

## 異常状態の自動定義による状況監視アプリケーションの支援

出内 将夫<sup>1</sup> 村上 朝一<sup>2</sup> 田丸 修平<sup>2</sup> 高汐 一紀<sup>2</sup> 徳田 英幸<sup>1,2</sup>

<sup>1</sup> 慶應義塾大学 環境情報学部 <sup>2</sup> 慶應義塾大学大学院 政策・メディア研究科

本稿では、状況監視アプリケーションを支援するミドルウェア、SARADA を提案する。近年、ネットワークに接続可能なセンサや機器を用いて環境の異常を検知し、対処を行う状況監視アプリケーションが開発されている。状況監視アプリケーションが異常を判定するためには、環境に応じた判定基準の設定が必要であり、アプリケーション開発者やユーザにとって大きな負担となる。SARADA はセンサや機器から取得できるデータの履歴から異常の判定基準を動的に定義することで判断基準設定の負担を軽減する。評価として既存システムと SARADA との機能比較、判定基準の設定に必要な時間の計測を行い、既存のシステムに対して導入時の負担に関する優位性を示した。

## SARADA: Support for Application of Recognizing environment by Auto Detection of Anomalies

Masao IDEUCHI,<sup>1</sup> Tomokazu MURAKAMI,<sup>2</sup> Syuhei TAMARU,<sup>2</sup>  
Kazunori TAKASHIO<sup>2</sup> and Hideyuki TOKUDA<sup>1,2</sup>

<sup>1</sup> Faculty of Environmental Information, Keio University

<sup>2</sup> Graduate School of Media and Governance, Keio University

This paper introduces "SARADA", the middleware for situated applications. The applications that detect and handle anomalies by networked sensors and appliances are being developed. To recognize anomalies, these applications need to define anomalies independent of environmental differences. Making such a definition is a burden for applications developer or user. SARADA reduces this burden by defining anomalies from history sensor and application data. Thus SARADA supports the development and installation of the application of recognizing environment. For evaluation, we compare SARADA with existing systems and measure the time needed for creating a definition, and show advantages to the existing systems.

### 1 はじめに

近年、ネットワーク接続性を持つセンサや機器が増加している。それらのセンサや機器を用いて環境をモニタリングし、異常があった場合、異常への対応やユーザへの通知を行う状況監視アプリケーションが開発されている(図1)。状況監視アプリケーションの例としては、カメラや人感センサを用いて侵入者を検知し通報する防犯警備システムや、位置センサや機器の使用状態から、居住者の生活状態を割り出し、介護に役立てる遠隔介護システムなどが挙げられる。

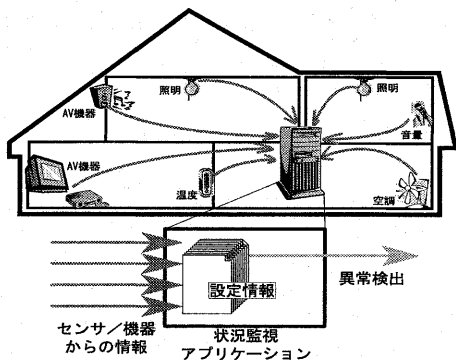


図1: 状況監視アプリケーションの概念図

ユビキタスコンピューティング環境[1]では、ネットワークに接続された様々なセンサや機器が、様々なアプリケーションから利用可能になる。また、その利用形態もアプリケーションがセンサや機器に対して個々に対応するのではなく、センサから得ら

れる値を抽象化する手法[2]や、情報家電におけるJavaプラットフォームの搭載[3]などによって、多様なセンサや機器を統一的に扱えるようになると考えられる。

このような環境下で、状況監視アプリケーションはネットワークを介して、様々な既存のセンサや機器を活用できる。そのため、状況監視アプリケーション導入に必要な、センサや機器の新たな設置にかかる費用が削減できる。また、多くの機器同士が連携することで様々な異常を検出できる。例えば、ガスの使用量の計器とガス使用機器の状態を監視することによるガス漏れ検知や、情報家電の動作パターンを監視することによる住人の危機状態の検知、緊急通報などが可能になる。

次節では、本研究の問題意識である、ユビキタスコンピューティング環境における状況監視アプリケーション実現のための課題を述べ、本研究の目的、機能要件について述べる。その後3節でアプローチ、4節で本システムで用いるアルゴリズムについて述べる。5節では、本システムの設計、6節で実装について言及する。7節で評価を行い、8節でまとめる。

