Feature Extraction in Wireless Personal and Local Area Networks

29. October 2003, Singapore

Rene Mayrhofer
Institut für Praktische Informatik
Johannes Kepler Universität Linz, Austria
rene@soft.uni-linz.ac.at

Content

Context awareness

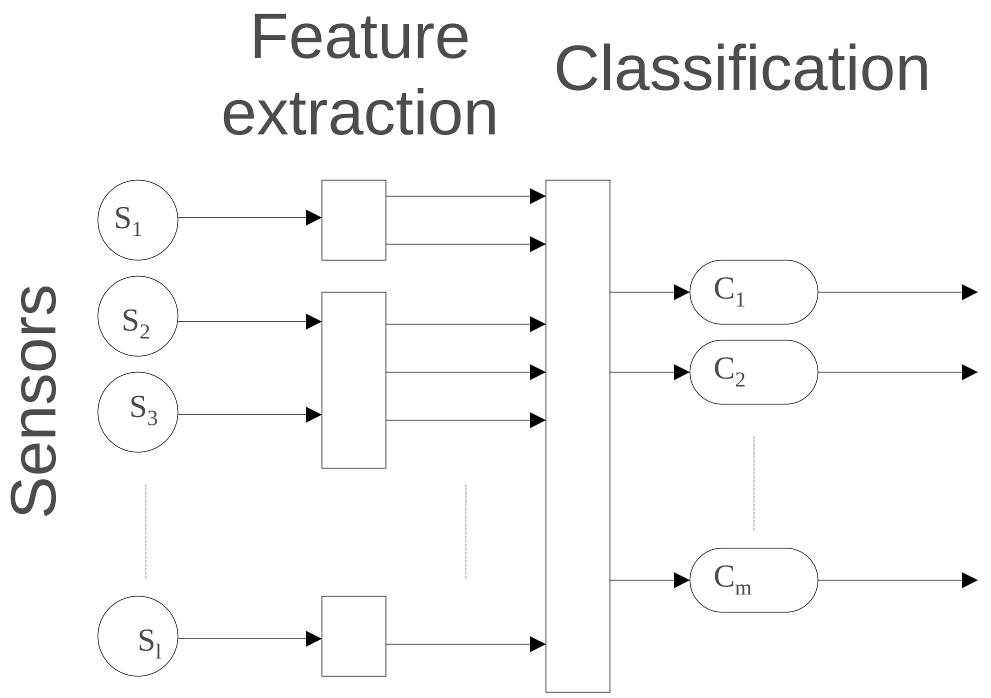
- Many definitions for context, e.g. by Dey as

  any information that can be used to characterize the situation of an entity, where an entity can be a person, place or a physical or computational object

- Context has many aspects
- Using multiple simple sensors seems more reasonable to capture different aspects of context (cf. Gellersen et.al.)

Content

Architecture



$<s_1, s_2, ..., s_l>_t \longrightarrow <f_1, f_2, ..., f_n>_t \longrightarrow <c_1, c_2, ..., c_m>_t$

Input vector (Sensor vector) — Feature vector — Class vector

Content

- Introduction
- Architecture
- Step 1: Sensor data acquisition
- Step 2: Feature extraction
- Step 3: Classification
- First results

Sensors for (mobile) information appliances

Typical „sensors“ available for monitoring the user context:

- Time
- Application/Window manager
- Brightness
- Microphone
- Bluetooth
- Wireless LAN
- Docked / undocked

Other suitable sensors can be connected:

- GPS
- GSM
- Compass
- Accelerometer
- Tilt sensor
- Temperature sensor
- Pressure sensor

Sharing of sensor data between appliances

Content

Feature Extraction: Different Types of Features

- Raw sensor data is transformed into more meaningful features
- Feature extraction exploits domain-specific knowledge
- Multiple features extracted from a single sensor

⇒ **High-dimensional input vectors**

- Different types of features:
  - Numerical (continuous) sensors: e.g. brightness sensor
  - Numerical (discrete) sensors: e.g. number of access points in range
  - Ordinal sensors: e.g. day of week
  - Nominal sensors: e.g. WLAN-SSID, list of Bluetooth devices in spatial proximity

