Activity recognition in multiple contexts for smart-house data

Kacper Sokol, Peter Flach

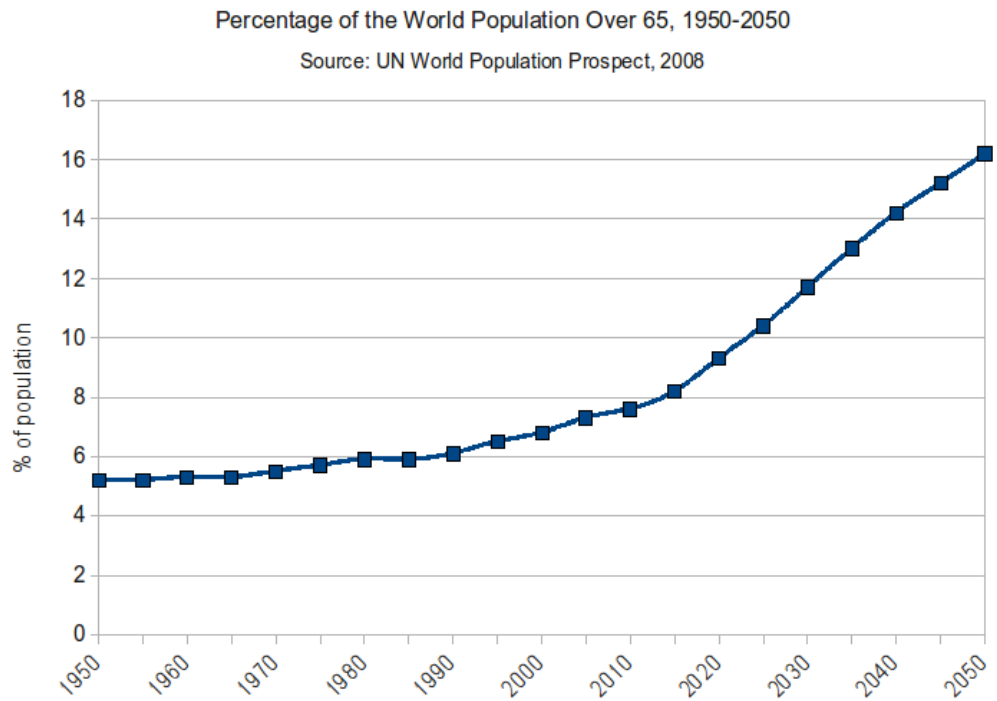
Intelligent Systems Laboratory
University of Bristol

Outline

Healthcare issues

The challenges:

- Global population health issues
- Ageing population
- Increasing healthcare costs
- Decreasing quality of life



SPHERE IRC

The Technology

- Sensors development for smart house applications
- **Use acquired information** to identify medical or well-being issues: predict falls, detect strokes, analyse eating behaviour, and detect periods of depression or anxiety

The Approach

Collaboration of clinicians, engineers, designers and social care professionals as well as members of the public to develop helpful technologies:

- Focus on real-world technologies acceptable in people's homes
- Address real healthcare problems in a cost effective way

