラスの実装

本項では，複数の種類が存在するセンサデータを統一的に取り扱うためのデータ構造と関数群を実装したセンサデータ操作クラスの実装について述べる．

### 4.6.1 センサデータ操作の統一化

本研究では焦電型赤外線センサと，2 軸の加速度センサのデータを取り扱っている．今回は 2 種類のセンサであるが，今後の研究で新しくセンサが必要となってくる可能性もある．それぞれのセンサデータに対して一つ一つデータ構造と関数群を定義する設計は，センサを増やした際にセンサデータを扱う部分以外のコードに対して手を加える必要が出てくる．そこで，本システムはセンサデータを統一的に管理，取得できる SensorData クラス，そして，SensorData のインスタンスを利用し，どのようなセンサデータもプログラム中で統一的に扱うことのできるインタフェースを提供する SensorDataConverter クラスを実装した．

#### SensorData クラス

SensorData クラスは様々な種類のセンサデータを統一的に管理し，取得することを目的としたデータ構造クラスである．メンバ変数は private で宣言された sensorID, dataType, dataValue, inferenceNodeName の 4 つである．sensorID はデータを送信してきたセンサの ID を，sensorType はデータのタイプをそれぞれ保存している．inferenceNodeName は，そのセンサデータが推論モデル上のどの推論ノードの値として利用されるかを示している．SensorData クラスは表 4.5 のように実装されている．

表 4.5　SensorData クラス

| アクセス | 変数/メソッド |
|---|---|
| private | int sensorID |
| private | int dataValue |
| private | String dataType |
| private | String inferenceNodeName |
| public | SensorData(int id, String type, int value) |
| public | boolean findInferenceNode(TableModel table) |
| public | int getSensorId() |
| public | int getSensorValue() |
| public | String getSensorType() |
| public | String getInferenceNodeName() |

#### SensorDataConverter クラス

SensorData クラスのインスタンスを利用して統一的にセンサデータの抽象化などの操作を行なう関数群を SensorDataConverter として実装した．SensorDataConverter クラスは，センサデータを推論ノードの状態として抽象化を行なってくれる dataToLabel(SensorData data) 関数を static 関数として実装した．

dataToLabel にはどのような SensorData のインスタンスのタイプが何であるか意識しなくても，引数として渡すと，そのタイプに対応した推論ノードの変数を返してくれる．実装は表 4.6 のようになっている．

表 4.6　SensorDataConverter クラス

| アクセス | メソッド |
|---|---|
| private | String dataToLabel |

### 4.6.2　センサデータ統一操作クラスの利用方法

センサノードから送られてくる文字列，”id=2,photo=150,accelX=550,accelY=550”には 3 つの違う種類のデータが含まれている．文字列を解析し，3 つのデータに分けてから推論に利用するまでのコードの例を以下に示す．このように，センサデータ操作クラスを利用することでセンサデータを統一的に扱うことができるため，処理を分ける必要がなくなることがわかる．

センサデータ操作クラスの利用例

```
/*3 つのセンサデータをパースする*/
SensorData[] sensorData = SensorDataParser.parseDotDataString(dataString);

/*3 つのセンサデータを推論に利用できるラベルに抽象化*/
for(int i=0; i<sensorData.length; i++){

        /*センサデータに対応する推論ノードを探す*/
        boolean isDataUsed = sensorData[i].findInferenceNode(sensorTable);

        /*推論ノードで使われているセンサデータであれば*/
        if(isDataUsed){

                /*推論に利用できるラベルデータに抽象化する*/
                String dataLabel = SensorDataConverter.dataToLabel(sensorData[i]);

                /*推論エンジンに証拠変数としてセット*/
                engine.setEvidence(sensorData[i].getInferenceNodeName, dataLabel);
        }
}
```

## 4.7 本章のまとめ

本章では身体異常状態抽出手法を用いて，トイレにおける利用者の身体異常に自動的に検知し，通報する危機回避システムの概要を説明し，設計，実装について述べた．本システムは複数の小型のセンサ，センサ受信機，計算サーバ，インタフェースパネルの 4 つのハードウェア構成となっている．また，本システムは推論モデル，センサデータと推論ノードの対応がすべて XML 形式で外部記述されているため，モデルの変更や場所の変更に臨機応変に適応できるシステムとなっている．

# 第 5 章

# 評価

本章は，本研究における身体異常状態の検知手法の評価を行う．

## 5.1　ベイズ近似推論の収束サンプリング回数評価

本研究において身体異常状態の抽出手法に利用するアルゴリズムに Likelihood Weighing を採用していることを述べた．Likelihood Weighting は，ベイジアンネットワークにおける近似推論であり，近似推論は一定サンプリング回数のコンピュータシュミレーションを行い，その結果から事象発生確率を求める．そのため，サンプリング回数が少なければ求められた結果は信頼の置けるものとはいえない．しかし，サンプリング回数が多ければ，計算にそれだけ時間がかかり実時間内に計算を終わらせることが出来なくなってしまう．そこで，身体異常状態の推論モデルを計算する上で，実時間内に妥当な収束値を求めることが出来ることを実証するために，身体異常状態の推論モデルを利用し，サンプリング回数と結果データの収束の関係を調査した．そして，収束が見られた最小のサンプリング回数を，本研究のサンプリング回数として利用する．

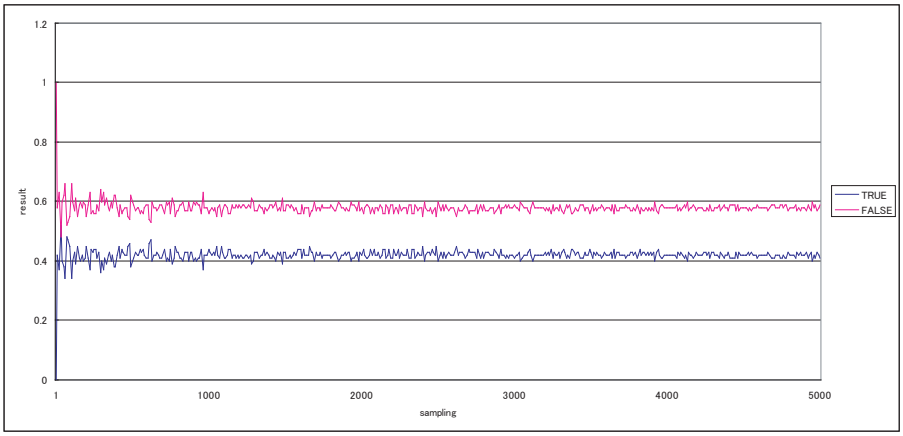


図 5.1　サンプリング回数と計算結果

図 5.1 は任意の値を証拠として全ての推論ノードに設定し，5000 回まで 0 回から 10 回ずつサンプリングレートを増やして得られた通常状態 “true”, “false” の確率結果である．グラフを見ると 1000 回までは毎回の計算結果が大きくばらついていることがわかるが，以後，収束しているかのように見受けられる．

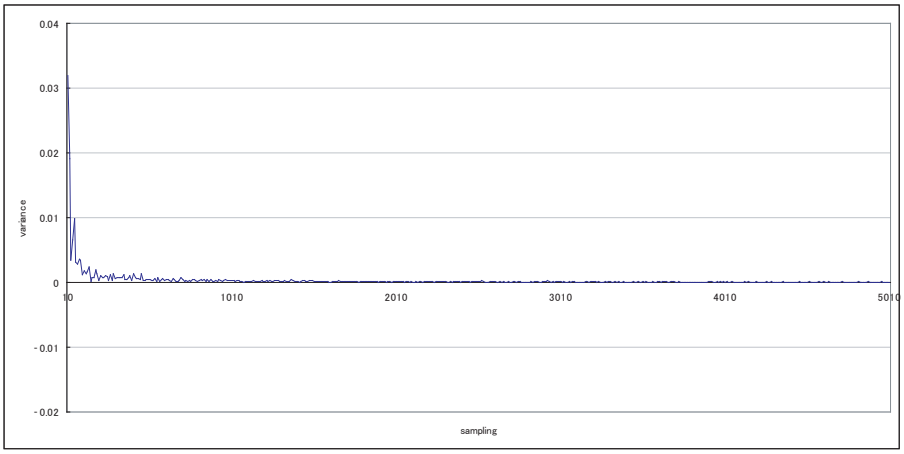


図 5.2　サンプリング回数と計算結果の分散

図 5.2 は，同じく 5000 回まで 10 回から 10 回ずつサンプリングレートを増やしながら，同じサンプリングレートで 10 回ずつ計算させ，その中での値の分散を算出した結果をグラフ化したものである．図を見ると，2000 回以降で分散が収束していることが顕著にわかる．2000 回のサンプリングと，結果の算出の該当部分の実行にかかった時間も 2000 回で 20ms であり，十分に実時間内に計算処理を終えることのできる時間であるため，本研究では 2000 回のサンプリングレートを採用する．

## 5.2　身体異常状態の抽出精度評価

Life$^2$Guard システムの身体異常検知精度を，実際の家庭のトイレを利用して評価を行なった．実験はどのくらいの精度と，どれくらいの経過時間で身体異常を検知できるかに加え，通常状態を異常と誤検知してしまう誤検知率も計る．その結果と考察を本項でまとめる．

### 5.2.1　実験環境

実験を行なった環境と被験者のデータは以下にまとめる．推論サーバには WindowsXP が搭載された IBM 社製 ThinkPad T42p を利用した．CPU は Intel Pentium M 2.00GHz のモデルである．スペックを表 5.1 にまとめる．また，被験者は，表 5.2 にある身長 170cm 代の 20 代男性を用いた．

表 5.1　推論サーバデータ

| 項目 | 説明 |
|---|---|
| モデル | IBM ThinkPad T42p |
| プロセッサ | Intel Pentium M 2.00GHz |
| メモリ | 2GB |
| OS | Windows XP |
| HD | 80GB |

表 5.2　被験者データ

| 項目 | 説明 |
|---|---|
| 性別 | 男性 |
| 年齢 | 22 歳 |
| 身長 | 173cm |

アクチュエータには Sony 社製の Vaio TypeU を利用している．Vaio TypeU には図 5.3 にあるようなカメラが内臓されている．そのため，今回の実験はカメラを使っていないことを利用者に対し明確に示すため，カメラ部分にシールを張ることでカメラにカバーをして行なった．シールを貼り，カメラ部分を物理的に隠した様子を図 5.4 に示す．

図 5.3　Vaio TypeU の内臓カメラ

図 5.4　Vaio TypeU の内臓カメラを隠した様子

### 5.2.2　ソフトウェア設定

推論のサンプリング回数には前項で導いた最適なサンプリング回数である 2000 回を用いる．表 5.3 には学習データとして利用したデータ数をまとめた．推論モデルには，便座が上がっている状態と，下がっている状態の 2 種類のモデルに，それぞれ 15 回分の通常状態を学習させ，条件付確率を設定したものを利用する．センサデータのフレーム数で数えると便座が上がっている状態は 982 フレーム，下がっている状態は 621 フレームである．

