Context reasoning technologies in ubiquitous computing environment

Sun Jie, Wu ZhaoHui

College of Computer Science,Zhejiang University,Hangzhou, Zhejiang, China 310027
{sunjie,wzh}@zju.edu.cn

Abstract. Context-aware computing is a hot area in ubiquitous computing. There are several challenges to be covered. This paper focuses on context reasoning, which means deducing higher context from raw sensor data. The context reasoning problem is discussed on two different levels: context inference/recognition concerning the generation of context atoms from raw sensor data, and context reasoning concerning the composition of context atoms and deduction of higher-level context. In this paper, we discuss some commonly used reasoning technologies in context-aware systems, including rule-based logics and machine learning methods. Besides, a clustering algorithm, the Symbol Clustering Map, is introduced to learn the current context.

1 Introduction

Mark Weiser first introduced the term ubiquitous computing in 1991[1]. The ability to compute anywhere, anytime in any way is the distinct characteristic. The use of context as an input is one of the major characteristics of ubiquitous computing and the research on this topic is context-aware computing. Context-aware computing is a computing paradigm in which applications can sense, react and adapt to their environment.

In a context-aware system, the most important information is the state of the environment, the devices and the users, which we call context. Context is any information that can be used to characterize the situation of an entity. [2]

For context-aware computing, there exist many challenges to be covered, such as: context acquisition, context modeling, context reasoning, context distribution and utilization. Context acquisition focuses on how to deal with heterogeneous, mass and ambiguous sensor data and gain useful information. Context modeling is to specify the intrinsic properties of and relationship among contexts. Context distribution use some subscription/ publishing and event triggering mechanism to distribute context throughout the system. The utilization of context is different in the existent projects. Context can be utilized in different granularity or level, such as the application level, the device level and the component level.

From the 80’s to now, many projects are developed or in the progress, such as [3, 4, 5], which lay emphasis on the infrastructure of context acquisition, utilization, representation and model of context. However, little emphasis has been laid on context reasoning.

The reasoning technology has a long history in AI. Most context-aware projects pick some technology of AI to do simple reasoning. In our opinion, although AI and context-aware computing are similar in the sense of discovering and reasoning of knowledge, they are two different domains. The requirements and characteristics of context-aware system must be mentioned. In this paper we present the reasoning in context-aware system and introduce an algorithm Symbol Clustering Map (SCM) improved by us.

The remainder of this paper is organized as follows: in section 2 we list the related works applied in context-aware systems. Section 3 gives an introduction of context reasoning. In section 4 we introduce the algorithm of SCM. Section 5 shows the experiment and the result. We conclude the paper in section 6.

2 Related works

The reasoning technologies generally used in context-aware system include rule-based logic, Bayesian network and neural network.

FOL is the most commonly used logic in context-aware projects. The typical one is GAIA [4]. It represents context as first-order predicates in the form of Location (chris, entering, room 3231) and uses rules to infer higher level context. Many other logics are applied, such as fuzzy logic used in [6], description logic used in [7] and temporal logic used in [8]. In most cases logic is powerful for reasoning with context knowledge, but its power is weakened when dealing with discrete unprocessed sensor data.

Bayesian network is a probability-based reasoning method. CORTEX [5] applies Bayesian network to context fusion and the conditional rules are used. [9] uses a naïve Bayes classifier to recognize raw context and reason higher-level context from context atoms. [10] takes advantage of Bayesian network to predict the next room a user will visit. Bayesian network is useful to model the uncertain nature and reason the probabilistic occurrence of a context.

Neural network is commonly used in context recognition and prediction. [11] combines KSOM and probability model to learn the user’s activity. In [12] neural network is used to learn and predict user’s context. [13] uses SOM to recognize different context.

For the development of ontology theory, many context-aware systems adopt ontology-based context model, such as CoolAgent [14], which use RDF to model context and Prolog Forward Chaining to reason context. Many middlewares support OWL-based context modeling and reasoning.

3 Context reasoning

The research of context reasoning can be divided into two levels: the lower level and the higher level, which we call context inference and context reasoning respectively. There are different challenges and problems in the two levels.

3.1 Context inference

This is the lower level of context reasoning, which lay emphasis on generating the current context of the user. We acquire the context of the user and the environment from sensors. We use a vector $S_1 \times S_2 \times \cdots \times S_n$ to denote the sensor readings at time t: $< s_1 \times s_2 \times \ldots \times s_n > \in S_1 \times S_2 \times \ldots \times S_n$.

In order to process the raw sensor data into context information, we need to solve the challenges such as: how to process the sensor data? How to deal with the conflictive information from multiple sources? How to match the raw data into context? We will discuss these in detail.

