Searching for Temporal Patterns in AmI Sensor Data

Romain Tavenard[1,2], Albert A. Salah[1], Eric J. Pauwels[1]

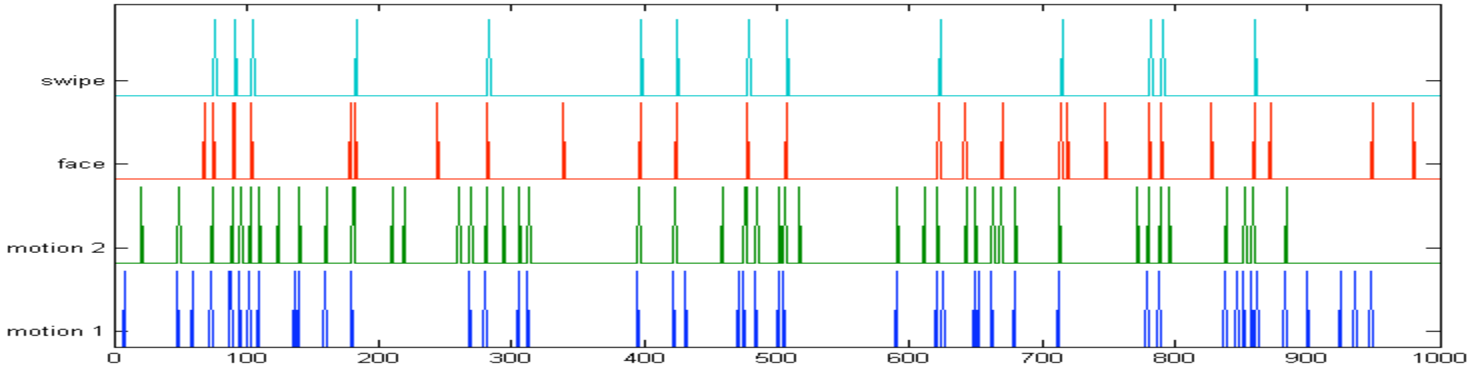
[1] Centrum voor Wiskunde en Informatica, CWI
Amsterdam, The Netherlands
[2] IRISA/ENS de Cachan, France

Overview

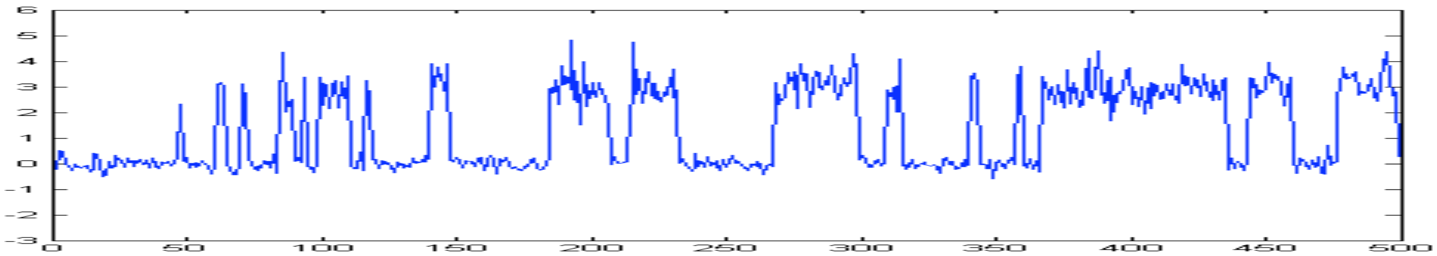
- Learning and Mining Temporal Patterns
- Different Approaches
  - Markov Models
  - Eigenbehaviours
  - Compression-based Approaches
    - Lempel-Ziv
    - Active Lempel-Ziv
    - Lempel-Ziv-Welch
- T-Patterns
  - Basic Algorithm
  - Proposed Approach

Learning and Mining Temporal Patterns

- Sensors in a dense network will typically exhibit
  - Baseline activity interspersed with bursts of activity (spikes); e.g. interruption sensors,



-- Switching between different states



Learning and Mining Temporal Patterns

- Temporal Patterns: Informative correlations between the activities (both across time and sensors!) due to underlying unobserved physical causes:

- Why interesting?
  - Layout discovery and self-calibration for plug-and-play devices: Correlations used to define proximity (context) in appropriate space (e.g. spatial or connectivity);
  - Increases robustness: Confidence in weak or ambiguous sensor signal will be bolstered when supported by expected activity in related sensors;
  - Anticipation and attention for resource management: Once temporal patterns have been detected they may be used to predict future events: expectation failure sparks increase in attention (temporal pop-out);
  - Personalisation and adaptivity: AmI system will have to adapt factory-settings to user preferences, based on the recurrence of stable usage patterns.

Temporal Patterns: A Prototypical Example

Movement patterns detected by low-level, low-resolution sensors (interruption sensors)

Advantages:

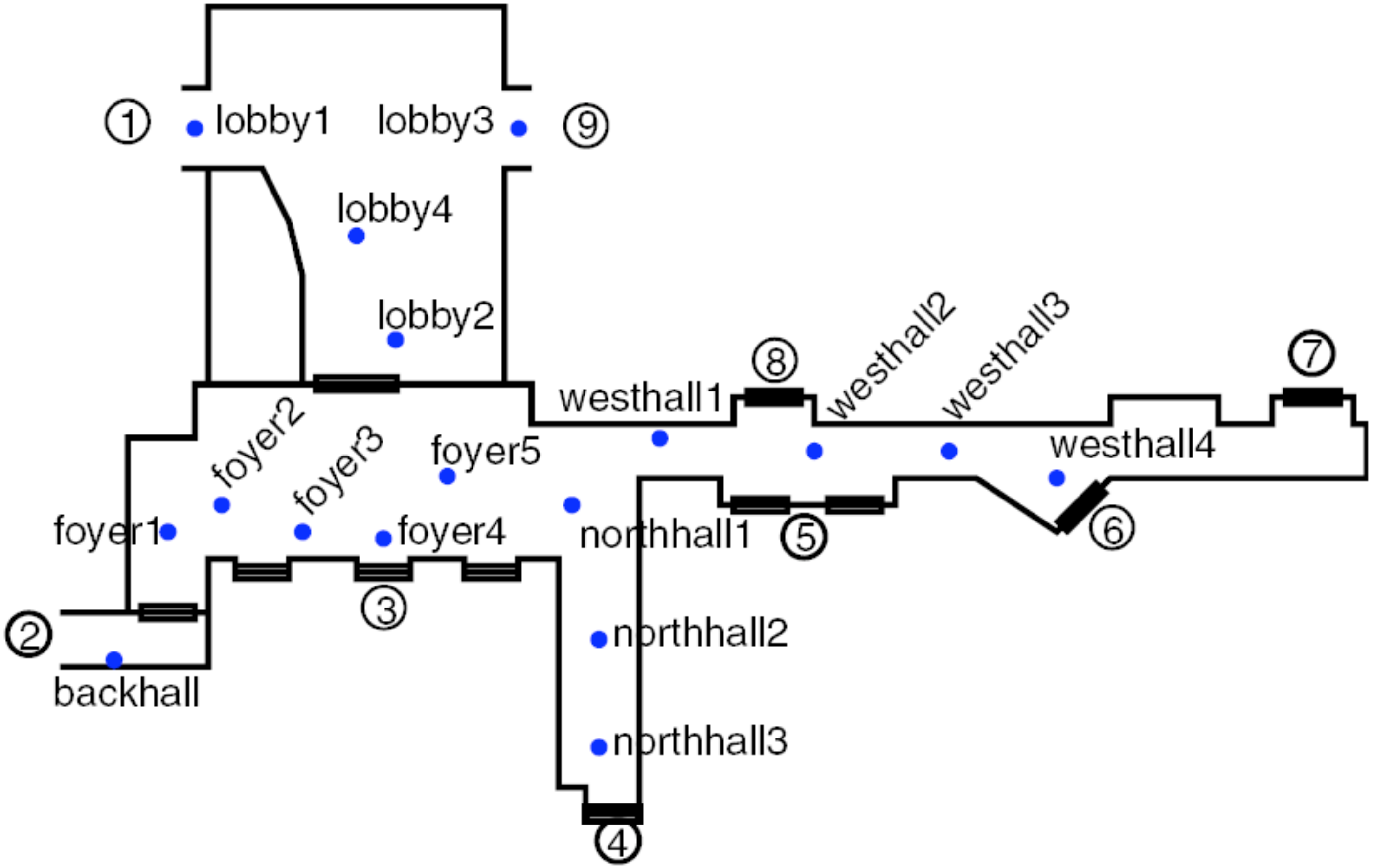
- Cheap, dense network possible,
- Minimally intrusive

Aims:

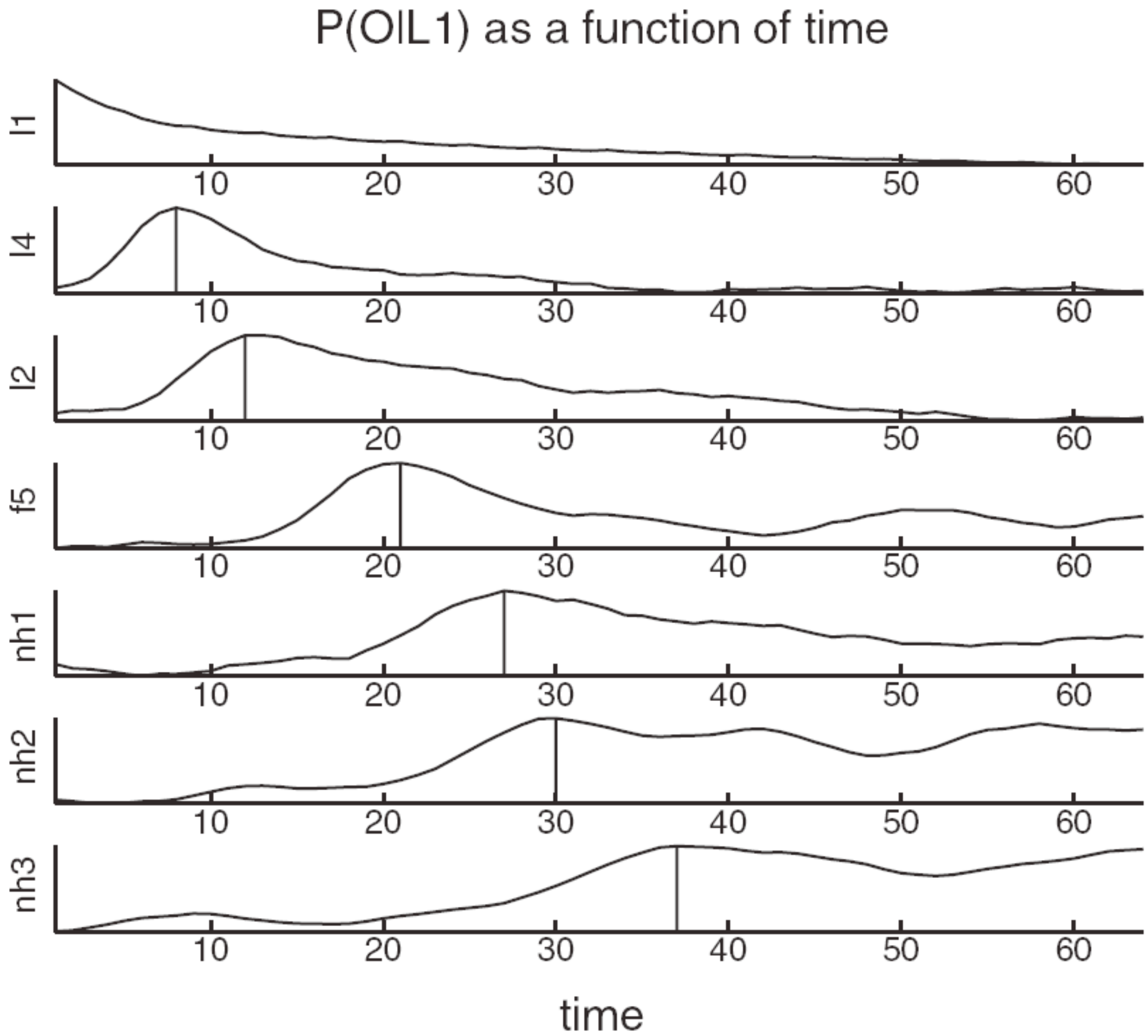
- Lay-out discovery: Compute correlation peaks in activation to infer distance, then use MDS to reconstruct (approximately) geographical layout
- Find similar sensor sequences and combine them to estimate HMMs characterizing various activities

Layout Discovery and Self-Calibration

- MERL office layout (C.R. Wren et.al., 2006)



Layout Discovery and Self-Calibration



Overview

- Learning and Mining Temporal Patterns
- **Different Approaches**
  - **Markov Models**
  - Eigenbehaviours
  - Compression-based Approaches
    - Lempel-Ziv
    - Active Lempel-Ziv
    - Lempel-Ziv-Welch
- T-Patterns
  - Basic Algorithm
  - Proposed Approach

HMM-based Clustering

- Basic Idea: Define similarity of temporal sequences in terms of the similarity of the underlying Markov models that generate them. Sequences can have different lengths!

- Juang and Rabiner (1985)
- Smyth (1996)
- Wren et al. (2006)

- Extensions to Hierarchical HMMs

HMM-based Clustering

- Illustrative Example: 20 temporal sequences (possibly of different length) generated by one of 2 HMMs

Slow dynamics: 2-state transition matrix: $A_1 = \begin{pmatrix} 0.6 & 0.4 \\ 0.4 & 0.6 \end{pmatrix}$

Fast dynamics: 2-state transition matrix: $A_1 = \begin{pmatrix} 0.4 & 0.6 \\ 0.6 & 0.4 \end{pmatrix}$

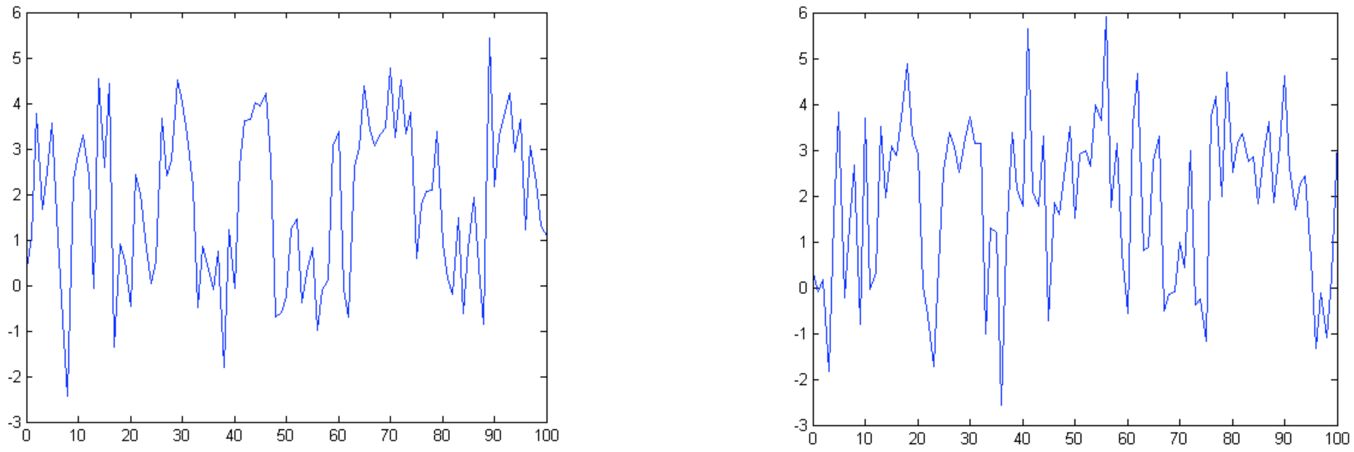
Gaussian Emissions:

state 1: $\mu_1 = 0 \quad \sigma_1 = 1$

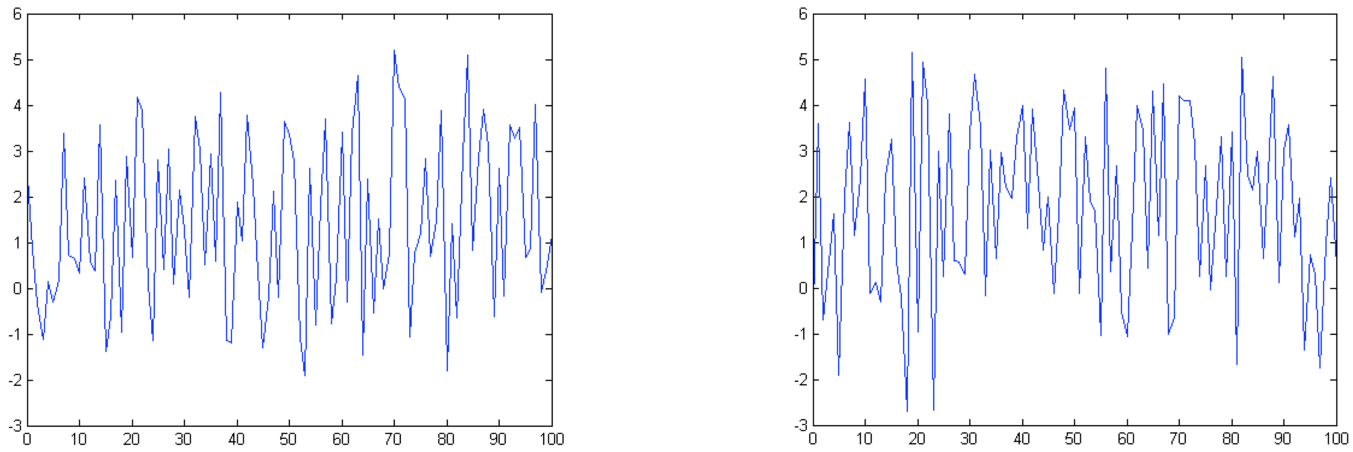
state 2: $\mu_1 = 3 \quad \sigma_1 = 1$

HMM Models

A1



A2



HMM-based Clustering

Assume K (known!) underlying HMMs each with m states emitting a Gaussian signal

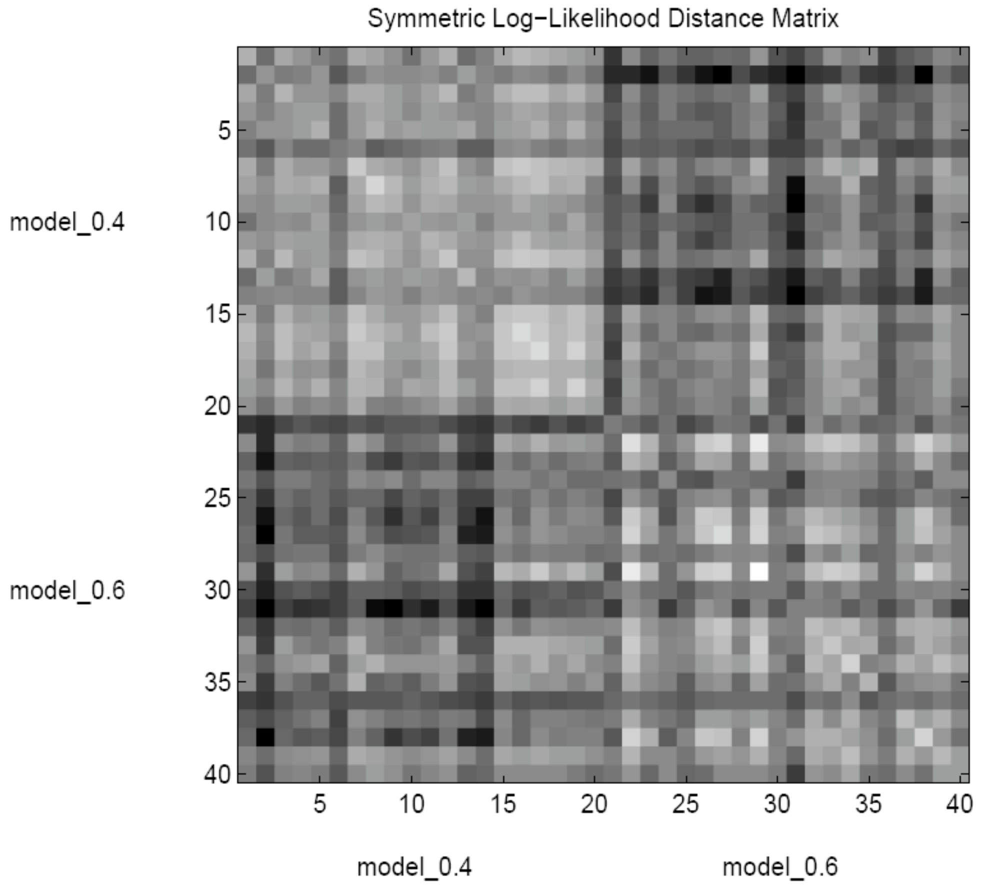
INPUT: N sequences $S_1, \ldots .S_N$, and parameter K and m, each sequence consists of observations $S_i = (x_{i1}, x_{i2}, \ldots, x_{it})$