Feature Extraction: Handling Nominal features

- Usual way to handle nominal input dimensions: code *as binary inputs*
  - E.g. every department has its own WLAN ESSID configured for its access point $\Rightarrow$ by querying the currently used ESSID, a device can query the department it is currently in
  - Create a (binary) feature for each department, which assumes 1 if the matching ESSID is found and 0 if is it not found
  - Feasible for small sets of (known) nominal values
  - Infeasible for large/infinite sets: e.g. WLAN MAC addresses would need 2^{48} feature dimensions

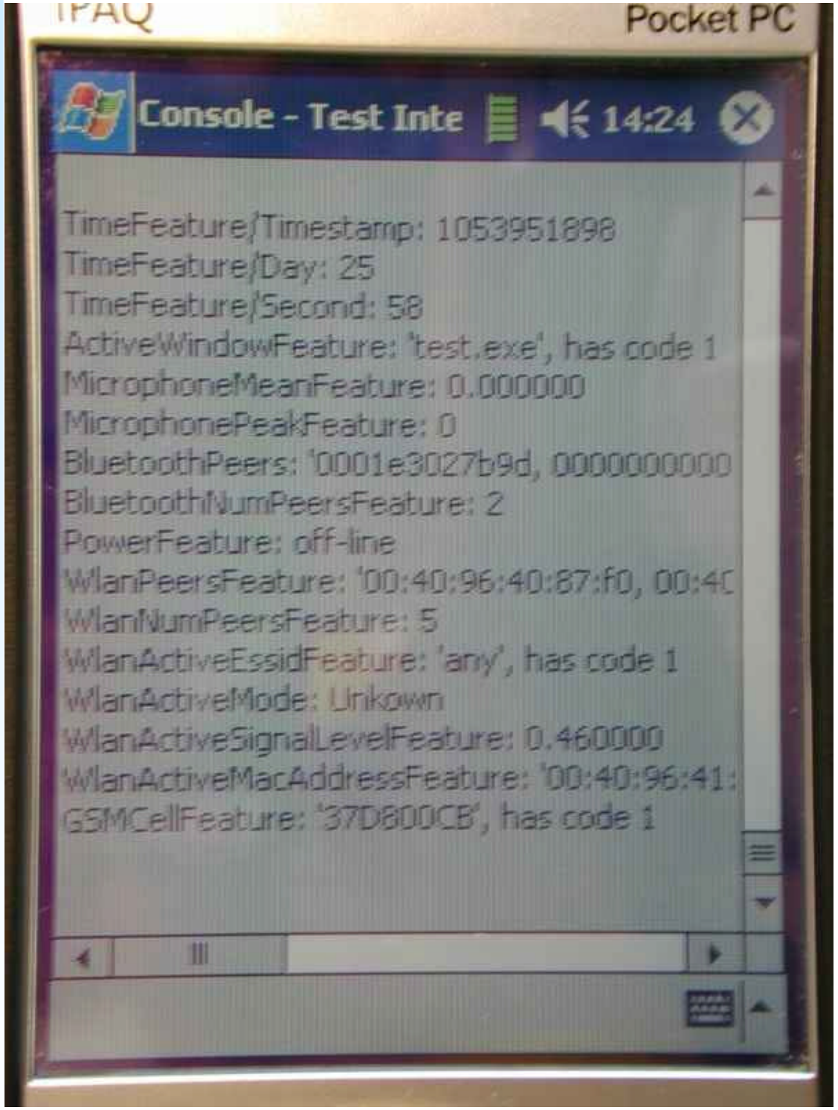
- But: only two operations necessary on each feature for most classifiers:
  - Distance metric
  - Adaptation operator

  $\Rightarrow$ When each feature dimension implements these operations, nominal values can be used directly for classification

Feature Extraction: currently implemented features

- Bluetooth:
  - Number of MAC addresses in range (reachable via inquiry): numerical (discrete)
  - List of MAC addresses in range: reachable via inquiry: nominal
  - List of MAC addresses in range: layer 2 connection can be established with configurable maximum “ping” time and minimum link quality: nominal
- WLAN (802.11b):
  - Currently used ESSID: nominal
  - Current mode (managed, ad-hoc, master): nominal
  - Current signal level / link quality (only for managed mode): numerical (continuous)
  - Access point MAC address associated to (only for managed mode): nominal
  - Number of MAC addresses in range (access points or devices in ad-hoc mode): numerical (discrete)
  - List of MAC addresses in range (access points or devices in ad-hoc mode): nominal

Feature Extraction on a current PDA



- With current computational power easily possible
- Every feature implements distance metric and adaptation operator

⇒ On-line classifications possible with any combination of features (user-selectable and loaded dynamically during startup)