### 2 問題意識と目的

本節では、まず研究の問題意識を述べ、目的を述べる。

#### 2.1 問題意識

状況監視アプリケーションが、センサや機器によって取得された情報から異常を判定するためには、異常の判定基準が必要である。しかし、異常状態を判定するための基準の設定は、センサが設置されている場所や、機器を使用している状況や使用頻度によって影響を受けるため、環境に応じた設定が必要となる。既存の状況監視アプリケーションではセンサや機器の種類が固定で、判断基準も統

一されているため、センサや機器を新たに購入し、適切な設置位置を考慮しなければならない。また、既存のセンサや機器を用いる場合に、手作業で環境に応じた設定を行うことは、増加し続けているセンサや機器の数から考慮すると、アプリケーション開発者やユーザにとって大きな負担となる。

## 2.2 目的

本研究の目的は、異常の検知に用いる判定基準を、環境に応じて自動生成することにより、状況監視アプリケーションの開発、導入を支援することである。これによって、既存の状況監視アプリケーションに必要な、新たなセンサや機器の購入や設置場所の決定などの制約が無くなる。また、既存のセンサや機器を用いる場合にも、異常を判定する基準の設定の手間を省くことで、アプリケーション開発者もしくはユーザにかかる負担を軽減する。

## 2.3 機能要件

目的を達成するために必要な機能要件を述べる。機能要件として、環境適応性、自動生成機能、可読性の3つを挙げる。

### ● 環境適応性

環境に応じた異常状態を判定する基準の設定が可能であること。これにより、センサが設置されている場所や機器の使用している状況、使用頻度などを考慮した異常の判定が可能となる。

### ● 自動生成機能

異常状態の判定に用いる基準の設定が自動的に生成されること。これにより、ユーザやアプリケーション開発者が行う手作業による基準の設定の手間を省き、状況監視アプリケーションの導入コストを軽減する。

### ● 可読性

アプリケーション開発者もしくはユーザにとって、自動生成された設定が理解可能である、あるいは理解可能な形に翻訳できること。これにより、自動的に生成された異常状態を判定する基準の設定をアプリケーション開発者やユーザが確認したり編集したりできる。

## 3 アプローチ

本研究では、センサや機器によって取得された情報の履歴から、自動的に異常の判定基準を決定し、アプリケーションに対して異常を通知するミドルウェア、SARADA(Supporting system for Applications of Recognizing environment by Auto Detection of Anomalies)を構築する。

以下に、機能要件を満たすために本研究が用いる手法について説明する。まず、環境適応性を実現するための、履歴情報を用いた定常状態の定義について述べ、次に、定常状態の定義を行うアルゴリズムについて、自動生成機能と解釈性の観点から比較検討を行う。

### 3.1 履歴情報を用いた定常状態の定義

まず、履歴情報を利用する根拠を述べ、次に利用方法について述べる。ここで、センサや機器によって取得され、環境の状態を表現する情報を、**環境属性**と定義する。環境属性の例には、温度や湿度、音量、光量、機器の使用状態などが挙げられる。

ある環境において、異常状態の判定に用いる基準の設定を環境に応じて行うためには、その環境の状態を表す情報を利用するのが適切である。センサや機器を用いて得られる環境属性がとる値を蓄積した履歴情報は、センサが設置された位置や、機器の普段の使用状態を反映している。この履歴情報を異常状態を判定する基準の決定に用いることで、環境に応じた設定が可能になる。

センサや機器によって得られる環境属性がとる値を履歴として蓄積し、普段センサが示している

値や機器の状態からセンサや機器の定常状態を定義する。センサや機器から得られる環境属性がとる値が、この定常状態から外れた場合に、異常状態と判定される。

### 3.2 定常状態の定義に用いるアルゴリズム

本研究では、履歴を用いた決定木学習アルゴリズムを利用することにより、自動生成機能と可読性を実現する。

ある環境の状態が与えられた場合、それが定常か異常かの判定は、分類問題[4]と捉えることができる。そこで、本研究ではデータマイニングや機械学習で用いられている手法に注目し、機能要件に挙げた自動生成機能と解釈性を満たすアルゴリズムを応用する。

