A Relational Approach to Sensor Network Data Mining

Floriana Esposito, Teresa M.A. Basile,
Nicola Di Mauro, and Stefano Ferilli

Abstract. In this chapter a relational framework able to model and analyse the data observed by nodes involved in a sensor network is presented. In particular, we propose a powerful and expressive description language able to represent the spatio-temporal relations appearing in sensor network data along with the environmental information. Furthermore, a general purpose system able to elicit hidden frequent temporal correlations between sensor nodes is presented. The framework has been extended in order to take into account interval-based temporal data by introducing some operators based on a temporal interval logic. A preliminary abstraction step with the aim of segmenting and labelling the real-valued time series into similar subsequences is performed exploiting a kernel density estimation approach. The prposed framework has been evaluated on real world data collected from a wireless sensor network.

1 Introduction

Sensor networks represent a powerful technology able to monitor many situations in the physical world including health, agriculture, emergency management, micro-climate and habitat, or earthquake and building health [3, 4, 11, 23]. The main objective of sensor networks is knowledge-gathering: each component (sensor node) acts to maximize its contribution by adding details and precision in order to completely outline the monitored situation and, by cooperating with the others, to understand phenomena *in situ* and in real time. Sensor nodes are small electronic components made up of a processing element, some measurement devices and a (wireless/wired)

Floriana Esposito · Teresa M.A. Basile · Nicola Di Mauro · Stefano Ferilli
Department Of Computer Science,
University of Bari, Italy
e-mail: {esposito,basile,ndm,ferilli}@di.uniba.it

communication device. They are able to gather different types of information from the environment, such as temperature, light, humidity, radiation, the presence or nature of biological organisms, geological features and more. As a consequence, a great amount of data is available that if analyzed in an appropriate way might help to automatically and intelligently solve a variety of tasks thus making the human life more safe and comfortable.

A sensor network is usually made up of a set of nodes spatially distributed in the environment, i.e. each node i is located in an environment at the position p_i, and senses a set of properties $\mathcal{P}_i$ at every time instance t. In other words, each sensor produces a continuous time series describing its reading over time, hence we have an observation at every instant of time. Furthermore, the data generated by sensor nodes involved in a sensor network are type-related (the humidity depends on the temperature), time-related (the temperature may change over time) and spatio-related (topological arrangements of the sensors in the network). All these relations could be easily represented by using an interval-based relational language, such the one proposed in this work, as opposed to the point-based approach, trying to shift the basic time-series description language to a higher one.

Finally, a set of events on different dimensions (time, space, etc.) can take place in the physical environment that could influence the sensor behaviour and hence the observation, thus the contextual information could be taken into account as well. Hence, temporal and context-based relations must be combined into a heterogeneous language and mined with appropriate techniques in order to obtain useful knowledge.

In the last decade, some approaches were proposed to face the problem of extracting knowledge from sensor data. They focused either on the data representation (e.g., sensors clustering, discretization) or knowledge extraction (association rules, sequential patterns). Nevertheless, they usually do not consider contextual information or they generally consider events occurring in one dimension only or in a time instant, while, in some applications, like in sensor networks, data are environment related, high dimensional and may occur in time intervals.

In this work the exploitation of a relational language to describe the temporal evolution of a sensor network along with contextual information is proposed. Furthermore, it is presented the use of relational learning techniques to discover interesting and more human readable patterns relating spatio-temporal correlations with the contextual ones.

As regards the relational language, it is based on the work described in [9] where the authors proposed a framework for mining complex patterns, expressed in first-order language, in which events may occur along different dimensions.

Here, that framework is extended in order to take into account interval-based temporal data along with contextual information about events occurring in the environment. The extension concerns the introduction of interval-based operators based on the Allen's temporal interval logic [5] in the sequences. Specifically, firstly an abstraction step with the aim of segmenting and labelling the real-valued time series into similar subsequences is performed exploiting a kernel density estimation approach. Then the integration of such a new knowledge along with the relative interval-based operators, able to deal with it, in the relation pattern mining framework is carried out. Finally, in order to evaluate the validity of both the abstraction step and the general framework, an experimental session on real world data collected from a wireless sensor network deployed in the Intel Berkeley Research Lab [15] is presented.

2 Relational Pattern Mining

The framework we present in this chapter is based on the work described in [6, 9] where the authors proposed an algorithm for mining complex patterns, expressed in first-order language, in which events may occur along different dimensions. Specifically, multi-dimensional patterns were defined as a set of atomic first-order formulae in which events are explicitly represented by a variable and the relations between events were represented by a set of dimensional predicates. The algorithm has been extended in order to take into account interval-based temporal data. Finally, an automatic discretization algorithm based on the concept of kernel density estimation has been introduced and evaluated.

Datalog [28] is the language used as representation language for the domain knowledge and patterns. Sequences and patterns are represented by a set of logical atoms[1]. Thus, a *relational sequence* may be defined as an ordered list of atoms separated by the operator $<$: $l_1 < l_2 < \cdots < l_n$, while a relational pattern is defined as follows:

Definition 1 (Subsequence [16]). Given a sequence $\sigma = (e_1 e_2 \cdots e_m)$ of m elements, a sequence $\sigma' = (e'_1 e'_2 \cdots e'_k)$ of length k is a *subsequence* (or a pattern) of the sequence σ if:

1. $1 \leq k \leq m$
2. $\forall i, 1 \leq i \leq k, \exists j, 1 \leq j \leq m : e'_i = e_j$
3. $\forall i, j, 1 \leq i < j \leq k, \exists h, l, 1 \leq h < l \leq m : e'_i = e_h$ *and* $e'_j = e_l$.

Example 1. Let us now introduce an example to better explain the representation language and the operators introduced in the following. Suppose to have a wireless sensor network deployed inside a building, for example a conference building, to monitor people's motion over space and time and, accordingly, the temperature, light and voltage in the rooms.

