Unsupervised Learning of Sensor Topologies for Improving Activity Recognition in Smart Environments

Niall Twomey[a], Tom Diethe[a], Ian Craddock[a], Peter Flach[b]

[a]*Department of Electrical and Electronic Engineering, University of Bristol.*
[b]*Department of Computer Science, University of Bristol.*

Abstract

There has been significant recent interest in sensing systems and 'smart environments', with a number of longitudinal studies in this area. Typically the goal of these studies is to develop methods to predict, at any one moment of time, the activity or activities that the resident(s) of the home are engaged in, which may in turn be used for determining normal or abnormal patterns of behaviour (*e.g.* in a health-care setting). Classification algorithms, such as Conditional Random Fields (CRFs), typically consider sensor activations as features but these are often treated as if they were independent, which in general they are not. Our hypothesis is that learning patterns based on combinations of sensors will be more powerful than single sensors alone. The exhaustive approach – to take all possible combinations of sensors and learn classifier weights for each combination – is clearly computationally prohibitive. We show that through the application of signal processing and information-theoretic techniques we can learn about the sensor topology in the home (*i.e.* learn an adjacency matrix) which enables us to determine the combinations of sensors that will be useful for classification ahead of time. As a result we can achieve classification performance better than that of the exhaustive approach, whilst only incurring a small cost in terms of computational resources. We demonstrate our results on several datasets, showing that our method is robust in terms of variations in the layout and the number of residents in the house. Furthermore, we have incorporated the adjacency matrix into the CRF learning framework and have shown that it can improve performance over multiple baselines.

Keywords: Machine Learning, Digital Signal Processing, Smart Homes, Activity Recognition, Activities of Daily Life, Unsupervised Learning, Meta Learning.

1. Introduction

The concept of a 'smart home' is that a number of different sensor technologies may be combined to build a picture of how we live in our homes. This information can then

Email addresses: niall.twomey@bristol.ac.uk (Niall Twomey), tom.diethe@bristol.ac.uk (Tom Diethe), ian.craddock@bristol.ac.uk (Ian Craddock), peter.flach@bristol.ac.uk (Peter Flach)

be used to detect medical or health-care issues. For example, such technology could help by predicting falls, detecting strokes so that help may be summoned (requiring real-time responses), analysing eating behaviour and whether people are taking prescribed medication, or detecting periods of depression and anxiety so that interventions using computer based therapy could be put in place (off-line assessment).

The Sensor Platform for HEalthcare in a Residential Environment (SPHERE) Interdisciplinary Research Collaboration (IRC) has developed a multi-modality sensing platform for collecting data from 100 houses in the Bristol area [1, 2, 3]. The overall architecture, which follows a clustered-sensor approach, is currently installed and running in a real house in Bristol (the SPHERE House). We currently do not have long-term data available from the SPHERE house that is available for public release, so instead focus on publicly available activity recognition and smart-home data [4, 5]. Since 2007, the Centre for Advanced Studies in Adaptive Systems (CASAS) research group has been collecting data from homes with various different sensor layouts and differing numbers of residents (see *e.g.* [6, 7, 8]).

Even though activity recognition is a sequential classification problem, the requirement for models to explicitly model the sequential nature of the data (with *e.g.* Hidden Markov Models (HMMs) or Conditional Random Fields (CRFs)) can be mitigated by automated segmentation of sensor data, assuming that sufficiently expressive features can summarise a wider context. Applying classification algorithms to segmented windows has obtained state-of-the-art performance in activity recognition with accuracy in the region of 80-90% depending on the labelled activities and segmentation algorithm [9, 10, 11, 12, 13, 14, 15]. Attribution of activities to residents is still a challenging task, in particular for bathroom-based activities. Many segmentation algorithms have been investigated, with naïve segmentation incorporating sliding time/size windows over sensor events, and more sophisticated approaches utilising classifiers/rules [16, 17, 18, 19, 20], and practitioners have investigated a wide range of classification models applied, *e.g.* Logistic Regression (LR), Support Vector Machines (SVMs), decision trees [9, 10, 11, 12, 13, 14]. In real-time/online learning settings, structured models provide a natural means of projecting historic context to newly arrived sensor events without having to construct and wait for the arrival of the full data segments. Such approaches make a prediction for each new example and are, in general, harder classification tasks which we investigate here.

A set of sensors that are binary in nature (*e.g.* on/off motion sensors or open/close door sensors), may be represented with logical conjunctions of the sensor states. This is a reasonable feature extraction methodology as complex activities will generally not be identifiable with isolated sensor activations. It is also reasonable to prefer that these conjunctions are derived from sensors that are geographically close, because long distance conjunctions are unlikely to be meaningful. As an example, it would seem unreasonable for the probability of a 'cooking' activity to be increased by the detection of motion in a distant bathroom. In this work we describe methodologies to automatically learn the topology of sensing environments. With these techniques complex features can be extracted from sensor data, and so-called 'spurious correlations' such as the one mentioned above can be eliminated by using the learnt topology as a means of regularising the adjacency matrix. Finally, we believe that adjacency matrices provide a facility for constructing high-level features that are suitable for

transfer learning between smart houses. We will see later that in general, learnt adjacency matrices are densely connected within rooms, but sparsely connected between rooms. Consequently, the adjacency matrix facilitates the extraction of important localised information from specific jurisdictions of the smart home that are only weakly connected to the layout of the residence.

A Radial Basis Function (RBF) kernel can be seen as computing an exhaustive set of sensor conjunctions (up to infinite degree due to the 'kernel trick') [21]. However, kernel methods are a poor choice in this setting due to the streaming nature of the data. If sensor combinations are considered exhaustively during training, classifiers might put weights on spurious sensor combinations caused by residents active in different parts of the house. If there are common patterns of activity in distant regions of the house (such as one resident often working in their bedroom whilst another is watching television in the sitting room), a classifier would be unable to discriminate between these and would learn weights that incorrectly respect this spurious correlation; a clear example of over-fitting. Secondly, during testing, if the same pattern of sensor activation were seen, the classifier would be unable to distinguish which resident(s) are responsible for the observed patterns of activation. This problem is magnified if the labelling or annotation of the data is imperfect (annotation is well-known to be a difficult task in the smart home activity recognition community [22]), meaning that there may be ambiguity both in terms of which resident is active, and in terms of which activity is being performed.

If we have access to the sensor locations within the house, such as may be seen in Figure 1, we could imagine constructing an *adjacency matrix* of neighbouring sensors. Even with perfect knowledge of sensor locations within the house, we argue that it is a non-trivial effort to create a meaningful adjacency matrix automatically or by hand. For example, simply thresholding the Euclidean distance between pairs of sensors would create an adjacency that does not respect walls and other constraints on the movement of individuals throughout the house. Even a hand-crafted adjacency matrix respecting the house layout will likely not be perfect, since, for example, the exact positions of large pieces of furniture will not be known, and indeed these may change over time. There are other confounding factors, for example constructing hand-crafted adjacency matrices is a time consuming process that does not scale well with large sets of residences, and we often have poor information regarding the jurisdiction of sensors in the smart home. Instead we advocate automatic learning of these adjacency matrices directly from the raw sensor measurements as people go about their Activities of Daily Living (ADL). In principle, this topology could also be updated online and should adapt if the configuration of the sensors, the house or furniture layout, or change in residents behaviour.

Once we have a spatial map of the sensor network, we can leverage this in several different ways. First and foremost, we may be able to create features that are based on groups of sensors that are close in terms of the topology of the house. This would mean that we could reduce the risk of spurious correlations such as those mentioned previously. If needed, the spatial map could also allow tracking of the residents, which would allow us to separate the streams of sensor data that are related to each resident. One could also potentially use knowledge of the topology to help design better sensor placements. In this paper we focus

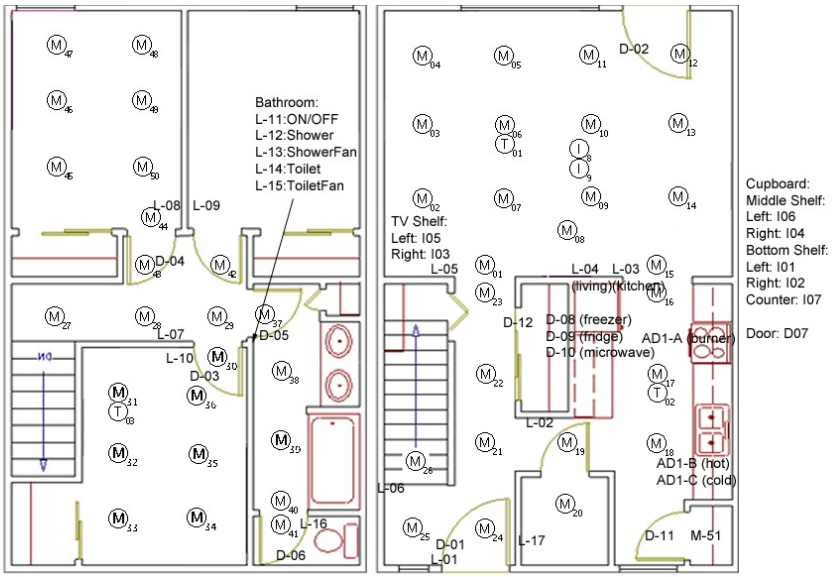


Figure 1: Sensor layout of the CASAS `twor.2009` dataset.

on the first two of these: generating features based on groups of sensors that improves the representation and hence performance on the activity recognition task.

Note that one can in principle take every possible pair, triplet, and higher order conjunctions of sensors (usually referred to as 'N-grams' in the text classification literature), of which the conjunctions found by the adjacency method would be a subset. This would be equivalent to using a polynomial kernel of degree N in a kernel-based classifier [21]. Since the set of possible N-grams at any given point in time is limited to the power-set of size N of observed sensor combinations, this exhaustive approach is actually feasible to compute for reasonable N. However, as we will show, this method is liable to over-fitting due to spurious correlations.

1.1. Our contribution

Specifically, we provide two methods that automatically learn about the topology of the sensor network of three smart environments. We demonstrate that these methods yield adjacency matrices visually appear sensible, and that the approaches generalise over different testbeds. We introduce a new Receiver Operating Characteristic (ROC) [23] based assessment of adjacency matrices, and employ this to numerically evaluate the performance against a hand-constructed adjacency matrix for one of our test scenarios.