応用を検討したアルゴリズムは、数量化法アプローチ[5]、ベイジアンネットワーク[6]、ニューラルネットワーク[7]、決定木[8]の4つである。それらを、履歴からの自動的な定常状態の定義、人間への可読性について比較し、その結果を表1に示す。

### ● 数量化アプローチ

ある1つの環境属性が他の環境属性とどの程度相関があるかを、数式化することで表現する。この数式は常にデータ全体の規則性を表すため、局所的な規則性に対応できない。局所的な規則性とは、例えば人感センサが部屋内にあった場合、部屋内に人がいるときの規則性、またはさらに条件が加わって、部屋内に人がいて、かつ照明が消えているときの規則性などの、条件に制約を与えた場合の規則性のことである。

また、全ての属性を数値として扱うため、属性が数値で表せない離散値を取る場合、属性が取る値と同じ数の変数を用いることになり、変数の数は膨大になる。さらに定常状態の定義を、それぞれの変数に係数を持つ項の総和を表す数式で表現する。各項の係数が相関の正負を表し、2つの属性間の関連を捉えることはできる。しかし、全体との関連を把握するためには数式全体を理解する必要があり、人間が解釈するのは困難といえる。

### ● ベイジアンネットワーク

複数の属性間の因果関係を条件付き確率で表したグラフ構造を構築する。この方法は、履歴データから各属性同士の因果関係を記述できるが、初期のグラフ状態を設定する必要がある。ベイジアンネットワークのアルゴリズム単独では、この初期設定については定義されていないため、初期のグラフ状態の設定が必要となる。初期のグラフ状態の設定に学習アルゴリズムを用いれば自動的な定常状態の定義は可能であり、様々なアルゴリズムが検討されている。

可読性については、グラフ構造における連結が因果関係の有無を表すため、属性同士の因果関係は直感的に理解可能である。しかし、条件付き確率を表す場合、因果関係が数値の表を用いて表現され、定常状態か異常状態かの判定は複数の数値を考慮して判断しなければならず、人間が解釈するには問題がある。

### ● ニューラルネットワーク

入力と出力の属性間の写像関係を表現するグラフ構造を構築できる。このグラフ構造は、初期設定に関連のない属性を連結しても、学習が進むにつれて、関連の深さによって連結の重みが調整されるため、自動的な構築が可能である。しかし、自動的に定常状態を定義するためには、入力と出力のセットを与える必要があり、履歴データに加えて、それが異常か定常かを表すデータも必要である。履歴には定常状態のみが含まれるとは限らないので、自動的な定常状態の定

表 1: 定常状態の定義に用いる手法とその比較

| 用いる手法       | 自動的な定常状態の定義 | 可読性 |
|-------------|-------------|-----|
| 数量化法アプローチ   | △           | ×   |
| ベイジアンネットワーク | ○           | △   |
| ニューラルネットワーク | △           | ×   |
| 決定木         | ○           | ○   |

義は難しい。

また、学習後のネットワークそのものが定常状態を表しているが、ネットワーク内部はブラックボックスともいえる複雑な構造となるため、可読性は認められない。

#### ● 決定木

複数の属性間の関係を木構造で表現する。この方法は、条件付き確率を考慮した履歴データの分類を自動的に行い、履歴データ以外の情報や初期設定も必要としない。

また、この木構造はトップダウンに見ていくと、フローチャートと類似しており、人間が理解しやすい形となっている。

以上の4つのアルゴリズムの比較から、本研究では、決定木で用いられている手法を応用した。

### 4 異常状態判定アルゴリズム

本節では、木構造を用いた異常状態の判定手法について述べる。まず、木構造による定常状態の表現手法について解説し、次に定常状態と異常状態の判定方法を述べ、最後に動的な木の作成方法について述べる。

#### 4.1 木構造による定常状態の表現

本研究では、定常状態を木構造で表現する。まず、決定木学習における木構造について説明し、次に、本研究での応用について述べる。

決定木は、木構造で分類問題を表現する。木構造のうち、終端となる点は“葉”であり、それ以外の点は“節”である。節と節、あるいは節と葉を結ぶ辺を“枝”と呼ぶ。節には、分類を行う質問文が配置され、辺には、節における質問文の答えが配置される。葉には、分類の終端となるクラスが配置される。クラスとは、データ全体の分類に用いた属性がとる値である。決定木の例を図2に示す。