1. Fit an HMM to each sequence S_i $(i=1..N)$,
   initialize using uniform transition matrix and Gaussian parameters derived from m groups obtained by applying k-means to sequence data x_{ij}
2. Call M_i the HMM model fitted to sequence S_i, compute the likelihood of every other sequence S_j wrt M_i and define the similarity between S_i and S_j

$$Sim(S_i, S_j) = \log P(S_i \mid M_j) + \log P(S_j \mid M_i)$$

HMM-based Clustering

3. Use the log-likelihood distance matrix to cluster sequences in K groups (e.g. using hierarchical clustering)



HMM-based Clustering

4. Finally, for each of the K clusters, fit a separate HMM model, this time trained on all the sequences assigned to this cluster.

$$\hat{A}_1 = \begin{pmatrix} 0.580 & 0.402 \\ 0.420 & 0.598 \end{pmatrix} \qquad \hat{\mu}_1 = \begin{pmatrix} 2.892 \\ 0.040 \end{pmatrix} \qquad \hat{\sigma}_1 = \begin{pmatrix} 1.353 \\ 1.219 \end{pmatrix}$$

$$\hat{A}_2 = \begin{pmatrix} 0.392 & 0.611 \\ 0.608 & 0.389 \end{pmatrix} \qquad \hat{\mu}_2 = \begin{pmatrix} 2.911 \\ 0.138 \end{pmatrix} \qquad \hat{\sigma}_2 = \begin{pmatrix} 1.239 \\ 1.339 \end{pmatrix}$$

It is then possible to use these parameter values to initialize and train a composite HMM using all available sequences.

Overview

- Learning and Mining Temporal Patterns
- Different Approaches
  - Markov Models
  - **Eigenbehaviours**
  - Compression-based Approaches
    - Lempel-Ziv
    - Active Lempel-Ziv
    - Lempel-Ziv-Welch
- T-Patterns
  - Basic Algorithm
  - Proposed Approach

Eigenbehaviours (Eagle & Pentland)

- Eagle & Pentland: *Eigenbehaviors: Identifying Structure in Routine* (Ubicomp, Proc. Royal Soc 07)

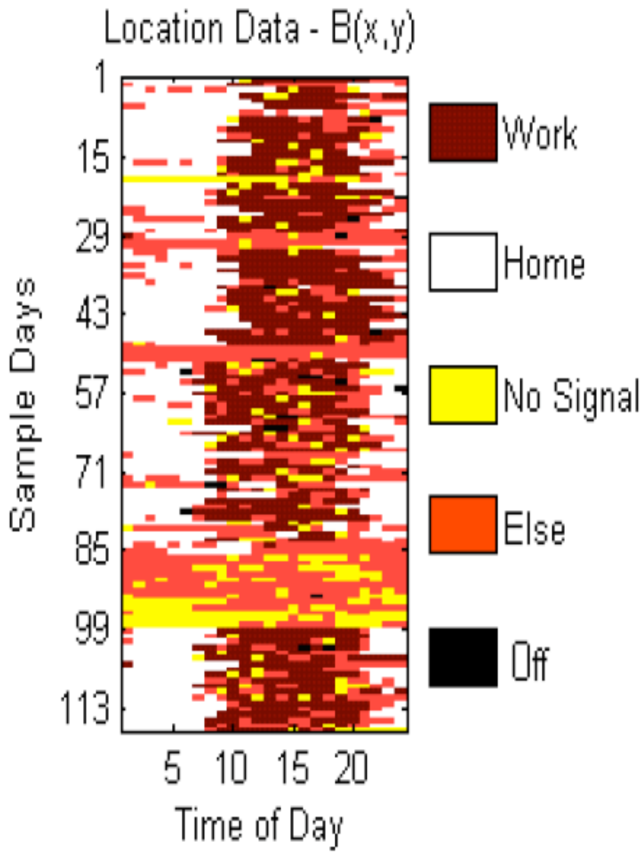
- Starting point: Markov models are ill suited to incorporate temporal patterns across different timescales.

- Reality Mining Dataset: Uncovering temporal patterns in cellphone logs of 100 MIT subjects:
  - 75 techies: faculty and students (both freshmen and seniors)
  - 25 MBA-students
  - Each subject equipped with Nokia smart phone logging:
    - Call logs, Bluetooth devices in proximity, cell-tower IDs, application usage, phone status (e.g. charging or idle)
  - Total of 450,000 hours of data on users' location, proximity, communication and device usage.

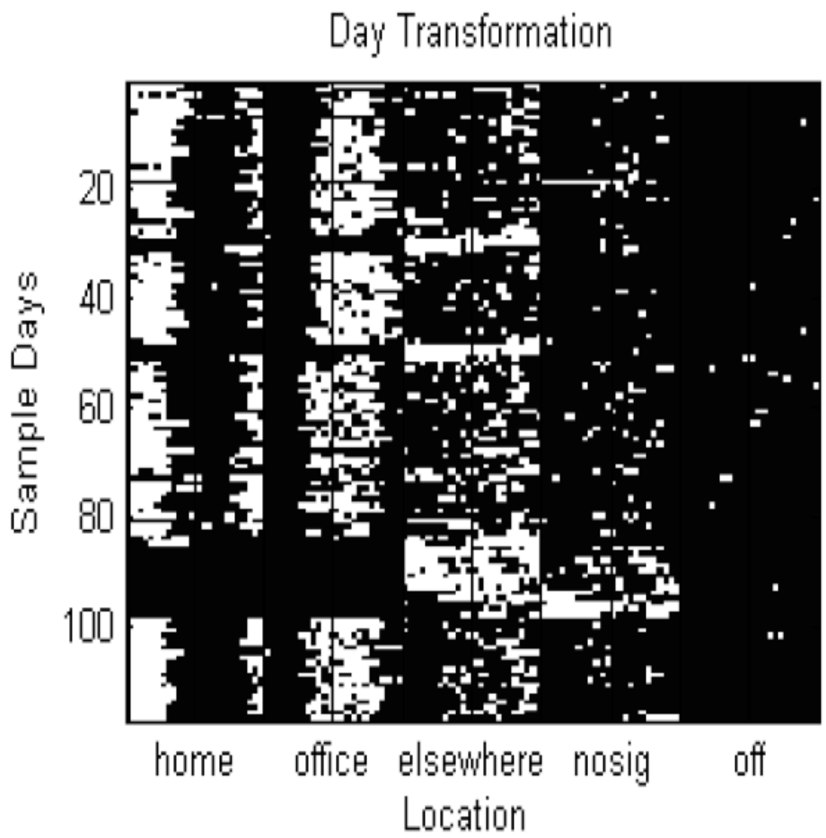
Eigenbehaviours (Eagle & Pentland)

Daily location logging for one individual

L = Multi-label location data



B = binary location data



Eigenbehaviours (Eagle & Pentland)

B = binary behaviour matrix for individual: size $= D \times H$ where $H = 24n$

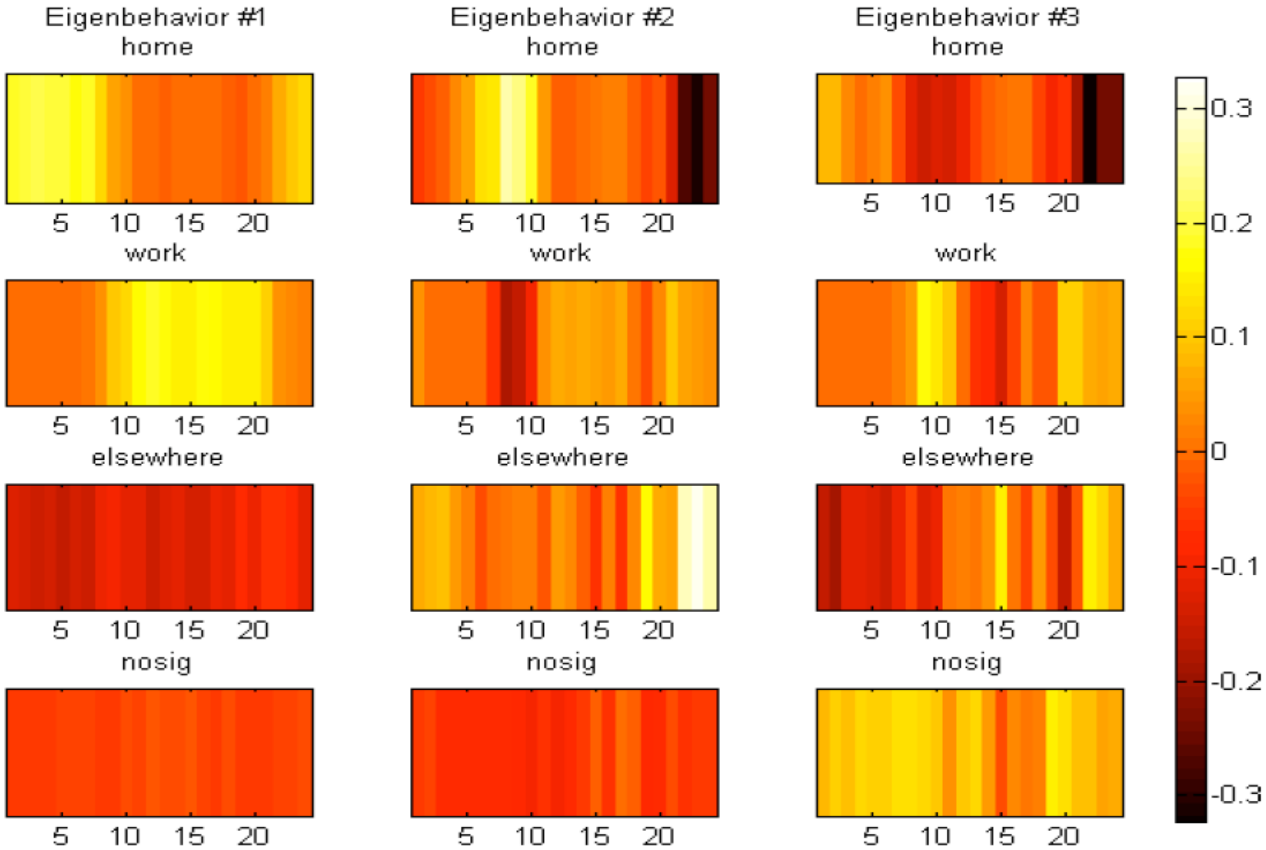
Covariance matrix: $C = \frac{1}{D}(B - \bar{b})^T(B - \bar{b})$