[1] An atom $p(t_1, \ldots, t_n)$ is a predicate symbol p of arity n applied to n terms t_i (constants or variables).

In this setting, a 1-dimensional sequence, specifically a time dimension sequence, could be:

```
move(user1,room5) < in(user1,room5) < talk(user1,room5,machine_learning)
 < leave(user1,room5) < move(user1,room4)
```

and a possible pattern could be `move(X,Y) < talk(X,Y,Z)`.

As one can note, the exploitation of just one dimension is not sufficient to describe the environment, indeed information about sensors or events occurring in the environment cannot be easily represented. Hence, some modifications have to be introduced as reported in the following. □

In order to make the framework more general, the concept of *fluents* has been considered. Let a sequence be an ordered succession of events, a fluent is used to indicate that an atom holds for a given event. In this way we are able to distinguish in the sequence, and hence in the pattern, *dimensional* and *non-dimensional* atoms. Specifically, the first ones refer to the dimensional relations between events involved in the sequence while the non-dimensional atoms introduce an event and the objects involved in it (fluent atoms) or the properties and the relations of the objects already introduced by an event (non-fluent atoms).

Example 2. The introduction of such kind of atoms allows one to introduce the events occurring in a situation, as reported in the following:

```
move(entering1,user1,room5) (entering1 < entering2)
move(entering2,user1,room4) near(room5,room4)
```

It denotes a 1-dimensional relational sequence with three non-dimensional atoms (i.e., `move(entering1,user1,room5)`, `move(entering2,user1,room4)` and `near(room5,room4)`) and one dimensional atom. Specifically,

- `move(entering1,user1,room5)` denotes the fluent `move(user1,room5)` at the event `entering1`,
- `move(entering2,user1,room4)` denotes the fluent `move(user1,room4)` at the event `entering2`,
- `(entering1 < entering2)` indicates that the event `entering2` is the direct successor of `entering1`, and
- `near(room5,room4)` represents a generic relation between the objects `room5` and `room4`.

The choice to add the event as an argument of the predicates is necessary for the general case of n-dimensional sequences with $n > 1$. In this case, indeed, the operator $<$ is not sufficient to express multi-dimensional relations and we must use its general version $<_i$. Specifically, $(e_1 <_i e_2)$ denotes that the event e_2 is the next successor of the event e_1 in the dimension i, where i could be, for example, time or space. Hence, in our framework a multi-dimensional sequence is supposed to be a set of events, and a sequence of events corresponds to each dimension. □

However, the $<_i$ operator is not sufficient to represent the knowledge in the patterns. Hence, a further generalization of the framework consisted in the possibility to represent multi-dimensional relational patterns by introducing some dimensional operators able to describe general event relationships: a) $<_i$, *next step on dimension* i; b) $\lhd_i$, *after some steps on dimension* i; and c) $\bigcirc_i^m$, *exactly after* m *steps on dimension* i.

Now, the general definition of subsequence (Def. 1) can be cast in this new framework by modelling the *gaps* represented by the third condition of Definition 1 with the $\lhd_i$ and $\bigcirc_i^m$ operators as reported in the following definition.

Definition 2 (Multi-dimensional relational pattern). A multi-dimensional relational pattern is a set of atoms, involving k events and regarding n dimensions, in which there are non-dimensional atoms and each event may be related to another event by means of the operators $<_i$, $\lhd_i$ and $\bigcirc_i^m$, $1 \leq i \leq n$.

Example 3. With such extensions concerning both the representation language and the operators, it is possible to better represent the environment as reported by the following example.

location(sensor1,room5) $<_{topology\_r5}$ location(sensor2,room5) $<_{topology\_r5}$
location(sensor3,room5) ... location(sensor1,room4) $<_{topology\_r4}$
location(sensor2,room4) $<_{topology\_r4}$ location(sensor3,room4) ...
move(entering1,user1,room5) activity(talking,user1,room5)
move(leaving,user1,room5) (entering1 $<_{time}$ leaving)
(entering1 $<_{time}$ talking) (entering1 $<_{spatial}$ entering2)
(talking $<_{time}$ leaving) move(entering2,user1,room4)
(talking $<_{time}$ entering2) activity(coffee_break,user1,room4)
(leaving $<_{spatial}$ entering2) (entering2 $<_{spatial}$ coffee_break)

and the corresponding temporal patterns that may be true when applied to it could be:

- move(entering1,user1,room5) (entering1 $<_{time}$ talking) activity(talking,user1,room5)
- move(entering1,user1,room5) (entering1 $\lhd_{time}$ leaving) move(leaving,user1,room5)
- move(entering1,user1,room5) (entering1 $\bigcirc_{time}^2$ entering2) move(entering2,user1,room4) if we consider 2 as hours

Note that the $<_i$ will describe the dimensional characteristics in the sequences, while all the three dimensional operators $<_i$, $\lhd_i$, $\bigcirc_i^m$, will be used to represent the discovered patterns. □

In particular, we are interested in mining maximal frequent patterns. Thus, let σ a sequence and p a pattern of σ. The frequency of p in σ is the number of different mappings from elements of p into the elements of σ such that the conditions reported in Definition 1 hold, and, p is *maximal* if there is no pattern p' of σ more frequent than p and such that p is a subsequence of p'.

Since the sequences and patterns are represented as a set of logical atoms, the frequency of a pattern over a sequence can be calculated as the number of subtitutions θ_i such that p subsumes σ, i.e. $p\theta_i \subseteq \sigma$ where subsumption and substitution are defined as follows.

Definition 3 (Subsumption). A a set of logical atoms c_1 θ-subsumes a set of logical atoms c_2 if and only if there exists a substitution θ such that $c_1\theta \subseteq c_2$.