We then describe how these can be used to construct restricted feature functions for a CRF classifier. We demonstrate that the use of adjacency matrices can be viewed as a means of regularisation in the classification problem and perform empirical evaluation showing that the classifiers that use adjacency matrices outperform the baseline methods. We show that this approach can assist in learning under ambiguous labelling without having to resort to manually segmenting data, which can greatly reduce the amount of pre-processing time expended by researchers.

In summary, our work demonstrates that adjacency matrices that accurately describe the effective topology of smart environments can be estimated from raw sensor data, and that informing learning routines about this effective topology can improve activity recognition performance.

2. Materials and Methods

We first present details of the datasets being studied, and then describe the feature representation, pre-processing, and activity models that we will be using in our experiments.

2.1. CASAS Datasets

We have analysed datasets from the CASAS research group [1] [6]. We primarily focus on the `twor.2009` dataset because it poses a difficult multi-class and multi-resident problem with a high proportion of the data annotated. The data was collected over a period of approximately three months, during which two residents were living in the apartment. In total, 13 activities are labelled, eight of which are personalised to specific residents.

The testbed floor-plan and sensor layout are shown in Figure 1. This testbed consists of six families of sensor: 1) 51 motion sensors; 2) 9 door sensors; 3) 7 light switch sensors; 4) 2 water flow sensors; 5) 1 stove-top burner sensor; and 6) 1 item sensor. The motion sensors are Passive Infra-Red (PIR) sensors that are distributed evenly throughout the testbed, and these are dispersed at distances of approximately two metres from one another. Door sensors can be found at the entrances to the house, on the doors to rooms, and on the cupboards in the kitchen and wardrobes in the bedrooms. These sensors will trigger as residents perform their ADLs.

An example of the data captured is shown in Table 1 which presents four columns that describe the events as they occurred in the datasets 1) the time-stamp; 2) the sensor identifier; 3) the sensor state; and 4) the change in activity label (if appropriate). Each row in the table is collected when the resident has interacted with the sensor, and as such the data are atomic and asynchronous.

One of the long-term objectives of our research involves the wide-scale deployment of sensor networks in residential environments [24]. As such, we would prefer to learn activity recognition models with minimal pre-processing efforts. Many datasets provide raw data files in the format of Table 1. However, this representation can lead to somewhat ambiguous sensor-activity associations, *e.g.* does the sensor activation 'm33 on' from Table 1 belong to the `R1_Work` activity or to the `R2_Sleep` activity? In order to help classification models answer these kinds of questions, small samples of pre-segmented activities are sometimes provided alongside the raw data files (as is the case with many of the CASAS datasets), where the interwoven activities `R1_Work` and `R2_Sleep` have been separated by hand. One of the objectives and contributions of this paper, however, is that our techniques can learn about the topology of the home, which when used to inform classification models, can automatically achieve this segmentation which in turn can reduce the burden of hand-segmenting these datasets.

[1] `http://ailab.wsu.edu/casas/datasets/`

Table 1: Example of a sensor event file.

Date/Time	Sensor ID	State	Activity Label
...	...	...	...
2009-02-02 07:15:16	m16	on	R1_Work begin
2009-02-02 07:15:22	m14	off	
2009-02-02 07:15:23	m31	on	R2_Sleep begin
2009-02-02 07:15:28	m33	on	
...	...	...	...
2009-02-02 07:20:55	m14	on	
2009-02-02 07:20:56	m30	off	R2_Sleep end
2009-02-02 07:21:00	m31	on	
2009-02-02 07:21:03	m15	off	R1_Work end
...	...	...	...

Table 2: Steady-state and state-change representations.

	Steady-state				State-change			
Raw data	m01	m02	m03	m04	m01	m02	m03	m04
m01 on	1	1	1	0	1	0	0	0
m02 off	1	0	1	0	0	-1	0	0
m03 off	1	0	0	0	0	0	-1	0
m04 on	1	0	0	1	0	0	0	1
m02 on	1	1	0	1	0	1	0	0
m01 off	0	1	0	1	-1	0	0	0
m03 on	0	1	1	1	0	0	1	0

2.1.1. Feature Representation

A number of different representations have been used by the CASAS research group in the past. We focus on two specific representations in this paper which we denote as the steady-state and state-change representations. Both representations result in a matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$ where m is the number of examples, and n is the dimensionality of the data.

The construction of the steady-state representation relies on the fact that when sensor i turns 'on', its value is set to 1 and remains at this value until the 'off' state occurs (for doors, 1 represents 'open' and 0 represents 'closed'). Correspondingly, with the state change representation a value of 1 is set only at the instant the sensor has turned on, a value of -1 is given only when the sensor turns off, and values of 0 indicate no change in the sensor's state. Examples of both representations are shown in Table 2.

We furthermore encode the day and time into the feature vector by introducing supplementary Boolean columns into the matrix $\mathbf{X}$. Seven columns are introduced to represent the

day of the week, and 24 columns are introduced to identify the hour of day. We note that introduction of such columns is largely arbitrary and likely to be a sup-optimal means of modelling periodic context of sensors and activities. We draw attention to the potential use circular statistics for modelling time of sensor activations and activities for activity recognition problems [25]. The dimensionality of the matrix $\mathbf{X}$ grows quickly with the inclusion of a greater number of sensors. However, we can achieve very good compression by encoding the matrix in sparse matrix formats.

2.1.2. Pre-processing

Note that not all sensors behave in the same way: for example a motion sensor will tend to provide an 'on' signal when motion is detected, and then only provide an 'off' signal if motion is no longer detected (and hence can be 'on' for quite some time) whereas a door sensor has 'open' and 'close' states that remain as they are until the next activation.

In all experiments, we convert the raw data into an absolute time format, where we consider a time slice to contain an event if the time-stamp is within a time window of one second. We chose the time window as this seems to be a reasonable compromise between aggregation power and the decision horizon (a longer time window would mean that a training/testing example would contain more discrete activations, but it would also mean that decisions would be delayed until that time had passed).

There were a number of issues in the data. For example sometimes a sensor would be reported as turning 'on' when the last seen state was also 'on' (or 'off' when last seen as 'off'), and in some cases sensors would never turn 'off'. In such cases it was deemed that the sensor would have returned to its 'off' state after 40 seconds. We selected this value with reference to the distribution of the length of activations over the dataset as much less than 5% of the sensor activations lasted over 40 seconds.

We also treat the different families of sensors differently. For example, we use the steady-state representation for the motion sensors, whereas we use the state-change representation for the door sensors. We argue that the steady-state value of a door is not always meaningful (one might equally well study or sleep with the door open or closed), but we believe the state change of a door sensor is significantly more meaningful owing to the fact that a resident was explicitly required to interact with the sensor at that time.

The resulting steady-state matrix is a sparse matrix with 7,862,131 rows and 65 columns with 2,904,702 nonzero elements (99.4% sparse). We also used a sensor activation matrix (the positive version of the state-change matrix described in Table 2 which has 7,862,130 rows and 65 columns with 440,416 nonzero elements (99.9% sparse).

2.2. Activity Model

We will see later how we wish to enhance the feature representation by automatically learning feature functions that respect the topological adjacencies of the home environment. Therefore, rather than evaluating a number of possible classification methods, we focus on the utility gained by incorporating these methods within the CRF model as previous work [26, 18, 27, 28, 29] has shown the utility of CRF for classification of activities of daily living.

CRFs are a structured classification model that learn the conditional distribution of label sequences. We restrict ourselves to linear-chain CRFs here. Let us formalise our notation by defining a training example, $\mathbf{x} \in \mathbb{R}^{m \times n}$, as a sequence of m sensor events ($\mathbf{x} = \{\mathbf{x}_i\}_{i=1}^{m}$), and $\mathbf{y} \in \mathcal{Y}$ to be the corresponding sequence of associated labels where $\mathcal{Y}$ is the set of labels. Each $\mathbf{x}_i$ consists of n attributes and $\mathbf{x}_{i,j}$ describes the j^{th} attribute of the i^{th} example. We will sometimes adopt the following notation to select, for example, motion sensor m16 explicitly: $\mathbf{x}_{i,\text{m16}}$.

The general model of the linear chain CRF is defined by

$$p\left(\mathbf{y}|\mathbf{x}\right) = \frac{1}{Z(\mathbf{x})} \exp\left\{\sum_{j=1}^{N} \mathrm{F}_j(\mathbf{x}, \mathbf{y})\right\}, \tag{1}$$

where

$$Z(\mathbf{x}) = \sum_{\mathbf{y}' \in \mathcal{Y}} \exp\left\{\sum_{j=1}^{N} \mathrm{F}_j(\mathbf{x}, \mathbf{y}')\right\}, \tag{2}$$

is termed the partition function and allows the CRF predictions to follow a true probability distribution by summing over all combinations of $\mathbf{y}'$, and

$$\mathrm{F}_j\left(\mathbf{x}, \mathbf{y}\right) = \sum_{i=1}^{M} \lambda_j f_j\left(\mathbf{y}_{i-1}, \mathbf{y}_i, \mathbf{x}, i\right) \tag{3}$$

is the weighted sum of feature functions, f_i, of the sequence. Feature functions define a template of tests conditioned on label hypotheses that slide over the sequences. Each feature function takes as arguments the previous and current labels ($\mathbf{y}_{i-1}$, $\mathbf{y}_i$), the full sequence of observations ($\mathbf{x}$) and the current position of the sequence (i). The final parameter informs the model of the index of the current in the label being classified and is required as the CRF may utilise features from the entire sequence of observation.

In many applications the output of these feature functions is binary, although this is a limitation of CRFs models. An example of a feature function for activity recognition in the `twor.2009` dataset may be ($u, v \in \mathcal{Y}$):

$$f_j\left(u, v, \mathbf{x}, i\right) = \begin{cases} 1 & \text{if } u = v = \texttt{cook} \wedge \mathbf{x}_{i,\text{m16}} = \text{`on'} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

This feature function returns 1 if there is currently motion in the kitchen and if the previous and current labels are both `cook`. The feature functions can be more expressive than sensor activations taken in isolation, as they may consist of a conjunction of tests that span multiple time steps and multiple attributes. A natural question to ask is how to determine these feature functions for activity recognition and we discuss this and similar matters in the next section.