Eigenbehaviours defined as eigenvectors: $Cu_k = \lambda_k u_k$

Eigenbehaviours (Eagle & Pentland)

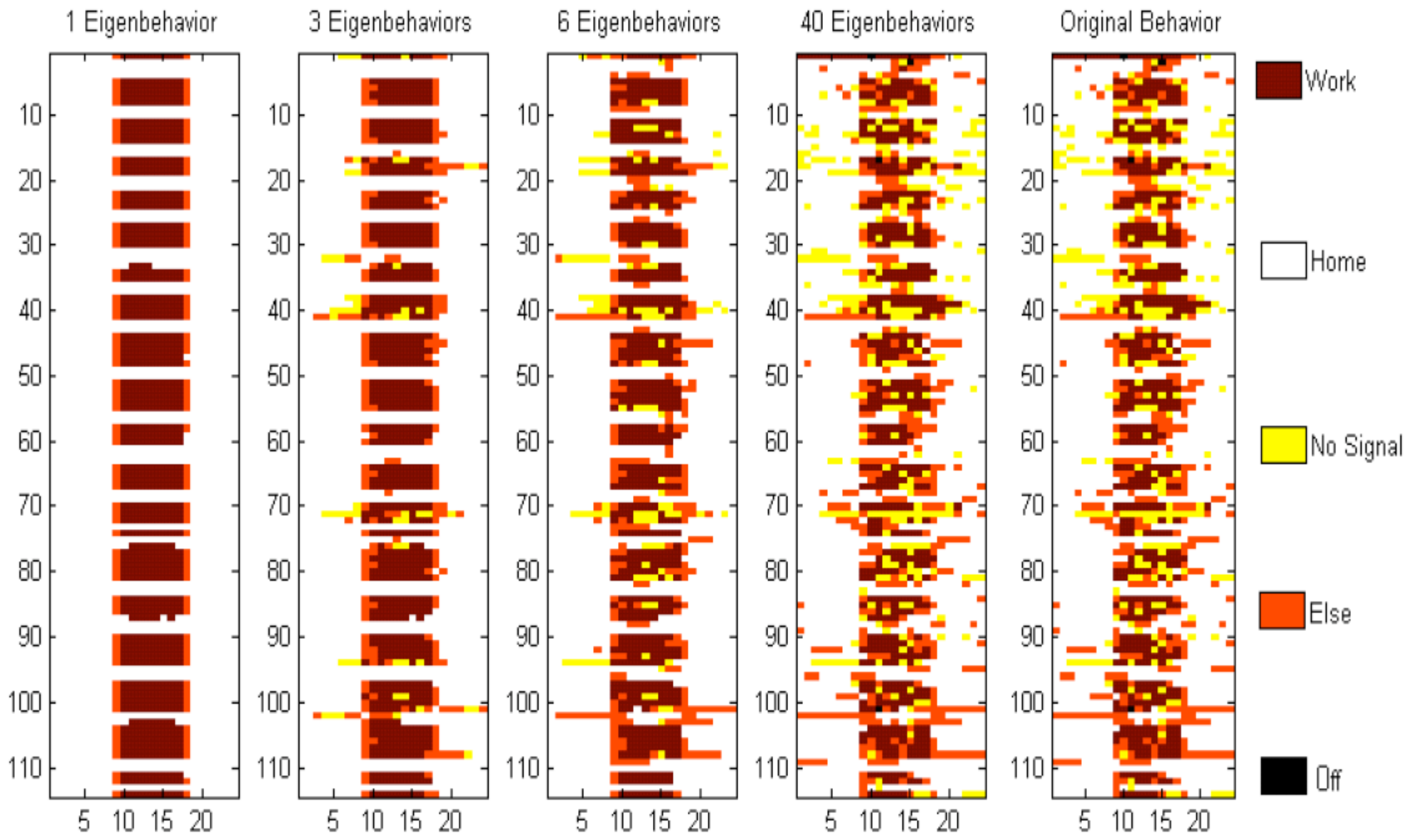
- Principal eigenvectors(behaviours) for 1 individual (long-term correlation!)

Week day | Weekend day | Poor reception



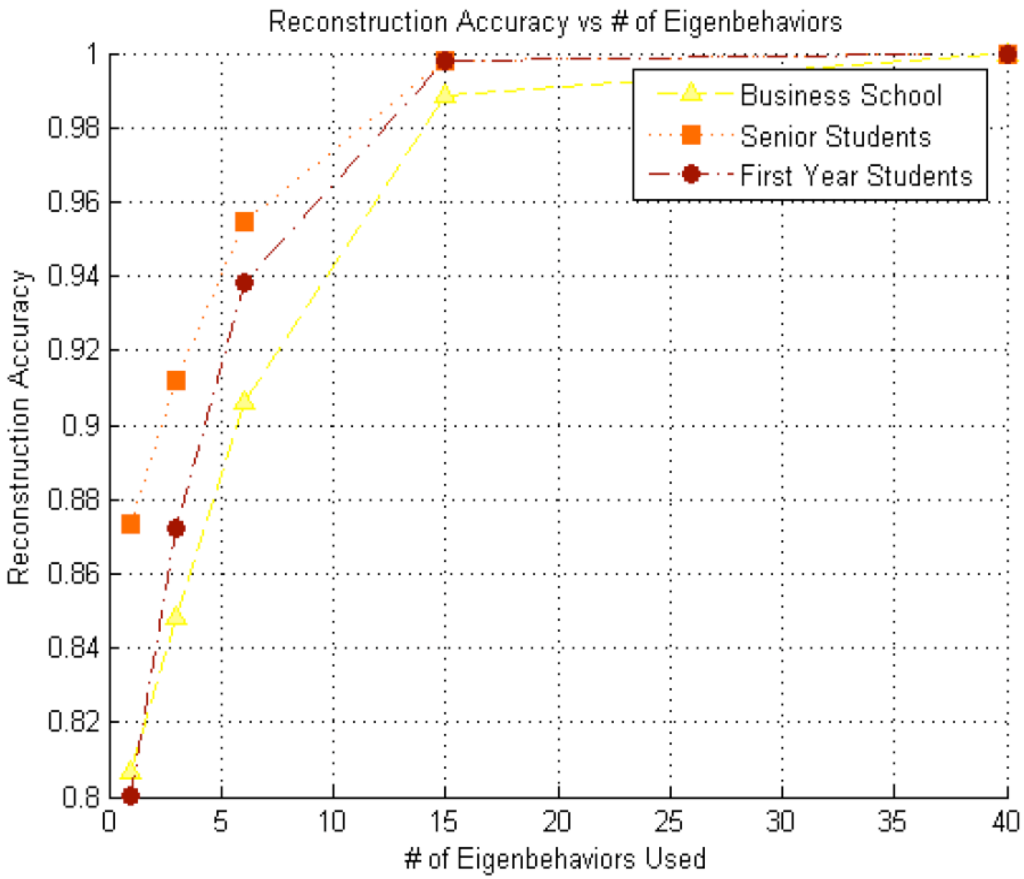
Eigenbehaviours (Eagle & Pentland)

Reconstruction of individual behaviour as function of eigen vectors

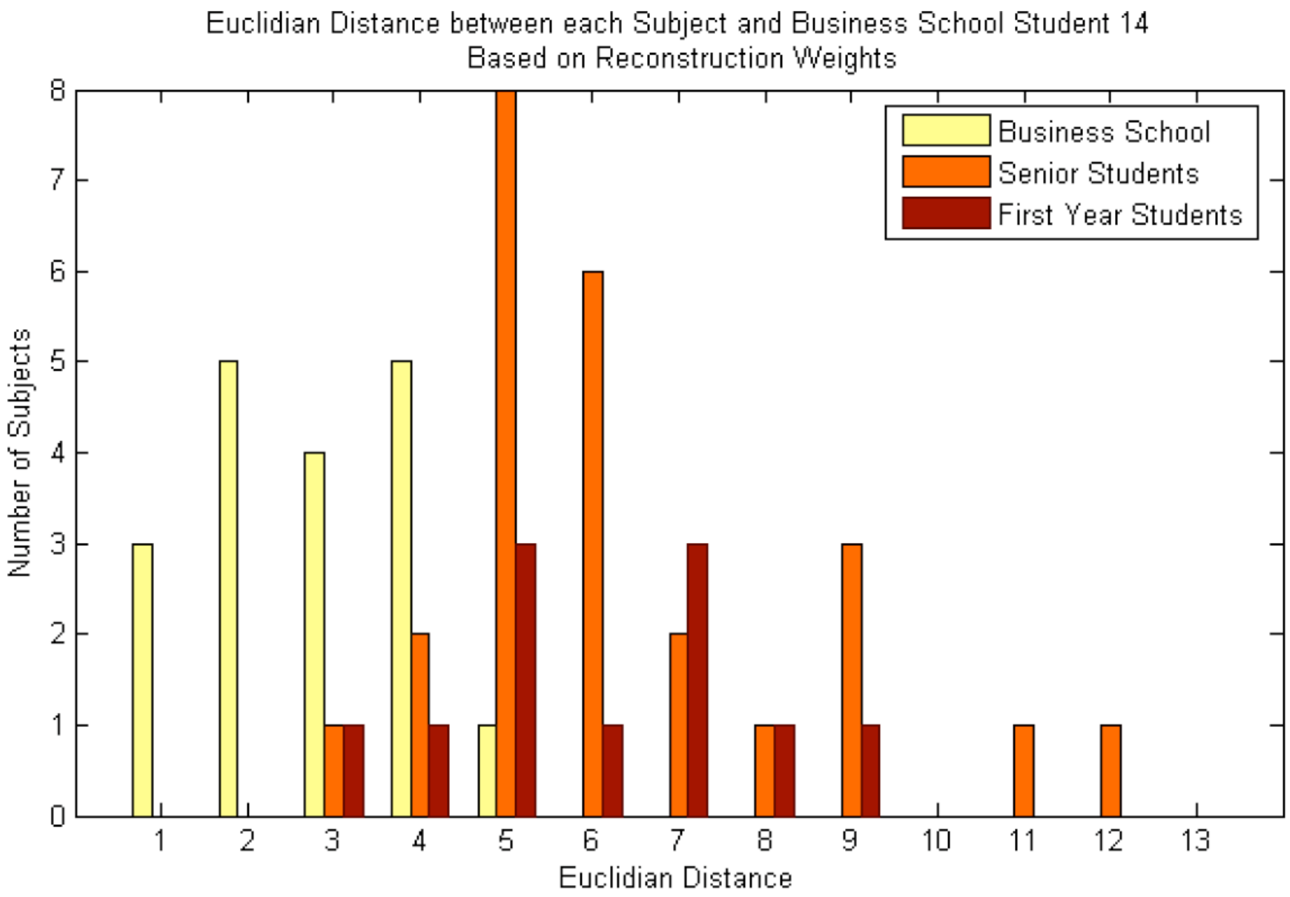


Eigenbehaviours (Eagle & Pentland)