(1) The sensor data preprocessing:

The sensor data are enormous, heterogeneous and nonstandard, which makes it hard to harness these data directly. So before the further process we must use different technologies to preprocess the sensor data. We reduce the data amount by sampling and using average or variance over a time to substitute the raw data. We rescale the sensor range to smooth the data diversity.

(2) Sensor fusion

Sensor fusion aims at integrating the information from multiple sensors. There are three kinds of fusion: competitive fusion, complementary fusion and cooperative fusion. The competitive fusion combines sensor data that represent the same measurement to reduce uncertainty and resolve conflicts. It is a basic fusion type. The fusion function always takes the form of weighting average. The complementary fusion combines incomplete sensor data do not dependant on each other directly to create a more complete model. The cooperative fusion combines sensor observations that depend upon each other to deduce higher-level measurement. The commonly used sensor fusion methods include classical inference, Bayesian inference, Dempster-Shafer theory, voting and fuzzy logic.

(3) Context matching

The raw sensor data is not sufficient for the context-aware system. We must derive knowledge about the user/device/environment context from it. The following steps are applied: feature extraction, classification, labeling.

From raw sensor data, specific data values will be extracted, which are called features. After the sensor data preprocessing, the average $\bar{s}$ or deviation σ can be used as features for time series of numerical data. We use a feature vector $< f_1, f_2, \ldots, f_n >_t \in F_1 \times F_2 \times \ldots F_n$ to denote the feature at time t.

Classifying the extracted features into clusters is to find the common pattern of the feature vectors. Using the pattern we can classify the feature vectors into clusters, in which each cluster is a context class. Each context class is assigned a certain membership $c_i : F_1 \times F_2 \times \ldots \times F_n \to C^m$. The commonly used clustering algorithms include Kohonen Self-Organizing Map (SOM), the Recurrent Self-Organizing Map (RSOM), K-Means clustering, Hartigan's sequential leader clustering, neural gas, and growing neural gas (GNG).

For each clustering, a descriptive name must be assigned for users to utilize context. Labeling maps context classes into a set of names: $C^m \to N$, where N is name of semantic meaning. The name comes from the ontology library which is domain-specific. This is valuable in context sharing and cooperation.

In many papers research on context recognition are described. We think that the two are the same thing in respect that they are all deducing context atoms from sensor data.

3.2 Context reasoning

This is the higher level of context reasoning, which lay emphasis on deducing higher level context from lower level context. The reasoning technologies taken in context-aware system can be grouped into two categories: logic and learning mechanism.

Logics are very powerful tools for reasoning with context knowledge, and they are sufficient for general pervasive context aware systems .The system can reason about context using rules written in different logics such as first order logic, temporal logic, and description logic. FOL is the most classic logic. Temporal logic extends conventional proposition logic in adding special operator that can specify the objects to behave as time progresses. Using temporal logic the specification of an object can define the attribute and relation of the past, present and the future. Description Logics are subsets of first order logic, and can be viewed as providing semantics for much network-based object-orientated formalism. They serve to disambiguate imprecise representations that these formalisms permit. Description Logics are the most expressive decidable logics with classical semantics, and usually are used in ontology reasoning.

Learning mechanisms are also generally used in context reasoning such as neural networks and Bayesian network. Neural networks are a classic learning mechanism which will converge to a stable state after a period of training. The network will output the state of the system whenever the contexts are input into the network. SOM is the most popular unsupervised neural network that can adjust the weight coefficient of the neuron to simulate the input signal pattern in an adaptive way. Bayesian network focuses on the representation and reasoning of uncertain knowledge. It is a directed acyclic graph with a conditional probability table associated with every node

SCM is a kind of learning mechanism, too. SCM is an unsupervised clustering and recognizing algorithm of symbol string. We use SCM in our system for the following advantages:

Autonomy: the algorithm is unsupervised and does not need the user interaction. The initialization can be of any random state and the system input samples of the environment automatically.

Noise resistance: the signal is sampled in the real world and so the noise is inevitable. SCM works well with noise and the performance will not be affected.

Simplicity: the algorithm must be simple enough to avoid heavy operations on the feature vectors.

SCM are presented as an algorithm for the unsupervised clustering of symbol string data based on adaptive learning and the application into context-aware system are also mentioned as a method of context recognition in [15, 16]. However, there are some problems in practice: the node strings can not converge to a stable state and the algorithm does not take into account the different source of context. So we improve the algorithm in the computation of similarity and the updating rules. The following sectors will describe the algorithm of SCM in detail.