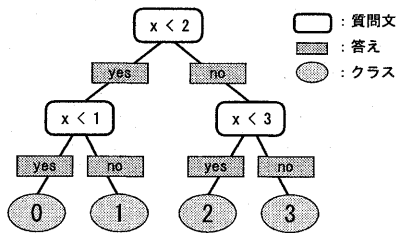


図 2: 決定木の例: 1~4 の数当て問題

本研究では、決定木を構築するために、まずクラスとして環境属性の一つを選び、節にはその他の環境属性、枝には分岐元の節が表す環境属性がとる値とした。そして、全ての属性を順にクラスとした木の構築を行い、現在の状態と比較し、定常状態に定義されていない属性があれば異常と判定する。図3に定常状態を表す木の例を示し、異常を判定する過程を説明する。

この例では、“扇風機の電源状態”という環境属性の値が ON か OFF かをクラスとすることで、環境の定常状態を定義した木を表している。また、定常状態の定義に用いられている環境属性は、“部屋の温度”、“部屋内の人の有無”、“人が部屋内にい

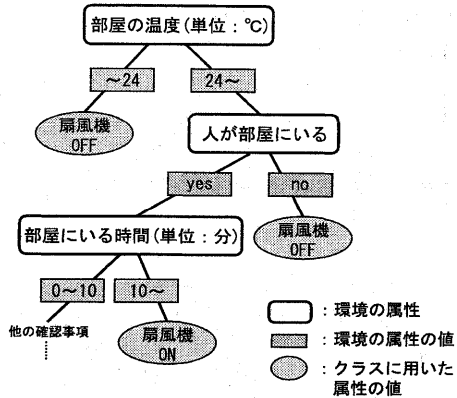


図 3: 扇風機の電源の状態を異常判定に用いた木

る時間”の3つであり、それらの属性が表す節から下に伸びる枝によって、属性の値が表される。

この図の読み方は、上から順にそれぞれの節を現在の状態に対する質問文と見立てて、その答えを表す枝へと進むことである。この作業をクラスに辿り着くまで繰り返す。この例だと、まず現在の環境属性の中から、最初の節である“部屋の温度”の属性に注目し、24℃未満だった場合は左の枝へ、24℃以上だった場合は右の枝へと進む。24℃未満だった場合はクラスへと到達するので、現在の環境属性の中から“扇風機の電源状態”を確認する。24℃以上だった場合は、次の節の属性に注目し、属性の値による条件分岐を繰り返していく。

#### 4.2 木構造を用いた定常/異常の判定

まず、定常状態と判定する場合について述べる。木を辿った結果、クラスに辿り着き、そのクラスが現在の状態に反していなければ、定常状態である。例えば現在の状態が、部屋の温度が23℃で、扇風機の電源がOFFである場合、4.1項の例に用いた木では、部屋の温度の判定だけでクラスに至るため、他の属性に関わらず、定常状態と判定される。また、分類が進み、十分な量のデータが無くなった場合、ある節以降は未定義とすることがあるが、未定義に至った場合も、定常状態とする。

次に、異常と判定される場合について述べる。異常と判定される要因は、大きく分けて2つ存在する。1つ目は、木を最後まで辿っていった結果、現在の状態と定常状態に定義されている結果が反した場合である。2つ目は、4.3項で詳しく述べる枝刈りや、前例がないなどの理由で、木を辿っている途中で枝が見つからなくなった場合である。

#### 4.3 木の動的作成方法

SARADA は決定木構築アルゴリズム [9] を利用し、動的に定常状態を表現する木を作成する。作成の過程を以下に説明する。

##### ● 数値データの離散化

扱う属性が数値データであった場合、枝の生成方法を統一し、木の作成を効率良く行うため、あらかじめ閾値を決定して、離散的な値として扱う。閾値を一定間隔で区切ってしまうと、データの分割が過度に行われてしまう可能性があるため、SARADA では、図4のように、データの平均と標準偏差を用いてデータを分割し、データの分割数が多くなり過ぎることを防いだ。

##### ● 節の決定と枝の生成

任意の一つの属性をクラスとし、履歴データを効率良く分類できる他の属性を探し出す。一番効率良く分類できる属性が節となり、属性がとる値によって枝を生成し分類を行う。分類の