Behaviour reconstruction accuracy versus number of eigenbehaviours



Eigenbehaviours (Eagle & Pentland)



Overview

- Learning and Mining Temporal Patterns
- Different Approaches
  - Markov Models
  - Eigenbehaviours
  - **Compression-based Approaches**
    - Lempel-Ziv
    - Active Lempel-Ziv
    - Lempel-Ziv-Welch
- T-Patterns
  - Basic Algorithm
  - Proposed Approach

Compression Based Approaches

- Lempel-Ziv (LZ78)
  - Find a dictionary of patterns
  - Seek the dictionary that allows the best compression
- Lempel-Ziv-Welch (LZW)
  - Use basic events to bootstrap dictionary
- Active Lempel-Ziv (Active LeZi)
  - Use a sliding window to extract all possible sequences of a given length

LZ78

Algorithm 1 LZ78

```
dictionary ← ∅
w ← ∅
while v = next symbol != ∅ do
   if w.v in dictionary then
      w ← w.v
   else
      add w.v to the dictionary
      emit (w.v)'s index (for compression purpose only)
      w ← ∅
   end if
end while
```

LZW

Algorithm 3 LZW

```
dictionary ← pre-defined set of symbols
w ← ∅
while v = next symbol != ∅ do
   if w.v in dictionary then
      w ← w.v
   else
      add w.v to the dictionary
      emit (w.v)'s index (for compression purpose only)
      w ← v
   end if
end while
```

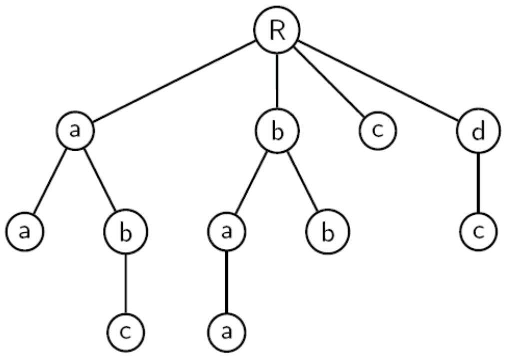
Active LeZi

Algorithm 2 Active LeZi

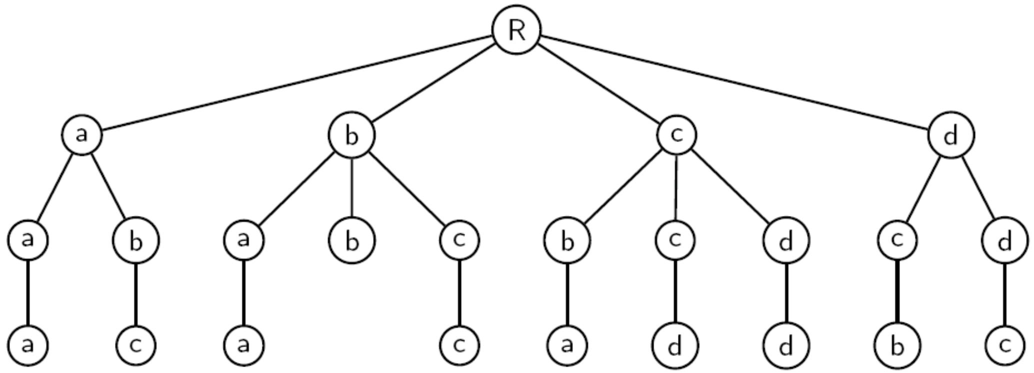
```
dictionary ← ∅
patterns ← ∅
w ← ∅
window ← ∅
Max_LZ_length ← 0
while v = next symbol != ∅ do
   if w.v in dictionary then
      w ← w.v
   else
      add w.v to the dictionary
      if length(w.v) > Max_LZ_length then
         Max_LZ_length ← length(w.v)
      end if
      w ← ∅
   end if
   window ← window.v
   if length(window) > Max_LZ_length then
      delete window[0]
   end if
   patterns ← patterns ∪ all possible subsequences of window
end while
```

Compression Based Algorithms - Example

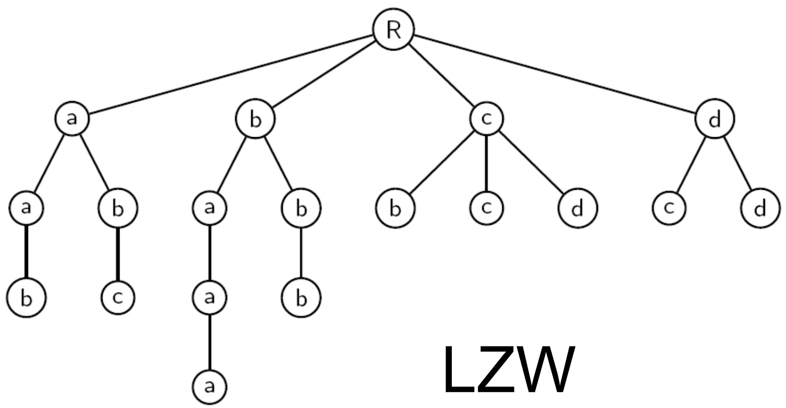
- Sample sequence : $aaababbbbaabccddcbaaa$



LZ78



Active LeZi



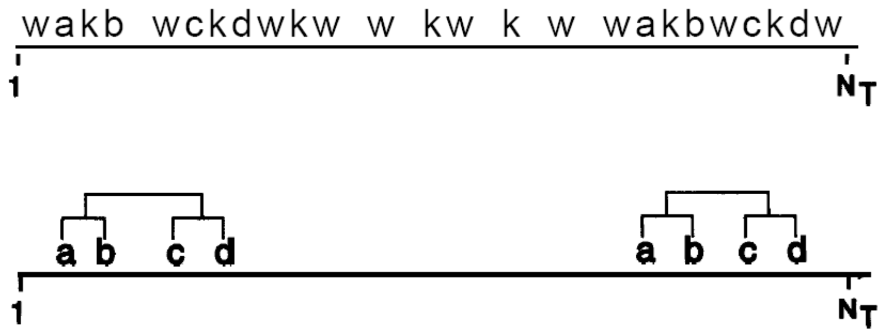
LZW

Overview

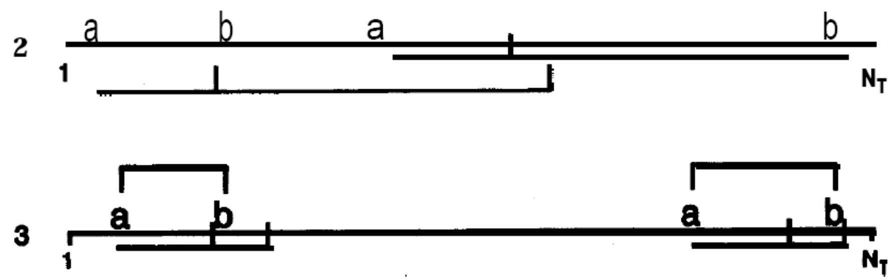
- Learning and Mining Temporal Patterns
- Different Approaches
  - Markov Models
  - Eigenbehaviours
  - Compression-based Approaches
    - Lempel-Ziv
    - Active Lempel-Ziv
    - Lempel-Ziv-Welch
- **T-Patterns**
  - Basic Algorithm
  - Proposed Approach

T-Patterns for Symbolic Time Series

Data stream = time series of symbols (e.g. labeled events)



Explicit modeling of inter-event time intervals:
Critical Interval [t1,t2] for (A,B) event: if A occurs at t0 then there are significantly more B occurrences in [t0+t1, t0+t2]:



T-Patterns

- Magnusson proposed T-patterns for mostly social sciences applications

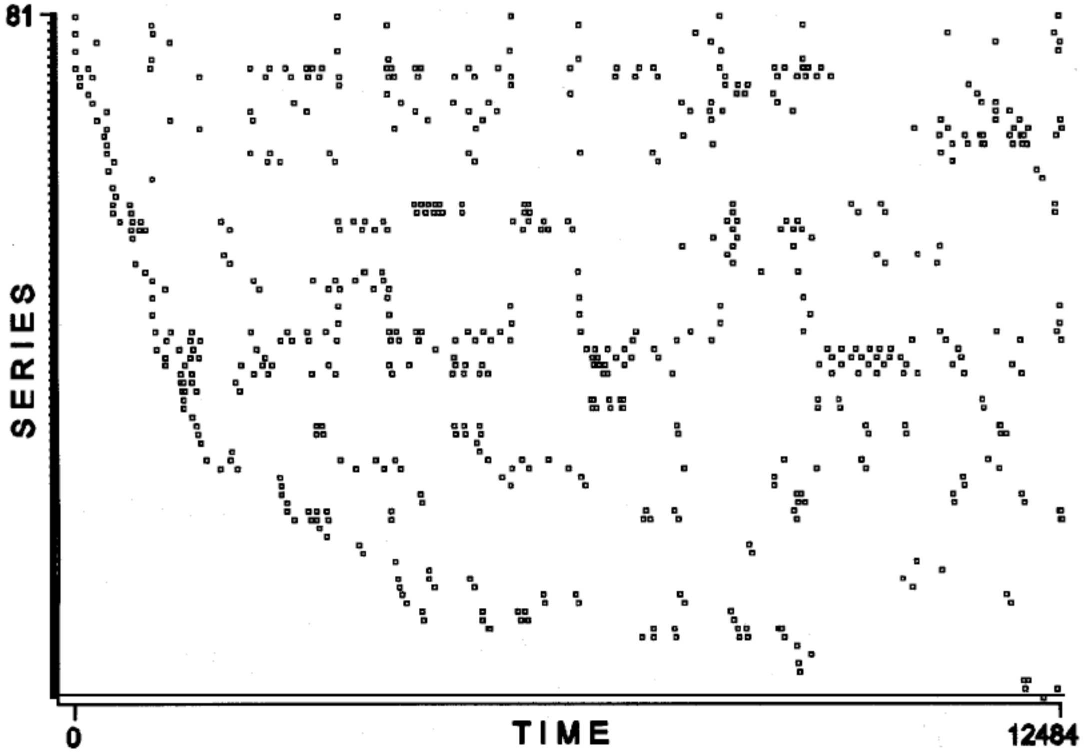
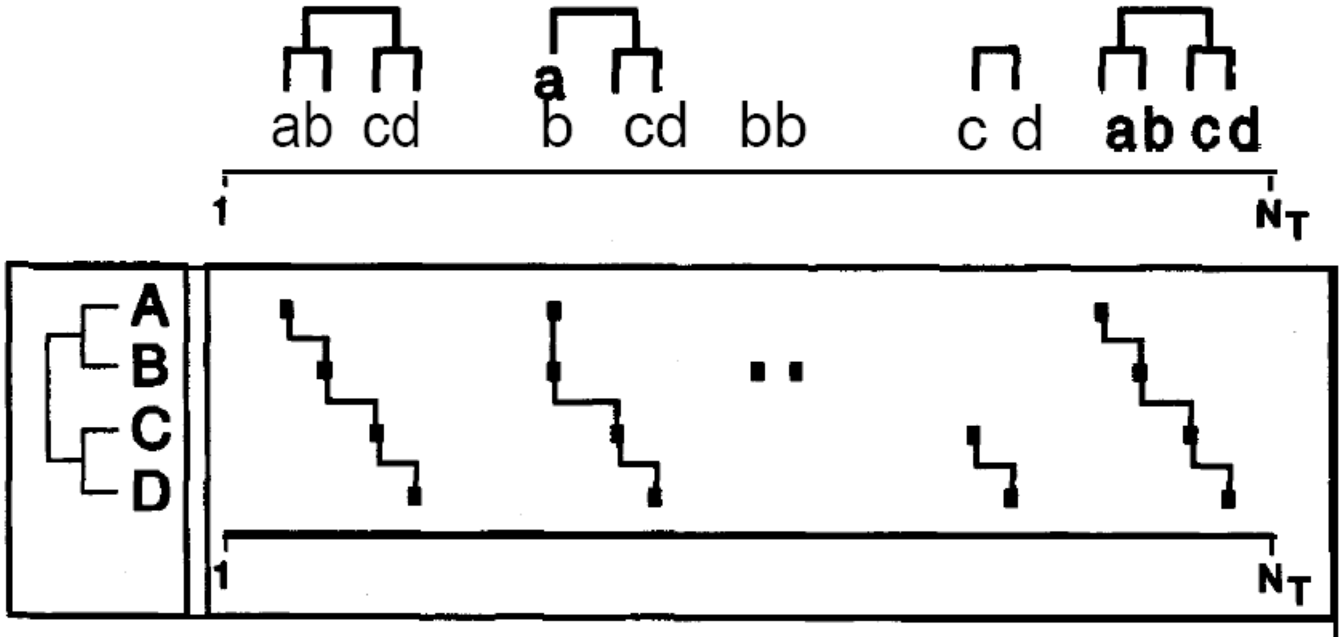


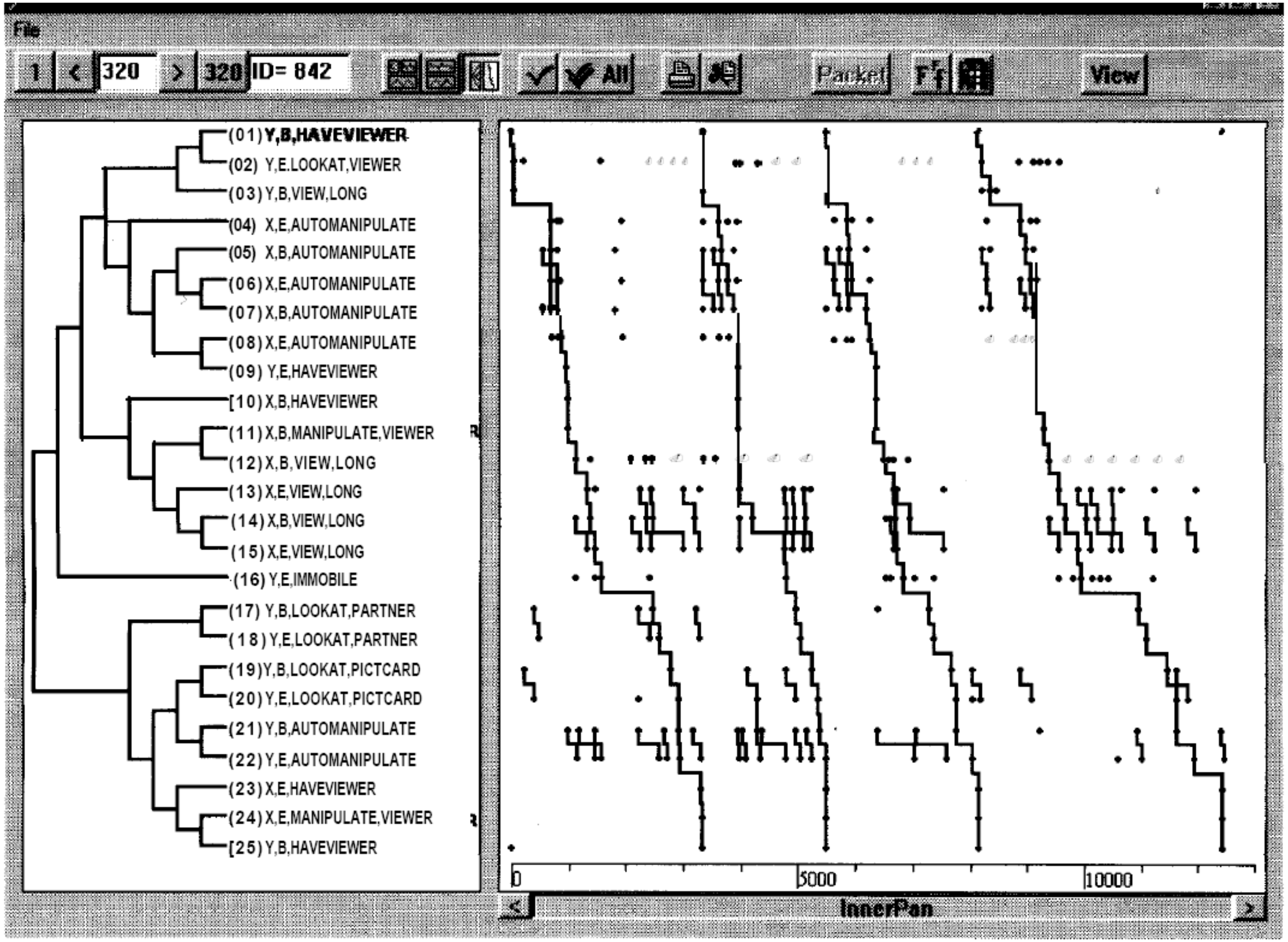
Figure 5. This figure shows a behavior record of the type described in the text. It consists of 81 series of occurrence times (1 for each coded event type) arbitrarily ordered according to their first occurrence time. The behavior was coded from a digitized video recording of approximately 13:52 min of continuous object play between two 5-year-old children (see the text). Time is in 1/15 sec (i.e., in digital video frames).

T-Patterns

- Software package THEME
  - Systematically search for critical intervals for all AB pairs
  - Hierarchical search: Assign new label to most significant pair and resume search



T-Patterns



File
1
320
320
ID= 842
All
Packet
View
(01) Y,B,HAVEVIEWER
(02) Y,E,LOOKAT,VIEWER
(03) Y,B,VIEW,LONG
(04) X,E,AUTOMANIPULATE
(05) X,B,AUTOMANIPULATE
(06) X,E,AUTOMANIPULATE
(07) X,B,AUTOMANIPULATE
(08) X,E,AUTOMANIPULATE
(09) Y,E,HAVEVIEWER
(10) X,B,HAVEVIEWER
(11) X,B,MANIPULATE,VIEWER
(12) X,B,VIEW,LONG
(13) X,E,VIEW,LONG
(14) X,B,VIEW,LONG
(15) X,E,VIEW,LONG
(16) Y,E,IMMOBILE
(17) Y,B,LOOKAT,PARTNER
(18) Y,E,LOOKAT,PARTNER
(19) Y,B,LOOKAT,PICTCARD
(20) Y,E,LOOKAT,PICTCARD
(21) Y,B,AUTOMANIPULATE
(22) Y,E,AUTOMANIPULATE
(23) X,E,HAVEVIEWER
(24) X,E,MANIPULATE,VIEWER
(25) Y,B,HAVEVIEWER
0
5000
10000
InnerPan

T-Patterns

Confidence testing for critical interval (CI) for A→B event:

- Find all A-events and search for first B-event and record its lagtime t
- Assume number of B-events in interval [t1,t2] trailing A equals N_{AB}
- Is this significantly different from expected value (if we assume independence)? Compute p-value:

$$
\begin{aligned}
p &= P(N_{AB} \text{ B-events or more} \mid \text{ A, B are independent}) \\
&= 1 - P(\text{strictly less than } N_{AB} \text{ B-events} \mid \text{ A, B are independent}) \\
&= 1 - \sum_{k=0}^{N_{AB}-1} P(\text{exactly } k \text{ B-events} \mid \text{ A, B are independent}) \\
&= 1 - \sum_{k=0}^{N_{AB}-1} \binom{N_A}{k} (1-\pi_0)^k \pi_0^{(N_A-k)}.
\end{aligned}
$$