A substitution θ *is defined as a set of bindings* $\{X_1 \leftarrow a_1, \ldots, X_n \leftarrow a_n\}$ *where* $X_i, 1 \leq i \leq n$ *is a variable and* $a_i, 1 \leq i \leq n$ *is a term. A substitution* θ *is applicable to an expression* e*, obtaining the expression* $e\theta$*, by replacing all variables* X_i *with their corresponding terms* a_i*.*

Example 4. Given the following sequence:
$S =$ `p(e1,a) q(a,t) q(a,s) (e1 < e2) p(e2,b) q(b,a)`
and the pattern
$P =$ `p(E,X) q(X,Y)`
there are 3 way to instantiate P from S in such a way that to different terms correspond different objects, i.e. :

1. $\theta_1 = \{$`E/e1, X/a, Y/t`$\}$,
2. $\theta_2 = \{$`E/e1, X/a, Y/s`$\}$,
3. $\theta_3 = \{$`E/e2, X/b, Y/a`$\}$.

However, since θ_1 and θ_2 map the same constants to the variables of `p(E,X)` (the first literal of the pattern), the frequency of P on S is equal to 2. □

2.1 The Algorithm

The algorithm for frequent multi-dimensional relational pattern mining is based on the same idea of the generic level-wise search method, known in data mining from the Apriori algorithm [2]. The generation of the frequent patterns is based on a top-down approach. Specifically, it starts with the most general patterns of length 1 generated by adding to the empty pattern a non-dimensional atom. Then, at each step it *specializes* all the frequent patterns, discarding the non-frequent patterns and storing the ones whose length is lesser than a user specified parameter *maxsize*. Furthermore, for each new refined pattern, semantically equivalent patterns are detected, by using the θ_{OI}-subsumption relation, and discarded.

In the specialization phase, the refinement of patterns is obtained by using a refinement operator ρ that maps each pattern to a set of specializations of the pattern, i.e. $\rho(p) \subset \{p' | p \preceq p'\}$ where $p \preceq p'$ means that p is more general of p' or that p subsumes p'.

The algorithm uses a background knowledge $\mathcal{B}$ (a set of Datalog clauses) containing the sequence and a set of constraints that must be satisfied by the generated patterns. In particular $\mathcal{B}$ contains the following predicates:

- `maxsize(M)`: maximal pattern length (i.e., the maximum number of non-dimensional predicates that may appear in the pattern);
- `minfreq(m)`: this constraint indicates that the frequency of the patterns must be larger than m;
- `dimension(next_i)`: this kind of atom indicates that the sequence contains events on the dimension i. One can have more that one of such atoms, each of which denoting a different dimension. In particular, the number of these atoms represents the number of the dimensions.

2.1.1 Constraints

Furthermore the background knowledge contains some constraints that are useful to avoid the generation of unwanted patterns. Specifically they are:

- `negconstraint([`$p_1, p_2, \ldots, p_n$`])`: specifies a constraint that the patterns must not fulfill, i.e. if the clause $\{p_1, p_2, \ldots, p_n\}$ subsumes the pattern then it must be discarded. For instance, `negconstraint([p(X,Y),q(Y)])` discards all the patterns subsumed by the clause {`p(X,Y),q(Y)`};
- `posconstraint([`$p_1, p_2, \ldots, p_n$`])`: specifies a constraint that the patterns must fulfill. It discards all the patterns that are not subsumed by the clause $\{p_1, p_2, \ldots, p_n\}$;
- `atmostone([`$p_1, p_2, \ldots, p_n$`])`: this constraint discards all the patterns that make true more than one predicate among $p_1, p_2, \ldots, p_n$. For instance, `atmostone([red(X),blue(X),green(X)])` indicates that each constant in the pattern can assume at most one of `red`, `blue` or `green` value.

Hence, the solution space is pruned by using some positive and negative constraints specified by the *negconstraint* and *posconstraint* literals. The last pruning choice is defined by the *atmostone* literals. This last constraint is able to describe that some predicates are of the same type.

2.1.2 Improving Efficiency

In order to avoid the generation of patterns containing not linked variables we used the classical types and modes declaration:

- `type(p)`: denotes the type of the predicate's arguments p;
- `mode(p)`: denotes the input output mode of the predicate's arguments p.

In this way we improve the efficiency of the algorithm, since it does not generate patterns containing unrelated atoms. These classical mode and type declarations specify a language bias indicating which predicates can be used in the patterns and to formulate constraints on the binding of variables.

Finally, the background knowledge contains the predicate `key([`$l_1, l_2, \ldots, l_n$`])` specifying that each pattern must have one of the predicates $l_1, l_2, \ldots l_n$ as a starting literal. Since each pattern a) must start with a non-dimensional predicate, or with a predefined key, and b) its frequency must be less than the sequence length, the frequency of a pattern can be defined as follows.

Definition 4 (Pattern Frequency and Support). Given a relational pattern $P = (p_1, p_2, \ldots, p_n)$ and S a relational sequence, the *frequency* of the pattern P is equal to the number of different ground literals used in all the possible $\mathrm{SLD_{OI}}$-deductions of P from S that make true the literal p_1. The *support* of P on S is equal to the frequency of the pattern $\{p_1\}$ over the frequency of the pattern P.

Mining from more than one sequence the support is calculated as the number of covered sequences over the total number of sequences.

In order to improve the efficiency of the algorithm, for each pattern $P = (p_1, p_2, \ldots, p_n)$ the set Θ of the substitutions defined over the variables in p_1 that make true the pattern P is recorded. In this way, the support of a specialization P' of P is computed by firstly applying a substitution $\theta \in \Theta$ to P'. It is like to remember all the keys of a table that make true a query.

3 Interval-Based Relational Sequences

In this section we present the extension of the framework to the case of relational sequences including interval-based dependencies. Furthermore, since we are working on real-valued time series, an approach to subdivide/discretize the series into similar subsequences is presented. In particular, the aim is to segment a signal (assigning data to discrete categories) by looking for a sequence of measurements over which a property holds and to label this segment. After this discretization process, some interval relationships may be introduced in order to better describe the evolution of the data along the time.