4 The algorithm of SCM

The basis of SCM is using symbol sets to represent the state of the system. We mention that context can be easily and completely represented by symbol strings. For example, we enter a shop. If we represent each type of commodities as $\psi_i \in \Psi$, in which Ψ is the summary of commodities in the shop and $\Psi = \{\psi_1, ..., \psi_N\}$, the shopping basket of customer i can be represented by a symbol string: $S_i = \{s_1, s_2, ..., s_n\}, s_j \in \Psi, j = 1, ..., n(i)$. It is a simple, natural and general form to represent a context by a symbol or a symbol string. We describe the idea in a formalized way in the first subsection and the algorithm in the second subsection.

4.1 The symbol strings of context

With the symbol string, we can describe the two most important properties of context:

(1) Hierarchical

The context is hierarchical essentially.

$$S_j^i = \{s_{j,1}^i, s_{j,2}^i, \dots, s_{j,N_j}^i\} \tag{1}$$

We use S_j^i to denote the context state of context source j at level i., in which every context includes N_j properties. The context at level 0 is the raw sensor data.

The higher level context is the composition of the lower level context. The context at level i+1 can be deduced from the context at level j.

$$S_j^{i+1} = \{s_1^i, s_2^i, \ldots, s_{N_{i+1}}^i\} \tag{2}$$

(2) Temporal

The output of a context source will vary with the time. We call the output of the context source varying with time the state of context. We use $S_j^i(t)$ to denote a context state, in which the meaning of superscript and subscript is same as above.

We can fusion the context of different sources at the same time to deduce new context. This is the most general reasoning in context-aware system. The order of symbols in the sequence does not matter.

$$S_j^{i+1}(t) = \{s_1^i(t), \ldots, s_n^i(t)\} \tag{3}$$

We can fuse the context of the same sources during the sequential time to deduce new context. This is often referred to as context prediction. The order of symbols in the sequence is of great importance.

$$S_j^{i+1}(t_{l+m}) = \{s_j^i(t_1), s_j^i(t_2), \ldots, s_j^i(t_{l+m})\} \tag{4}$$

Using the symbol string to describe the properties above, we gain the context states sequence of context source j at level i:

$$\{s_j^i(t_1), s_j^i(t_2), s_j^i(t_3), \ldots\} \tag{5}$$

4.2 The algorithm

The basic structure of SCM is a two dimension lattice with M*M nodes. There are a symbol string $S_k(t)$ and a weight vector $X_k(t)$ associated with each node k, where $S_k(t) = \{s_1^k(t), s_2^k(t), \ldots, s_n^k(t)\}$, $s_j^k(t) = (s_{j,1}^k, s_{j,2}^k, \ldots, s_{j,n(k,j,t)}^k)$. Here the superscript is used to denote the current node k. $s_{j,i}^k$ is the ith context state from source j at time t. $n(k, j, t)$ is the number of symbols of context source j of node k at time t. Initially $s_k(t)$ is a string of single symbol selected randomly and $n(k, j, t)$ =1. $X_k(t) = \{x_1^k(t), x_2^k(t), ..., x_n^k(t)\}$, $x_j^k(t) = (x_{j,1}^k, x_{j,2}^k, ..., x_{j,n(k,j,t)}^k)$. Each $x_{j,i}^k \in [0,1]$ and is directly associated with $s_{j,i}^k$. The two sets are initialized to random values.

As other neural networks, the training similarly consists of two steps: 1) finding the best-matching unit for the input and 2) adaptation of the models.

The input string at time t is $\hat{S}(t) = \{\hat{s}_1(t), \hat{s}_2(t), ..., \hat{s}_{n(t)}(t)\}$. For each node we calculate the similarity between the input string and the node string using a function,

$$A_k(t) = \frac{(\sum_{i=1}^{\hat{n}(t)} \sum_{j=1}^{n(k,t)} \hat{\delta}_{k,i}(t)\delta_{k,j}(t))^2}{n(k,t)\hat{n}(t)}, \text{ where}$$

$$\delta_{k,j}(t) = \begin{cases} 1, s_j(t) \in S_j^k(t) \\ 0, s_j(t) \notin S_j^k(t) \end{cases}$$, and $\hat{\delta}_{k,j}(t)$ is defined in the same way.