Each feature function has a weight, λ_j, associated with it and these weights are learnt

through optimisation procedures such as Stochastic Gradient Descent (SGD) or other quasi-Newtonian methods [30]. We have employed an adaptive sub-gradient algorithm "ADA-Grad" [31] to optimise our CRF models as this algorithm provides strong convergence guarantees, converges very quickly and is appropriate for online estimation, in particular for sparse data. The partial derivative of the likelihood function, $\mathcal{L}(\mathcal{D})$, is taken with respect to the j^{th} attribute which yields

$$\frac{\delta}{\delta\lambda_j}\mathcal{L}(\mathcal{D}) = \widehat{\mathbb{E}}[f_j] - \mathbb{E}[f_j], \tag{5}$$

where $\widehat{\mathbb{E}}[f_j]$ is the expected value under the empirical distribution of the feature f_j, and $\mathbb{E}[f_j]$ is the expected value under the model distribution of f_j [32]. Equation 5 will equal zero when learning has converged, and so the CRF will have learnt a model in which $\widehat{\mathbb{E}}[f_j] = \mathbb{E}[f_j]$. This implies that the model has been calibrated against the training data.

2.3. Learning Feature Functions

2.3.1. Background

Our classification models, as a preliminary step, exploit the freedom permitted by CRFs and enhance the feature space from the steady-state representation described earlier. This is achieved by decomposing the feature functions into a product of two separate feature functions, one of which depends on $\mathbf{x}$ and the other on $\mathbf{y}$:

$$f_j(\mathbf{y}_{i-1}, \mathbf{y}_i, \mathbf{x}, i) = f_{x,j}(\mathbf{x}, i) \times f_{y,j}(\mathbf{y}_{i-1}, \mathbf{y}_i, i) \tag{6}$$

Here $f_{x,j}$ and $f_{y,j}$ are the functions which only consider the observations and the labels respectively. As an example, the feature function shown previously in Equation 4 can be represented as

$$f_{x,j}(\mathbf{x}, i) = \begin{cases} 1 & \text{if } \mathbf{x}_{i,\text{m16}} = \text{`on'} \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

$$f_{y,j}(u, v, i) = \begin{cases} 1 & \text{if } u = v = \texttt{cook} \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

Using the steady-state feature representation, sets of sensors may be concurrently active. Our feature generation method first selects the activated elements of $\mathbf{x}_i$ ($s = \{j | \mathbf{x}_{i,j} = \text{`on'}\}$) and then enumerates over the possible conjunctions of activated sensors. This yields a set of feature functions that depend on $\mathbf{x}$. These $f_{x,j}$ are then generalised over pairs of tags ($u, v \in \mathcal{Y}$) as these examples occur in the training data. Together, these create f_j.

To enumerate the conjunctions of the set of activated sensors, s, we compute its power-set (2^s). For example, if $s = \{\text{m15}, \text{m16}\}$, the set of features considered by our learning algorithm

are

$$2^{\{\mathrm{m15,m16}\}} = \{\{\}, \{\mathrm{m15}\}, \{\mathrm{m16}\}, \{\mathrm{m15}, \mathrm{m16}\}\}. \tag{9}$$

Each individual element in this power-set has an associated weight whose value is optimised in training. Note also that we include the empty set in our enriched representation. Feature functions must dependent on at least one $\mathbf{y}$ in order to contribute to inference, but no such requirements are imposed on $\mathbf{x}$. Therefore, we can consider the weights associated with the empty set of features as 'bias' terms. The utility of this approach is furthermore strengthened by considering the day and time values (in this case d1 and h1 respectively):

$$2^{\{\mathrm{h1,d1,m11}\}} = \{\{\}, \{\mathrm{h1}\}, \{\mathrm{d1}\}, \cdots, \{\mathrm{h1}, \mathrm{d1}, \mathrm{m11}\}\}. \tag{10}$$

With this representation we obtain a rich set of features that describe the activity comprehensively. Time dependence is partially encoded using the original set of activated sensors s. Our representation, by virtue of constructing features that explicitly consider time-based parameters with every sensor event, should learn a set of specialised classification weights that improve classification performance. Indeed, because we can view our enhanced feature functions as logical tests, our approach corresponds to utilising a polynomial kernel on the steady-state feature representation [21, 33]. However, due to our sparse input data, feature function indexing scheme and for reasons discussed later, we will instead call this approach the power-set method.

For a given set of activated sensors, s, its power-set will consist of $2^{|s|}$ elements and so its feature representation grows exponentially with respect to $|s|$. In general, given our sparse dataset, only a small number of sensors are activated together. Occasionally, however, more than ten sensors may be active at the same time. Computing the exhaustive power-set and learning weight parameters for all possible feature combinations becomes prohibitively time consuming as this can result in a maximum of 2^N elements ($N > 51$ for the `twor.2009` dataset once the date and time features are incorporated). Therefore, we introduce a cardinality-restricted power-set, $\mathcal{P}_c\left(2^s\right)$, in which elements whose cardinality is greater than c are removed. This can be interpreted as setting the degree parameter in polynomial kernel methods.

$$\mathcal{P}_c\left(2^s\right) = 2^s, \text{ s.t. } |z| \leq c, \ \forall \ z \in 2^s \tag{11}$$

2.3.2. Imposing Topological Constraints on Feature Functions

So far we have described how we enhance our feature representation, but the naïve implementation of this may learn spurious associations in the data. Consider an example where two residents are active in the smart home: one resident is in his bedroom working and the other is in the kitchen preparing a meal. While this combination may occur frequently in the dataset we would prefer that our model would only construct conjunctions within localised

ALGORITHM 1: Split a sequence of sensor activations into parallel and contiguous streams.

Input: Input, $\mathbf{x}$; adjacency matrix, $\mathbf{A}$; cardinality, c.
Output: Contiguous sensor activations, `sequences`.

```
sequences ← [ ]
while x is not empty do
    sequence ← [ ]
    seed ← first element of x
    N ← A_seed                                  // The neighbourhood N around seed.
    for i ← 1 to |x| do
        s ← {}
        repeat
            v ← x_i ∩ N          // Select the sensors that intersect with N.
            if |v| ≠ 0 then
                N ← N ∪ A_v      // Grow N to accommodate the intersection.
                s ← x ∪ v                       // Accumulate sensors within N.
                remove v from x_i
        until |x_i ∩ N| = 0
        sequence ← [sequence, P_c(2^s)]
        N ← A_{v_i} ∀ v_i ∈ x
    sequences ← [sequences, sequence]
return sequences
```

regions of concentrated activity (*i.e.* we would prefer two parallel streams of independent sensor activations here: one for the bedroom only, and the second for the kitchen only). The reason for this preference is that, in general, we do not believe that sensor activations within the kitchen should necessarily influence the classifier's belief about activities being performed in the bedroom. Defining ground truth functional regions in the smart home has been used by a number of researchers [34, 35] in the smart environments, but to our knowledge these have not been explicitly to determine sensor adjacency from data.

We present our partitioning and tracking method in algorithm 1. This algorithm assumes that an adjacency matrix, $\mathbf{A}$, is available and we describe two methods that can derive this in the next section. We start by randomly selecting a sensor activation, x, from the first row of the sequence $\mathbf{x}$ (*i.e.* $x \subset \mathbf{x}_1$). Then, by querying the adjacency matrix, the sensors which surround x are added to a set $\mathcal{N}_i$. The purpose of $\mathcal{N}_i$ is to define a neighbourhood of sensors that surround a region of localised activity at a particular time point. We then remove x from $\mathbf{x}_1$, and repeatedly select the remaining components of $\mathbf{x}_1$ that intersect with $\mathcal{N}_i$. This algorithm will, by using the adjacency matrix, separate out distant (in terms of adjacency) sequences of parallel sensor activations in the smart home. The principal utility of this is that when multiple residents are performing activities concurrently, the sensor activations in the kitchen, for example, can be filtered so as not to affect the predicted activities in the bedroom. As this algorithm reduces the effect of irrelevant sensors in prediction, we view

our approach as a form of graph-based regularisation.

3. Unsupervised Learning of Adjacency Matrices

In this section we describe two methods for automatically constructing adjacency matrices – one based on Cross Correlation, and the other based on Mutual Information between pairs of sensors. These are two of many possible methods that have been chosen from signal processing and information theory respectively, but as we will see both methods are able to achieve good performance.

3.1. Cross Correlation based adjacency

Cross Correlation (XC) is a measure of similarity of two waveforms. By permitting a time lag, l, between the signals, the cross correlation for lag l between two signals X and Y can be computed with

$$\begin{aligned} C_{X,Y,l} &= \frac{\mathbb{E}\left[X_i - \mu_X\right]}{\sigma_X} \frac{\mathbb{E}\left[Y_{i+l} - \mu_Y\right]}{\sigma_Y} \\ &= \frac{\mathbb{E}\left[(X_i - \mu_X)(Y_{i+l} - \mu_Y)\right]}{\sigma_X \sigma_Y}. \end{aligned} \tag{12}$$

where $\mu_X = \mathbb{E}[X]$, $\sigma_X = \sqrt{\mathbb{E}\left[(X - \mu_X)^2\right]}$ and similarly for Y.

3.1.1. Cross Correlation for sparse binary/ternary data

Assume now that our sensor readings are arranged in a matrix $\mathbf{X} \in \mathbb{R}^{m \times n_x}$ with the m readings in rows and n_x sensors in columns. We are then correlating this with another matrix $\mathbf{Y} \in \mathbb{R}^{m \times n_y}$ with the same number of readings but a possibly different number of sensors. In our case, $\mathbf{Y}$ will be a time-lagged version of $\mathbf{X}$ so $n_x = n_y$. We further consider two cases: that the matrix consists of the sensor 'on' events only and hence is binary $(0, 1)$; the matrix consists of sensor 'on' and 'off' events and hence is ternary $(-1, 0, 1)$.

The naïve computation of correlation between two signals requires that first the mean is subtracted from each signal, and then the inner product is normalised by the product of the standard deviations. For sparse binary data and ternary data, these computations would be extremely slow, since removing the mean would make the data matrices dense, and then the computation time would be $O(mn_xn_y)$.