Content

- Introduction
- Architecture
- Step 1: Sensor data acquisition
- Step 2: Feature extraction
- Step 3: Classification
- First results

Classification: Introduction

- Classifies feature vectors and finds common patterns in sensor data
- Different types of classification algorithms
  - Type (partitioning / hierarchical)
  - Soft / hard classification
  - Supervised / unsupervised

- Requirements for classifying user context in information appliances:
  - On-line learning
  - Adaptivity
  - Variable number of classes and variable topology
  - Soft classification
  - Noise resistance
  - Limited resources
  - Simplicity
  - Interpretability of classes / protection of data privacy

Classification: Algorithms

Algorithm	Network topology	Topology preserving	Competitive
SOM	fixed	yes	soft
RSOM	fixed	yes	soft
K-Means	fixed	no	hard
Leader	variable	no	hard
Growing K-Means	variable	no	hard
Neural Gas	variable	no	soft
Neural Gas + Competitive Hebbian Learning	variable	yes	soft
Growing Neural Gas	variable	yes	soft
Incremental DBSCAN	variable	No	hard

More details in „*Recognizing and Predicting Context by Learning from User Behavior*“, Proceedings ot the International Conference on Advances in Mobile Multimedia (MoMM 2003)

Content

- Introduction
- Architecture
- Step 1: Sensor data acquisition
- Step 2: Feature extraction
- Step 3: Classification
- First results

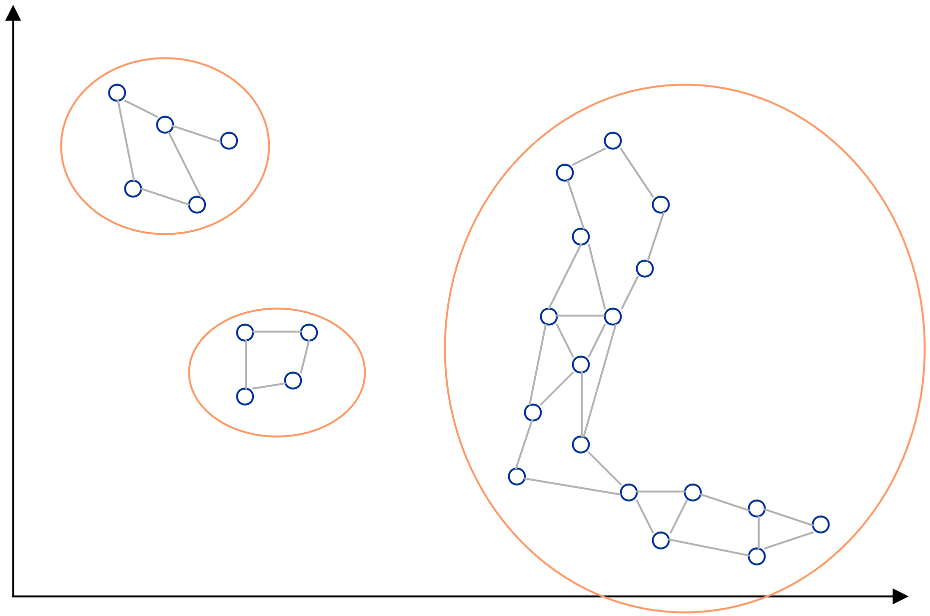
First Results: artificial data

- Artificial test data generated for testing classification with nominal features
  - 3 string features
  - Each input vector has different sets of values (sets of strings)
  - Feature 1: 5 different, but short and similar values (active application)
  - Feature 2: 11 different, clearly distinguished values with significantly different lengths (WLAN ESSID)
  - Feature 3: 30 different values with same length (12 hexadecimal digits), ca. 50 % of the values have one of two common prefixes (WLAN MAC)
- Only one iteration of a context transition sequence in artificial logs ⇒ classification algorithm is trained by presenting the same sequence multiple times (batch learning)