After the calculation, a node with the most similarity is selected as the winner node, which is the one that most matches the input and can best describes the input:

$$v(t) = \arg \max_{1 \le k \le M \times M} A_k(t).$$

The updating rules for the node string $S_k(t)$ and the weight vector $X_k(t)$ are:
For each node k=1,2,…,M*M,

(1) If $s_{j,i}^k(t) \in \hat{S}(t)$, $x_{j,i}^k(t) = x_{j,i}^k(t) + \alpha(t)h_m(dl)(1.0 - x_{j,i}^k(t))$

(2) If $s_{j,i}^k(t) \notin \hat{S}(t)$, $x_{j,i}^k(t) = x_{j,i}^k(t)(1.0 - \alpha(t) \mid h_m(dl))$

(3) If $\hat{s}_j(t) \notin s_j^k(t)$ and $h_m(dl)$ >0, $n(k, j, t) = n(k, j, t) + 1$,

$s_{j,n(k,j,t)}^k(t) = \hat{s}_j(t)$, $x_{j,n(k,j,t)}^k(t) = \alpha(t)h_m(dl)$

(4) If $x_j(t) < \beta(t)$, $n(k, j, t) = n(k, j, t) - 1$.

In which the function $h_m(dl)$ is Mexican hat:

$$h_m(i) = (1 - a * b * i_2)\exp(-a * i^2),$$

and the parameter dl is the Euclidean distance between the winner node v(t) and the node k being updated.

In this algorithm we take the context type into account and the matching of symbols must happen between contexts of same type. This guarantees the clustering of training data with the same environment information.

5 The experiment

The experiment shows how we use symbol string clustering algorithm to recognize and deduce the higher level context.

Firstly we gain context data from sensors. We equip six kinds of sensors, including temperature, humidity, light, microphone, accelerometer and a cell phone with GSM positioning service. We define six types of context atom classes: temperature, humidity, light, loudness, speed and location. For each context atom we define a symbol which is unique as shown in table 1:

Table 1. Context atoms and the symbols assigned

Context atom class	sensor	Context atom
temperature	temperature	1=hot,2=warm,3=cool,4=cold,5=Unknown
Humidity level	Humidity	6=humid, 7=normal, 8=dry, 9= Unknown
Loudness level	microphone	10=loud,11=normal,12=murmuring,13=silent,14= Unknown,
Light type	Ambient light	15=bright, 16=normal, 17=dim, 18=dark,19= Unknown
Speed level	accelerometer	20=fast,21=normal,22=low,23=walk fast,24=walk slowly,25= still,26= Unknown
location	GSM	27=home, 28=workplace, 29=marketplace,30=Unknown

Here we assign unique symbol for each atom in respect that the context recognizing is done through symbol matching between the node string and the input string. Hence the similarity is calculated and the winner node is selected. So the symbol assigned to each context must be unique. Here we use integer for its simplicity. For the future research on context uncertainty, we leave a value Unknown to represent the occasion the sensor data is ambiguous or vague. The speed is divided into six levels, which the first three are used to denote the speed of user in the vehicle, such as the bus, while the next three are used to denote the walk speed. As for the location we only emphasize three positions and the left can be extended easily.

The experiment focus on four scenarios: at working, shopping, in street and at home. Each of them involves different activities as shown in table 2:

Table 2. The experiment scenarios

scenario	activity
1:working	Sitting
	Walking around
	Discussion
2:shoppingl	Walking around
	Stand still
	Conversation
3:in the street	By bus
	Walking
	Run
4:home	Sitting
	Cooking
	Watching TV

Each activity will last for 2 minutes and will be repeated 10 times. So there will be 1200 symbol strings for each activity. For the whole experiment the sum will be 14400 symbol strings.

We use a lattice of 15×15 nodes. The node strings and the associated weight vectors are randomly initialized. The resulting clustering distribution is shown in figure 1 and figure 2:

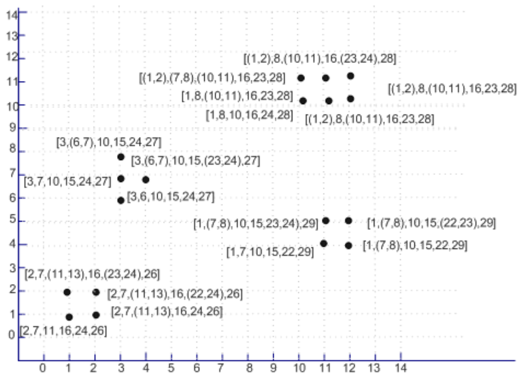

Fig. 1. The experiment result: symbol strings for nodes of the SCM. The nodes converge to 4 clusters corresponding to the 4 scenarios.