表 5.3　学習データ

| モデル | 利用回数 | 学習フレーム回数 |
|---|---|---|
| 便座上 | 15 | 982 |
| 便座下 | 15 | 621 |

### 5.2.3　検出目標状態の定義

本実験では，身体異常状態の精度検知に用いる状態を気絶と痙攣と設定し，それぞれをどれ程の精度で，発生からどれくらいの経過時間で検知できるかどうかを計ることを目的としている．またそれぞれの状態を利用者が座っている時，倒れている時の両方で行い，それぞれの精度，検知時間についても見当を行なった．

図 5.5　座って気絶している状態の例

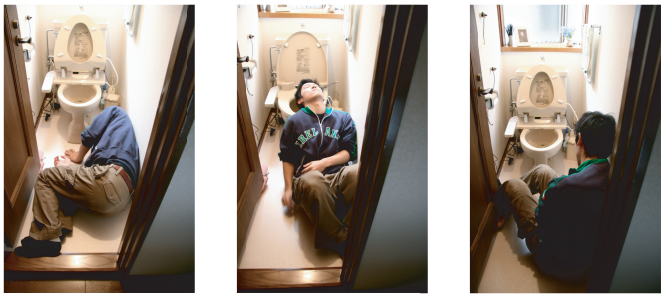

図 5.6　倒れて気絶している状態の例

気絶は，利用者が脱力し動かなくなっている状態とし，痙攣では，利用者は移動せず，その場で四肢，胴体，またはその何れかを断続的に動かしている状態のことを指す．それぞれ座った状態では，利用者は便座に座った状態でその動作を行い，倒れた状態は床の上に横たわりながらその状態を取る．図 5.5 と図 5.6 では，それぞれ実験に利用した座った状態と倒れた状態での気絶の例である．

検知時間の最大許容値を 1 分とし本評価を行なった．1 分という時間は，身体異常状態で想定される最悪な事態である心停止が起こった際にでも AED(自動体外式除細動器) を利用することで，80%〜90% の確率で心拍を再開できる時間である．図 5.7 に示す心停止から除細動までの経過時間と生存確率の関係のグラフ [19] を示す．心拍の再開の可能性は 1 分ごとに 7〜10% ずつ低下する．近年，AED は無資格の一般市民でも利用できるようになったため，本システムと一緒に家庭に AED を設置することで，1 分以内に身体異常状態を検知できた場合，私的空間で心停止が発生してしまった最悪の場合でも高い確率で救うことが出来ることになる．更に，心拍が停止する以前の段階で，利用者が苦しんでいる状態や，痙攣している状態を異常として発見することができれば，救助できる確率は更に高いものとなる．

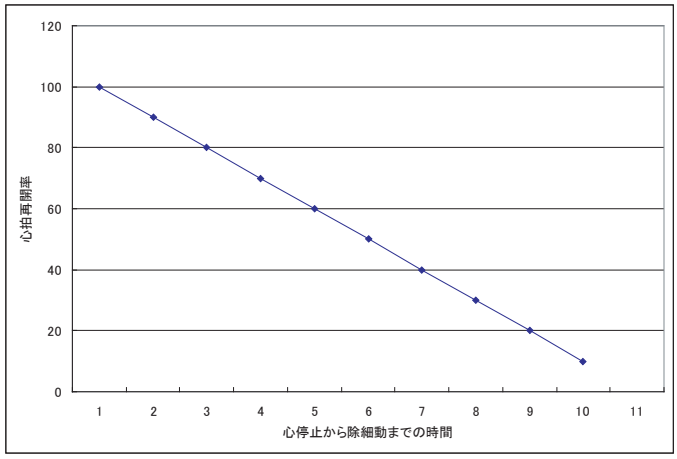


図 5.7　心停止から除細動までの時間と心拍再開率の関係

### 5.2.4　実験手順

実験は事前に設定した手順に基づき行なった．まず，身体異常検知の制度を計る実験の手順を以下に示す．身体異常検知精度を測る実験では，1 分間の制限時間を設け，その中で検出できるか否かを確認した．また，時間内に検出できた場合は，経過時間を確認した．

身体異常検知精度実験の手順

1. **開始スイッチを入れる**
2. **ドアを開けて入室する**
3. **入室して 5 秒〜15 秒ほど通常の行動を取る**
4. **異常状態を再現する**
5. **1 分間を制限時間として異常を検知するまでその状態を取る**

次に，誤検知率を算出するための実験の手順を示す．誤検知の算出は，被験者に通常の行動をトイレの中で行なってもらい，その中でシステムが身体異常を誤って検出してしまうかどうかを確認する．

誤検知率算出実験の手順

1. **開始スイッチを入れる**
2. **ドアを開けて入室する**
3. **通常の行動を取る**
4. **異常と誤検知された場合は中断し誤検知カウントを上げる**

### 5.2.5　実験結果

上記の手順で実験を行なった結果，以下の実験結果が得ることが出来た．実験の結果を表にまとめる．

表 5.4，5.5 は倒れて気絶している状態の精度と結果のデータである．まず，倒れた状態で気絶している状態は，60 秒以内という制限時間は 20 回の試行に対して全ての試行において正しく身体異常を検知した．30

表 5.4　倒れて気絶している状態の検知制度 (左:60 秒, 右:30 秒)

| | 検知 | 試行 | 精度 |
|---|---|---|---|
| 便座上 | 10 | 10 | 1.0 |
| 便座下 | 10 | 10 | 1.0 |
| 総合 | 20 | 20 | 1.0 |

| | 検知 | 試行 | 精度 |
|---|---|---|---|
| 便座上 | 10 | 10 | 1.0 |
| 便座下 | 7 | 10 | 0.7 |
| 総合 | 17 | 20 | 0.85 |

表 5.5　倒れて気絶している状態の検知結果

| 試行回数 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 平均 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 便座上 | 24 | 17 | 16 | 25 | 27 | 20 | 13 | 16 | 25 | 18 | |
| 便座下 | 28 | 32 | 37 | 30 | 33 | 28 | 26 | 15 | 29 | 28 | 24.35 |

秒に制限時間を縮めてみたところ，そのうち 3 回だけ検知をすることが出来なかった．結果，30 秒以内では 85% の検知精度であった．また，平均は異常状態発生から検知までの経過時間の平均は 24.35 秒であった．倒れた状態での気絶は便座が上がっている状態，下がっている状態，両方の状況で起こる可能性があるため，それぞれ 10 回ずつで 20 回の試行とした．

表 5.6　座って気絶している状態の検知精度 (左:60 秒, 右:30 秒)

| | 検知 | 試行 | 精度 |
|---|---|---|---|
| 便座下 | 20 | 20 | 1.0 |
| 総合 | 20 | 20 | 1.0 |

| | 検知 | 試行 | 精度 |
|---|---|---|---|
| 便座下 | 19 | 20 | 0.9 |
| 総合 | 19 | 20 | 0.9 |

表 5.7　座って気絶している状態の検知結果

| 試行回数 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 平均 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 便座下 | 29 | 28 | 30 | 29 | 26 | 12 | 27 | 12 | 25 | 25 | |
| | 51 | 30 | 15 | 27 | 30 | 37 | 24 | 18 | 29 | 28 | 26.6 |

座って気絶している状態に関してもほぼ確実に取得できた．表 5.6, 5.7 に座って気絶している状態の検出精度と結果を示す．これを見ると 20 回の試行における全てに対して 60 秒以内に検知できたことがわかる．さらに，30 秒以内で検出できた回数は 18 回で全体の 90% になる．状態の発生から平均 26.6 秒後に身体異常と検知されている結果となった．検知までの経過時間平均は 26.6 秒であった．尚，座っている際の気絶は便座が降りている状態が前提であるため，便座が上がった状態での試行は行なっていない．

表 5.8　倒れて痙攣している状態の検知精度 (左:60 秒, 右:30 秒)

| | 検知 | 試行 | 精度 |
|---|---|---|---|
| 便座上 | 20 | 20 | 1.0 |
| 便座下 | 20 | 20 | 1.0 |
| 総合 | 20 | 20 | 1.0 |

| | 検知 | 試行 | 精度 |
|---|---|---|---|
| 便座上 | 7 | 20 | 0.7 |
| 便座下 | 2 | 20 | 0.2 |
| 総合 | 9 | 20 | 0.45 |

表 5.9　倒れて気絶している状態の検知結果

| 試行回数 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 平均 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 便座上 | 11 | 20 | 26 | 31 | 26 | 27 | 33 | 28 | 34 | 20 | |
| 便座下 | 36 | 17 | 31 | 48 | 11 | 33 | 44 | 31 | 31 | 33 | 31.5 |

次に，倒れた状態での痙攣においても 20 回の試行と 60 秒の制限時間では確実に検知する事ができた．その具体的な結果を表 5.8，5.9 に記す．しかし，気絶とは違い，制限時間を 30 秒とした結果，20 回中で 9 回しか検知する事ができなかった．30 秒の制限時間を設けた際の検知制度は 45% であった．発生から検知までの平均経過時間は 31.5 秒である．倒れた状態での痙攣に関しても，便座が上がっている状態，下がっている状態，両方の状況で起こる可能性があるため，それぞれ 10 回ずつで 20 回の試行の試行を行なった．

表 5.10　座って痙攣している状態の検知精度 (左:60 秒, 右:30 秒)

| | 検知 | 試行 | 精度 |
|---|---|---|---|
| 便座上 | 2 | 20 | 0.1 |
| 総合 | 2 | 20 | 0.1 |

| | 検知 | 試行 | 精度 |
|---|---|---|---|
| 便座下 | 0 | 20 | 0.0 |
| 総合 | 0 | 20 | 0.0 |

表 5.11　座って気絶している状態の検知結果

| 試行回数 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 平均 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 便座下 | 43 | x | x | x | x | x | x | x | x | 47 | |
| | x | x | x | x | x | x | x | x | x | x | |

しかし，最後の座った状態での痙攣を取得する実験の結果は 60 秒以内の制限時間に身体異常を検知できた回数は 20 回の試行中わずかに 2 回で，30 秒以内に限っては 1 回も検知することができなかった．60 秒間の制限時間においても検知精度は 10% ということになる．この実験も，便座に座っている状態が前提であるため，便座が上がった状態での試行は行なっていない．便座が下がった状態のみで 20 回の試行を行なった結果である．表 5.10，5.11 にその精度と結果を記す．