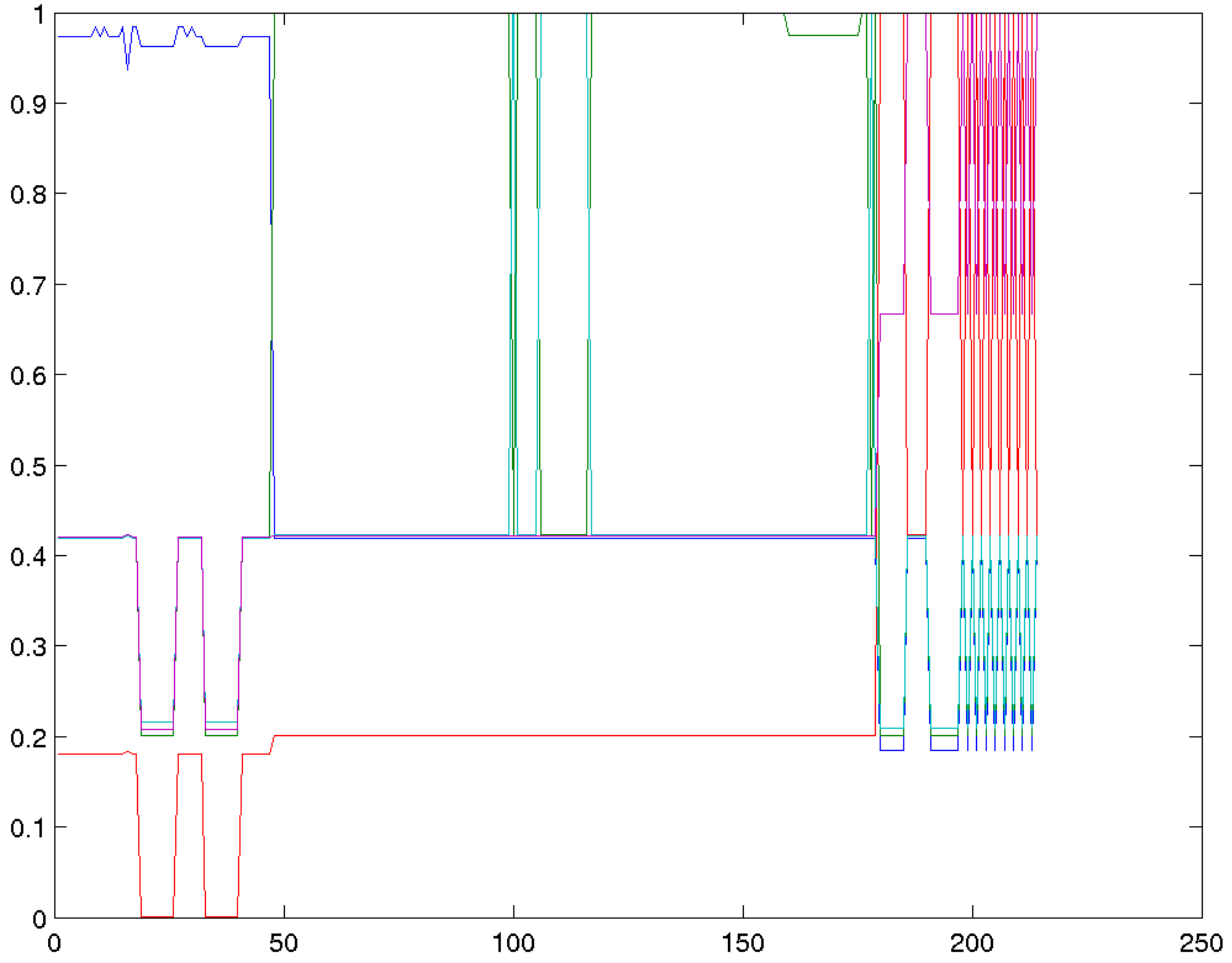
- Results:
  - 0,008 overall classification error (average distance of input data points to cluster centers) after only 10750 samples
  - 15 clusters automatically recognized, aggregated to 5 meta-clusters

First Results: explanation

- Meta-clusters formed out of clusters with edges
- Cyclic, undirected graphs consisting of components



First Results: artificial data



First Results: real-world data

- Test data gathered over 10 days for a selection of features:
  - Timestamp
  - Bluetooth: list of MAC addresses in spatial proximity
  - Bluetooth: number of peers in spatial proximity
  - Wireless LAN: list of MAC addresses in spatial proximity (clients associated to a nearby access point)
  - Currently active application
- Distance metrics and adaptation operators implemented in straight-forward ways (e.g. Hamming distance for lists of MAC addresses)
- Implementation of “Lifelong Growing Neural Gas (*LLGNG*)” as classification algorithm
- One-pass (online) run through the log file yields (multiple test runs):
  - ~ 40 clusters
  - ~ 6 meta-clusters
  - < 0.12 overall classification error