The Learning Problem

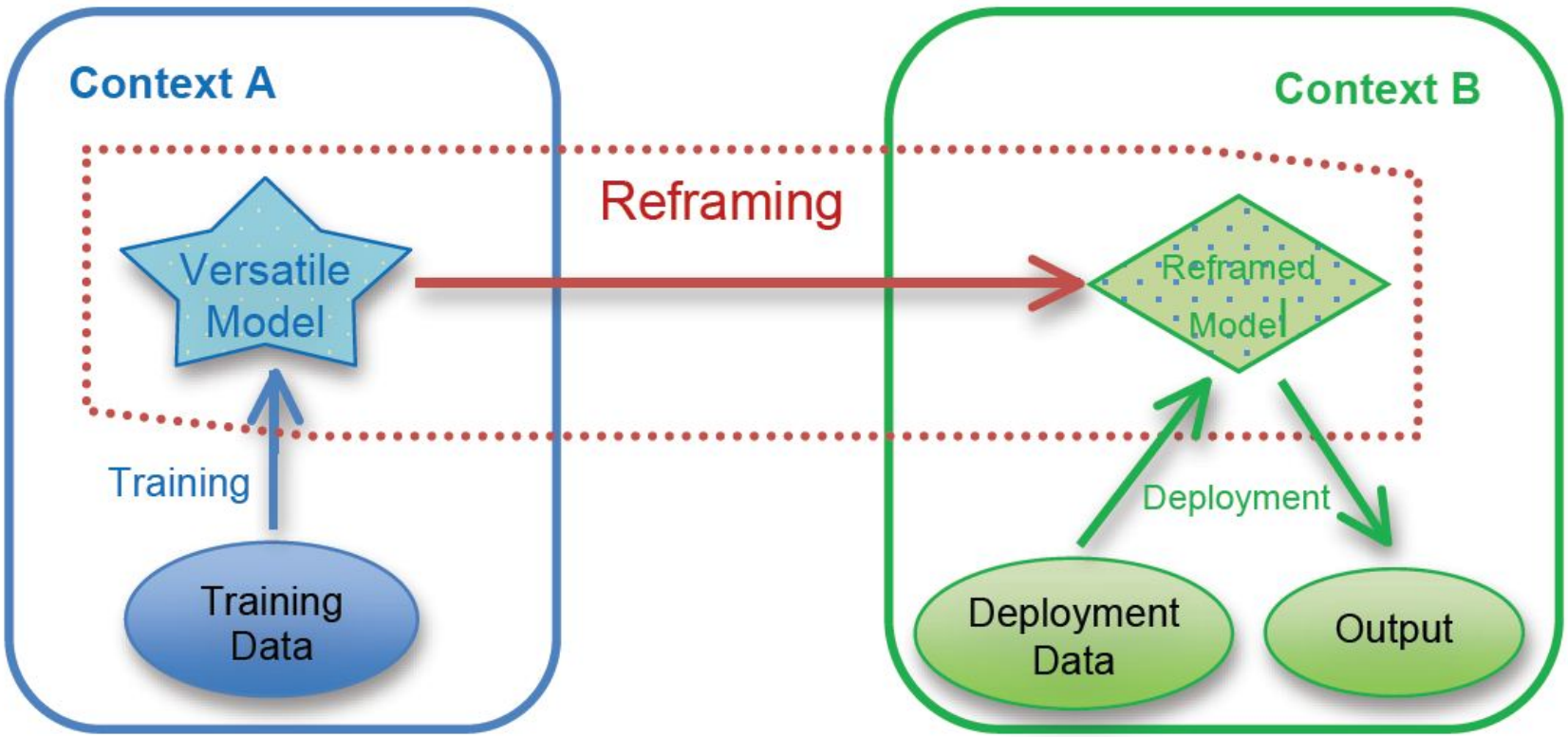
A smart home is an intelligent residential platform which collects and utilises the diverse data generated in this environment (such as sensor data, video and audio streams), the discovery/recognition of activities of daily living is an essential function of a smart home. Most existing work in this area use black-box methods which cannot be inspected by a human expert.

We aim to deploy the SPHERE technology to **100** homes. . . which are different! Therefore black-box methods are of little use.

REFRAME

Develop a set of techniques — broadly called *reframing* — to allow *model reuse* instead of retraining. One of the approaches is to account for a *context variation* in the model.

Each house being different context, the approach allows us to use only *one* model for 100 houses.



Plan

Smart House Sensor Data

Motion sensors, door sensors and light switches, provide binary readings, whereas temperature sensors, battery sensors provide continuous values. In this work we focus on binary sensors.