表 5.12　気絶状態の検知精度 (左:60 秒, 右:30 秒)

| 検知 | 試行 | 精度 |
|---|---|---|
| 40 | 40 | 1.0 |

| 検知 | 試行 | 精度 |
|---|---|---|
| 35 | 40 | 0.875 |

表 5.13　痙攣状態の検知精度 (左:60 秒, 右:30 秒)

| 検知 | 試行 | 精度 |
|---|---|---|
| 22 | 40 | 0.55 |

| 検知 | 試行 | 精度 |
|---|---|---|
| 0 | 40 | 0.00 |

表 5.12，5.13 は総合的に 2 つの身体異常状態がどれ程の精度で検知できているかをまとめたものである．気絶は 40 回の試行に対して 60 秒以内には全てにおいて検知できている．また，30 秒以内でも 35 回検知して

いる．一方，痙攣状態は同じく 40 回の試行に対して，60 秒以内には 22 回，30 秒以内では 0 回の検知しかできなかった．

### 5.2.6 実験考察

今回の実験において，気絶は利用者の姿勢に関わらずかなりの精度で検知できることがわかった．経過制限時間を 60 秒とした場合では 100% の検出精度が達成され，30 秒とした場合でも 87.5% の高精度で検出できている．検知までの経過時間のばらつきは，それまでの利用者の行動に依存するものであるが，利用者がそれまでどのような行動を取っていても平均して約 30 秒で検知できることがわかった．しかし，気絶状態は一定時間動きがない状態として定義できるため，単一の広域視野のセンサと述語論理でも抽出できてしまう．そのため，本研究の効果は気絶状態の検知精度では測れない．

一方で，単一の広域視野のセンサと述語論理では検出が難しい痙攣状態の検知精度を見ると，倒れて痙攣をしている場合は発生から 60 秒以内であれば 100% 検知できているが，座った状態で痙攣している場合は 10% しか検出することができなかった．そのため，痙攣という状態は発生から 60 秒までの間に 55% でしか検知できていないことになる．

この結果は利用者が座った状態で痙攣している時のセンサデータのパターンが通常な状態に似ているため起こったと考えられる．結果，便座に利用者が座っている通常状態と座って痙攣している異常状態ではセンサデータのパターンに十分な差異を出す事ができず，検知漏れが生じてしまう．この 2 つの状態の差異を出すために考えられる手段は 3 つ考えられる．

- 設置するセンサの数を増やす
- 推論ノードの変数の種類を増やす
- 設置するセンサの種類を増やす

まず，一つ目の設置するセンサの数を増やす対策について説明する．これは現在利用している 4 つのセンサだけでは存在するはずである座った状態での痙攣と通常状態との間でセンサデータのパターンの差異を発見できていない可能性があるためである．センサの数を増やすことで，取得できる情報の粒度が向上し，それに比例してパターンの数も増える．また，4 つでは四肢や胴体動きの有無しか取得できなかったが，センサの数を増やすことで，それらの移動が取得できるようになる．結果としてこの 2 つの状態のセンサデータのパターンの違いを見出すことができ，利用者が座っている状態でも痙攣していることを検知できるようになるという仮説である．しかし，設置するセンサの数を増やすことはコスト面からも，推論の速度の問題からも多ければよいわけではないことは自明である．

また，推論ノードの変数の種類を増やすという対策は，利用者が動いていないことの方が身体異常に近いだろうという観念に基づき構築された現在のモデルの修正を意味する．現在はセンサが動きを検知していない継続時間という推論ノードがモデルに含まれているが，センサが動きを検出している継続時間という推論ノードはない．センサが動きを検出していない継続時間とセンサが動きを検出している継続時間は一つがわかればもう一方もわかるという変数ではない．そのため，痙攣のように動いているにも関わらず身体の異常を表すような状況には動きを検出していない継続時間という推論ノードだけでは不十分であることがわかる．

最後に，設置するセンサの種類を増やすことは，焦電型赤外線センサで取得可能な動きの有無以外にも情報が取得可能になるということである．これは，動きの有無以外に 2 つの状態に対する決定的な相違点が存在する可能性を考慮し提案している．その 1 つに便座に座っている利用者の重心の変化などが考えられる．

このような対策，それぞれに対し改善の見込みはある．まずは，コストのかからない推論ノードの変数の種類を増やすことからはじめ，設置するセンサの数と種類の増加の 2 つの対策については，コストも考慮し，種類と数を最適なものにする必要がある．

今回の実験の成果は決してよいものとはいえないが，今回，既存の方法では検知できなかった倒れた状態での痙攣は発生から 60 秒以内に 100% 検知できることがわかった．この結果は本研究が採用した手法で身体異常を検知できることの可能性を示す結果である．そのため，この方法を基礎に研究を進めることができることを十分に示している．今後は上記の 3 つの対策をうまく組み合わせて更なる精度を向上させることが課題となる．

## 5.3 ユーザビリティ評価

本項では $\text{Life}_2$Guard システムのユーザビリティを定性的に評価した結果をまとめる．評価は実験室の環境ではなく，現在，居住空間として利用されている家庭のトイレを利用した．50 代の主婦，70 代の主婦の 2 人に協力を依頼し，2〜3 回，本システムの動作しているトイレを利用してもらった感想を述べてもらった．実験の環境や構成やソフトウェア設定に関しては精度評価を測った時点のものと変わらない．

### 5.3.1 実験結果

50 代の主婦からは以下のような 4 つの意見が得られた．

- 始めのうちはセンサが気になっていたが，何回かのうちに慣れてきた．
- 普通に使用しているだけなのに，トントントン (安全確認音) がなるとびっくりしてあせる．
- カメラで見られているような感覚はないので，安心だ．
- 掃除や水濡れにセンサが大丈夫なのか気になる．

また，70 代主婦からの意見は以下の 4 つである．

- 無視すればよいとわかっていても，ちょっとした動きでトントントン (安全確認音) がなってしまわないか気になってしまう．
- センサも説明を聞いていれば気にならない．
- 動作が映像に映らないので安心である．
- 異変があったら知らせてくれるので心強い．

### 5.3.2 実験考察

システムを利用してみた被験者からの意見を検証すると，誤検知に対する対処の問題と，センサデバイスに対する意識の問題の両方が伺える．まず，検知漏れに対する問題に関しては，それぞれの被験者は敏感になっているように感じられる．確認音が鳴るとあせる，鳴らないように気をつけるという利用者のシステムに対する意識は，利用者がサービスの存在を意識しているということである．サービスへ対する過剰の意識は，利用者のトイレの利用にストレスを与えてしまうため良い結果とはいえない．現在の誤検知の確認メッセージに無機質なノック音を利用したことが，このような利用者の焦りを招く原因とも考えられる．誤検知の確認メッセージは日本語で，利用者に心配を与えないインタラクションに改善する必要がある．また，ボタンを押すと

いう行為は物理的な距離を解決しなくてはならないため，声によるインタフェースの採用も検討できる．次に，センサデバイスに対する利用者の意識であるが，事前にセンサの種類に関して説明を行なったところ，感想からは過剰に意識していた様子は見られなかった．本研究で用いたセンサはプロトタイプであるり，パッケージもしていない上にやや大きめである．しかし，この状態にも関わらず，利用者のセンサに対する意識が低かったことは良い実験結果であるといえる．センサは，製品化フェーズにおいては壁に埋め込まれる，もしくはデザインし，パッケージングされたものを利用することを念頭にしているため，今回得られた意見がさらに改善される可能性が大きい．

## 5.4 本章のまとめ

本章では，まず今回の実験で利用するベイジアンネットワークのモデルを用いて Likelihood Weighting のアルゴリズムの性能評価を行なった．実験の中では，シュミレーションを行なう回数と，その結果導かれた結果ノードの確率の値の収束度合いの関係について調査を行なった．結果，速度的にも，制度的にも 2000 回が最適な回数であり，実時間推論を十分に実現できることを結論付けることができた．次に，身体異常状態の抽出精度の評価を行なった．対象とした身体異常状態は，気絶と痙攣で，それぞれ倒れている状態，座っている状態において実験をした．両状態での気絶，そして倒れている状態での痙攣に関しては良い実験結果が得られたが，座っている状態での痙攣に関しては，通常状態との差異をシステムは検知することができず，ほとんど検出する事が出来なかった．結果，いくつかの課題も浮き彫りになったが，その対策も考察することができた．また，単一のセンサと述語論理では検出することの難しい倒れた状態での痙攣がほぼ確実に取得できたことは，本研究の更なる可能性を示す上で重要な結果である．

# 第 6 章

# 関連研究

本章では，本研究の関連研究を紹介し，本研究との差分の解説を行なう．まず，「センサとコンテクストアウェア・アプリケーション」の代表的な手法と具体的な研究について触れ，その後で，本研究と目標を同じくする 3 つの研究の解説を行なう．

## 6.1 センサとコンテクストアウェア・アプリケーション

一つの研究分野としてセンサとコンテクストアウェアアプリケーションという分野がある．この分野では，その時々の人間の振る舞いや，場の状況にあったサービスを実行してくれるアプリケーションを作成するために，センサデータから人の振る舞いや場の状況を認識する手法の研究が盛んに行なわれている．本研究も，私的空間の中の利用者の身体異常を検知し，助けを求めるアプリケーションに有用性を見出し，プライバシーを侵害しないセンサデータから身体異常を検知する手法を提案した．本項では，センサを利用したコンテクストアウェアアプリケーションと、コンテクスト抽出手法の研究を行なっている本研究の関連研究を紹介し，本研究との相違点も説明する．

### 6.1.1 ウェアラブルコンピュータによる実現手法の研究

小さなコンピュータを身にまとうことで，利用者がコンピュータの支援を受けることができるウェアラブルコンピューティングの概念を用いて，センサとコンピュータを身にまとい，コンテクストアウェア・アプリケーションを開発している研究は多数存在する．

その中でも 1970，1980 年代からバックパック型のウェアラブルコンピュータを提案してきた Steve Mann は彼の Smart Clothing に関する論文 [20] の中で，コンピュータの小型化は，彼が提案し続けてきた大きなウェアラブルコンピュータをいつでも継続的に身に着けることを可能にしたと述べている．その後で，それらのウェアラブルコンピュータが 1 つのユニットに統合され，携帯電話などの一般的に既に人々が持ち歩いているものに埋め込まれることで予期せぬ効果が期待できると述べている．さらにそれらに様々な生体センサが埋め込まれることで，個人健康管理システムなどへの応用が可能であると述べている．生体センサつきの服などで，患者の医療情報が取得でき，それを，その服内に保存できることから，セントラルサーバも不要になり情報の管理が分散して現在騒がれているプライバシーの問題も解消されるとしている．彼の SmartClothing の提案の実装には Smart underware, Smar Eyeglasses，そして図 6.1 の Smart Shoes などがある．