Summary

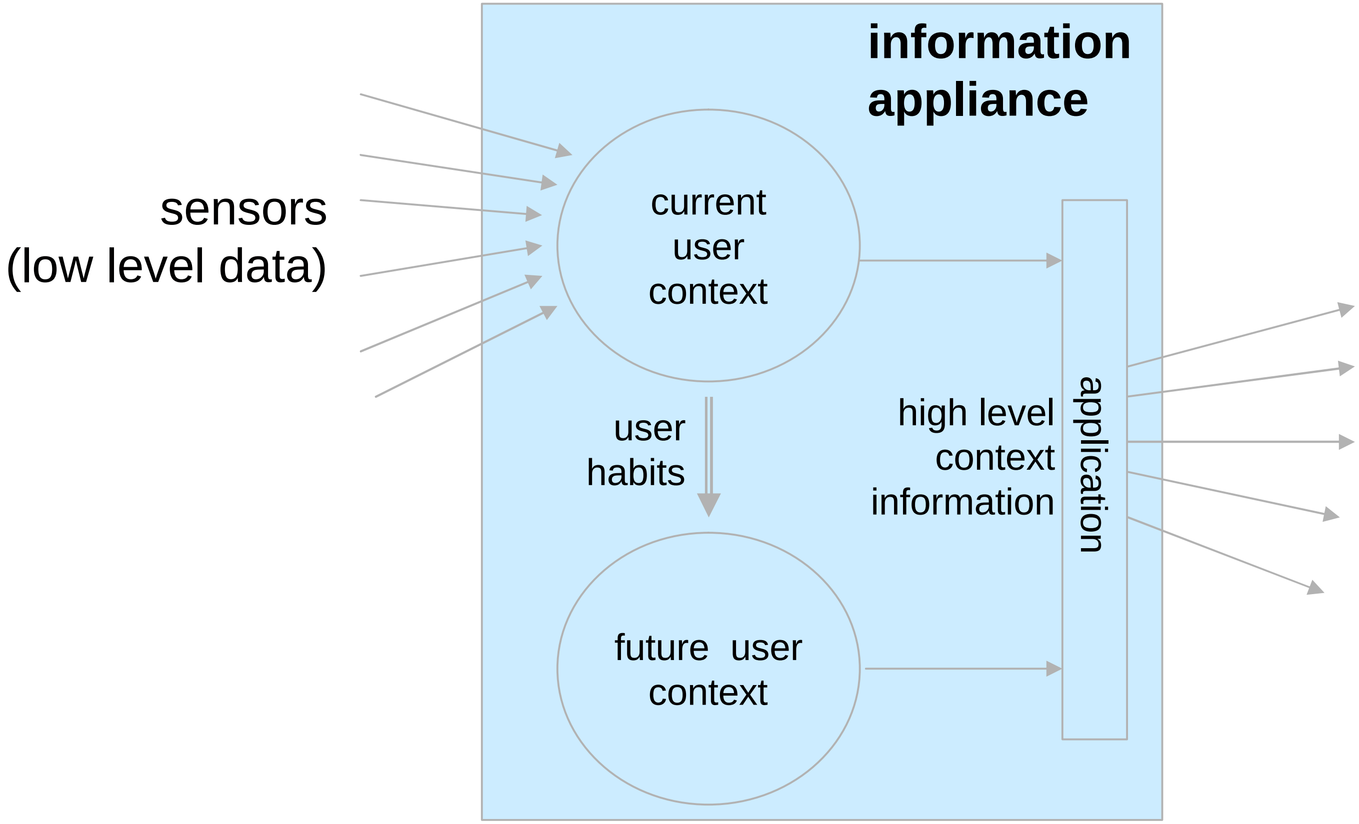
- Architecture allows recognition of user context from raw, heterogeneous sensor data in an un-supervised, non-obtrusive way in 3 steps:
  - Sensor data acquisition
  - Feature extraction
  - Classification
- Simple interfaces between the architecture layers
- Multiple, simple sensors instead of a single, complex one
- Heterogeneous feature vector handled via specific implementation of:
  - distance metric
  - adaptation operator

  for each feature
- Arbitrary types of features (also nominal and ordinal) can be handled without introducing numerous (possible infinitely many) input feature dimensions