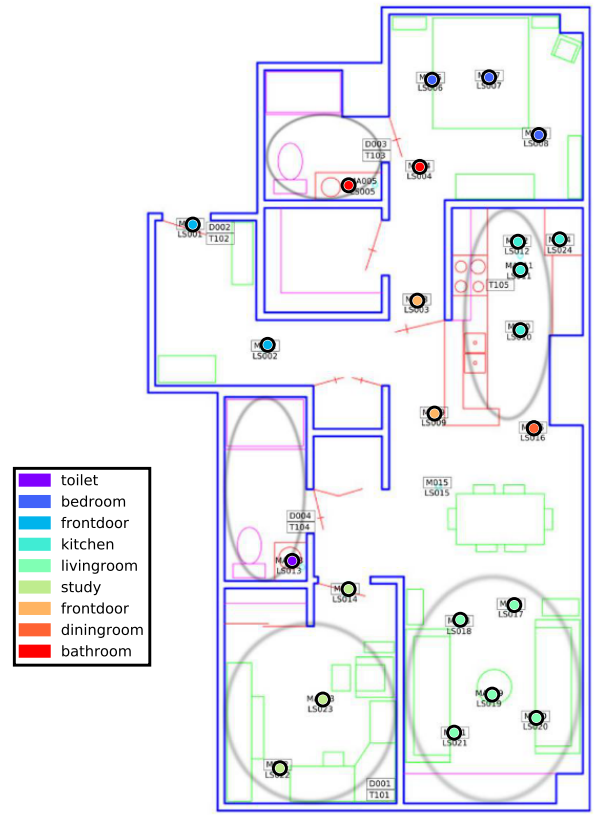
```
...
happensAt(sensor(m06, on) , 1472761057).
happensAt(sensor(m12, on) , 1472718332).
happensAt(sensor(m03, off), 1472719054).
happensAt(sensor(m06, off), 1472719161).
...
```

We want to predict the location of a resident half-an-hour in advance, hence we introduce the following annotations for every time point:

```
...
not_visiting_room_in_the_next_time_block(living_room, 1472761057).
not_visiting_room_in_the_next_time_block(living_room, 1472761058).
not_visiting_room_in_the_next_time_block(living_room, 1472761059).
not_visiting_room_in_the_next_time_block(living_room, 1472761060).
...
not_visiting_room_in_the_next_time_block(living_room, 1472719159).
not_visiting_room_in_the_next_time_block(living_room, 1472719160).
not_visiting_room_in_the_next_time_block(living_room, 1472719161).
...
```

A Smart Home:

Locations and Sensors



Simple Event Calculus

In most of the cases smart house data is in form of a time series consisting of sensor activations. The events are usually not evenly spaced through time as they are caused by a change in sensor state. This property of time in our data causes representational difficulties well known in logic.

Simple Event Calculus is a formalisation of representing actions and their effects in logic. This framework allows for *events* (sensor state changes) occurring in a single time point to activate or terminate *fluents* — properties of system that persist through time. Additionally, event calculus provides an approach to evaluate the model over time and not events what guarantees best possible model for a set of events.

To solve the task we implement *Simple Event Calculus* framework in *Aleph* Inductive Logic Programming system.

Features

Our smart house spatio-temporal data can be characterised by the following features:

- Location-based features:
  - ⋆ being in a given room,
  - ⋆ not being in a given room,
  - ⋆ visiting a room in a sliding time window of fixed length,
  - ⋆ sequence of visited rooms.
- Time-based features:
  - ⋆ date: year, month, day;
  - ⋆ time: hour, minute, second;
  - ⋆ timestamp derivatives: season of the year, day type, week number, week day, time of the day (morning, afternoon, etc.).

Finally, we use some additional features neither related to time nor place like type of day encoding contextual information.

As we are using Event Calculus framework all of our features are represented as *fluents*:

```
holdsAt(sensor(m06, off), 1472761056).
holdsAt(sensor(m06, on) , 1472761057).
...
holdsAt(sensor(m06, on) , 1472719160).
holdsAt(sensor(m06, off), 1472719161).
...
```

Plan

Multiple Contexts

Two different context types to consider: *house-oriented* and *occupier-oriented*. In our work we use the latter where we identify the following working patterns of the resident:

- normal,
- part-time,
- shifts.

We can also observe that these contexts, as well as many others, are non-atomic and can be divided into smaller components; here with daily resolution. Therefore, we have 5 different types of day:

- `working_day`,
- `working_night`,
- `working_morning`,
- `working_afternoon`,
- `free_day`.

Among house-oriented contexts we can name the following:

- house layout,
- geographic location,
- time zone.

Variable Context Time Sequence

The smart house produces data in a form of an observable binary vector $\mathbf{x}_t = (x_{t1}, \ldots, x_{tn})$ where $x_{ti} \in \{0, 1\}$ is a state of a sensor i at time point t for n motion sensors; these create $[\mathbf{x}_1, \ldots, \mathbf{x}_T] \in \mathcal{X}$, the sequence of interest.