また，Jonny Farringdon，Andrew J Moore らは Sensor Badge[21] という 3 軸加速度センサを搭載した小

図 6.1　SmartShoes

型のコンピュータを開発している．それを持ち歩くことで，座っている，立っている，横になっている，歩いている，走っているといった人間の行動を取得可能にした．同研究者らは，さらに Sensor Jacket と呼ばれる曲がり具合を計るセンサをジャケットに織り込んだデバイスを開発し，利用者の姿勢や振る舞いを取得する研究に取り組んでいる．これらのデバイスの利点は，環境側へデバイス設置などを求めず，いつでもどこでも利用できるということと延べられている．

デバイスの開発ではなく，振る舞いや状況のコンテクストを把握する手法に注力した研究には Jonathan Lester, Tanzeem Choudhury らの研究 [13] がある．彼らは 3 軸加速度センサを 2 つと，超音波センサを，手首とひじに着用して，工事作業で利用されるコンテクスト 9 種類 (hamering, sawing, filling, drilling, sandin, gripping, tightening a screw. tightening a vice and opening a drawer) の状況を抽出している．ユーザに依存する実装では 70%, ユーザに適応された実装では 66.1%, ユーザに非依存な実装であれば 65.5% の精度で結果が得られている．Jani Mantyjarvi，Johan Himberg らは，腰に 2 つの加速度センサを取り付け，あらかじめ定義されたオフィス環境での行動 4 種類 (waling in a corridor, walking up and down stairts, opening doors) を抽出する研究をしている．成果は 83%〜90% の精度で抽出できたとしている．

上記の 4 つの研究は，複数のセンサを身にまとわなくてはならなかったり，センサを付ける場所が指定されていたり，また，あらかじめ指定された物を着なければならないものであった．複数のセンサを使ったり，利用者に制約を課すことで，システムの精度は向上し信頼性を確保できるが，実用性が下がってしまう．そこで，D.Minnen, T.Starner らは，様々な健康管理アプリケーションに適応できる信頼性と実用性のある人間の振る舞い検知システム [22] を作成した．彼らは信頼性と実用性の確保を実現するために以下の 3 条件を設定した．

- 身体上の 1 つセンサからのデータのみを必要とする．また，その位置も指定しない．
- ある一個人だけでなく，誰でもが利用できる．個人への適応は精度向上の範囲に収める．
- 安価なセンサの集合からでも有効に動作させる．

これらを実現させたシステムを実際に動作させ評価を行なったところ，8 つの振る舞いを 12 の被験者から約 90% の確率で計算させることができたと書かれている．今後，この研究で利用したようなセンサが携帯電話などの現在に持ち歩かれているデバイスに埋め込まれることで，一般的に普及させることができると評価している．

これらの研究から，ウェアラブルコンピューティングにより利用者の振る舞いや状況を検知し，サービスを行なうことは，信頼性があり実現できることが理解できる．ウェアラブルコンピューティングでは，利用者から直接必要な情報を取得できるので、システムがデータを解析し振る舞いや状況を検知する上出はとても都合が良い反面，システムの利用者にとっては制約が多く，負担を要する手法である．その為，一般的に家庭で，

町で，誰もが利用しているサービスとすることは難しい．一方で病院での治療やゲームのコントロールなど，その負担を容認できるだけの利点や，強制力のある状況，または着用することが負担とならない状況にとても有効である．

D.Minner, T.Starner らの研究のように，負担を極力軽くして振る舞いや状況を検知を実現しようとする研究が進み，携帯電話など，私たちが普段持ち歩くデバイスでそれが実現できるようになれば，実用的だが，本研究のように日常的に万一のための身体状況のモニタリングを行なう場合，24 時間 365 日の観察が必須となってくる．本研究はトイレや風呂場などのプライベート空間を想定しているため，公共においては常に身に着けていることが前提とされる衣服や，携帯電話でさえ，身に着けていない環境も想定される．さらに生体センサを利用すると，センサを肌に装着し，文字どおり肌身離さず装着していなければならない．そのため，そのような場合は，人間が何かを装着してそこから情報を得る手段は不適切といえる．このような場合は，環境から利用者をモニタリングする方法を考えなくてはならない．

### 6.1.2 空間に配置された環境情報センサによる実現手法の研究

ウェアラブルコンピューティングでは，精度や信頼度の高い振る舞い検知や状況検知などのコンテクスト抽出ができる一方で，利用者への制約や負担が大きくなってしまう．そのため，利用者ではなく，環境情報センサを主に扱い，データを収集する研究も行なわれている．その一つが Stephen S. Intille, Kent Larson[8] らの研究で，彼らは自然な形で利用者からセンサデータを収集することを目的としたセンサキット MITes を開発している．MITes には 5 つのウェアラブルセンサ (onbody acceleration, heart rate, ultra-violet radiation exposure, RFID reader wristband, and location beacons) と，さらに 6 つの環境情報センサ (movement, movement tuned for object-usage-detection, light, tempereture, proximity and current sensing in eletric appliances) を含んでいる．また，Stephen S. Intille らは，Mites を含むセンサを利用し，これまで実験室ベースであったコンテクストアウェアの研究を実際に人が住む居住空間としてのアパートを利用して行なった．アパートには扉の開け閉めや，電気の点灯消灯，電子機器のオンオフのわかる on/off スイッチ，温度センサ，湿度センサ，気圧センサ，電流センサ，水流センサ，ガスセンサなど 125 個が取り付けられ，リモコンや箒，引き出しなど利用者が操作することのできる日用品の監視を行なっている．図 6.2 は部屋のセンサ設置例である．

図 6.2　An apartment with sensor installed

## 6.2 類似研究

本項では，コンテクストアウェア・アプリケーションの中でも，本研究と同じ健康リスクを管理するアプリケーションの代表的な研究を 3 つとりあげ，本研究との共通点，相違点を明らかにする．そのうち 1 つの研究では，プライバシーの保護を研究の主題としているため，本研究の取り上げているプライバシーとの理解の違いについても説明する．

### 6.2.1 Sensorized Elderly Care Home の研究

Toshio Hori, Yoshifumu Nishida らは，社会の高齢化に伴う，高齢者の介護の人手不足，また，介護時における過度の巡回における非効率に問題意識を持ち，センサネットワークを利用した，高齢者の介護の効率化を図るためのシステム [23] を構築した．システムは介護の必要な高齢者が事故がよく起こる場所に入ったら，それを介護師に知らせることで，巡回の効率を上げる手助けを行なう．このシステムは，超音波受信機を介護施設に取り付け，Ultra Badge と呼ばれる超音波発信器を用いて，高齢者の位置を把握している．Ultra Badge は高齢者の車椅子に装着されている．彼らは予め介護師に聞き取りを行い，事故が起こりやすい場所をベッド脇，トイレ，階段と絞るり，その前提に基づいてシステムの実装を行なった．

論文内では既存の利用者のモニタリング手法であるカメラと映像を用いた手法は，本研究と同じくプライバシーの問題から論外としている．さらに，“embedded distributed sensor network for a home for the aged” と埋め込み型で環境がモニタリングするという目標も本研究と共有できる．しかし，実際は車椅子にセンサを取り付けることから利用者に制約を課している点があり，完全に環境側からのモニタリングを実現できていない．この研究の場合は，利用シーンが高齢者の介護施設と限定されていること，また，対象となる利用者が動く際には必ず車椅子を利用しなくてはならないという前提条件があるため，この手法は妥当であるといえる．本研究の場合は，一般家庭で，利用者の層も幅が広いため，そのような前提条件がないため，この手法は利用できない．

また，大きな相違点として，この研究におけるコンテクストは場所であることに対して，本研究におけるコンテクストは身体状態である．場所コンテクストは超音波センサのデータを抽象化しているが，決まった公式に従い抽象化しているため，データのインプットに対して一意のコンテクストがあり，絶対度が極めて高いコンテクストである．そのため，確率的な処理を行なうことも必要としない．場所と比較すると，本研究の扱う身体異常コンテクストは抽象度が高く，確率的な処理を必要とする．これらのことから，両研究の間ではコンテクスト対象の違いから取るべきアプローチが大幅に相違する．

### 6.2.2 House_n の研究

MIT の Emmanuel Manguia Tapia らは，“tape and forget” というセンサ設置に関する標語を設定し，低レベルで，どこにでも利用できるセンサを用いて人間の家庭内での行動を把握するシステム [7] を研究，構築している．彼らは，日常生活を営む上で不可欠な日常動作を表す ADLs(日常生活動作),Instrumental ADLs(手段的日常生活動作) が身体状態の変化を検知する上で重要な指標の一つであるという医療専門家の見地から，それらをシステムが検知して，医療アプリケーションに利用することを目指し研究開発を進めている．その応用事例の一つとしてアンチエイジングのための生活モニタリングシステムを開発した．コンテクスト抽出のアプローチとして彼らは代表的な 6 つのアプローチを取るとしている．そのアプローチと，採用した理由として

述べられていることを以下にまとめる．

- **Supervised lerning**
  家やアパート，またそれらの家具のレイアウトはばらばらであり，利用者の行動パターンも同じことはありえない．そのため，同じ振る舞いも，センサデータとしてみたときは，環境が違うだけで全く違う値となって現れる．教師あり学習を利用し，明確な学習期間を設定することで，そのような問題を回避する．
- **Probabilistic classification**
  曖昧かつノイズの多く含む複数のセンサからのデータを解析するためには確率的に処理することで効率よく分類することができる．
- **Model-based lerning**
  モデルベースは一定量のサンプルデータから，数学的な確率モデルを作成するために，サンプルデータをそのまま保存しておく必要がないため，利用者のプライバシーを守ることができる．
- **Sensor location and type independent**
  理論上，アルゴリズムでどのセンサをどこに置くかを指定する必要はなく，指定しないことでセンサの設置の時間的コストが劇的に下がり，実用性が向上する．
- **Real-time performance**
  家庭内での利用では，リアルタイムに振る舞いを検知することによるメリットはとても大きい．
- **Online lerning**
  新しいデータをサンプルデータとして利用し，モデルを更新していくことによって，システムは利用者の習慣的な行為の変化にも対応できる．

アルゴリズムは, ベイジアンネットワークを利用し，モデルの形態として，本研究が採用した Naive Bayes を利用している．論文内で，彼らは Naive Bayes は全ての結果ノードが全ての原因ノードに独立的に依存しているため，現実を扱う問題領域においては効率が悪いとされていた Naive Bayes を事前に検証し，この領域で効率がよいことが証明したとしている．

この研究では，Berkeley Motes や Smart-ITS[24] などのセンサは様々な目的で利用を可能にしているためコストパフォーマンスに優れないとして，オブジェクトの状態を見張る機能だけを搭載した安価なワイヤレスセンサを利用している．このセンサをドア，窓，タンス，引き出し，電子レンジなど，家具や家電に装着し，データを収集している．結果、この研究の成果として，ADLs に属する行為とされる toileting，bathing，grooming などのコンテクストを最高 89% の精度で抽出している．