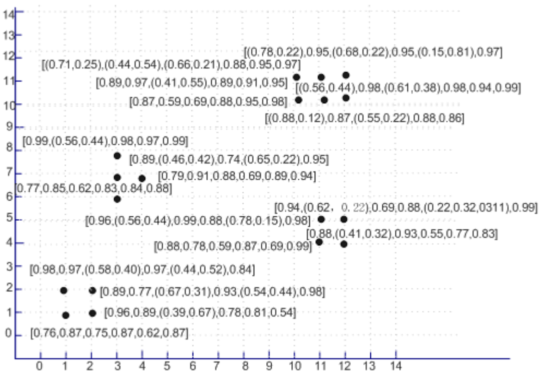

Fig. 2. The experiment result: the corresponding weight vectors associated with the symbol strings of each node of the SCM.

6 Conclusion

Context reasoning is one important problem in the research of context-awareness, but it is not laid enough emphasis on. Although it is similar in the sense of knowledge discover and learning with AI, the difference must be specified.

This paper discusses context reasoning in two levels: the lower level is context inference/recognition, which focuses on generating context atoms from raw data, and the higher level is context reasoning, which focuses on the composition of context atoms and deducing context of higher abstract level.

We describe the technologies used in context-aware systems and introduce an improved algorithm SCM used to adaptively learn the current context. The algorithm does not need any user intervention and is suitable for mobile devices. SCM has been applied in real-time recognition. It is proven feasible to separate different context scenarios using SCM.

References

1. Mark Weiser: The Computer for the 21st Century. Scientific American, (1991) 94-100
2. Anind K. Dey: Understanding and Using context. Personal and Ubiquitous Computing, Vole 5, Issue 1, (2001).4-7.
3. Anind K. Dey: The Context Toolkit: Aiding the Development of Context-Aware Applications. Workshop on Software Engineering for Wearable and Pervasive Computing , Limerick, Ireland (2000).
4. Anand Ranganathan, Roy H. Campbell: An infrastructure for context-awareness based on first order logic. Personal and Ubiquitous Computing , Vol. 7 , Issue 6 (2003).
5. Gregory Biegel, Vinny Cahill: A framework for developing mobile, context-aware applications. In Second IEEE International Conference on Pervasive Computing and Communications (2004) 361.
6. Mantyjarvi, Jani, Seppanen: Adapting applications in handheld devices using fuzzy context information. In Interacting with Computers, 15 (4) , (2003)521-538.
7. Xiao Hang Wang, Da Qing Zhang, Tao Gu, Hung Keng Pung: Ontology Based Context Modeling and Reasoning using OWL. Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops, (2004)18.
8. Hongwei Zhu, Stuart E. Madnick, Michael D. Siegel: Reasoning About Temporal Context Using Ontology and Abductive Constraint Logic Programming. proceeding of Principles and Practice of Semantic Web Reasoning: Second International Workshop, PPSWR 2004 , (2004)90-101.
9. Panu Korpipaa, Jani Mantyjarvi, Juha Kela, Heikki Keranen, Esko-Juhani Malm: Managing Context Information in Mobile Devices. Pervasive compuoting, July-September Vol. 2, No. 3 (2003) 42-51.
10. Jan Petzold, Andreas Pietzowski, Faruk Bagci, Wolfgang Trumler, Theo Ungerer: Prediction of Indoor Movements Using Bayesian Networks. First International Workshop on Location- and Context-Awareness (LoCA 2005), Oberpfaffenhofen, Germany (2005).
11. Kristof Van Laerhoven, Ozan Cakmakci: What Shall We Teach Our Pants?. Proceedings of the 4th IEEE International Symposium on Wearable Computers (2000).
12. Jong-Hwa Choi, Soon-yong Choi, Dongkyoo Shin, Dongil Shin: Research and implementation of the context-aware middleware based on Neural network. In proceeding of AIS 2004, Jeju Island, Korea, (2004)295-303.
13. Kristof Van Laerhoven, Kofi Aidoo: Teaching Context to Applications. Personal and Ubiquitous Computing, Vol.5, Issue 1, (2001) 46 – 49.
14. Harry Chen, Sovrin Tolia, Craig Sayers, Tim Finin, and Anupam Joshi: Creating Context-Aware Software Agents. First GSFC/JPL Workshop on Radical Agent Concepts, Greenbelt, MD, USA (2001).
15. John A. Flanagan: Unsupervised Clustering of Symbol Strings. In proceeding of International Joint Conference on Neural Networks (IJCNN 2003), Portland, Oregon, USA, (2003) 3250-3255.
16. J. Himberg, J. A. Flanagan, J. MÄantyjÄarvi: Towards context awareness using Symbol Clustering Map. In Proceedings of the Workshop for Self-Organizing Maps 2003 (WSOM2003), Kitakyushu, Japan, (2003) 249–254.