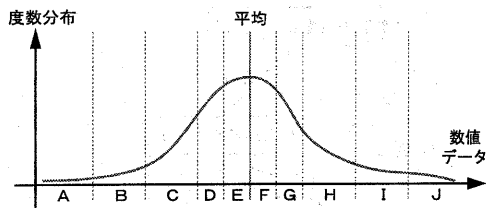


図 4: 数値データの離散化の例

効率を定める基準には、決定木でも用いられている情報利得比を用いた。情報利得比を用いることにより、ある属性で分割することによってどの程度分類が進んだか、を表す尺度に対して、その属性が持つ分割数による補正がかかる。これにより、分類の効率が良い属性であり、かつ過度の分割を行わない属性が選ばれることで、枝の数が少ない、簡潔な木が作成可能となる。

上記の方法で生成されたそれぞれの枝に対して、該当する履歴データを振り分け、分類する。それぞれの枝で分類された履歴データをもとに上記と同様の作業を行い、次の節を決定し、枝を生成する。これを再帰的に行うことで、木を作成する。

#### ● 枝刈り／未定義の枝

枝を生成する際、データ量が少なく、誤差の範囲に収まる枝に対しては、枝刈りを行い、過度の分類を防ぐ。この作業を行うことにより、データに含まれる誤差や例外値の枝は排除される。

また、決定木本来のアルゴリズムでは、未定義の枝は存在しないが、分岐した枝全体のデータ量自体が少ない場合、定常状態とは言い難いため、その枝は未定義とした。

## 5 設計

本節では、まず SARADA が想定する環境を述べる。次にシステムの構成と動作について述べる。

### 5.1 想定環境

SARADA が想定する環境は、家やビルなどの建物内に以下の3つの要素が設置され、それらがネットワークを介して通信できる環境である。

#### ● センサ

位置情報、温度、湿度や音量など、環境属性がとる値を取得する。

#### ● 機器

照明、AV 機器やエアコンなど、電源の ON、OFF、その他の動作状況を取得する。

#### ● サーバ

PC やワークステーションなど、SARADA が動作する端末である。部屋もしくは建物内に1つ存在する。

SARADA が動作中に新たなセンサや機器が設置されることも考えられるため、センサや機器の数の動的な変化を想定する。

### 5.2 システム構成

まず、構成図を図5に示し、システム全体の動きを説明する。

SARADA の動作は大きく2つに分けられる。履歴を蓄え、木を作成する動作と、作成された木をもとに異常状態を判定する動作である。

前者は、データ取得部、履歴管理部と木作成部が行う。センサや機器からデータを取得する度に各部が動作し、木を更新する。

後者は、判定部と通知部が行う。センサや機器からデータを取得し、環境属性がとる値が変化する度に異常状態の有無を木をもとに判定し、異常があればアプリケーションに通知する。

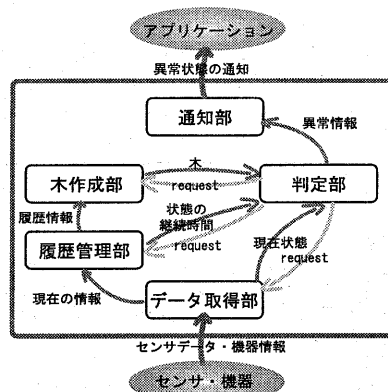


図 5: システム構成図

異常状態の判定を開始する時間は、システムが起動してから、木の作成にかかる時間やその木の信頼性を評価し、動的に決定できることを目指している。

### 5.3 各部の動作

SARADA を構成する5つのモジュールについて、それぞれの動きを説明する。

#### ● データ取得部

センサや機器からデータを取得し、履歴管理部に渡す。取得したデータを数値データ、非数値データに分け、それぞれに適した処理を行う。

#### ● 履歴管理部

データ取得部から受け取ったデータを履歴として蓄積する。数値データの場合は統計的な手法を用いて離散化する。ここで、データがある値を取り続けている時間を計算し、それぞれ一つのデータとして扱う。これにより、一時的な値だけでなく、値の変化や継続時間など、時間を考慮した定常状態の定義が可能になる。木作成部に渡す情報はデータそのものではなく、それぞれのデータの種類の応じた規則に従って、分類を行った離散値をとるデータを渡す。