彼らの研究と，本研究では，分野，手法などは大きく類似している．まず，この研究が取っている 6 つの主要アプローチのうち，Sensor の設置位置の問題である “Sensor location and type independent” という項目以外は全て同じである点が上げられる．本研究では，求めるコンテクストが 1 つであるため，センサの種類をピンポイント型の焦電型赤外線センサと限り，実装を行なっている．また設置位置に関してはコストの面から最適な設置場所のガイドラインを設けている．両研究の決定的な違いは，求めるコンテクストの階層の違いである．彼らの研究が目標とするコンテクストは全て利用者の行動である．抽出した行動の組み合わせが意味することは解析せず，そこは医療分野の専門家に任せている．一方，本研究では抽象度の低いコンテクストを取得はしていない．本研究では，行動という抽象度コンテクストは扱わず，その上の概念階層である “身体異常” を対象としている．身体異常を通常状態の補集合として扱っているため，身体異常の定義には専門家の見地は必要ない．

また，本研究では Mica2Dot というリプログラム可能な比較的高価なセンサノードを利用していることから，コストパフォーマンスには優れない点が上げられるが，センサ自身はプロトタイプであるため，低価格化の余地は十分に残されている．実際に利用しているセンサは 1 種類のセンサのみであるため，単一利用を前提としてセンサノードを専用に作ることでコスト，そして大きさは大幅に下げられると考えられる．

### 6.2.3 uEyes の研究

Takuo Suganuma らは，プライバシーを考慮した優しい行動の監視システムとして uEyes[25] を実装し，研究を行なっている．uEyes は実空間上の被監視者の状況と，監視者と被監視者の社会的関係を考慮することで，利用者が求める質とプライバシー保護を満たす優しい行動監視システムを実現するとしている．その応用事例として，彼らは高齢者向けの監視システムを開発した．このシステムは常に監視対象の高齢者をカメラを用いて映像をストリーミング配信しているが，被監視者の現在の状況と，それを見る監視者と高齢者の社会的関係 (家族，隣人など) を考慮し，映像にフィルタをかけ，見せる情報量を変化させるというものである．実行すると，図 6.3 のようになり，4 つの関係がそれぞれ異なる利用者に対してそれぞれ異なる画質の映像を配信していることがわかる．

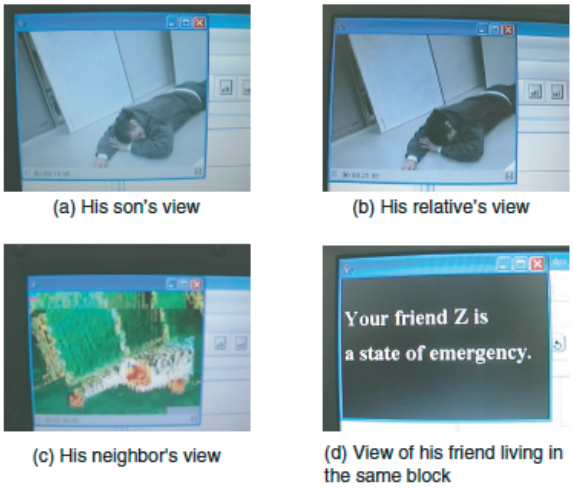


図 6.3　uEyes の実行画面

高齢の父親のことをシステムを利用して監視する息子の例が論文中に掲載されている．父親が寝ている場合，環境情報センサの値から利用者が寝ていることをシステムが把握し，父親のプライバシーを守るために映像のクオリティを下げて，各監視者に配信する．その際は監視者の社会的関係を考慮し，クオリティを決定する．また，被監視者の身体に危険がある状況では，センサがその状況を検知し，システムは高画質な映像を監視者に流す．この場合はプライバシーよりも緊急度が優先されるためである．このように uEyes は両利用者のニーズにあった QoS とプライバシーでリアルタイム映像を配信できるシステムである．

このシステムを実現させるには利用者が何も負担せずに，全自動で利用者の状況と社会的関係を抽出しなくてはならない．彼らは「Real Space Understanding Mechanism」と称し，それを実現している．その機構には，被監視者と監視者の社会的関係を抽出する「Human Relation Recognition Function」と，被監視者の状況を推測する「Situation Recognition Function」という 2 つの機能があり，いづれもオントロジーの概念を

用いて推測を行なっている．

この研究と本研究とでは，プライバシーの理解の仕方が異なっている．この研究におけるプライバシーは家族間においてはある程度共有が可能なものとして取り上げているが，本研究におけるプライバシーは家族内においても共有をすることを望まないものとして扱っている．そのため，本研究においてはプライバシーの開示レベルを分けることは出来ない．また，カメラの設置も本研究では認めていない．両研究の間で監視範囲が大きく違うために，このような違いが生じている．

また，社会的知識としてのオントロジーを用いたコンテクスト抽出は矛盾などは顕著に検出できるが，個人差の吸収が出来ない．その点で，個人個人で幅広く違う習慣が大きく関係する日常生活において，オントロジーを用いることは難しいと考える．そのため，本研究ではその個人差の吸収をすることも可能である確率的推論手法を採用している．

## 6.3 本章のまとめ

本章では，まず現在盛んに行なわれているウェアラブルコンピューティング技術を用いたコンテクスト抽出の研究を紹介した．この技術を用いたシステムは，コンテクスト抽出精度，システムの信頼度が上がるという利点があるという一方，システムを利用するためには，利用者がある程度の負担を受け入れなくてはならないという弱点があることを述べた．

次にウェアラブルコンピューティング技術を用いないコンテクスト抽出手法として，環境情報センサを利用した手法を紹介した．この方法では，利用者は何も負担せず，普段通りの行動を取ることが出来る．さらに、このような手法を用いた研究を紹介し，環境情報センサを利用したコンテクスト抽出でも信頼性を確保することが十分に可能であることを参照した．

また，本研究と目的や手法を共有する研究を 3 つ紹介し，それらの研究との相違点を述べ，本研究の抽出対象としているコンテクストの層の抽象度が高く，推論を行なう際に技術的課題があることを確認した．

# 第 7 章

# 結論

本章は，本研究の総括として，本研究で得たことをまとめ，本研究の貢献，そして課題と今後の展望を明確にする．

## 7.1 本研究のまとめ

現在，社会の高齢化に伴い，血圧の変動などにより身体異常を引き起こす現象が増加している．また，この被害を大きくしている要素として，身体異常が発生する場所に問題があることが考えられる．このような突発的な身体異常は気温差の大きい風呂場やトイレといった場所で多く発生しており，このような場所はほとんどの場合は 1 人で利用するため，発見が遅れてしまうのである．上記から利用者が 1 人でいる場所での身体異常を検知するシステムの開発が急務であることがわかる．しかし，トイレや風呂場といった場所は，プライバシーに対する意識が高く，利用者はそのような部屋の中を見られたくはない．また，そのような場所ではカメラのような映像として情報を記録するデバイスの存在に対しても嫌悪感を持つことは自明である．そのため，映像や画像を利用したシステムは，このような私的空間の監視に利用することは望めない．

そこで，カメラ以外の環境情報を取得するセンサを利用し，このような私的空間を監視し身体異常を検知するシステムを開発した．本研究はそのシステムを実現する上で必要な身体異常検知手法を提案，実装することである．研究の焦点は，システムを実現する上で考えなければならない以下の 4 つとした．

1. 環境情報センサによる身体センシング
2. 動作信頼性の確保
3. 実時間推論の実現
4. 誤検知通報の回避

環境情報センサによる身体センシングには，体の位置と動きの有無を把握でき，即時伝達性があり，環境によるノイズが少なく，更にコストも安いピンポイント型の焦電型赤外線センサを複数利用する．またベイジアンネットワークのアルゴリズムを利用し，データの欠落にも耐えうるエンジンを実装することで，動作信頼性を確保した．実時間推論の実現には，近似推論アルゴリズムとモデルの単純化による実現手法をとる．最後に誤検知による通報回避は確認メッセージで利用者に問いかけと，利用者のボタンによるインタラクションを用いることで，負担を最小限にしながら解決している．

上記にまとめた手法を用いて実際の居住空間のトイレを利用し実験を行なったところ，座って，および，倒れた状態での気絶と倒れた状態での痙攣を高い精度で検知することができた．しかし，座った状態での痙攣

は，通常状態とセンサデータのパターンが類似しているため，判別がほぼできない結果が得られた．

## 7.2 本研究の貢献

本研究は複数の環境情報センサと，ベイジアンネットワークの近似推論を行なうことで，利用者の負担なく実時間内に身体異常を検知することができるという可能性を実証する事が出来た．この手法を用いることで，利用者に負担を課す生体センサを利用せずとも身体異常を検知できることから，日常生活での健康リスクの回避を目指すアプリケーションを今後広く実現，普及する上で大きな貢献を残すことができた．さらに，環境情報センサの数や種類が限られている場合，通常状態と異常状態の差分をシステムが学習できないこともわかり，この目標における今後の課題を発見することができた．

## 7.3 今後の課題と展望

今回の研究の成果は，論文内で提案してきた私的空間で身体異常を検知するための手法が強力であり，日常生活の中で起こる健康リスクを避けるために十分に活用できる可能性を示せたことである．今回の成果から，本研究の今後の課題・展望として，3 つ上げることができる．

1. 身体異常検知精度の向上
2. インタフェース部分の整備
3. 他の私的空間への技術応用

今後の展望として最重要であり，まず第一に上げられるのは，この手法を基礎に，次はどのような状況でも高い精度で身体異常を検知する具体的な方法を考え，実装することである．検知精度の向上への対策として，ひとつに様々なセンサ利用することがある．今回は，利用者が起こす行動以外の環境の変化を検知してしまう危険性を考慮し，赤外線センサのみを利用したが，それだけでは不十分であることが分かった．そのため，今後は様々なセンサを赤外線センサで取れないデータを補足するものとして利用していく．また，設置する数も必要に応じて増やして，今回取れなかった身体異常と通常状態の差異をセンサデータに学習させることができるようにさせる．

次に，インタフェース部分の整備も重要な課題となる．センサのインタフェースがプロトタイプであるため，写真にもあった様に基盤がむき出しで，重々しく感じられる．サービスの存在を利用者に忘れさせ，普段どおりの利用モデルの中で，必要なときだけアクチュエートされるユビキタスサービスに向けて，センサインタフェースの整備は重要である．また，誤検知回避手法における利用者への状況確認メッセージと，それに対する応答のインタラクションも考慮する必要がある．現在の効果音と，それに対するボタン入力インタフェースではなく，自然言語による音声確認と，自然言語による音声応答でより自然で利用者に優しいインタフェースを実現していきたい．