3.1 Abstracting Using Kernel Density Estimation

A method to segment a sequence is to iteratively merge two similar segments based on the squared error minimization criteria. Another approach is using clustering, by firstly finding the set of subsequences with length w, by sliding a window of width w, and then clustering the set of all subsequences. A different symbol is associated with each cluster. Other approaches are based on using self-organizing maps.

The segmentation process has been obtained by adopting an unsupervised discretization method that uses non-parametric density estimators, as proposed in [7]. The algorithm searches for the next two sub-intervals to produce, evaluating the best cut-point on the basis of the density induced in the sub-intervals by the current cut and the density given by a kernel density estimator for each sub-interval. It uses cross-validated log-likelihood to select the maximal number of intervals.

Two classical techniques for unsupervised discretization are equal-width and equal-frequency binning, where continuous intervals are split into sub-intervals providing them the width or the frequency parameter. However, they require that the values follow a uniform distribution, with low accuracy in case of skew data. The method used here exploits density estimation methods to select the cut-points and their number is computed by cross-validating the log-likelihood.

3.1.1 Simple Binning and the Naive Estimator

The histogram, or simple binning, is the oldest and most popular density estimator. Given a set of training instances $x_1, \ldots, x_N$, let x_0 be an origin and w be the bin (class) width. The intervals (or bins) may be defined as follows

$$I_j = [x_0 + jw, x_0 + (j+1)w), j = 0, 1, \ldots$$

for which the histogram counts the number of instances x_i falling into each I_j, as reported in Figure 1. This procedure replaces the training data $x_1, \ldots, x_N$ with the smaller set $c_1, \ldots, c_g$, where c_j is the corresponding class (label) of the interval I_j.

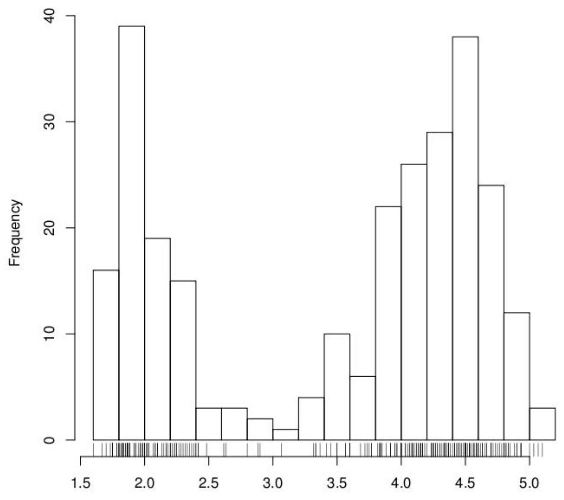

Fig. 1 Histogram for eruptions of the Old Faithful geyser in Yellowstone National Park, Wyoming, USA

In particular, let k be the number of intervals (bins) and $\mathbf{1}_{I_j}$ be the indicator function[2], the density function $\hat{f}(x)$ is computed by

$$\hat{f}(x) = \left(\sum_{i=1}^{M} \mathbf{1}_{I_i}(x) \sum_{j=1}^{N} \mathbf{1}_{I_i}(x_j) \right) (Nw)^{-1}.$$

[2] $\mathbf{1}_{I_j}(x)$ is equal to 1 when $x \in I_j$, 0 otherwise.

From the definition of a probability density, for any given h, it is possible to estimate the probability $P(x-h < X \leq x+h)$ by the proportion of the observations falling in the interval $(x-h, x+h]$. The naive estimator is given by choosing a small number h and setting

$$\hat{f}(x) = \frac{1}{2nh}[\text{no. of } X_i \text{ falling in } (x-h, x+h)].$$

3.1.2 The Kernel Density Estimator

The naive estimator is not a continuous function and hence it is interesting to consider its generalization. In particular, it is useful to consider the kernel estimator, using a smooth kernel function $K(\cdot)$, defined as

$$\hat{f}(x) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x-X_i}{h}\right),$$

$$K(x) \geq 0, \int_{-\infty}^{+\infty} K(x)dx = 1, K(x) = K(-x).$$

The kernel function used in this paper is the Epanechnikov kernel function [22], see Figure 2, defined as

$$K(u) = \frac{3}{4}(1-u^2)\mathbf{1}_{|u|\leq 1}.$$

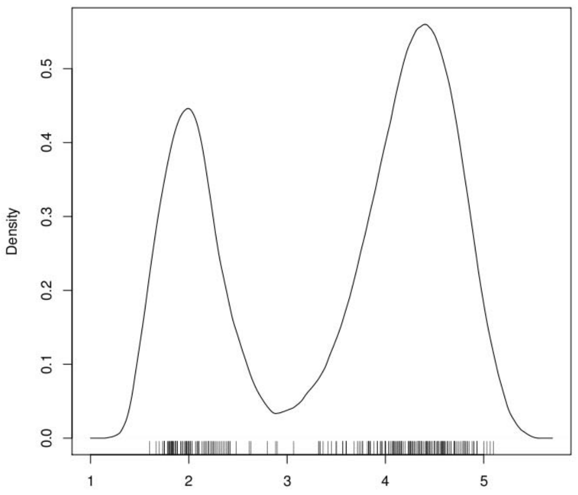

Fig. 2 Epanechnikov kernel (bandwidth = 0.2) for eruptions of the Old Faithful geyser in Yellowstone National Park, Wyoming, USA

3.1.3 The Scoring Function

The goal of discretization is to produce sub-intervals whose induced density over the instances best fits the available data. The cut points are the middle points between the instance values. On the other hand, the choice of the interval that should be split next, among those produced at a given step, is driven by an objective function capturing the significant changes of density in different separated bins.

All the possible cut-points are considered, and a score to each sub-interval is assigned. Given a single interval to split, any of its cut-points produces two bins and thus induces, upon the initial interval, two densities, computed using the simple binning density estimation formula. Every sub-interval produced has an averaged binned density that is different from the density estimated with the kernel function. The less this difference is, the more the sub-interval fits the data well, i.e. the better this binning is, and hence there is no reason to split it.