#### ● 木作成部

履歴管理部から渡された分類済みのデータを用いて、定常状態を表す木を作成する。

#### ● 判定部

木作成部が作成した定常状態を表す木と、現在の状態や現在の状態が継続している時間を用いて、異常状態の判定を行う。異常と判定された場合、通知部に異常情報を渡す。

#### ● 通知部

アプリケーションに対して、異常を通知する。通知する内容には、異常と判定された理由となる木の情報や統計情報を含める。

## 6 実装

本節では、実装について述べる。まず、実装の概要と実装環境について述べ、次に実装において工夫した点について述べる。

### 6.1 実装の概要

SARADA のプロトタイプを実装した。開発言語には、モジュールのプラットフォーム独立性を確保するために、Java 言語を用いた。

今回実装したモジュールは、データ取得部、履歴管理部、木作成部、判定部の一部である。履歴から異常の判定基準を自動生成し、判定基準から異常を検知する機能を実現した。

表 2: 実現手法による状況監視アプリケーションの比較

| 比較項目     | SARADA | 既存の手法 | 手動設定 |
|----------|--------|-------|------|
| 環境適応性    | ○      | △     | ○    |
| 判定基準の設定  | ○      | ×     | ○    |
| 設置場所の工夫  | ○      | ○     | ○    |
| 導入コスト    | ○      | △     | △    |
| 設備投資の必要性 | ○      | ×     | ○    |
| 設定の手間    | ○      | ○     | ×    |
| 可読性      | ○      | ○     | ○    |

## 6.2 取得データの抽象化

データ取得部と履歴管理部が、取得データを抽象化することにより、木作成部では、データ形式の相違を透過的に扱える。

まず、連続値と離散値の相違を透過的に扱う機構について説明する。データ取得部ではセンサや機器を扱うモジュールとは独立した、データを表現する Value クラスが定義した。Value クラスは木作成部が必要とする、離散化したデータを返すメソッドを持つ。次に、履歴管理部はデータ取得部から渡されるデータの種類ごとに履歴を管理する ValueManager クラスが定義した。ValueManager クラスは、管理するそれぞれの Value に対して、データを離散化して格納するメソッドを持つ。

次に、実際にセンサや機器のモジュールからデータを取得する過程について説明する。センサや機器を扱うモジュールは、取得するデータが連続値か離散値かにより、Value クラスを継承した ContiguousValue クラスと DiscreteValue クラスのどちらかを生成する。これを、ValueManager クラスを継承した ContiguousValueManager クラスと DiscreteValueManager クラスのどちらかに格納し、離散化を行う。これにより、木作成部では離散化したデータのみを扱うことが可能である。

今後もし、連続値や離散値に関わり無く、特殊な処理が必要なデータを扱うことになった場合でも、Value クラスと ValueManager クラスを継承して離散化を行うことで、木作成部には一切の変更を加えずにシステムを再構築することが可能である。

## 6.3 センサや機器の抽象化

データ取得部でセンサや機器を扱うモジュールにおいて、インターフェースを定義した。これにより、Java で書かれた既存のモジュールがこのインターフェースを実装することで SARADA に簡単に組み込むことができる。また、そのセンサや機器が連続値と離散値のどちらを生成するかを指定するだけで、それぞれに合った Value クラスが生成可能となる。

## 7 評価

SARADA の定性的評価と実験で得られたデータを用いた定量的評価を行う。まず、状況監視アプリケーションを他の手法で実現した場合との機能比較を行う。次に、今回測定を行った実験環境を説明し、その結果得られた定常状態の定義について考察する。

### 7.1 機能比較

状況監視アプリケーションの性能を、SARADA、既存の手法、手動で設定を行う場合について比較する。既存の手法とは、利用するハードウェアの種類を制限し、設置場所を工夫することで、状況監視アプリケーションを実現することを指す。比較の結果を表 2 に示す。

まず、環境適応性について比較する。SARADA は、センサや機器からの情報の履歴を用いて環境に応じた判定基準を設定することが可能である。また、センサや機器の設置場所を工夫することも可

能である。既存の手法では、センサの種類に制限を設けたり、異常の判定基準が予め設定されたりしている。そのため、センサや機器の設置場所を工夫することしかできず、環境適応性を十分に満たしているとはいえない。手動で判定基準を設定する手法では、人間が判断することで、環境に応じた判定基準の設定や設置場所の工夫が可能である。