最後に，今後は他の場所においての身体異常も検知できるよう，様々なモデルを考えていく．Life$^2$Guard システムは，推論モデルを変化させることにより，容易に他の場所へ適応できるようフレームワーク化されているため，対象の変更は容易に可能である．今回，私的空間の代表例としてトイレという限定された空間を対象に推論モデルを立て，Life$^2$Guard システムを利用した．次はトイレ以外にも身体異常検知の需要がある私的空間で，Life$^2$Guard システムのフレームワークを利用して適応させていく．結果，浴室，寝室，リビングなど，あらゆる場所でプライバシーを守りながら身体異常を検知できようになることを目標とする．

# 謝辞

本研究の機会を与えてくださり，絶えず丁寧なご指導を賜りました，慶應義塾大学環境情報学部教授徳田英幸博士に深く感謝致します．また，貴重なご助言を頂きました應義塾大学環境情報学部准教授高汐一紀博士に深く感謝致します．慶應義塾大学徳田研究室の諸先輩方に折にふれ貴重なご助言，ご指導を頂きました．特に，慶應義塾大学政策・メディア研究科特別研究講師中澤仁博士，慶應義塾大学政策・メディア研究科後期博士課程由良淳一氏には絶え間ざるご指導と励ましを賜りました．ここに感謝の念を表します．また，私が学部 2 年時にコンピュータサイエンスへの興味を与えてくださった慶應義塾大学政策・メディア研究科特別研究講師岩井将行博士，常に研究の方向を示してくださった慶應義塾大学政策・メディア研究科後期博士課程門田昌哉氏，厳しくも有意義な助言をいつも与えてくれた慶應義塾大学政策・メディア研究科修士山崎俊作氏，ありとあらゆる面で研究生活を支えてくれた慶慶應義塾大学政策・メディア研究科修士課程徳田英隼氏に，心から感謝したします．最後に，研究生活を支えてくれた家族，研究の日々を i208 で共に過ごした慶應義塾大学環境情報学部 4 年伊藤友隆氏，河田恭兵氏，橋爪克弥氏，角田龍二氏，KMSF,HORN 研究グループ，その他多くの友人に深く感謝し，謝辞と致します．

2008 年 2 月 7 日

生天目直哉

# 参考文献


[1] 東邦ガスホームページ一 2005 年 12 月号 5, 2005.

[2] 厚生労働省 人口動態統計, 2000.

[3] ミシェルフーコー, 田村淑. 監獄の誕生 監視と処罰. 新潮社, 1977.

[4] Terazawa Masahiko, Yamazawa Kazumasa, Takemura Haruo, and Yokoya Naokazu. Estimating areas of moving objects for a remote surveillance system using omnidirectional sensors. Technical report, The Institute Of Electronics, Information And Comunication Engineers, 2001.

[5] Rosalind Picard and Jennifer Healey. Affective wearables. In *Proceedings of the First International Symposium on Wearable Computers*, 1997.

[6] Ryan Aipperspach, Elliot Cohen, and John Canny. Modeling human behavior from simple sensors in the home. In *Pervasive 2006 Proceedings*, 2006.

[7] Emmanuel Munguia Tapia, Stephen S. Intille, and Kent Larson. Activity recognition in the home using simple and ubiquitous sensor. In *Pervasive 2004 Proceedings*, 2004.

[8] Stephen S. Intille, Kent Larson, Emmanuel Munguia Tapia, Jennifer S. Beaudin, Pallavi Kaushik, Jason Nawyn, and Randy Rockinson. Using a live-in laboratory for ubiquitous computing research. In *Pervasive 2006 Proceedings*, 2006.

[9] 吉崎賢介. 「救命の連鎖」の確保と実現に向けて.

[10] Mihai Marin-Perianu and Paul Havinga. D-fler: A distributed fuzzy logic engine for rule-based wireless sensor network. In *Ubiquitous Computing Systems 2007*, pp. 86–101, 2007.

[11] M. Ozawa, Y. Kawahara, N. Kawanishi, and H. Morikawa. A user posture inference scheme for resource limited microprocessors. In *Ubiquitous Computing Systems 2007 Demo*, 2007.

[12] K. Aidoo K. Van Laerhoven and S. Lowette. Real-time analysis of data from many sensors with neural networks. In *International Symposium on Wearable Computing 2001*, 2001.

[13] Jonathan Lester, Tanzeem Choudhury, and Gaetano Borriello. A practical approach to recognizing physical activities. In *PERVASIVE 2006*, pp. 1–16, 2006.

[14] Crossbow mica2 dot, 12 2007.

[15] 松下電工 焦電型 mp モーションセンサ napion, 12 2007.

[16] Sony japan, 2007.

[17] Masaya Kadota. 確率を用いた適応的なコンテクスト解釈系に関する研究, 2004.

[18] Kevin Korb and Anne Nicholson. *Bayesian Artificial Intelligence*. CRC Pr I Llc, 2003.

[19] 高野照夫, 高山守正. 日本医師会インターネット生涯教育講座 救急診療における処置と連携, 2004.

[20] Steve Mann. Smart clothing: The wearable computer and wearcam. *Personal Technologies*, March

1997. Volume 1, Issue 1.

[21] Jonny Farringdon, Andrew J. Moore, Nancy Tilbury, James Church, and Pieter D. Biemond. Wearable sensor badge and sensor jacket for context awareness. In *ISWC*, pp. 107–113, 1999.

[22] David Minnen, Thad Starner, Jamie A. Ward, Paul Lukowicz, and G. Troster. Recognizing and discovering human actions from on-body sensor data. In *ICME 2005*, pp. 1545–1548, 2005.

[23] Toshio Hori, Yoshifumi Nishida, Hiroshi Aizawa, Shin'ichi Murakami, and Hiroshi Mizoguchi. Distributed sensor network for a home for the aged. In *2004 IEEE International Conference on Systems, Man and Cybernetics*, pp. 1577–1582, 2004.

[24] University of karlsruhe teco smart its, 12 2007.

[25] Takuo Suganuma, Kazuhiro Yoshida, Yoshikazu Tokairin, Hideyuki Takahashi, Kenji Sugawara, and Norio Shiratori. A ubiquitous watch-over system based on environmental information and social knowledge. In *Ubiquitous Intelligence and Computing*, 2007.

# 付録 A

# ベイジアンネットワーク

本研究では利用者の行動や，環境の状況，そして最終的には身体異常状態を抽出するため，複数のセンサデータのパターンの解析を行う．本章では，本研究で利用する解析手法であり，状態解析の分野で広く用いられているベイジアンネットワークについて詳しく解説を行う．

## A.1　ベイジアンネットワーク

私たちは普段の生活の中で，何かの問題において判断をしたり，決断する場面がしばしばある．しかし，私たちはそれらの判断や決断に必要な情報すべてを知った上で，答えを出しているわけではない．そのような場合は，答えを出す上で必要な情報の中で，いくつかにおいては必ずといっていいほど不確実な要素を含んでいるといえる．例えば，研究室に差し入れのドーナツを買っていく場合でも，研究室に今日何人の人がいるかということはわからない．そのような場合，私たちは今日の天気や曜日，直近の発表の有無などの情報から，研究室に今日何人いるかを推測し，ドーナツを買う個数を決定する．

ベイジアンネットワークとは，このような不確実な要素を扱い，確率を用いて答えを推論することができる人工知能システムである．不確実性は一般に 3 つに分類することができ，それらすべてをベイジアンネットワークは扱い，私たちが実世界で判断や決断をするように，確率を用いて答えを導き出してくれる．その不確実性の分類を以下に示す．

1. **無知による不確実性**
   明確な答えはあるのだが，答えを判断するに当たって必要な知識の不足により生じる不確実性の分類である．例を挙げると，初対面の人の年齢である．初対面の人の年齢は，その人の生まれた年などの知識がないために判断できない．
2. **無作為による不確実性**
   無造作に答えが選ばれるため，知識を利用して判断することのできない不確実性の分類である．例えば，さいころを振ったときに出る目の数などが，この不確実性の分類に入る．さいころを転がしたとき，出目は無造作に選ばれるため，予測して言い当てることができない．
3. **あいまいさによる不確実性**
   答えの判断基準があいまいで，答えを出すことができないことによる不確実性の分類である．例えば，夕食がおいしかったか，そうではないか，ということは明確な基準がないため，答えを出すことができない．

### A.1.1 ベイジアンネットワークの構造

ベイジアンネットワークは，不確実性を伴う問題領域をグラフ構造として表現し，解くことができる．ベイジアンネットワークのグラフの要素は，問題領域内の個々の事象を表すノードと，それぞれの事象同士の関係性を表す有向リンクによって表現される．関係性の詳細情報は，条件付確率のテーブルによって定義されている．そのグラフの有名な例を図 A.1 に示す．このグラフは患者が肺がんであるかどうかという問題領域を表したものである．

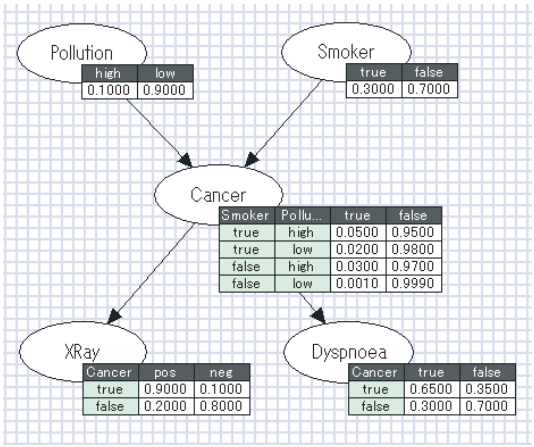


図 A.1　肺がんにおけるベイジアンネットワーク

### A.1.2 ノードと変数

ノードは問題領域内の個々の事象を表す．肺がんであるかどうかを扱う問題領域においては，先ず，肺がん (Cancer) であるかどうかが目的のノードとなり，肺がんに影響を及ぼす事象である公害 (Pollution)，喫煙 (Smoking)，そして，肺がんであることが影響を及ぼすレントゲン (XRay)，呼吸困難 (Dyspenoea) が周辺のノードとなる．さらにこれらの事象を表すノードはその状態を変数として保持する．変数として取りえる値は，boolean 値や，状態を表すラベル，また整数値などがある．肺がん問題領域における事象の変数を以下の表 A.2 に示す．

表 A.1　肺がん問題におけるノードと変数

| ノード名 | 変数名 |
|---|---|
| 公害 (Pollution) | {Low,High} |
| 喫煙 (Smoking) | {true,false} |
| 肺がん (Cancer) | {true,false} |
| 呼吸困難 (Dyspenoea) | {true,false} |
| レントゲン (XRay) | {neg,pos} |