Defining $\mathbf{s}_x = \mathbf{1}'_m\mathbf{X}, \boldsymbol{\mu}_x = \frac{\mathbf{s}_x}{m}, \boldsymbol{\mu}_y = \frac{\mathbf{s}_y}{m}$ where $\mathbf{1}_m$ is the vector of all ones of length m, we can compute the numerator simply as:

$$\mathbf{N} = \mathbf{X}'\mathbf{Y} - \left(\boldsymbol{\mu}'_x\boldsymbol{\mu}_y\right) \tag{13}$$

If we now define $\odot$ as the Hadamard product between two vectors or matrices, $\mathbf{Z}_x = \sqrt{|\mathbf{X}| - (\boldsymbol{\mu}_x \odot \boldsymbol{\mu}_x)}$ and $\mathbf{Z}_y = \sqrt{|\mathbf{Y}| - \left(\boldsymbol{\mu}_y \odot \boldsymbol{\mu}_y\right)}$, then we can compute the denominator as:

$$\mathbf{D} = m\mathbf{Z}'_x\mathbf{Z}_y \tag{14}$$

Note here that by taking the absolute values of the matrices $\mathbf{X}$ and $\mathbf{Y}$, the computation of the standard deviation will be correct for both binary and ternary data.

And finally the correlation is given by $\mathbf{C} = \mathbf{N} \oslash \mathbf{D}$ where $\oslash$ is the element-wise division operator, *i.e.* (for vectors) $\mathbf{x} \oslash \mathbf{y} \equiv (x_1, \ldots, x_n) \odot (\frac{1}{y_1}, \ldots, \frac{1}{y_n})$. For our sparse data the complexity of this method is $O(nnz(\mathbf{X})nnz(\mathbf{Y})$ where $nnz(\cdot)$ is the number of nonzero elements in the matrix. Since the sensor data is extremely sparse after converting into the time base (typically $\sim 99.9\%$ sparse for the 'sensor on' events and $\sim 99.8\%$ sparse for the on and off events), this represents a significant speedup. Note also that a fast update scheme could be created since the new denominators will scarcely differ from the old ones and so, to a high degree of accuracy, need not be recomputed at all, and the numerator update is simply done by adding a row to $\mathbf{X}$ and $\mathbf{Y}$. For the purposes of this study the computations were fast enough since computing the full Cross Correlation on the entire `twor.2009` dataset on a standard desktop PC (Intel i7 processor) took less than two seconds.

3.1.2. Creating the adjacency matrix from the Cross Correlations

Consider two adjacent functioning motion sensors. If an individual were to pass them back and forth at a constant velocity, and the motion sensors were trigger at an exact distance[2], we would find that the XC between these sensors would be equal for positive and negative lags. Since for the purpose of determining adjacency we do not actually care which direction the person is moving in, we can treat positive and negative lags equally. However, we argue that the larger the lag, the less likely the correlation is a 'true' correlation, in the sense that it is caused by the same person moving past the sensors.

Hence we now have a set of correlation coefficients at different lag values from $-L : L$ where we define L as the maximum permitted lag, giving a total of $2L + 1$ lags. For each pair of sensors, there are various ways of summarising this information, such as:

- The maximum: $\hat{C}_{X,Y} = \max_l C_{X,Y,l}$
- The absolute maximum: $\hat{C}_{X,Y} = \max_l |C_{X,Y,l}|$
- The mean: $\hat{C}_{X,Y} = \frac{1}{L} \sum_l C_{X,Y,l}$
- The median: $\hat{C}_{X,Y} = \text{med}_l\, C_{X,Y,l}$
- Weighted sum: $\hat{C}_{X,Y} = \sum_l w(l) C_{X,Y,l}$ for some weighting function $w(l) : \mathbb{R} \to \mathbb{R}$

We evaluated each of these selection methods, as well as the individual lags separately. For the weighted sum, we used the following weighting function:

$$w(l) = \exp(-\lambda |l|), \quad l = -L : L. \tag{15}$$

This increases the importance of correlations found at shorter lags exponentially over both directions in time. Since time has been discretised to one second, a value of $\lambda = 1$ gives a

[2] although this assumption is violated, in practice the variations should average out

reasonable shape to the weighting function (the effect of lags diminishes to almost zero at $l = \pm 5$). This will be referred to henceforth as the Weighted XC (WXC) method. We found the WXC method to be the most promising way of summarising the lagged XC information, and as such will only report results using this method.

Finally, the adjacency is formed using function $A_{i,j} = u_\beta(Ci, j)$ where $u_\beta(\cdot) : \mathbb{R}^{n\times n} \to \{0, 1\}^{n\times n}$ is the Heaviside step function [36] for a given threshold β. It remains to determine the optimal threshold β for a given dataset. This depends on the density of sensor nodes, as well as the distance between the nodes: for a densely populated sensor layout, one would expect a more densely connected adjacency matrix. We use the heuristic that on average a sensor node will be connected to itself and a few other neighbours, and hence choose the first β that results in an an average graphical connectivity of less than five. Empirically, this was sufficient on the three datasets that we tested on (see subsection 4.1) which are of differing natures, so this heuristic seems reasonable. Note also that if we plot the proportion of rows in the steady-state representation of the `twor.2009` dataset as a function of the number of simultaneous activations, we see that most events involve 4 or fewer sensors, which gives further justification to this method.

3.2. Mutual Information Based Adjacency Learning

Information theory is the field which quantifies the information content, or entropy, of random variables. The entropy, H, of a discrete random variable, X, is calculated with the following equation:

$$H(X) = -\sum_{x\in X} P(x) \log_2 P(x). \tag{16}$$

The most frequent interpretation of this quantity is that it estimates the number of bits required, on average, to encode the values of X into a message. As this value is derived from probabilistic quantities, it is possible to also compute the joint and conditional entropy values of two random variables, X and Y, respectively with

$$H(X, Y) = -\sum_{x\in X}\sum_{y\in Y} P(x, y) \log_2 P(x, y), \tag{17}$$

$$H(X|Y) = -\sum_{x\in X}\sum_{y\in Y} P(x, y) \log_2 \frac{P(x, y)}{P(y)}. \tag{18}$$

The quantities from Equations 16, 17 and 18 can be visualised in the form of the information diagram shown in Figure 2. The shared mutual dependence between two discrete random variables, denoted as $I(X; Y)$, is shown in Figure 2 and is known as Mutual Information (MI). This can be calculated with the identities from Equations 17 and 18 giving the following equation

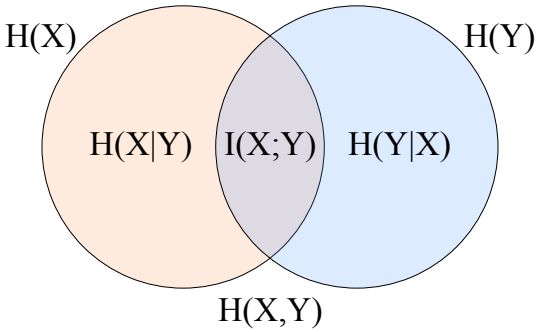


Figure 2: Venn diagram of information theoretic quantities for two dependent random variables X and Y.

$$I\left(X;Y\right) = \sum_{x \in X} \sum_{y \in Y} P(x,y) \log_2 \frac{P(x,y)}{P(x)P(y)} \tag{19}$$

which can be decomposed to a number of equivalent formulations including

$$I\left(X;Y\right) = \mathrm{KL}\left(P\left(X,Y\right) \| P\left(X\right) P\left(Y\right)\right) \tag{20}$$

where KL is the Kullback-Leibler divergence [37] between two probability distributions. With this interpretation, we can understand MI as being a measure of the similarity of the probability distributions $P(X,Y)$ and $P(X)P(Y)$, *i.e.* a measure of the extent to which X and Y are dependent. MI is a non-negative metric. Larger values indicate more pronounced mutual dependence between the random variables, and MI will be zero if and only if there is no shared information between the two random variables. MI is also a symmetric measurement because $I\left(X;Y\right) \equiv I\left(Y;X\right)$.

The intuition behind using MI for learning adjacency matrices is that, using the steady-state sensor representation, neighbouring sensors should depict similar activation patterns, *i.e.* the sensors should both be 'on' and 'off' at similar times; in effect we hypothesise that the state of one sensor should inform us about the states of its neighbours. Therefore, one might expect neighbouring sensors to yield a larger MI than that calculated for non-adjacent sensors.

Given a set of sensors, $\mathcal{S}$, the pairwise MI between all pairs of elements in this set is computed according to Equation 19. The computed values are used to populate the matrix $\mathbf{A} \in \mathbb{R}_{+}^{M \times M}$ in which $\mathbf{A}_{i,j} = I\left(\mathcal{S}_i;\mathcal{S}_j\right)$. The values taken by the elements of this matrix are positive but are not subject to an intrinsic upper bound. Intuitively, the larger the value the more information is shared between the sensors.

3.2.1. Adjacency Threshold Selection

Previously we stated our preference that only adjacent sensors contribute in the decision making process, and in order to accommodate this, a function which performs a threshold is

constructed $t : \mathbb{R}^{M\times M} \rightarrow \{0,1\}^{M\times M}$. We term $\mathbf{A}$ as the soft adjacency matrix from the MI method and $t(\mathbf{A})$ is its associated hard adjacency matrix. Note that the method outlined for the WXC adjacency methods is not appropriate here since the elements of the WXC matrix are normalised and bounded (between $+1$ and -1) whereas those from the MI are not.

Our threshold selection method exploits the understanding that, on average, pairs of sensors will be non-adjacent. A naïve threshold selection method may consider the mean of the whole matrix, but this may not elicit a viable threshold because the magnitude of the elements depends strongly on how many of their instances were observed. Therefore, our threshold selection method derives a threshold matrix $\mathbf{T} \in \mathbb{R}^{M\times M}$ that accommodates the relative range of values in soft adjacency matrices. We first compute the vector of expected values along the rows and columns of the soft adjacency matrix, *i.e.* $\mathbb{E}[\mathbf{A}_{i,:}]$ and $\mathbb{E}[\mathbf{A}_{:,j}]$, *i.e.* the average information shared between a sensor and $\mathcal{S}$. The elements of this vector represent the expected MI between the specific sensors and the remaining sensors. To determine whether sensors $\mathcal{S}_i$ and $\mathcal{S}_j$ are adjacent we compute the mean of their expectations

$$\mathbf{T}_{i,j} = \frac{\mathbb{E}[\mathbf{A}_{i,:}] + \mathbb{E}[\mathbf{A}_{:,j}]}{2} \tag{21}$$

and perform the test

$$t(\mathbf{A}_{i,j}) = \begin{cases} 1 & \text{if } \mathbf{A}_{i,j} > \mathbf{T}_{i,j} \\ 0 & \text{otherwise} \end{cases} \tag{22}$$

4. Experiments

In this section we discuss the experiments we use to evaluate the algorithms discussed in the earlier sections.