次に、導入コストについて比較する。導入コストとは、状況監視アプリケーション導入の際に必要な設備投資や設定の手間を指す。SARADA では、環境に既存のセンサや機器を利用できるため、新たなセンサや機器の購入は必要としない。また、システムが自動的に判定基準を設定するため、状況監視アプリケーション導入の際に手間がかからない。既存の手法では、センサや機器の設置が必要となる。判定基準の設定は、アプリケーション内部に定義されているため、必要ない。手動で判定基準を設定する手法では、既存のセンサや機器を利用することで、新たにセンサや機器を購入する必要は無い。しかし、判定基準の設定を行わなければならないので、設定の手間がかかる。

さらに、可読性について比較する。既存の手法や、手動での設定では、設定を行っているのは人間であるため、可読性を満たしている。SARADA では、決定木を応用した手法を取ることで、人間にとって理解可能な記述が可能である。

以上から、SARADA はユビキタスコンピューティング環境での状況監視アプリケーションにおいて、ユーザやアプリケーション開発者にかかる負担を軽減したといえる。

### 7.2 定常状態の定義

まず、実験環境の説明と SARADA が定常状態を定義するのに必要となった時間の計測結果を示し、定常状態の定義が収束することを示す。次に、実際に定義された定常状態を表現する木を示し、結果の考察を行う。

SARADA の動作実験を慶應義塾大学徳田研究室の実験施設である、SSLab[10]で行った。SSLab には多様なセンサや機器が設置されており、多種の環境属性を取得できる。また、SSLab では日常的に会議や実験などの活動が行われている(図 6)。そのため、SSLab は SARADA の測定を行う環境として適切である。

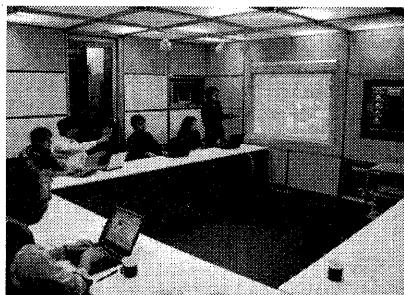


図 6: ミーティング中の SSLab の風景

使用したセンサや機器、扱った環境属性を表 3 に示す。

図 7 は、SARADA が木の作成を始めてから、定義した定常状態を表す木に含まれる枝の本数と、それに必要とした時間の関係を表すグラフである。

木の作成が始まってから 300 分程度で枝の本数は 154 本となり、その後は、2 日後に計測を終えるまで枝の数に変化は見られなかった。また、木を作成し始めたのは、履歴データが蓄積されてから 300 分ほど経過してからである。このグラフを、デー

表 3: 使用したセンサや機器と環境属性

| センサや機器    | 通信方法    | 環境属性              |
|-----------|---------|-------------------|
| DA100     | RS-232C | 室内の温度             |
| WebCamera | USB     | 明るさの変化            |
| RF-Code   | ネットワーク  | 部屋内に存在する人の ID セット |
| 部屋の照明     | ネットワーク  | 照明の状態             |

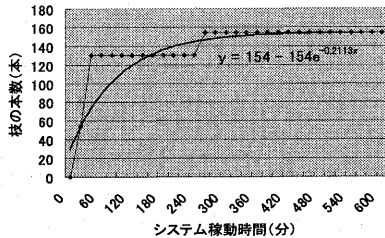


図 7: 定常状態を表現する木が持つ枝の本数の推移

タの数を独立変数, 枝の数を従属変数とした関数と考えと,

$$y = 154 - 154e^{-0.2133x}$$

で近似可能である。初項である 154 は今回定義された最終的な枝の本数を指し, 2 項目はデータ数が多くなるにつれて 0 に近づくため, 定常状態が収束することが分かる。

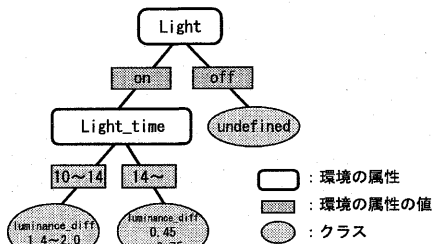


図 8: SARADA によって定義された木

収束後の木の一つを図 8 に示す。この木は, カメラによって得られる明るさの差分をクラスとして作成した木である。この木は, ライトが ON になってから 10~14 秒後はカメラによって得られる明るさの差分の値が 1.4~2.0 であり, 14 秒以降ではその値が 0.45~0.75 であれば定常状態であることを示している。