公害は，その度合いを表す {low,high} で表され，喫煙，肺がん，呼吸困難は発生の有無を表す {true,false}

で表される．また，レントゲン結果は {neg,pos} で表現される．

### A.1.3 リンク構造

リンクはノード間のつながりを表す．ベイジアンネットワークはリンクでつながれたノードのネットワークである．リンクは方向性を持っており，矢印の根元にあるノードは原因を表し，矢印の指す先にあるノードが結果を表す．このノード同士の関係で，原因を表すノードは，親ノードと呼ばれ，結果を表すノードは子ノードと呼ばれる．肺がんの問題領域では，肺がんであるかどうかの診断に影響を及ぼす原因は公害と喫煙であるため，これら 2 を表すノードから，肺がんノード向かってリンクが張られる．また，肺がんはレントゲン結果と呼吸困難の発生に影響を及ぼすため，肺がんノードからレントゲン結果と呼吸困難を表すノードへ向かって矢印が伸びている．

### A.1.4 条件付確率テーブル

親ノードがある状態であったとき，子ノードがある状態を取る確立を条件付確率という．親ノードの取りえるそれぞれの状態における，子ノードの条件付確率を示した表を条件付確率テーブルという．図 A.1 の肺がん問題領域におけるレントゲン結果の条件付確率テーブルを以下の図 A.2 に示す．

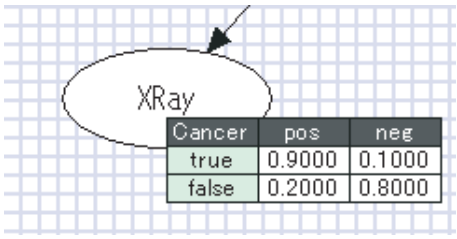


図 A.2　肺がん問題におけるレントゲン結果の条件付確率テーブル

この条件付確率テーブルは，肺がんであるときにレントゲンに映る確率 (pos) をである確率は 90% であり，映らない確率 (neg) は 10% であることを示している．また，肺がんでないときには，レントゲンに間違えて映る確率 (pos) は 20% であり，映らない確率 (neg) は 80% である．親ノードが 2 つ以上ある場合は，それらの親ノードの前通りの組み合わせに対して条件付確率テーブルが定義される．

## A.2 ベイジアンネットワークにおける推論

ベイジアンネットワークのモデルを用いることで，任意の事象の状態の発生確率を算出することができる．本項ではベイジアンネットワークの計算方法を解説する．

### A.2.1 推論パターンの分類

任意の事象の状態の発生確率を算出する推論のパターンには大きく分けて 2 種類存在する．一方は結果から原因を推論する診断的推論で，他方は原因から結果を推論する予測的推論である．

- **予測的推論**
  予測的推論は，ネットワーク構造の原因部分にあたるノードに対して証拠が与えられた状態で，結果部分に当たるノードである状態が発生する確率を算出するパターンである．肺がんの問題領域で，喫煙や公害といった原因が観測され，証拠として与えられた状態で，どれほどの確率でその患者は肺がんであるか，またその結果に伴い呼吸困難が起こったり，レントゲンに写ったりするかを推論する際は，このパターンの推論を行う．その推論の様子を図 A.3 に示す．
- **診断的推論**
  診断的推論は，ネットワーク構造の結果部分にあたるノードが証拠を与えられた状態で，原因部分のノードである状態が発生する確立を求める推論パターンである．肺がんの問題領域におけるレントゲン結果や呼吸困難という症状は結果である．そのため，レントゲン結果や呼吸困難が観測された状態で，患者が肺がんであり，さらに喫煙者であり，公害の多いところに住んでいる可能性を求める推論は診断的推論に当てはまる．その様子を図 A.4 に示す．

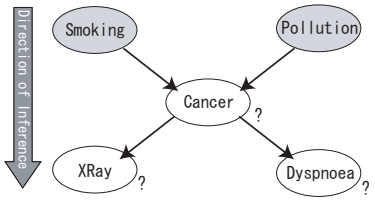


図 A.3　予測的推論の例

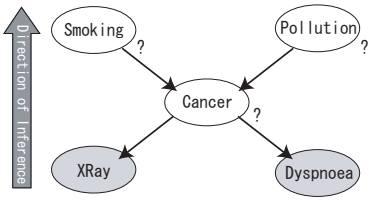


図 A.4　診断的推論の例

### A.2.2　結合確率の算出

これらの推論は，それぞれのノードにおける条件付確率を掛け合わせて行くことによって求められる．その例に図 A.1 の肺がん問題領域モデルにおいて，ある患者が喫煙者であり，公害レベルの高い地域に居住している場合を考える．その患者が肺がんである確率は，$P(Cancer = T \mid Smoker = T, Pollution = H) = 0.05$ であることが Cancer ノードの条件付確率テーブルから判断できる．さらに，同じように，肺がんである患者が呼吸困難になる確率は $P(Dyspnoea = T \mid Cancer = T) = 0.65$ であることがわかる．結果，公害レベルの高い地域に居住する喫煙者が肺がんになる確率は，$P(Cancer = T \mid Smoker = T, Pollution = H) \cdot P(Dyspnoea = T \mid Cancer = T)$ であり，$0.05 \cdot 0.65 = 0.325$ と求められる．Xn を 1 ノードとしてこれを数式に表すと，数式 A.1 以下のようになる．

$$P(X_0, X_1, .., X_n) = \prod_{i=0}^{n} P(X_i \mid Parents(X_i)) \tag{A.1}$$

### A.2.3　ベイズの定理

条件付確率を掛け合わせる上記の手法は，条件付確率が既知である必要があるため，予測的推論にのみ適応できる．しかし，診断的推論をする際，必要な条件付確率は定義されていない．ベイズの定理は，この診断的

推論をする際に必要な条件付確立をベイジアンネットワークにおいて既知の条件付確率から求める定理である．ベイズの定理を数式 A.2 に示す．

$$P(B \mid A) = \frac{P(A \mid B) \cdot P(B)}{P(A)} \tag{A.2}$$

肺がん問題領域において，ある患者が呼吸困難を起こした時に肺がんである確率は，既知である肺がんであるとき呼吸困難が起こる条件付確率と，肺がんである事前確率を掛け合わせた値を，呼吸困難が起こる事前確率で割ることで求められる．これを数式 A.3 に示す．ベイズの定理を利用した結果，ベイジアンネットワークにおいて診断的推論を行うことが可能となる．

$$P(Can = T \mid Dysp = T) = \frac{P(Dysp = T \mid Can = T) \cdot P(Can = T)}{P(Dysp = T)} \tag{A.3}$$

### A.2.4 推論アルゴリズムの分類

ベイジアンネットワークにおける推論アルゴリズムには 2 種類存在する．本項では，「厳密推論アルゴリズム」と「近似推論アルゴリズム」のそれぞれについて特徴を説明する．

- **厳密推論アルゴリズム**
  厳密推論は，既知の条件付確率と，ベイズの定理によって算出された条件付確立を計算式 A.1 に沿って厳密に計算し，結合確率を求める推論手法である．しかし，厳密推論は事象の数の増加，関係の複雑化に伴い計算量が大幅に増え，算出に長い時間を要する．
- **近似推論アルゴリズム**
  一方，近似推論は，BN のモデルに基づくシミュレーションを行うことにより，近似的な結合確率を求める手法である．そのため，複雑な BN においては近似推論は短い時間で結合確率の近似値の収束値を求めることができる．近似推論アルゴリズムには Logic Sampling や Likelihood Weighting などがあげられる．

## A.3 ベイジアンネットワークにおける学習

本項では，一般的な機械学習について触れ，ベイジアンネットワークにおける学習の種類と方法を解説する．

### A.3.1 機械学習

一定量の観測されたサンプルデータからシステム自身が何らかの規則や関係性，判断基準を見つけ出し，次第に最適な解法に近い解法を導き出していくことをを機械学習という．ベイジアンネットワークを用いる推論も，機械学習を行うことで，推論モデルや，条件付確率を学習し，推論の精度を次第に向上させることができる．

### A.3.2 学習の分類

ベイジアンネットワークにおける学習には,「グラフ構造の学習」と「条件付確率の学習」の 2 種類がある.以下でそれぞれを詳しく説明する.

- **グラフ構造の学習**
  グラフ構造の学習は，ある問題領域におけるベイジアンネットワークのノードで構成されるグラフ構造を抽出し，決定するための学習を指す．ベイジアンネットワークを利用した推論は，グラフ構造に従って行う確率を用いた推論であるため，あらかじめ定められた問題領域の関係性が必要不可欠である．人間がそれぞれの事象の関係性を的確に把握し，グラフ構造を定義することのできないような問題領域では，システムによるグラフ構造の学習により，正しい推論を行うにあたって必要な，ノード同士の関係を的確に示したふさわしいグラフ構造を決定することができる．
- **条件付確率の学習**
  ベイジアンネットワークのグラフ構造におけるそれぞれのノードの条件付確率を，サンプルデータから算出し，更新することで，推論精度を高めることができる．ある事象が起こった際のノードが取った状態の組み合わせ結果から，ある事象が起こった問題を確率的に学習できるため，ある問題領域内で，ある事象が発生する原因を人間が明確にわかっていない場合，条件付確率の学習を行うことで，その事象が起こる原因を確率的に学習することができる．