This sequence can be divided into a series of consecutive *non-observable* blocks $B = [B_{1m}, \ldots, B_{zT}]$ for $1 < m < z < T$ by a series of time points $[m, \ldots, z]$; and a block B_{ab} is defined as $B_{ab} = [\mathbf{x}_a, \ldots, \mathbf{x}_b]$ where $1 < a < b < T$. Each block can be characterised by some unique label $L_{ab} = L(B_{ab})$ from a finite set of labels $\mathcal{L} = \{\texttt{sleep}, \texttt{work}, \texttt{leisure}\}$.

A sub-sequence of events $[\mathbf{x}_r, \ldots, \mathbf{x}_s]$ for some $1 \leq r < s \leq T$ occurs in an implicit, *non-observable context* $C_{rs} = C([\mathbf{x}_r, \ldots, \mathbf{x}_s])$. We assume that every complete series $[\mathbf{x}_1, \ldots, \mathbf{x}_T]$ is a mixture of several contexts, here $\mathcal{C} = \{\texttt{working\_day}, \texttt{working\_night}, \texttt{working\_morning}, \texttt{working\_afternoon}, \texttt{free\_day}\}$.

We define the binary target variable to be $y_t \in \{0, 1\}$ — where $[y_1, \ldots, y_T] \in \mathcal{Y}$; and y_t indicates not being in a selected room (`living_room`) in the next $t + \delta$ time block for $\delta > 0$, here $\delta = 1$.

With the input space $\mathcal{X}$ as defined above our data are represented as $\mathbf{D} \coloneqq \mathcal{X} \times \mathcal{Y}$ and we want to learn a function $f : \mathcal{X} \to \mathcal{Y}$ describing these data.

Vanilla Model

Vanilla model — $y_{t+1} = \texttt{vanilla\_model}([\mathbf{x}_1, \ldots, \mathbf{x}_T])$ — is one that does not use the contextual information hidden in the data:

```
not_visiting_room_in_the_next_time_block(living_room,A) :-
 holdsAt(in_room(bedroom),A),
 holdsAt(time_block(3),A),
 holdsAt(day_number(1),A).

not_visiting_room_in_the_next_time_block(living_room,A) :-
 holdsAt(not_in_room(living_room),A),
 holdsAt(time_block(7),A).
```

Versatile Model

Versatile model — $y_{t+1} = \texttt{versatile\_model}(C_{tt}, [\mathbf{x}_1, \ldots, \mathbf{x}_T])$ — picks up patterns and parametrises itself based on context:

```
not_visiting_room_in_the_next_time_block(living_room,A) :-
 holdsAt(not_in_room(living_room),A),
 holdsAt(time_block(6),A),
 contex(working_day,A).

not_visiting_room_in_the_next_time_block(living_room,A) :-
 holdsAt(not_in_room(living_room),A),
 holdsAt(time_block(10),A),
 contex(free_day,A).
```

To arrive at the versatile model we need to learn over multiple targets: firstly, the `context`; then, include the learnt context predicates into the background knowledge and learn top level `not_visiting_room_in_the_next_time_block` predicate.

Performance Evaluation

	Rules:	Normal	Shift	Part	Merged	Versatile	Majority
Data:	Normal	**89.29**	72.62	76.19	90.48	*96.43*	75.00
	Shift	80.95	**82.14**	67.86	89.29	*90.48*	66.67
	Part	80.95	77.38	**85.71**	*97.62*	88.10	63.10
	Merged	83.73	77.38	76.59	***92.46***	91.67	68.25
	Shuffled	83.73	77.38	75.40	74.21	*84.76*	68.25

Table: Accuracies (in %) averaged over 3 realisations of given dataset for *vanilla*, *versatile*, and *majority class* approaches. Rows are data and columns are rule sets. Italic numbers indicate best rule set for given dataset and bold figures indicate test set accuracies.

Plan

Scope & Limitations

Our *versatile* model outperforms most of the single classic model; it performs close to the *Merged* model with fixed context order; and performs significantly better for variable context time series with randomised context sequence.

Unfortunately, the solution does not scale well. Given 60k sensor activations a day the learning process is extremely long.

Presented example was developed with a toy dataset which is relatively small and only represents basic features of Variable Context Time Sequence completion problem. The approach has yet to be investigated for artificial real-like data and genuine smart home data.