このカメラが設置されていた場所は, ライトの光が届く場所を写していた。そのため, ライトが ON になることによって, 明るさの変化に規則性が見られたことで, ライトが ON の木の分岐ができたと考えられる。ライトが OFF の場合には, 夜は明るさの変化はほとんど認められないが, 昼は日光により明るさが変化する。そのため, 明るさの変化にライトからの規則性が見られず, 今回使用したセンサや機器からは定常状態が定義することができなかったと考えられる。

## 8 まとめと今後の課題

本稿では, ユビキタスコンピューティング環境において, 状況監視アプリケーションを支援するミドルウェア, SARADA を提案した。SARADA は決定木学習アルゴリズムを応用し, 異常状態の判断基準を自動的に定義することで, 状況監視アプリケーションの支援を実現した。

今後の課題として, 連続値の離散化手法, 定常状態のパターンの変化への対応, 定常状態の収束時間の定式化が挙げられる。

今回, SARADA では連続値の離散化に統計的手法を用いた。これにより, データの分割数が過度になることを防ぐことはできたが, データの平均値が分割の閾値に与える影響が大きいため, データの分布状態によっては, 閾値の設定が粗過ぎることがあった。連続値の離散化には様々な手法 [11] が存在するため, 今後, それらを計算量や分割数などを考慮に入れ, SARADA に応用する。

定常状態が定義されてから, センサや機器の位置の変化や住人の生活規則の変化により, 定常状態のパターンが変わった場合に, 定常状態を新たに再定義する必要がある。今回, SARADA では古い履歴も新しい履歴も区別せず扱うので, 定常状態を定義し直すには時間がかかってしまうことが予想される。今後, 時間によるデータのクラスタリングや情報の新しさによる重みなどを考慮し, パターンの変化に対応する。

今回, 定常状態の収束を数式で表したが, 今後とも様々な環境で新たに計測を行い, データを収集することで, 定常状態が収束する時間を定式化し, システムが異常の判断を始める時間の決定に応用する。

## 参考文献

- [1] Weiser, M.: The computer for the 21st century, *Scientific American* 256(3), 94-104, September 1991.
- [2] UAHEarthSystemScienceLab: Sensor Modeling Language. <http://stromboli.nsstc.uah.edu/SensorML/>.
- [3] Aplix Corporation: JBlend. <http://www.jblend.com/>.
- [4] Clancey, W. J.: Classification Problem Solving, *Proceedings of the National Conference on AI*, pp. 49-55 (1984).
- [5] Hayashi, C.: The quantitative study of national character: Interchronological and international perspectives, *International Journal of Cultural Studies*, pp. 91-114 (1998).
- [6] D. Heckerman, J. Breese, K. R.: Decision-theoretic troubleshooting, *CACM*, pp. 49-57 (1995).
- [7] Hopfield, J. and Tank, D.: Neural computation of decisions optimization problems, *Biological Cybernetics*, pp. 141-152 (1985).
- [8] Quinlan, J. R.: Induction of decision trees, *Machine Learning vol.1*, Kluwer Academic Press, pp. 81-106 (1986).
- [9] Quinlan, J. R.: Discovering rules from large collections of examples: a case study, *Expert Systems in the Microelectronic Age*, Edinburgh University Press (1979).
- [10] T.Okoshi, S.wakayama, Y.Sugita, S.Aoki, T.Iwamoto, J.Nakazawa, D.Furusaka, M.Iwai, A.Kusumoto, N.Harashima, J.Yura, N.Nishio, Y.Tobe, Y.Ikeda and H.Tokuda: Smart Space Laboratory Project: Toward the Next Generation Computing Environment, *Proceedings of IEEE International Workshop on Networked Appliances(IWNA)*, pp. ??- (2001).
- [11] Dougherty, J., Kohavi, R. and Sahami, M.: Supervised and Unsupervised Discretization of Continuous Features, *International Conference on Machine Learning*, pp. 194-202 (1995).