4.1. Unsupervised Sensor Adjacency Learning

We appraise our adjacency learning approach on three CASAS smart-home datasets: `twor.2009`[3] [6], `tokyo`[4] [7] and `aruba`[5] [8]. We chose these three datasets in order to test the robustness of our adjacency learning algorithm in various scenarios, since they differ in layout, environment (home/work), and number of occupants.

The `twor.2009` dataset was obtained in a multi-resident environment in which two residents performed their normal daily activities. This smart home spans two stories and much of the dataset is annotated. This dataset forms the basis of our activity recognition experiments. The `tokyo` dataset consists of sensor events collected in a smart workplace. This testbed differs intrinsically from the `twor.2009` dataset in that the majority of the

[3] http://ailab.wsu.edu/casas/datasets/twor.2009.zip
[4] http://ailab.wsu.edu/casas/datasets/tokyo.zip
[5] http://ailab.wsu.edu/casas/datasets/aruba.zip

sensors are in a relatively open-plan space. Furthermore, nine employees participated in this dataset. Finally, the `aruba` dataset is a single-resident smart home in which the resident is regularly visited by their immediate family. One interesting aspect of this dataset is that are two categories of motion sensors: 1) localised motion sensors with a small sensing jurisdiction; and 2) wide-range motion sensors which cover a much wider jurisdiction of the floor-plan.

4.2. ROC Analysis of Adjacency Matrices

There are several ways to compare adjacency matrices for graphs with the "fixed cardinality vertex sequence property" (meaning that the number of vertices is fixed and the order is the same for both graphs) [38]. Given two adjacency matrices, Mantel's test of the null hypothesis of 'no association' between the two matrices is the most commonly used [39]. However, it has been established that the Mantel test lacks a clear statistical framework specifying fully the null and alternative hypotheses, and in particular can be flawed in the presence of spatial auto-correlations [40], which is certainly the case for this application.

This motivated to introduce the use of Receiver Operating Characteristic (ROC) analysis [23] to compare adjacency matrices. The WXC and MI adjacency learning routines yield real-valued numbers which we sort in ascending order as larger values indicate a higher likelihood of adjacency. Performing ROC analysis requires a binary adjacency matrix to be constructed that represents the true adjacency matrix, in which values of '1' indicate adjacency between sensors. Creating these matrices based only on the floor-plan of the testbeds is a subjective and potentially noisy process. In particular with the `tokyo` dataset we cannot know whether a row of desks might break the adjacency between pairs of sensors, and with the `aruba` dataset, we do not know the radius over which the two kinds of motion sensor are receptive. Hence, we have estimated the "ground truth" adjacency (if such a ground truth exists) only for the `twor.2009` dataset, henceforth termed the 'hand-crafted' adjacency matrix, and used this to estimate ROC metrics.

Hence we apply the ROC analysis to the `twor.2009` dataset, whilst we demonstrate the results of our adjacency learning approaches on the `aruba` and `tokyo` visually.

4.3. Activity Recognition

We assess our activity classification models on the `twor.2009` dataset and perform the following experiments:

1. Baseline classification without using power-set representation (`vanilla`);
2. Activity classification using exhaustive power-set representation (`exhaustive`);
3. Activity classification using hand coded adjacency matrix (`hand-coded`);
4. Activity classification using the adjacency matrix learnt from the WXC algorithm (`cross-correlation`); and
5. Activity classification with MI adjacency learning algorithm (`mutual-information`).

4.4. Performance Evaluation

Given a set of ground truth labels and classifier predictions, we can define predictions as being True Positives (TPs), True Negatives (TNs), False Positives (FPs), or False Negatives (FNs). By accumulating these over a dataset, we can compute various accuracy metrics, including precision, recall as follows:

$$\text{precision} = \frac{\#\text{TP}}{\#\text{TP} + \#\text{FP}} \tag{23}$$

$$\text{recall} = \frac{\#\text{TP}}{\#\text{TP} + \#\text{FN}} \tag{24}$$

Precision and recall are accuracy metrics, and these averaged by calculating their harmonic mean, which yields the F_1 score:

$$\text{F}_1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{25}$$

The F_1 score is a more appropriate performance evaluation metric than accuracy when dealing with unbalanced class labels [41], and in this work, precision, recall and F_1 are used to evaluate classification performance, *c.f.* [28].

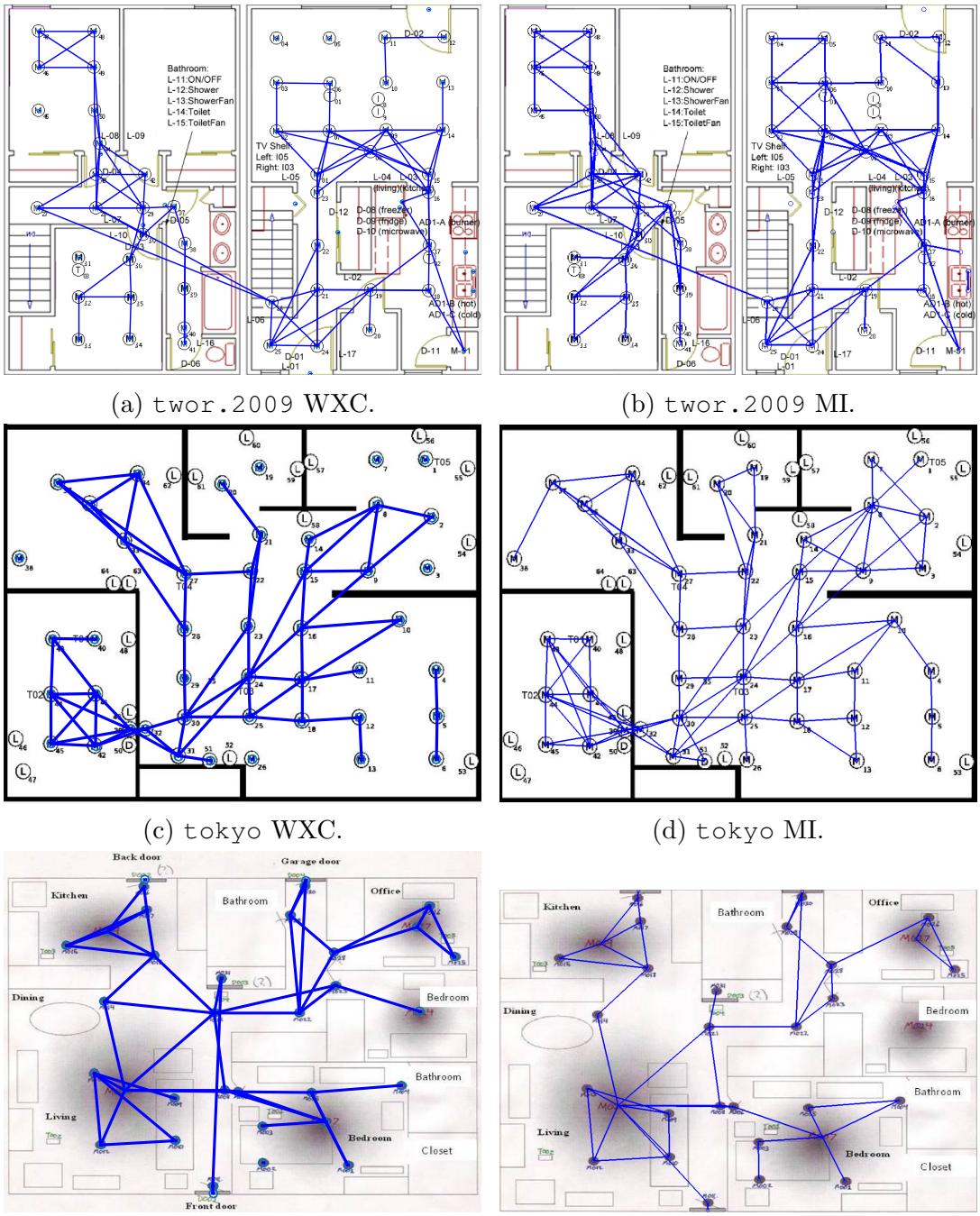
5. Results

5.1. Learning the Adjacency Matrix

5.1.1. Visual Assessment of Adjacency Matrices

Figure 3 shows the adjacency matrices that were learnt from the `twor.2009`, `tokyo` and `aruba` datasets. Figures 3a, 3c and 3e show the adjacency matrices that were learnt by the WXC approach, and Figures 3b, 3d and 3f show the matrices that were learnt from the MI methods. In these images, blue lines that link two sensors indicate that these sensors were deemed adjacent by the algorithm. We term the projection of the thresholded adjacency matrix to the floor-plan the 'adjacency map'.