# 付録 B

# BayesEngine 抽象クラスの実装

```
1  package jp.ac.keio.sfc.ht.namatame.bayes.algorithm;
2  import java.util.ArrayList;
3
4  /**
5   * LogicSamplingを用いてベイズ推論を行うクラス。
6   * @author Naoya Namatame
7   * @version 1.0
8   */
9
10 public abstract class BayesEngine {
11
12     protected NodeArrayList nodeList;
13     protected int nodeCount;
14     protected int topHierarchyLevel;
15
16
17     /**
18      * 引数なしコンストラクタ。各メンバ変数の初期設定をする。
19      */
20     public BayesEngine(){
21         nodeCount = 0;
22         nodeList = new NodeArrayList();
23         topHierarchyLevel = 0;
24     }
25
26
27     /**
28      * ベイジアンネットワーク上のノードをセットアップする。
29      * @param filePath ノードの定義XMLファイルへのパス
30      */
31     public void setUpNodes(String filePath){
32
33         System.out.println("program started.");
34
35         //XMLをパースしてノードを作成、設定
36         XMLParser parser = new XMLParser(filePath);
37         parser.parse();
38         parser.setUpNodes(this);
39
40         //テーブルの条件部分をセットアップ
41         for(int i=0; i<nodeList.size(); i++){
42             Node node = (Node) nodeList.get(i);
43             node.setProbabilityTable();
44         }
45
46         //構造を解析
47         this.analyseModel();
48
49     }
50
51
```

```
52      /**
53       * ベイジアンネットワークの構造を解析し、ノードの計算順序を把握する。
54       */
55      public void analyseModel(){
56
57          ArrayList bottomList = findBottomNodes();
58
59          for(int i=0; i<bottomList.size(); i++){
60              Node node = (Node) bottomList.get(i);
61              explore(node);
62          }
63
64      }
65
66
67      /**
68       * 新しくベイジアンネットワーク上にノードを作成する。
69       * @param nodeName ノードの名前
70       * @return 新しく生成したノードのインスタンス
71       */
72      public Node createNode(String nodeName){
73
74          Node node = new Node(nodeCount, nodeName);
75          nodeList.add(node);
76          nodeCount++;
77
78          return node;
79      }
80
81
82      /**
83       * ベイジアンネットワーク上で子ノードを持たない最下位レベルのノードを探索する。
84       * @return 最下位ノードのインスタンスのリスト
85       */
86      public ArrayList findBottomNodes(){
87
88          //親ノードになっているノードを見つけてフラグを立てる
89          for(int i=0; i<nodeList.size(); i++){
90              Node node = (Node) nodeList.get(i);
91
92              for(int j=0; j<node.parentList.size(); j++){
93                  Node parent = (Node) node.parentList.get(j);
94
95                  if(parent.isParent == false){
96                      parent.setIsParent(true);
97                  }
98              }
99          }
100
101         //親ノードになっていない、一番下のノードのリストを作成する
102         ArrayList list = new ArrayList();
103
104         for(int i=0; i<nodeList.size(); i++){
105             Node node = (Node) nodeList.get(i);
106             if(!node.isParent){
107                 list.add(node);
108             }
109         }
110         return list;
111     }
112
113
114     /**
115      * 再帰的に呼び出され、ベイジアンネットワーク上の各ノードに計算順序をつける。
116      * @param child 子ノードのインスタンス
117      */
118     public void explore(Node child){
119
120         for(int i=0; i<child.parentList.size(); i++){
121             Node node = (Node) child.parentList.get(i);
```

```
122
123             //既に設定されているノードのレベルが今回のレベルより低い場合は更新
124             if(node.hierarchyLevel < child.hierarchyLevel+1){
125                 node.setHierarchyLevel(child.hierarchyLevel+1);
126             }
127
128             //現在のトップレベルが今回のレベルより低いとき
129             if(topHierarchyLevel < node.hierarchyLevel){
130                 topHierarchyLevel = node.hierarchyLevel;
131             }
132
133             //再起
134             explore(node);
135         }
136
137     }
138
139
140     /**
141      * ベイジアンネットワーク上のノードに証拠を与える。
142      * @param nodeName 証拠となるノードの名前
143      * @param state 証拠となる事象の名前
144      * @return 正しくセットできたらTRUEを返す。
145      */
146     public boolean setEvidence(String nodeName, String state){
147
148         boolean isSet = nodeList.getNodeByName(nodeName).setEvidence(state);
149         return isSet;
150
151     }
152
153
154     /**
155      * ベイジアンネットワーク上のノードの証拠をクリアする。
156      */
157     public void clearAllEvidence(){
158
159         for(int i=0; i<this.nodeList.size(); i++){
160             Node node = (Node) nodeList.get(i);
161             node.clearEvidence();
162         }
163
164     }
165
166     /*
167      * 指定されたサンプリング回数のサンプリング処理を行なう
168      */
169     public void infer(int samplingRate){
170
171     }
172
173 }
```

# 付録 C

# LogicSampling クラスの実装

```
1  package jp.ac.keio.sfc.ht.namatame.bayes.algorithm;
2  import java.util.ArrayList;
3
4  /**
5   * LogicSamplingを用いてベイズ推論を行うクラス。
6   * @author Naoya Namatame
7   * @version 1.0
8   */
9
10 public class LogicSampling extends BayesEngine{
11
12     /**
13      * 引数なしコンストラクタ。スーパークラスのコンストラクタを呼び出す。
14      */
15     public LogicSampling(){
16         super();
17     }
18
19
20     /**
21      * LogicSamplingでベイズ推論の計算を開始する。
22      * @param samplingRate サンプル回数
23      */
24     public void infer(int samplingRate){
25
26         int sampleCount = 0;
27
28         //構造を解析
29         this.analyseModel();
30
31         //結果のホルダーをnewして初期化
32         for(int i=0; i<this.nodeList.size(); i++){
33             Node node = (Node) nodeList.get(i);
34             node.historicalStates = new String[samplingRate];
35         }
36
37         while(sampleCount < samplingRate){
38
39             //System.out.println("Count" + sampleCount + "==============");
40
41             LOOP1: for(int i=topHierarchyLevel; i>=0; i--){
42
43                 for(int j=0; j<nodeList.size(); j++){
44                     Node node = (Node) nodeList.get(j);
45
46                     if(node.hierarchyLevel == i){
47
48                         if(node.probabilityTable.conditionList.size() == 1){
49                             boolean evd = node.update(
50                                 (ArrayList) node.probabilityTable.probabilityList.get(0));
51                             node.setHistoricalStates(sampleCount);
```

```
52                              if(!evd){
53                                  //System.out.println("break");
54                                  sampleCount--;
55                                  break LOOP1;
56                              }
57                          }
58
59                          else{
60                              boolean evd = node.update(
61                                  (ArrayList) node.probabilityTable.probabilityList.get(
62                                          node.findCurrentConditions()));
63                              node.setHistoricalStates(sampleCount);
64                              if(!evd){
65                                  //System.out.println("break");
66                                  sampleCount--;
67                                  break LOOP1;
68                              }
69                          }
70                      }
71
72                  }
73              }
74
75              sampleCount++;
76          }
77
78      }
79
80
81      /**
82       * あるノードにおけるある事象の推論結果の確率を取得する。
83       * @param nodeName 確率が知りたいノードの名前
84       * @param state 確率が知りたい事象
85       */
86      public float getLikelihood(String nodeName, String state){
87          float likelihood = (float) 0.0;
88
89          likelihood = nodeList.getNodeByName(nodeName).getLikelihood(state);
90
91          return likelihood;
92
93      }
94  }
```

# 付録 D

# LikelihoodWeighting クラスの実装

```
1  package jp.ac.keio.sfc.ht.namatame.bayes.algorithm;
2  import java.util.ArrayList;
3
4  /**
5   * LikelihoodWeightingを用いてベイズ推論を行うクラス。
6   * @author Naoya Namatame
7   * @version 1.0
8   */
9
10
11 public class LikelihoodWeighting extends BayesEngine{
12
13
14     private float score[];
15
16     /**
17      * 引数なしコンストラクタ。スーパークラスのコンストラクタを呼び出す。
18      */
19     public LikelihoodWeighting(){
20         super();
21     }
22
23
24     /**
25      * LikelihoodWeightingでベイズ推論の計算を開始する。
26      * @param samplingRate サンプル回数
27      */
28     public void infer(int samplingRate){
29
30         int sampleCount = 0;
31
32         score = new float[samplingRate];
33
34         //結果のホルダーをnewして初期化
35         for(int i=0; i<this.nodeList.size(); i++){
36             Node node = (Node) nodeList.get(i);
37             node.historicalStates = new String[samplingRate];
38         }
39
40
41         while(sampleCount < samplingRate){
42
43             //System.out.println("Count" + sampleCount + "==============");
44
45             float currentScore = (float) 1.0;
46
47             for(int i=topHierarchyLevel; i>=0; i--){
48
49                 for(int j=0; j<nodeList.size(); j++){
50                     Node node = (Node) nodeList.get(j);
51
```

```
52                      if(node.hierarchyLevel == i){
53
54                          //条件部分がないノードであれば
55                          if(node.probabilityTable.conditionList.size() == 1){
56
57                              //evidenceが設定されていれば
58                              if(node.evidenceState != null){
59                                  node.setEvidenceAsCurrentState();
60
61                                  //スコアを求める
62                                  ArrayList table =
63                                      (ArrayList) node.probabilityTable.probabilityList.get(0);
64                                  for(int k=0; k<node.probabilityTable.stateList.size(); k++){
65                                      if(node.evidenceState.equals(
66                                          node.probabilityTable.stateList.get(k))){
67                                              //System.out.print(table.get(k) + " ");
68                                              Float caseScore = (Float) table.get(k);
69                                              currentScore = currentScore * (float)caseScore;
70                                      }
71                                  }
72
73                              }
74
75                              //evidenceが設定されていなければ
76                              else{
77                                  node.update(
78                                      (ArrayList) node.probabilityTable.probabilityList.get(0));
79                              }
80
81                              node.setHistoricalStates(sampleCount);
82                          }
83
84                          //条件部分があるノードであれば
85                          else{
86
87                              //evidenceが設定されていれば
88                              if(node.evidenceState != null){
89                                  node.setEvidenceAsCurrentState();
90
91                                  //スコアを求める
92                                  ArrayList table =
93                                      (ArrayList) node.probabilityTable.probabilityList.get(
94                                      node.findCurrentConditions());
95
96                                  for(int k=0; k<node.probabilityTable.stateList.size(); k++){
97                                      if(node.evidenceState.equals(
98                                          node.probabilityTable.stateList.get(k))){
99                                               //System.out.print(table.get(k) + " ");
100                                              Float caseScore = (Float) table.get(k);
101                                              currentScore = currentScore * (float)caseScore;
102                                     }
103                                 }
104                             }
105
106                             else{
107                                 node.update(
108                                     (ArrayList) node.probabilityTable.probabilityList.get(
109                                     node.findCurrentConditions()));
110                             }
111
112                             node.setHistoricalStates(sampleCount);
113                         }
114                     }
115
116                 }
117             }
118
119             score[sampleCount] = currentScore;
120             //System.out.println(sampleCount + ":" + score[sampleCount]);
121             sampleCount++;
```

```
122             }
123
124         }
125
126
127         /**
128          * あるノードにおけるある事象の推論結果の確率を取得する。
129          * @param nodeName 確率が知りたいノードの名前
130          * @param state 確率が知りたい事象
131          */
132         public float getLikelihood(String nodeName, String state){
133             float likelihood = (float) 0.0;
134             Node node = nodeList.getNodeByName(nodeName);
135             ArrayList states = node.probabilityTable.stateList;
136             float targetScore[] = new float[states.size()];
137             float totalScore = 0;
138
139             //初期化
140             for(int i=0; i<states.size(); i++){
141                 targetScore[i] = 0;
142             }
143
144
145             //ステートの数だけ
146             for(int i=0; i<states.size(); i++){
147                 for(int j=0; j<node.historicalStates.length; j++){
148
149                     //その条件のケースがあれば
150                     if(node.historicalStates[j].equals(states.get(i))){
151                         targetScore[i] += score[j];
152                     }
153                 }
154
155                 totalScore += targetScore[i];
156             }
157
158             for(int i=0; i<states.size(); i++){
159                 if(states.get(i).equals(state)){
160                     likelihood = targetScore[i]/totalScore;
161                 }
162             }
163
164             return likelihood;
165         }
166
167     }
```