Hence, at each step of the discretization process, we must choose from different sub-intervals to split. In every sub-interval we identify as candidate cut-points all the middle points between the instances. For each of the candidate cut-points c_i we compute a score as follows:

$$score(T) = \sum_{x_i < c_i} (p(x_i) - f(x_i)) + \sum_{x_i > c_i} (p(x_i) - f(x_i)).$$

The density functions p and f are respectively the kernel density function and the simple binning density function, computed as

$$f(x_i) = \frac{m_i}{wN},$$

where m_i is the number of instances that fall in the bin (left or right) containing x_i; and

$$p(x) = \frac{1}{nw} \sum_{i=1}^{n} K\left(\frac{x - X_i}{h}\right),$$

where we set the bandwidth h to the value of the binwidth w.

3.1.4 The Stopping Criterion

In order to avoid overfitting and to define a stopping criterion, the log-likelihood has been used to evaluate the density estimators. Given a density estimator g and a set of test instances $y_1, \dots, y_n$ the log-likelihood is computed as

$$LL(g|y_1, \dots, y_n) = \sum_{i=1}^{n} \log g(y_i).$$

In order to have an unbiased estimation of how the model fits the real distribution, a cross-validation has been used. In particular, for a histogram having $n_{j_{train}}$ training instances in the I_j interval, let $n_{j_{test}}$ be the number of testing instances falling into the same interval, w the bin width, and N be the total number of training instances. Then, the log-likelihood on the test data is computed by

$$LL = \sum_j n_{j_{test}} \log \frac{n_{j_{train}}}{wN}.$$

3.1.5 Abstraction Rules

After having presented a method to discretize the data, that translates the initial sequence (with real-valued elements) to a segmented sequence made up of symbols taken from a given alphabet, now we can formalize the abstraction rules.

Given a real-valued time series $(t_i, x_i)_{1 \leq i \leq n}$, $x_i \in \mathbb{R}$, the goal is to transform it into a discrete series $(t_i, c_i)_{1 \leq i \leq n}$, $c_i \in \{1, \ldots, C\}$. In the case of a sensor network, made up of n nodes, each node i, located in the environment at the position p_i, senses a set of properties $\mathcal{P}$ at every time instance t. Our approach is to define some abstraction rules useful to shift the basic sensor description language into a more general one. In particular each sensor produces a time series, describing its reading over time, that is then divided into intervals.

Let $\mathcal{C}$ denotes the set of possible properties or descriptive labels, such as "temperature is high". Having a time series $(t_i, x_i)_{1 \leq i \leq n}$, denoted by $(t, x)_{1\_n}$, an abstraction rule is a function $\phi_a((t, x)_{1\_n})$ returning a set of m consecutive intervals of the time series. In particular,

$$\phi_a((t, x)_{1\_n}) = \{\delta_a(l, t_i, t_{i+h}, c_k) | t_j \in I_k^a, i \leq j \leq i + h \wedge c_k \in \mathcal{C}\}_{1 \leq l \leq m}$$

where $\delta(k, t_i, t_{i+h}, c_k)$ denotes an interval starting from t_i and ending to t_{i+h}, and I_k^a represents the domain of values for the function ϕ_a associated to the label $c_k \in \mathcal{C}$ extracted using the discretization process presented below. For instance, for the temperature time series in the wireless sensor network domain we firstly compute its discretization obtaining the following intervals

$$I_t^{vl} = \{x | x < 13\}, \quad I_t^{l} = \{x | 13 \leq x < 22\}, \; I_t^{m} = \{x | 22 \leq x < 31\}$$
$$I_t^{h} = \{x | 31 \leq x < 40\}, \; I_t^{vh} = \{x | x \geq 40\}.$$

Then we define the abstraction function as

$$\phi_t((t, x)_{1\_n}) = \{\delta_t(l, t_i, t_{i+h}, c_k) | t_j \in \mathcal{D}_k^t, c_k \in \mathcal{C}_t\},$$

with labels set to $\mathcal{C}_t = \{$ *very_low*, *low*, *medium*, *high*, *very_high* $\}$.

3.2 Relational Interval Sequences

Now that we have discretized the time series into intervals, we can extend the definitions of both sequences and patterns to the case of interval-based relational sequences.

Definition 5 (Relational Interval Sequence). Given a set $\mathcal{T}$ of time series and the sets $\mathcal{C}_1, \ldots, \mathcal{C}_{|\mathcal{T}|}$ of descriptive labels, a *relational interval sequence* is a sequence of relational atoms

$$\delta_{a_1}(id_1, b_1, e_1, v_1), \delta_{a_2}(id_2, b_2, e_2, v_2), \ldots, \delta_{a_n}(id_n, b_n, e_n, v_n)$$

where $v_j \in \mathcal{C}_i$ is a descriptive label, b_j and e_j represent, respectively, the starting and ending time, $id_j \in \mathbb{N}$ represents the interval identifier, and δ_{a_j} is the corresponding name of the time series $a_j \in \mathcal{T}$. (The interval $\delta(id, b, e, v)$ can be written also by means of three literals as $\delta(id, v)$, *begin*(*id*,*b*), *end*(*id*,e)).

In particular, a relational interval sequence can describe several labeled interval sequences into a single one, enabling one to take into account the multivariate analysis in case of different time series. Relations between time intervals are described adopting the Allen's temporal interval logic [5], as reported in Figure 3.