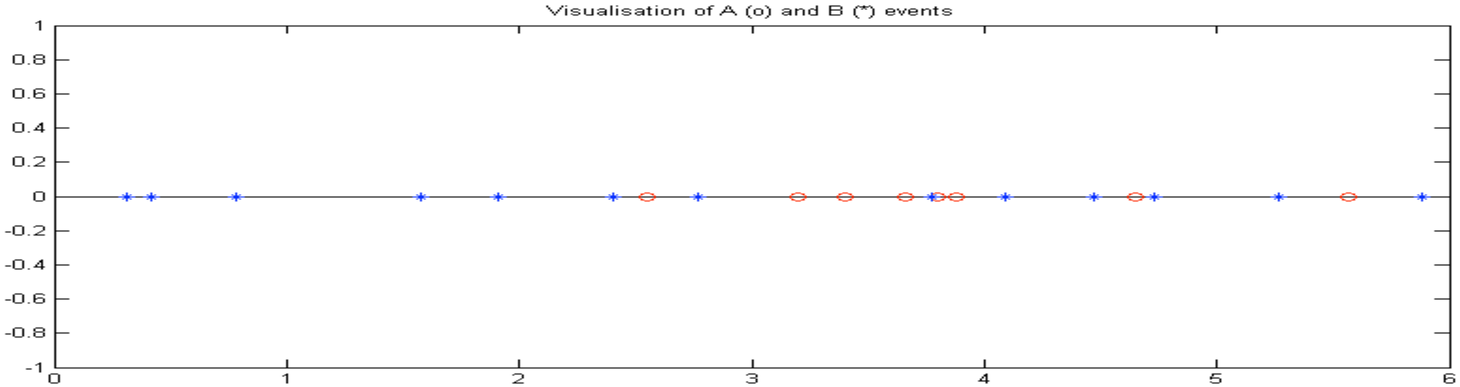
T-Patterns

- Criticisms:
- 1- repeated significance testing generates many spurious intervals (false positives!)
- 2- too slow for real-time operation in an AmI environment

- Proposed Modifications:
- 1- Start by testing independence between A and B process (as a whole);
- 2- If they are dependent, model B-lag times as 2-component GMM:
  - Peaked component: identifying typical lag-time (CI)
  - Broad component: collecting all unrelated B-occurrences

T-Patterns: Independence Testing

- Point processes A and B are *independent* if knowledge about A hasn't any prediction value for B and vice versa:

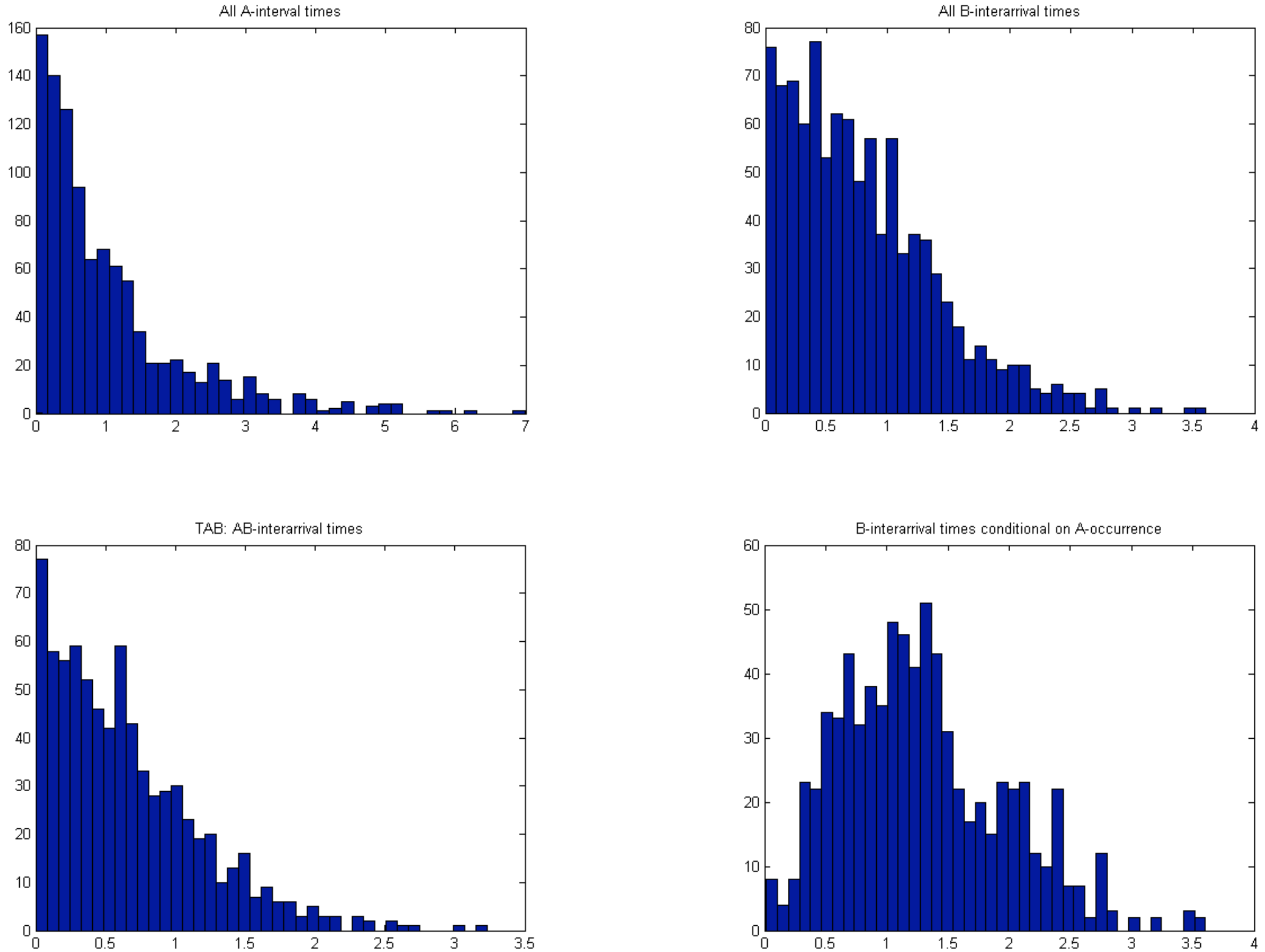


- To test this: Find for each A-event the two flanking B-events and compute A's relative position (between 0 and 1) in this B-interval:

$$T_{AB}(k) = B_{k^*} - A_k \quad \text{where} \quad k^* = \arg\min\{j \mid B_j > A_k\}$$

$$U(k) = \frac{T_{AB}(k)}{\tilde{T}_B(k)} = \frac{B_{k^*} - A_k}{B_{k^*} - B_{k^*-1}}$$

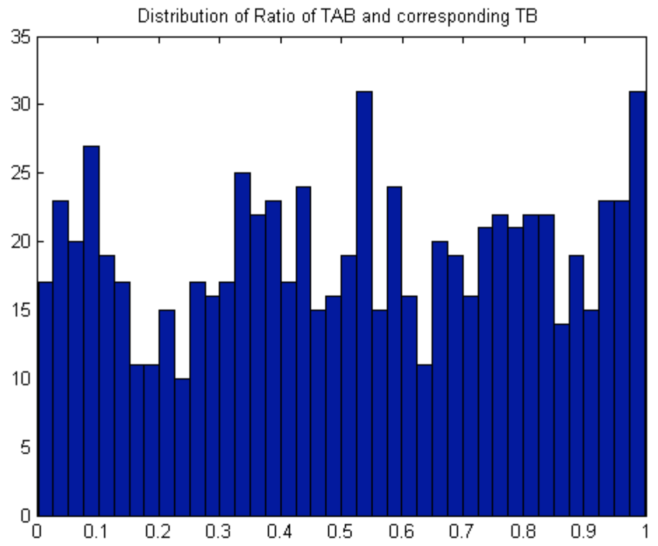
T-Patterns: Independence Testing

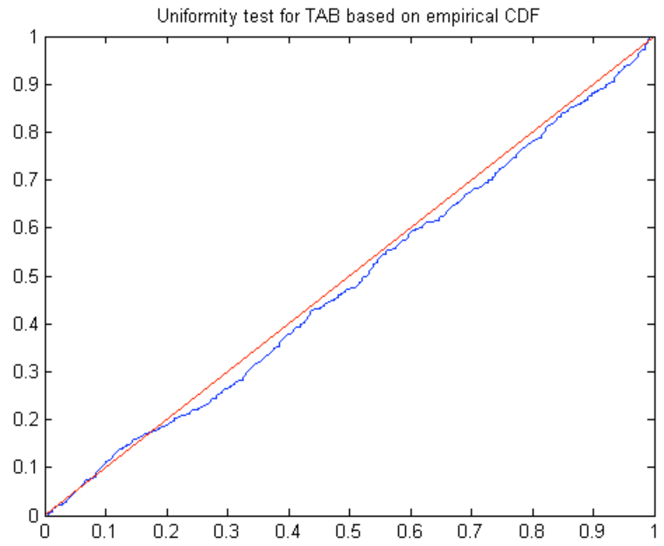


T-Patterns: Independence Testing

- If A and B are independent, then the ratio U(k) is uniformly distributed between 0 and 1:

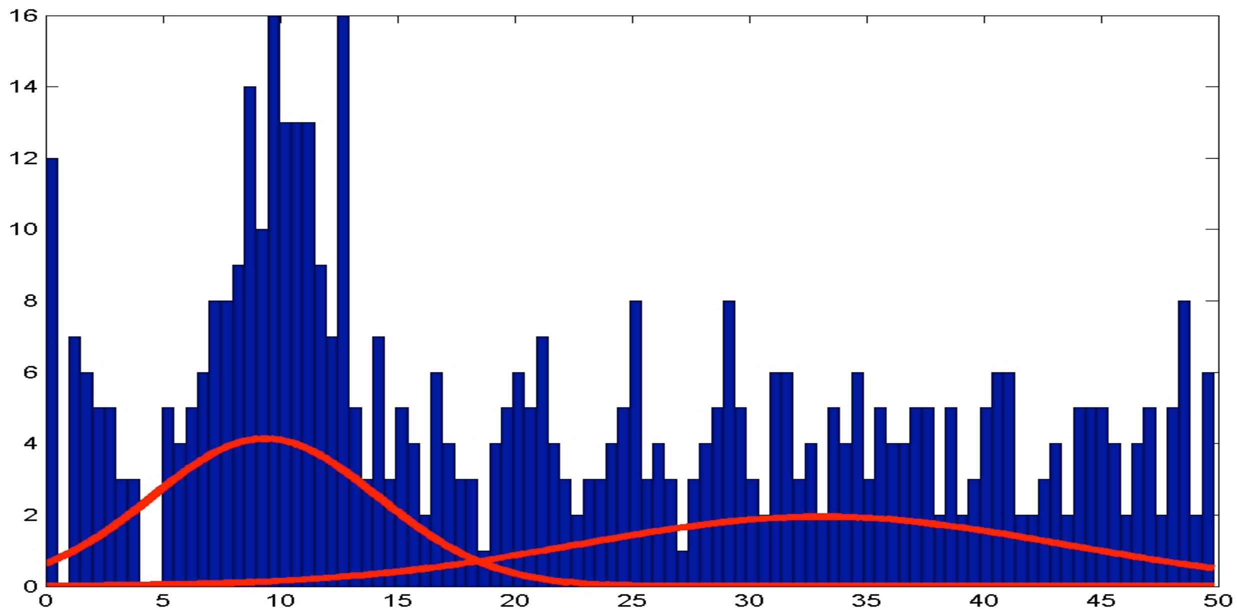
$$U(k) = \frac{T_{AB}(k)}{\tilde{T}_B(k)} = \frac{B_{k^*} - A_k}{B_{k^*} - B_{k^*-1}}$$



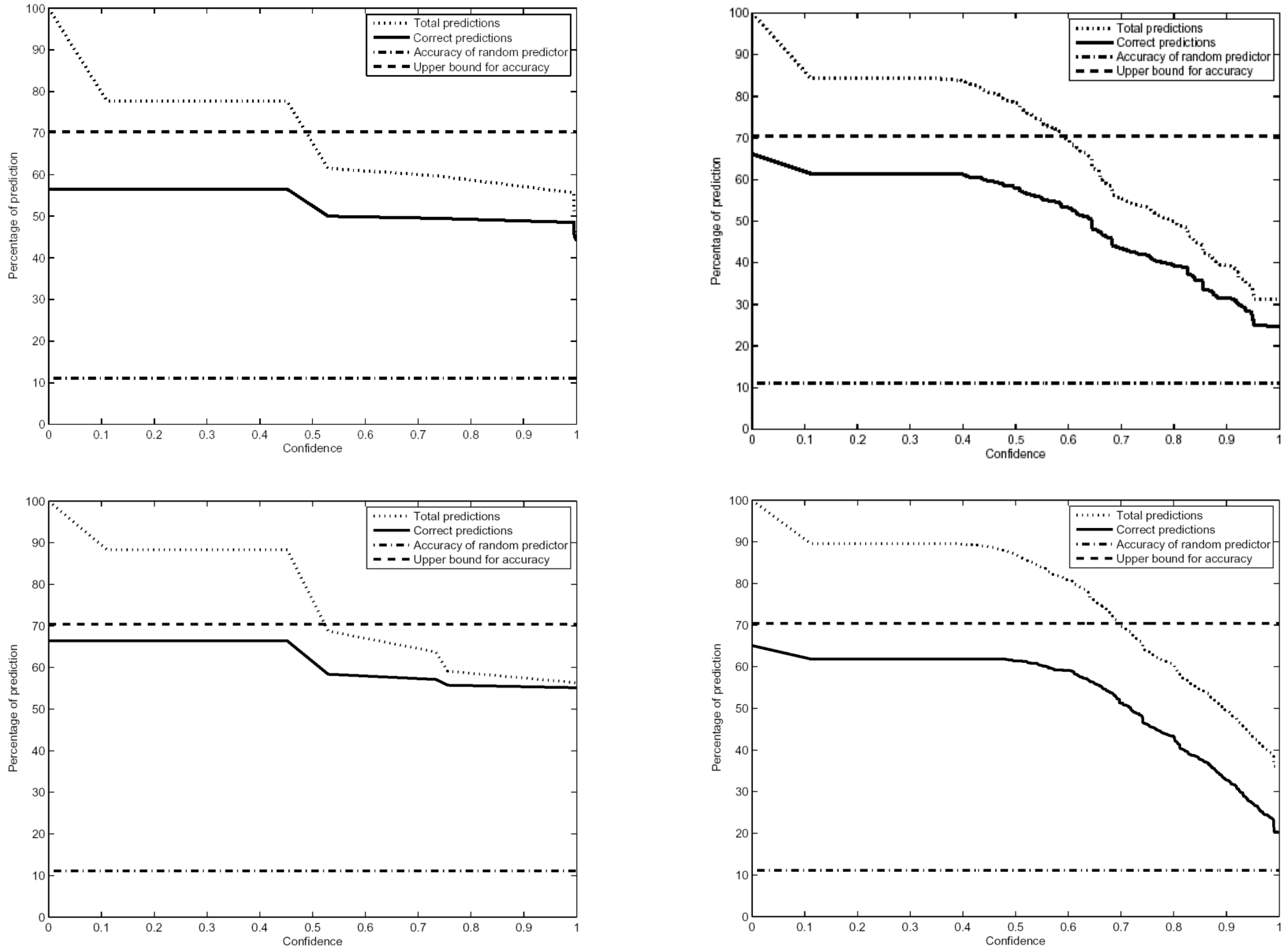


T-Patterns: Gaussian Mixture Modeling

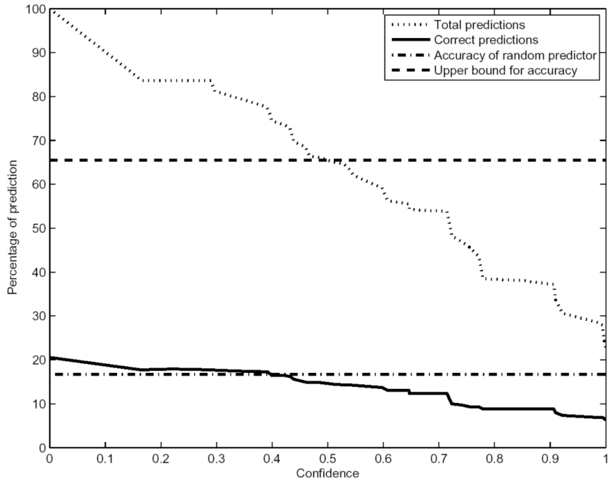
- The critical interval is given by the mean and standard deviation of a Gaussian component.
- The remaining events are modeled with a second, broader and flatter Gaussian.

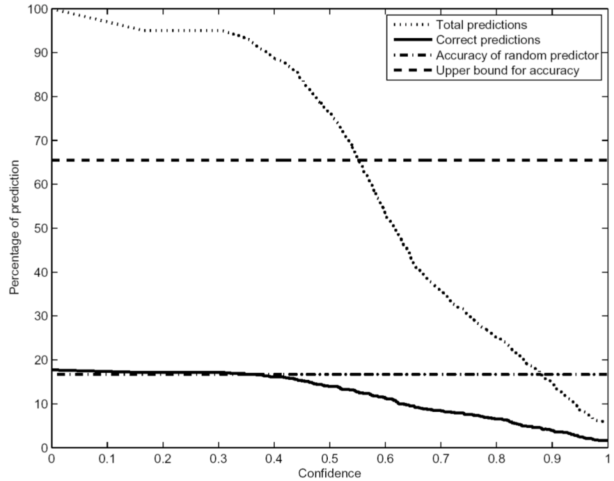


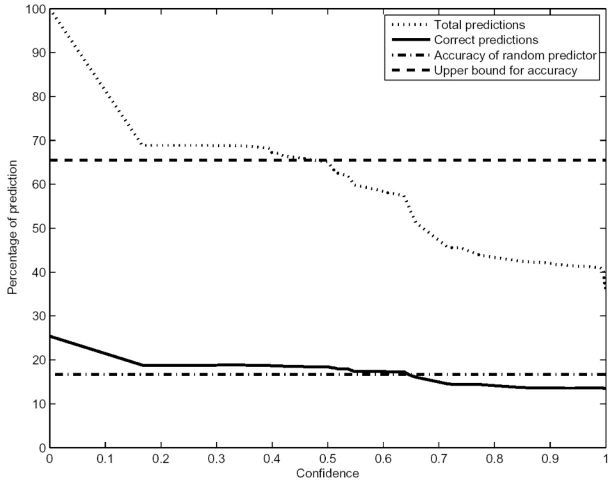
Simple Setup

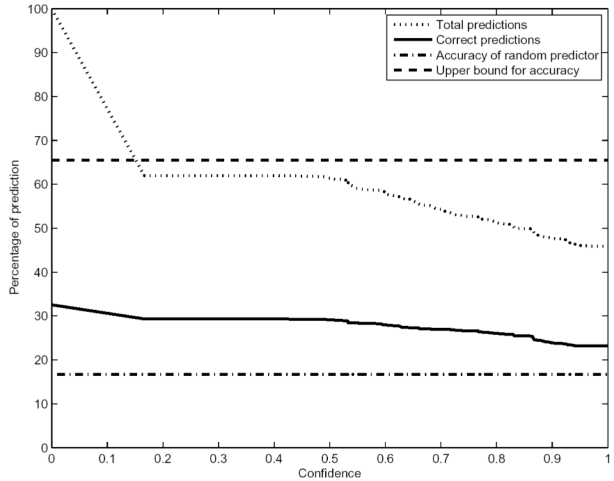


Difficult Setup









T-Patterns: Independence Testing

- Experiment: persons walking through home activating interruption sensors

	Layout 1		Layout 2	
	1 person	2 persons	1 person	2 persons
LZ	29.8	17.7	56.5	13.2
ALZ	21.1	18.8	66.4	19.6
LZW	28.9	22.0	60.5	15.1
T-patterns	28.8	17.1	61.5	24.2
GMM T-patterns	**34.8**	**29.3**	**61.9**	**48.3**

Table 1. Percentage correct predictions at the 20% confidence level.

Conclusions

- We adopt the T-pattern method to fast discovery of behaviour patterns in simple sensor data
- Temporal information is not discarded as in "next event" prediction approaches
- Dictionary-based simulation allows performance measurement
- Many possible applications
  - Behaviour analysis in sensing-endowed environments (e.g. smart homes, offices)
  - Automatic layout discovery
  - Anomaly detection
  - Process control and management

Thank you!