Upon visual inspection, the learnt adjacency maps seem reasonable for all testbeds. We draw the reader's attention to the `twor.2009` and `tokyo` datasets where we see that the discovered adjacency maps respect the topological boundaries imposed by the walls in the office. In particular with the `tokyo` dataset, the learnt adjacency map is interesting because a total of nine employees participated in the dataset. Furthermore, given that `tokyo` is an office environment, one might expect that many sensors throughout the testbed would be highly correlated during working hours, and this in turn might be expected to elicit unreliable adjacency maps. However, we have demonstrated that that our approaches are tolerant to this. In the lower right hand corner in the `tokyo` dataset we can see six sensors which are not densely connected together. We do not know why these sensors are not connected, but

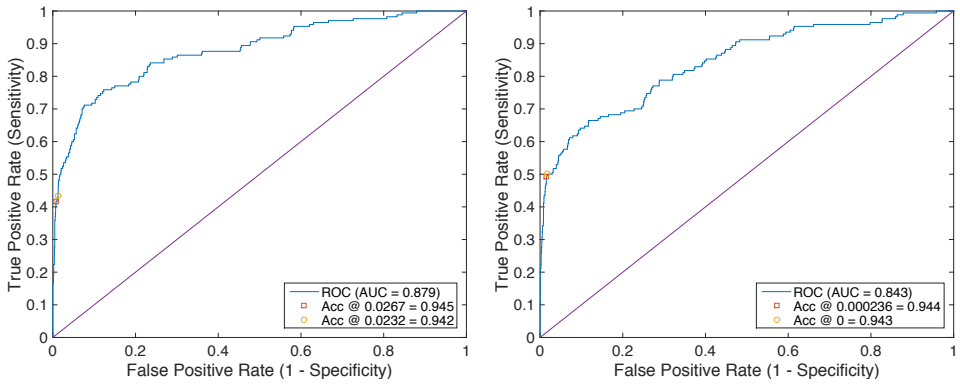


(a) `twor.2009` WXC. (b) `twor.2009` MI.

(c) `tokyo` WXC. (d) `tokyo` MI.

(e) `aruba` WXC. (f) `aruba` MI.

Figure 3: WXC (left column) and MI (right column) adjacency maps overlaid on the floor-plans for the `twor.2009` (top row), `tokyo` (middle row) and `aruba` (bottom row) datasets.



(a) WXC method. (b) MI method.

Figure 4: ROC curves for weighted adjacency matrices learnt from the WXC and MI methods. The optimal operating points and the operating point at the thresholds are shown on the plots. The red □ indicates the optimal threshold position, and the orange ○ shows the selected threshold.

we speculate that a row of office desks or cabinets might be found in this region and these may restrict motion locally.

Furthermore, note that in the `twor.2009` dataset there are two floors (the ground floor is the right half of the figure, and the upper floor is the left of the figure), and that the adjacency maps correctly show connections that span the floors from the top to the bottom of the stairs.

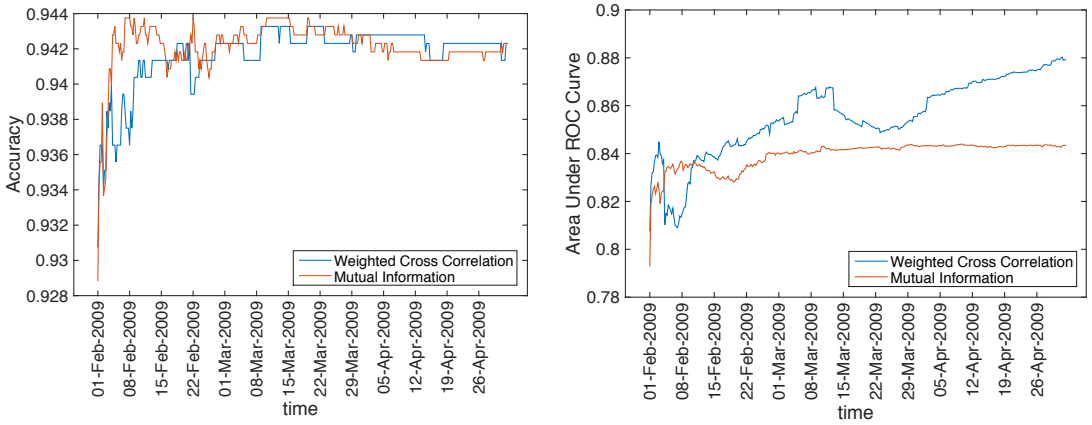
With the `aruba` testbed, we also see that the adjacency maps also seem reasonable. This testbed consists of a hierarchy of motion sensors (some have a much wider jurisdiction than others). The wide-coverage motion sensors are shown by the shaded regions in the floor-plan, and those with a narrower field of view are shown as circles. Our adjacency learning methods cope well with this family of motion sensors, and, in general, the 'narrow' motion sensors are reasonably clustered under the jurisdiction of 'wide' sensors.

The MI approach generally yields a greater number of connections, in particular near the living room and upstairs hallway in the `twor.2009` dataset. Visually, there is no clear indication of superiority of one method over the other. In the following section we will see how these methods compare when used to generate features for classification.

5.1.2. Performance on two resident dataset

We now consider the evaluation of the estimated adjacency matrices with ROC curves using the method described in subsection 4.2. Figure 4 shows the ROC curves generated from the weighted adjacency matrices. We can see visually that although the WXC method obtains a higher Area Under Curve (AUC), the accuracy at the selected thresholds are equivalent. We note that the threshold selection routines described in the previous section obtain near-optimal performance in both cases.

We additionally estimated the amount of sensor data required to obtain a stable adjacency matrix by selecting increasing proportions of dataset. Note that here we recompute the entire adjacency matrix with all of the data up till time t, rather than using online



(a) Accuracy for increasing dataset size.

(b) AUC for increasing dataset size.

Figure 5: Accuracy and AUC plots for the thresholded MI and WXC adjacency matrices computed for increasing dataset sizes (labels on x-axis are in weeks).

updates. Figure 5a shows the accuracy obtained for WXC and MI methods as the number of training examples increases, and we can see that the MI accuracy increases and settles to a steady state rapidly, while the WXC method requires more data to obtain the same accuracy. Figure 5b shows the AUC computed for increasing proportions of the data. The WXC method achieves a larger AUC with more data. This indicates that WXC adjacency matrices ultimately achieves better ranking of the pairwise adjacency than MI given a larger dataset.

Given the unsupervised nature of our adjacency learning techniques, achieving high performance against the hand-crafted adjacency matrix is very encouraging.

5.2. Activity Recognition

Table 3: Performance metrics obtained for all test scenarios, obtained in five-fold cross validation. In general, we can see that with all experiments, better performance is obtained (for precision, recall and F_1) when informing the classification model with an adjacency matrix.

Method	Precision	Recall	F_1 Score
`vanilla`	0.644	0.636	0.632
`exhaustive`	0.533	0.530	0.521
`hand-coded`	0.679	0.692	0.682
`mutual-information`	0.668	0.679	0.669
`cross-correlation`	0.673	0.684	0.675

Table 3 shows the precision, recall and F_1 scores that were obtained for each of the experiments investigated. We can see that the average scores for the `exhaustive` approach are significantly lower (by approximately 10-15%) than those obtained by other results.

Adjacency-informed CRF models outperform the `vanilla` and `exhaustive` models on all metrics. Interestingly, the hand-coded adjacency matrices perform best, with the hand-coded adjacency matrix obtaining the best results overall.

In order to see if the differences between the methods were significant, we first performed a Friedman test [42] on the F_1 scores [6]. The Friedman test was significant with $Q = 143.9, p < 10^{-10}$, and hence we performed Nemenyi[7] pair-wise *post-hoc* testing at a (fairly stringent) significance level $\alpha = 0.01$. The results are shown in Figure 6, and can be summarised as follows: the `exhaustive` method is significantly worse than all of the other methods, but we cannot reject the null hypothesis that there is no significant difference between the `vanilla`, `hand-coded`, `cross-correlation` and `mutual-information` methods.

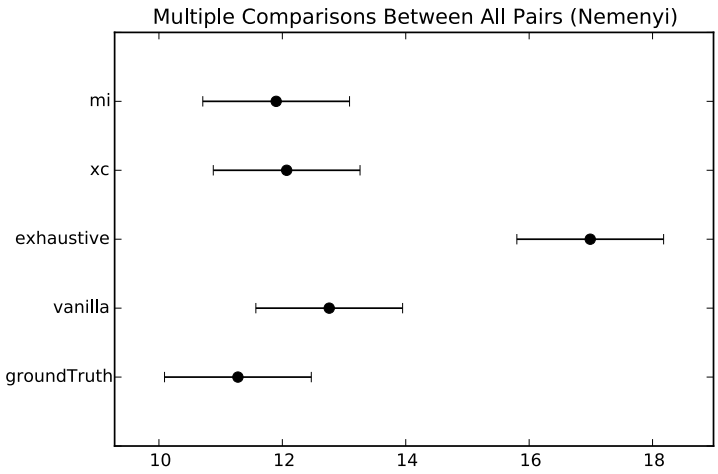


Figure 6: Multiple comparisons *post-hoc* Nemenyi Test.

6. Discussion

6.1. Unsupervised adjacency learning

We have presented two unsupervised learning algorithms based on the weighted cross correlation and mutual information for learning sensor topologies in smart environments. We have shown that both algorithms perform well and we have shown that the algorithms are general and perform well on different datasets.

Figure 3 shows the sensor linkage obtained with the binarised adjacency matrices of both techniques on three datasets. Upon visual inspection, the set of links obtained from

[6] The Friedman test is a non-parametric test that is similar to the parametric repeated measures Analysis of Variance (ANOVA) but does not make the normality assumption that ANOVA does

[7] Nemenyi test is a *post-hoc* test intended to find the groups of data that differ after a statistical test of multiple comparisons (such as the Friedman test) has rejected the null hypothesis that the performance of the comparisons on the groups of data is similar

the algorithm seem legitimate. It is particularly interesting that the methods work with the `tokyo` dataset (Figures 3c and 3d) because the data were obtained from a nine-person smart office. One might expect that spurious correlations would occur throughout the network as participants might adhere to similar patterns of behaviour (*e.g.* employees may tend to arrive, depart and have meals at similar times). This could obscure the true adjacency between sensors, but, on the basis of the output of our algorithms on the three datasets, our approach seems to be tolerant to this. It is also interesting that the `twor.2009` dataset did not yield spurious links, for example, between sensors in the two bedrooms. The most frequently occurring activities for both residents in this dataset are sleeping and working, and these all take place in the residents' bedrooms. Yet the adjacency learning routine never connected these sensors together.