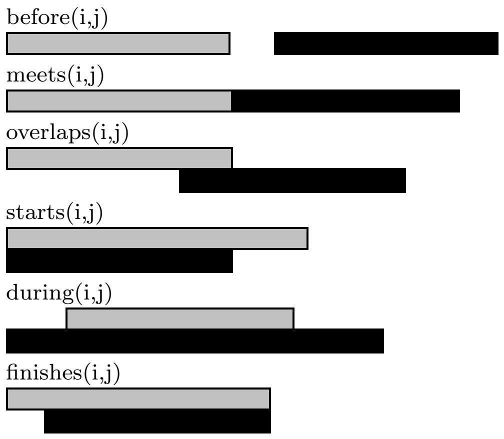

Fig. 3 Allen's temporal intervals [5]

Definition 6 (Relational Interval Pattern). Given $\mathcal{S}$, the set of interval relation symbols, a *relational temporal pattern* is a set of relational atoms

$$P = I \cup R = \{\delta_i(id_i, b_i, e_i, v_i)\}_{i=1\ldots n} \cup \{rel_j(id_j^1, id_j^2)\}_{j=1\ldots m}$$

where $rel_j \in \mathcal{S}$, and $\forall rel_j(id_j^1, id_j^2) \in R\ \exists \delta_h(id_h, b_h, e_h, v_h), \delta_k(id_k, b_k, e_k, v_k) \in I$ such that $id_j^1 = id_h$ and $id_j^2 = id_k$.

4 Related Work

The extraction of useful knowledge from raw sensor data is a difficult task and conventional tools might not be able to handle the massive quantity, the high dimensionality and the distributed nature of the data. For such reasons, in recent years a great interest emerged in the research community in applying data mining techniques to the large volumes of sensor data [1]. Specifically, the exploitation of data mining approaches could be a key issue for summarizing the data into events and for elaborating further adaptive tactical decisions or strategic policy.

Many works are presented in literature concerning the topic of distributed data mining [26], spatial data mining [10], and temporal data mining [27]. In a more general context, there is a growing interest in applying data mining techniques to sensor data [13] that operate mainly on a centralized data set, as we proposed, rather than providing mechanisms for in-network mining.

However, most of the existing techniques operate on an attribute-value descriptions of the (spatio, temporal and spatio-temporal) data and sensors involved in the network adopting a point-based event approach in order to discover useful patterns. On the other hand, very few works face the challenge of discovery temporal patterns using an interval-based approach but in some cases they do not use a relational language [17, 14, 21] and hence they cannot represent interval-based relations, and in other cases, even considering some kind of relations among temporal intervals [25] they are able to represent both intervals and their relations but they cannot completely describe the network, the sensors involved in it and the data (spatio, temporal and spatio-temporal) gathered from the sensors. Furthermore, spatial data mining and similarity-based approaches were designed to tackle into account complex representations with a relational language, however without considering temporal-based relations [8, 18, 12].

The work presented in this chapter can be related to that proposed in [19], optimized in [20]. In these works [19, 20], the authors represent a single sequence as a set of predicates and temporal relations. Each predicate is assumed to be hold in a given temporal interval, while the temporal relations are predicates expressing the Allen's temporal correlation between two predicates. Furthermore, each predicate is associated to a unique symbolic identifier indicating a specific temporal interval, and temporal relations are expressed between those identifiers. Hence, every time a predicate is used in a sequence, it is implicitly assumed that it corresponds to a fluent predicate. In this way, it is not possible to introduce predicates that only express a structural relation between objects, i.e. between sensors or (temporal, spatial) events involved in the network.

Furthermore, as reported in [19], the algorithm they presented is not applicable to real world problems due to its high complexity. Indeed, they specialize a pattern by adding a literal, or by variable unification, or by introducing k^n (where k is the number of different Allen's relation and n corresponds to the number of possible predicate pairs) temporal restrictions between predicate pairs leading to an exponential time complexity.

On the contrary, the framework we presented in this chapter can be used to solve complex temporal data mining tasks by using a relational interval-based description as shown by the outcome obtained by the application of the proposed framework to a real world wireless sensor network data (see Section 5). Furthermore, it is based on a powerful and general purpose multi-dimensional relational pattern mining system [9] and extends it with new dimensional operators thus allowing one to be able to represent and handle spatial (or other dimensional) information gathered form the network. Finally, the framework was extended to automatically provide an interval-based description of the temporal data.

As regards the language used to describe both sequences and patterns, it has some similarities with the Planning Domain Definition Language (PDDL) proposed in [24]. Adopting PDDL as a representation language could make our approach directly applicable to specific planning real world domains.

5 Experiments

In order to evaluate our approach, we used the data, freely available from [15], collected from a wireless sensor network made up of 54 Mica2Dot sensors deployed in the Intel Berkeley Research Lab and arranged in the laboratory as shown in Figure 4.

A sensor network node is a small autonomous unit, often running on batteries, with hardware to sense environmental characteristics, such as temperature, humidity and light. Such nodes usually communicate using a wireless network. A sensor network is composed of a large number of sensors deployed in a natural environment. The sensors gather environmental data and transfer the information to the central base station with external power supply.

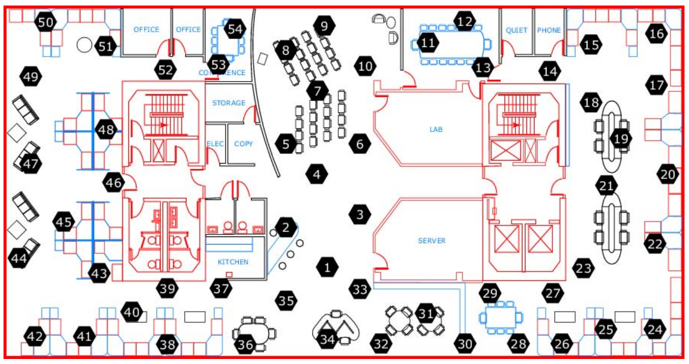

Fig. 4 Sensors in the Intel Berkeley Research lab

The 54 sensors have been monitored from February 28th to April 5th 2004, and the data, about 2.3 million readings, were collected using the TinyDB in-network query processing system, built on the TinyOS platform. Each sensor collected topology information, along with humidity, temperature, light and voltage values once every 31 seconds.