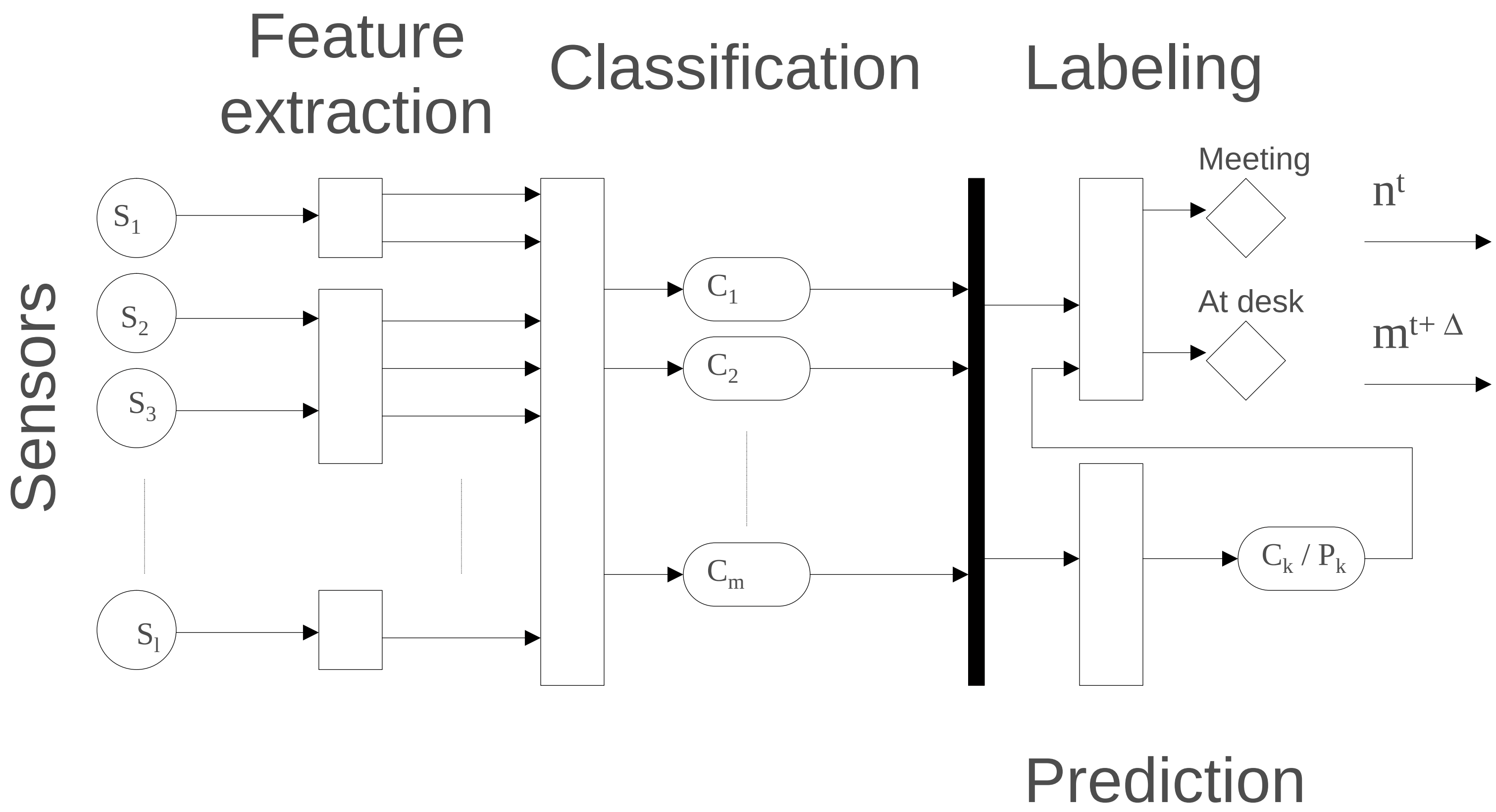
- First implementations of feature extraction and classification show promising results

Future Outlook

- Future research on non-obtrusive user interfaces for labeling user context necessary
- Prediction of context classes (by learning user habits) will allow the development of proactive, context-aware applications



Future Outlook



$\langle s_1, s_2, \ldots, s_l \rangle_t \rightarrow \langle f_1, f_2, \ldots, f_n \rangle_t \rightarrow \langle c_1, c_2, \ldots, c_m \rangle_t \rightarrow \langle c_1, c_2, \ldots, c_m \rangle_{t+\Delta}$

Input vector (Sensor vector) | Feature vector | Class vector | Future class vector

Thank you for your attention !