We draw attention to a number of other factors. Firstly, it is difficult to find any connection between pairs of sensors that are obviously representing an 'incorrect' correlation in the `twor.2009`, `aruba` and `tokyo` datasets (*i.e. false positive* links). Secondly, a number of 'orphaned' sensors (*i.e.* sensors which have no connections) can be found in the datasets (*i.e. false negative* links). We argue that false positive connections are much less desirable than false negative connections because false positive connections may lead to models that predict confidently based on spurious sensor activations (*e.g.* such a classifier might credit 'motion in living room and motion in bedroom' as deriving from the activity 'resident 1 is working'). Models with increasing connectivity of this variety will increasingly begin to resemble the `exhaustive` class. The presence of false negative connections in the adjacency will learn models will be more similar to the `vanilla` category, as the adjacency learning algorithm has deemed that those sensors are more likely to be independent than not. We can see from the results table earlier that in terms of subsequent classification accuracy `exhaustive`-type models are less desirable than `vanilla` models.

The primary existing method for comparing adjacency matrices, known as the Mantel test [39], has been shown to be flawed in the presence of spatial auto-correlations [40], which motivated our decision to introduce a new method for comparing the quality of adjacency matrices using ROC analysis. We applied this to the `twor.2009` dataset. To perform ROC analysis, a 'ground truth' adjacency matrix is required, and constructing this is a laborious and imprecise process as it was constructed without reference to the layout of the furniture in the home (*e.g.* knowing about the presence of a large table or couch in the floor-plan may better support the inclusion or exclusion of certain links). We maintain that our hand-crafted adjacency matrix is sufficiently accurate because, when incorporated into the CRF learning procedure, models using the hand-coded adjacency matrix performed best on all metrics.

Figure 5 shows that approximately 1–2 weeks of sensor data is sufficient to obtain a stable adjacency matrix with the `twor.2009` dataset (with regard to steady-state accuracy). However, it is worth stating that with less than one week of data, an accuracy of $\approx 92\%$ can be obtained indicating that the algorithm might cope with changing room layouts. While the accuracy remains approximately constant for increasing amount of data, the AUC for the WXC continues to increase with larger proportions of data. This indicates that the WXC adjacency learning yields a model which achieves a better ranking between sensors,

but, around the region of thresholding, both approaches exhibit similar performance. Correspondingly, however, it seems that sufficient information is obtained after approximately one month for the MI adjacency learning technique, as the accuracy and AUC remain relatively constant after this time.

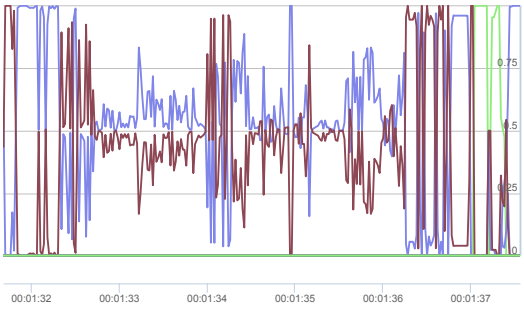
It is worth mentioning, however, that by virtue of the ROC analysis, we can measure the optimality of the threshold selection. Figures 4a and 4b show that the threshold selection routines described earlier selects values that are very close to the optimal values. This is an unexpected and pleasing result, and supports the threshold selection methods discussed previously. The optimality of the threshold selection here assumes equal costs for incorrect link predictions [43]. Earlier in this section we stated that false negative predictions are preferred over false positives predictions and so we have, in fact, shown that classification costs are non-uniform. Given the absence of objective ground truth labels, however, we argue that assuming uniform mis-classification costs is reasonable as we do not know whether the hand-crafted matrix consists of a greater number of false positive or false negative links.

We have observed in our analysis that learning adjacency matrices is more reliable between pairs of motion sensors than between different categories of sensors (*e.g.* motion sensors and door sensors). We believe two factors contribute to this. Firstly, the door sensors are always represented in the state-change representation which will have a significantly lower count of sensor activations than motion sensors. However, as MI can be interpreted as measuring the (in)dependence between random variables, it may not be so sensitive to variable imbalance. Therefore, we believe that the more significant cause of the absence of linkage between motion and door sensors is that when a resident passes by a door sensor (*e.g.* the door sensor in a cupboard in the kitchen) the door sensor will only be activated when the resident explicitly interacts with the door, as opposed to motion sensors which will always toggle when a resident passes. This implies that for some sensor varieties, there is a stochastic element that must be modelled. Our methods have not accounted for this, and so our algorithms specify that these sensors are independent to the others (*i.e.* they are orphaned). We leave the construction of algorithms that can cope with these scenarios as future work.

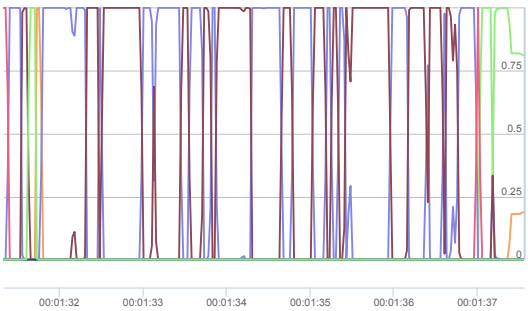
6.2. Classification performance

In the previous section, we have shown that incorporating adjacency matrices into the modelling framework improves classification accuracy and that the improvement is significant with respect to the `exhaustive` feature representation. Here we will will expand further on these points and discuss other means of gauging the performance of the classification model.

We first present the output of two classification models on a segment of the sleeping activity. Here, both residents are asleep in their bedrooms, and occasionally move during the night, and this is logged in the database. In order to demonstrate the utility of the adjacency matrix models, we searched for a period of time where only one resident tends to move at any given time. Given that there is motion only in Resident 1's room, then, one would expect a model to predict the activity 'Resident 1 Sleeping' with high confidence.



(a) Example predictions with `vanilla` classification model.



(b) Example predictions with `hand-coded` model.

Figure 7: Sample predictions for `vanilla` and `hand-coded` CRF models over the same time period. In both image the blue and brown traces represent 'Resident 1 Sleeping' and 'Resident 2 Sleeping' respectively.

In Figure 7, we show the output of the `vanilla` (Figure 7a) and `ground-truth` (Figure 7b) classifiers in this scenario. The blue lines indicate predictions favouring Resident 1 and the brown trace is related to Resident 2. Figure 7a shows that the classifier has insufficient evidence on which to distinguish between 'Resident 1 Sleeping' and 'Resident 2 Sleeping' and the classifier predicts the sleeping activity with probability ≈ 0.4 to 0.6. We believe that the reason for this behaviour is that during the training procedure, the classification algorithm was presented with many examples where both residents have concurrently been sleeping. Without reference to the sensor topology, the model was unable to distinguish between the two residents sleeping behaviour. The classification weights associated with the motion sensors, therefore, tend to be smaller, and the weights associated to the 'hour of the day' features are significantly higher. As a result of this, the classifier explains the motion sensor activations as resulting from both residents. For activities which frequently occur in isolation, this pathology does not occur.

However, the adjacency matrix based methods were able to separate concurrent sensor activations into contiguous sequences of parallel sensor activations. Using these alongside the constructed feature functions enabled the learning algorithm to reason about the most likely activity in a more representative manner during training. Therefore, when testing the model it can yield more confident predictions. This is illustrated in Figure 7b for the `hand-coded` adjacency matrix, where the probabilities toggle between 0 and 1 as movement occurs in each room.

The previous scenario demonstrated an example where the classifier learnt questionable relationships between sensors and activities (sensors in Resident 1's bedroom yielded significant probability mass on a prediction for Resident 2). This is similar to the scenarios described earlier in this paper regarding events in the kitchen affecting predictions in the bedroom. The previous example explained how this could occur with the `exhaustive` feature representation, but we have just shown that these pathologies can also occur with the 'simple' `vanilla` models. Figure 7 shows that we can mitigate against this by incorporating topological domain knowledge into the modelling routine. This can help predictive

confidence with both the `vanilla` and `exhaustive` models. It is worth reiterating that we have learnt these representations without having to resort to manually segmenting the sensor activations.

Furthermore, we observed that when two activities are performed by two residents, the `vanilla` model will occasionally predict only one activity as having occurred even when evidence has been observed for both activities (*e.g.* if both residents are sleeping, and if both residents are moving at the same time, the model might predict the sleeping activity only for one of the residents). We attribute this behaviour to the Viterbi algorithm [44] which can have the effect of smoothing away variation in the predictions [45]. We once more submit to the utility of the power-set representation which is shown upon inspection to be significantly less susceptible to this behaviour. This is due to the ability of being able to separate out functional areas of activity in the house, as enabled by the adjacency learning routines.

We have also observed similar pathologies to other researchers in the community in determining the resident in bathroom-based activities. These were the primary cause of the errors in prediction as our features are the instantaneous sensor events. We also observed other patterns which illustrate the difficulty involved with accurate annotation smart home datasets. During a period wholly annotated as 'Watching TV', we have observed that residents often enter the kitchen for a brief period of time. The classification models, upon seeing this, tend to predict a sequence of 'Watching TV', 'Meal Preparation' and when the resident returns to the living room, the prediction returns to 'Watching TV'. We do not find the prediction particularly representative here (as it could be argued that the resident is in fact 'Watching TV' and 'Preparing a Meal' concurrently), but nor do we believe that the annotation is particularly accurate either. Instead, we would favour a label which with demonstrate a superposition of both activities at the same time, but this is open for debate.

We believe that our techniques can be used by other sequence classifiers, *e.g.* HMMs, with the adjustments outlined here. Standard HMMs learn the joint distribution of the observations and labels, with the assumption that emission probabilities are unconditionally independent at all time points. This assumption is violated if structure is imposed on the features. General loopy inference techniques can perform inference on the new structures, but with a significant increase in time complexity. A number of methods, such as Tree-Augmented Naïve Bayes (TAN) [46], offer a middle-ground between assuming complete independence and dense dependence between features for such model classes as naïve Bayes and HMMs. It can be shown that by relaxing certain restrictions on the structure of the features, exact inference can still be achieved efficiently. If the connectivity of the adjacency matrices is appropriately restricted, we believe that the methods outlined in this paper could be applied to HMMs. CRFs (and non-sequential LR) are not subject to this consideration during inference because they model the conditional distribution of labels given the observations, *i.e.* $p(\mathbf{y}|\mathbf{x})$, and this form is agnostic to the particular form and structure of the features.