We selected the measurements (temperature, humidity, light and voltage) from the sensors 31, 32, and 34, for the time period from 2004-03-10 to 2004-03-13 corresponding to 16253 log rows. The aim is to discover some correlations between sensors and/or measurements useful for anomaly detection. For instance, there is a strong correlation between the temperature and humidity, as we can see from the Figure 5 that reports the corresponding graphs for the sensor 41. The first task is to discretize the time series corresponding to each information in order to obtain an interval-based temporal sequence, where each interval is labeled with a specific name.

The discretization step has been executed exploiting the functions ϕ_t, ϕ_h, ϕ_l, and ϕ_v with the corresponding domains $\mathcal{I}_i^j$, obtained with the algorithm presented in Section 3, where i is the time series name (temperature, humidity, light and voltage) and j is the descriptive label associated to the interval:

$$
\begin{array}{lll}
\mathcal{I}_t^{t1} = \{x | x < 18.9\}, & \mathcal{I}_t^{t2} = \{x | 18.9 \leq x < 21.3\}, & \mathcal{I}_t^{t3} = \{x | 21.3 \leq x < 25.8\}, \\
\mathcal{I}_t^{t4} = \{x | 25.8 \leq x < 30.6\}, & \mathcal{I}_t^{t5} = \{x | 30.6 \leq x < 31.1\}, & \mathcal{I}_t^{t6} = \{x | x \geq 31.1\} \\
\mathcal{I}_h^{h1} = \{x | x < 32.6\}, & \mathcal{I}_h^{h2} = \{x | 32.6 \leq x < 38.2\}, & \mathcal{I}_h^{h3} = \{x | 38.2 \leq x < 42.9\}, \\
\mathcal{I}_h^{h4} = \{x | 42.9 \leq x < 43.8\}, & \mathcal{I}_h^{h5} = \{x | 43.8 \leq x < 46.3\}, & \mathcal{I}_h^{h6} = \{x | x \geq 46.3\}, \\
\mathcal{I}_l^{l1} = \{x | x < 2.7\}, & \mathcal{I}_l^{l2} = \{x | 2.7 \leq x < 16\}, & \mathcal{I}_l^{l3} = \{x | 16 \leq x < 84.6\}, \\
\mathcal{I}_l^{l4} = \{x | 84.6 \leq x < 176.6\}, & \mathcal{I}_l^{l5} = \{x | x \geq 176.6\}, & \\
\mathcal{I}_v^{v1} = \{x | x < 2.5\}, & \mathcal{I}_v^{v2} = \{x | 2.5 \leq x < 2.6\}, & \mathcal{I}_v^{v3} = \{x | x \geq 2.6\}
\end{array}
$$

Adopting these functions we obtained a temporal sequence made up of 1249 intervals (138 for temperature, 427 for humidity, 117 for light and 612 for voltage). Then we added all the Allen's temporal relations between the intervals (836729 before, 1558 meets, 13714 overlaps, 122 starts, 11945 during, 134 finishes and 60 matches atoms) obtaining a relational sequence of about 868000 literals. The following literals represent a fragment of a sequence describing the relational representation of some time series, where each interval is described by three predicates

α(sensor, interval, label), begin(interval, s), end(interval, e)

where $\alpha \in$ {temperature, humidity, light, voltage}.

```
temperature(31,i1,it3). begin(i1,0). end(i1,22).
humidity(31,i2,ih5). begin(i2,0). end(i2,30).
...
starts(i1,i2). before(i2,i3). ...
```

Table 1 reports the results of the algorithm when applied on the sequence previously described and using two different values for the minimum support. The fourth column reports the number of patters belonging to all the possible

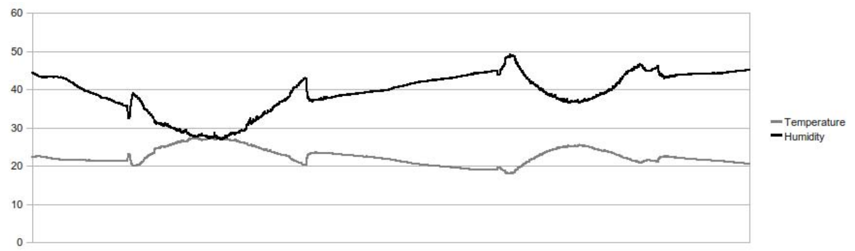

Fig. 5 Correlation between temperature (bottom) and humidity (top) time series

Table 1 Detailed results of the experiments

MinSupport	Level	Specializations	Candidates	Maximals	Time (secs)
10%	1	17	10	307	319.3
	2	166	40		
	3	666	170		
	4	2340	344		
	5	3864	200		
	6	1806	0		
15%	1	17	9	246	277.8
	2	150	34		
	3	568	141		
	4	1882	254		
	5	2784	141		
	6	1272	0		
20%	1	17	9	194	234.7
	2	150	33		
	3	555	122		
	4	1618	206		
	5	2293	55		
	6	494	0		

specializations whose support is greater than MinSupport. The fifth column reports the number of maximal patterns fulfilling all the constraints obtained by the algorithm.

Some interval-based patterns discovered by the algorithm and expressing the time correlation and the information correlation are:

`temperature(_,A,B), before(A,C), temperature(D,C,E),`
`18.95` $\leq$ `B < 21.35, 21.35` $\leq$ `B < 25.85, mote31(D)` $[s = 28.4\%]$,
`temperature(A,B,C), meets(B,D), temperature(A,D,E),`
`18.95` $\leq$ `C < 21.35, 21.35` $\leq$ `E < 25.85` $[s = 24.8\%]$,
`temperature(A,B,C), meets(B,D), temperature(A,D,E),`
`21.35` $\leq$ `C < 25.85, 18.95` $\leq$ `E < 21.35` $[s = 26.2\%]$.

6 Conclusion

In this chapter a relational language useful to describe the temporal nature of a sensor network has been proposed, and a relational learning technique able to discover interesting and more human readable patterns relating spatio-temporal correlations has been implemented.

The framework, already presented in [9] has been extended in order to take into account interval-based temporal data along with contextual information about events occurring in the environment. The extension concerns the introduction of interval-based operators, based on the Allen's temporal interval logic [5], in the sequences. Firstly, an abstraction step with the aim of segmenting and labelling the real-valued time series into similar subsequences is performed exploiting a kernel density estimator approach. The knowledge is enriched by adding interval-based operators between the subsequences obtained in the discretization step, and the relation pattern mining algorithm has been extended in order to deal with these new operators.

In order to evaluate the validity of both the abstraction step and the general extended framework, an experimental session on real world data collected from a wireless sensor network has been presented.

References

1. International Workshop on Knowledge Discovery from Sensor Data (Sensor-KDD) (2007-2008-2009)
2. Agrawal, R., Manilla, H., Srikant, R., Toivonen, H., Verkamo, A.: Fast discovery of association rules. In: Fayyad, U., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R. (eds.) Advances in Knowledge Discovery and Data Mining, pp. 307–328. AAAI Press, Menlo Park (1996)
3. Akyildiz, I., Su, W., Sankarasubramanian, Y., Cayirci, E.: A survey on sensor networks. IEEE Communication Magazine 40(8), 102–114 (2002)
4. Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E.: Wireless sensor networks: a survey. Computer Networks 38, 393–422 (2002)
5. Allen, J.: Maintaining knowledge about temporal intervals. Commun. ACM 26(11), 832–843 (1983)
6. Basile, T.M.A., Mauro, N.D., Ferilli, S., Esposito, F.: Relational temporal data mining for wireless sensor networks. In: Serra, R. (ed.) AI*IA 2009. LNCS, vol. 5883, pp. 416–425. Springer, Heidelberg (2009)
7. Biba, M., Esposito, F., Ferilli, S., Di Mauro, N., Basile, T.: Unsupervised discretization using kernel density estimation. In: Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI 2007), pp. 696–701 (2007)
8. Malerba, D., Lisi, F.: An ILP method for spatial association rule mining. In: Working notes of the First Workshop on Multi-Relational Data Mining, pp. 18–29 (2001)
9. Esposito, F., Di Mauro, N., Basile, T., Ferilli, S.: Multi-dimensional relational sequence mining. Fundamenta Informaticae 89(1), 23–43 (2008)

10. Ester, M., Kriegel, H.P., Sander, J.: Algorithms and applications for spatial data mining, vol. 1(Part 4), ch. 7, pp. 160–187. Taylor and Francis Group, Abington (2001)
11. Estrin, D., Culler, D., Pister, K., Sukhatme, G.: Connecting the physical world with pervasive networks. IEEE Pervasive Computing 1(1), 59–69 (2002)
12. Ferilli, S., Basile, T., Biba, M., Di Mauro, N., Esposito, F.: A general similarity framework for horn clause logic. Fundamenta Informaticae 90(1-2), 43–66 (2009)
13. Ganguly, A.R., Gama, J., Omitaomu, O.A., Gaber, M.M., Vatsavai, R.R.: Knowledge Discovery from Sensor Data. CRC Press, Inc., Boca Raton (2008)
14. Hoppner, F.: Learning dependencies in multivariate time series. In: Proc. of the ECAI Workshop on Knowledge Discovery in (Spatio-)Temporal Data, pp. 25–31 (2002)
15. Intel Berkeley Research Lab, `http://db.csail.mit.edu/labdata/labdata.html`
16. Jacobs, N., Blockeel, H.: From shell logs to shell scripts. In: Rouveirol, C., Sebag, M. (eds.) ILP 2001. LNCS (LNAI), vol. 2157, pp. 80–90. Springer, Heidelberg (2001)
17. Kam, P., Fu, A.W.: Discovering temporal patterns for interval-based events. In: Kambayashi, Y., Mohania, M., Tjoa, A.M. (eds.) DaWaK 2000. LNCS, vol. 1874, pp. 317–326. Springer, Heidelberg (2000)
18. Koperski, K., Han, J.: Discovery of spatial association rules in geographic information databases. In: Egenhofer, M.J., Herring, J.R. (eds.) SSD 1995. LNCS, vol. 951, pp. 47–66. Springer, Heidelberg (1995)
19. Lattner, A., Herzog, O.: Unsupervised learning of sequential patterns. In: ICDM Workshop on Temporal Data Mining: Algorithms, Theory and Applications (2004)
20. Lattner, A., Herzog, O.: Mining temporal patterns from relational data. In: Lernen Wissensentdeckung Adaptivität (LWA), GI Workshops, pp. 184–189 (2005)
21. Laxman, S., Unnikrishnan, K., Sastry, P.: Generalized frequent episodes in event sequences. In: 8th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining, Workshop on Temporal Data Mining (2002)
22. Li, Q., Racine, J.: Nonparametric Econometrics: Theory and Practice. Princeton University Press, Princeton (2007)
23. Mainwaring, A., Culler, D., Polastre, J., Szewczyk, R., Anderson, J.: Wireless sensor networks for habitat monitoring. In: Proceedings of the 1st International Workshop on Wireless sensor networks and applications, pp. 88–97. ACM, New York (2002)
24. McDermott, D., Hove, A., Knoblock, C., Ram, A., Veloso, M., Weld, D., Wilkins, D.: PDDL - The Planning Domain Definition Language. Yale Center for Computational Vision and Control (1998)
25. Papapetrou, P., Kollios, G., Sclaroff, S., Gunopulos, D.: Discovering frequent arrangements of temporal intervals. In: IEEE ICDM, pp. 354–361 (2005)
26. Park, B.H., Kargupta, H.: Distributed Data Mining: Algorithms, Systems, and Applications, pp. 341–358 (2002)
27. Roddick, J.F., Spiliopoulou, M.: A survey of temporal knowledge discovery paradigms and methods. IEEE Transactions on Knowledge and Data Engineering 14(4), 750–767 (2002)
28. Ullman, J.: Principles of Database and Knowledge-Base Systems, vol. I. Computer Science Press, Rockville (1988)