7. Conclusions and Future Work

We have described two methods that attempt automatically learn about the topology of the sensor network and have shown how adjacency matrices between sensors can be constructed from these. We demonstrated that the learnt adjacency matrices for both methods have good agreement with a hand-constructed adjacency matrix and that our threshold selection is near-optimal. We then outlined how the adjacencies can be used to construct feature functions for a CRF classifier. Our empirical evaluation showed that the CRF classifier that uses these adjacencies outperforms two baseline methods: a vanilla method that treats sensors as independent, and an exhaustive approach that takes the power-set of size N of all sensors. Finally, we have demonstrated that our approach mitigates against the requirement for segmenting data by hand.

Future work will introduce methods that can more satisfactorily cope with learning adjacencies between varieties of sensor families. An online learning framework [47] which will reason about the sensor topology and activities being performed will be investigated. Finally, future work will involve the extraction of features through time which we believe will be capable of describing trajectories within the residence. We expect this to improve classification performance of resident identification (*e.g.* for bathroom-based activities which are difficult to classify).

Acknowledgement

This work was performed under the Sensor Platform for HEalthcare in a Residential Environment (SPHERE) Interdisciplinary Research Collaboration (IRC) funded by the UK Engineering and Physical Sciences Research Council (EPSRC), Grant EP/K031910/1.

References

[1] P. Woznowski, X. Fafoutis, T. Song, S. Hannuna, M. Camplani, L. Tao, A. Paiement, E. Mellios, M. Haghighi, N. Zhu, G. Hilton, D. Damen, T. Burghardt, M. Mirmehdi, R. Piechocki, D. Kaleshi, I. Craddock, A multi-modal sensor infrastructure for healthcare in a residential environment, in: IEEE International Conference on Communications (ICC), Workshop on ICT-enabled services and technologies for eHealth and Ambient Assisted Living, 2015.

[2] N. Zhu, T. Diethe, M. Camplani, L. Tao, A. Burrows, N. Twomey, D. Kaleshi, M. Mirmehdi, P. Flach, I. Craddock, Bridging eHealth and the internet of things: The SPHERE project, IEEE Intelligent Systems 30.

[3] T. Diethe, P. A. Flach, Smart-homes for eHealth: Uncertainty management and calibration, in: NIPS 2015 Workshop on Machine Learning in Healthcare, 2015.

[4] T. Diethe, N. Twomey, P. Flach, Active transfer learning for activity recognition, in: European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, 2016.

[5] N. Twomey, T. Diethe, P. Flach, Bayesian active learning with evidence-based instance selection, in: Workshop on Learning over Multiple Contexts, European Conference on Machine Learning (ECML15), 2015.

[6] D. J. Cook, M. Schmitter-Edgecombe, Assessing the quality of activities in a smart environment., Methods Inf Med 48 (5) (2009) 480–5.

[7] D. J. Cook, A. Crandall, G. Singla, B. Thomas, Detection of social interaction in smart spaces, Cybernetics and Systems: An International Journal 41 (2) (2010) 90–104.

[8] D. J. Cook, Learning setting-generalized activity models for smart spaces, IEEE intelligent systems 2010 (99) (2010) 1.
[9] E. Kim, S. Helal, D. Cook, Human activity recognition and pattern discovery, Pervasive Computing, IEEE 9 (1) (2010) 48–53.
[10] N. Krishnan, D. J. Cook, Z. Wemlinger, Learning a taxonomy of predefined and discovered activity patterns, Journal of Ambient Intelligence and Smart Environments 5 (6) (2013) 621–637.
[11] E. Nazerfard, B. Das, L. B. Holder, D. J. Cook, Conditional random fields for activity recognition in smart environments, in: Proceedings of the 1st ACM International Health Informatics Symposium, ACM, 2010, pp. 282–286.
[12] A. Fleury, M. Vacher, N. Noury, SVM-based multimodal classification of activities of daily living in health smart homes: sensors, algorithms, and first experimental results, Information Technology in Biomedicine, IEEE Transactions on 14 (2) (2010) 274–283.
[13] P. N. Dawadi, D. J. Cook, M. Schmitter-Edgecombe, Automated cognitive health assessment using smart home monitoring of complex tasks, Systems, Man, and Cybernetics: Systems, IEEE Transactions on 43 (6) (2013) 1302–1313.
[14] I. Fatima, M. Fahim, Y.-K. Lee, S. Lee, A unified framework for activity recognition-based behavior analysis and action prediction in smart homes, Sensors 13 (2) (2013) 2682–2699.
[15] L. G. Fahad, A. Khan, M. Rajarajan, Activity recognition in smart homes with self verification of assignments, Neurocomputing 149 (2015) 1286–1298.
[16] J. Wan, M. J. O'grady, G. M. O'hare, Dynamic sensor event segmentation for real-time activity recognition in a smart home context, Personal and Ubiquitous Computing 19 (2) (2015) 287–301.
[17] D. J. Cook, N. C. Krishnan, Activity Learning: Discovering, Recognizing, and Predicting Human Behavior from Sensor Data, John Wiley & Sons, 2015.
[18] D. J. Cook, N. C. Krishnan, P. Rashidi, Activity discovery and activity recognition: A new partnership, Cybernetics, IEEE Transactions on 43 (3) (2013) 820–828.
[19] T. Gu, S. Chen, X. Tao, J. Lu, An unsupervised approach to activity recognition and segmentation based on object-use fingerprints, Data & Knowledge Engineering 69 (6) (2010) 533–544.
[20] P. Palmes, H. K. Pung, T. Gu, W. Xue, S. Chen, Object relevance weight pattern mining for activity recognition and segmentation, Pervasive and Mobile Computing 6 (1) (2010) 43–57.
[21] J. Shawe-Taylor, N. Cristianini, Kernel methods for pattern analysis, Cambridge university press, 2004.
[22] S. Szewcyzk, K. Dwan, B. Minor, B. Swedlove, D. Cook, Annotating smart environment sensor data for activity learning, Technology and Health Care 17 (3) (2009) 161–169.
[23] T. Fawcett, An introduction to ROC analysis, Pattern recognition letters 27 (8) (2006) 861–874.
[24] T. R. Diethe, N. Twomey, P. Flach, Sphere-a sensor platform for healthcare in a residential environment, Large-scale Online Learning and Decision Making Workshop.
[25] T. R. Diethe, N. Twomey, P. Flach, Bayesian modelling of the temporal aspects of smart home activity with circular statistics, in: European Conference on Machine Learning (ECML'15) (to appear), 2015.
[26] K.-C. Hsu, Y.-T. Chiang, G.-Y. Lin, C.-H. Lu, J. Y.-J. Hsu, L.-C. Fu, Strategies for inference mechanism of conditional random fields for multiple-resident activity recognition in a smart home, in: Trends in Applied Intelligent Systems, Springer, 2010, pp. 417–426.
[27] E. Nazerfard, B. Das, L. B. Holder, D. J. Cook, Conditional random fields for activity recognition in smart environments, in: Proceedings of the 1st ACM International Health Informatics Symposium, ACM, 2010, pp. 282–286.
[28] N. Twomey, P. Flach, Context modulation of sensor data applied to activity recognition in smart homes, in: Workshop on Learning over Multiple Contexts, European Conference on Machine Learning (ECML'14), 2014.
[29] J. D. Lafferty, A. McCallum, F. C. N. Pereira, Conditional random fields: Probabilistic models for segmenting and labeling sequence data, in: Proceedings of the Eighteenth International Conference on Machine Learning, ICML '01, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2001, pp. 282–289.
[30] T. Lavergne, O. Cappé, F. Yvon, Practical very large scale CRFs, in: Proceedings of the 48th Annual

Meeting of the Association for Computational Linguistics, 2010, pp. 504–513.
[31] J. Duchi, E. Hazan, Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization, The Journal of Machine Learning Research 12 (2011) 2121–2159.
[32] R. Klinger, K. Tomanek, Classical probabilistic models and conditional random fields, TU, Algorithm Engineering, 2007.
[33] Y. Goldberg, M. Elhadad, splitSVM: fast, space-efficient, non-heuristic, polynomial kernel computation for NLP applications, in: Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers, 2008, pp. 237–240.
[34] N. C. Krishnan, D. J. Cook, Activity recognition on streaming sensor data, Pervasive and Mobile Computing 10 (2014) 138–154.
[35] J. Wan, M. J. OGrady, G. M. OHare, Dynamic sensor event segmentation for real-time activity recognition in a smart home context, Personal and Ubiquitous Computing (2014) 1–15.
[36] M. Abramowitz, I. A. Stegun, Handbook of mathematical functions: with formulas, graphs, and mathematical tables, no. 55, Courier Dover Publications, 1972.
[37] S. Kullback, R. A. Leibler, On information and sufficiency, The annals of mathematical statistics (1951) 79–86.
[38] J. Richiardi, D. Van De Ville, K. Riesen, H. Bunke, Vector space embedding of undirected graphs with fixed-cardinality vertex sequences for classification, in: Pattern Recognition (ICPR), 2010 20th International Conference on, IEEE, 2010, pp. 902–905.
[39] M. Fortin, J. Gurevitch, Mantel tests: spatial structure in field experiments, Design and analysis of ecological experiments. Chapman and Hall, New York (1993) 342–359.
[40] G. Guillot, F. Rousset, Dismantling the Mantel tests, Methods in Ecology and Evolution 4 (4) (2013) 336–344.
[41] F. J. Provost, T. Fawcett, R. Kohavi, The case against accuracy estimation for comparing induction algorithms., in: ICML, Vol. 98, 1998, pp. 445–453.
[42] M. Hollander, D. A. Wolfe, E. Chicken, Nonparametric statistical methods, Vol. 751, John Wiley & Sons, 2013.
[43] C. Elkan, The foundations of cost-sensitive learning, in: International joint conference on artificial intelligence, Vol. 17, Citeseer, 2001, pp. 973–978.
[44] A. J. Viterbi, Error bounds for convolutional codes and an asymptotically optimum decoding algorithm, Information Theory, IEEE Transactions on 13 (2) (1967) 260–269.
[45] K. P. Murphy, Machine learning: a probabilistic perspective, MIT press, 2012.
[46] F. Zheng, G. I. Webb, Tree augmented naive bayes, in: Encyclopedia of Machine Learning, Springer, 2011, pp. 990–991.
[47] T. Diethe, M. Girolami, Online learning with (multiple) kernels: A review, Neural Computation 25 (3) (2013) 567–625. doi:10.1162/NECO_a_00406.