Managing Data in CGSNs

Sebastian Cartier,
Saket Sathe,
Dipanjan Chakraborty,
Karl Aberer
Content

1. CGSNs
2. Condense
3. Model Cover Estimation
4. Adaptive Methods
5. Datasets
6. Experiments
Motivation

- Community-driven Mobile Geo-Sensor Network

- Community-driven $\rightarrow$ No central authority
  - Different sensor quality
  - Different update rate
  - Unreliable readings
  - Uncontrollable movement of sensor nodes

- Irregular Data
  - Daytime, Season
  - Geographic situation

- Sensed Values
  - Pollution, Temperature, Radiation

- Challenge: Produce homogenous view on this data
Sensor Layer

- Deployment by data distributor
- Sensor readings are continuously updated in Database
- Each reading is represented in a tuple:
  \[ b_i = (t_i, x_i, y_i, r_i) \]

- Timestamp
- Position
- Reading value
Model Layer

- Abstraction level for raw data
- Model cover
  - More than one model
  - Single models are less complex
- Continuous update of models
- Model layer is main focus of this Project
Query Layer

- No direct access to raw data
- One or more model responsible for each query position
- Possible queries:
  - Single position
  - Continuous queries
  - Moving continuous queries
Problem Characterization

Model Cover Estimation

- One mathematical Model is not enough!
- Given: Region of interest $R$ and raw tuples of one time window $W_s$
- Partition of region $R$: $R_1, R_2, \ldots, R_p$
- Raw tuples are distributed among regions
- For each Region $R_\alpha$ we want to create a Model $M_\alpha
Problem Characterization

Model Cover Maintenance

- New points are streamed into the system: $W_{s+1}$
- Which models have to be updated
- Only update these Models
- The other models are still valid from last time window
- Reduce cost by adapting the model cover, instead of creating new model cover for each new time window
Adaptive K-Means

Adaptive Method

1. Select 2 region centers
2. Run Simple K-Means
3. Check for each region if error criteria is met
Adaptive K-Means

Adaptive Method

1. Select 2 region centers
2. Run Simple K-Means
3. Check for each region if error criteria is met
4. For each region, where error is too high:
   1. Select reading with highest error
   2. Create new region center
5. Jump to step 2, if new regions were created
## Datasets

<table>
<thead>
<tr>
<th></th>
<th>Records</th>
<th>Interval</th>
<th>Pollutant</th>
<th>Mounted on</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cabspotting</td>
<td>11 m</td>
<td>50 sec</td>
<td>-</td>
<td>Taxicab</td>
</tr>
<tr>
<td>Opensense Zurich</td>
<td>110 k</td>
<td>40 sec</td>
<td>Ozone</td>
<td>Public tram</td>
</tr>
<tr>
<td>Opensense Lausanne</td>
<td>70 k</td>
<td>60 sec</td>
<td>Ozone</td>
<td>Public bus</td>
</tr>
<tr>
<td>Safecast</td>
<td>970 k</td>
<td>5 sec</td>
<td>radiation</td>
<td>Car</td>
</tr>
</tbody>
</table>

- Cabspotting: only positioning data
- Zurich and Lausanne: clean environment
- Safecast: radiation is changing slowly and predictable in time
Error Analysis

- **Opensense Zurich**
  - H = 6 hours, P = 50
  - Random time windows
  - Plot normal percentage error

- **Safecast**
  - No significant difference with Opensense
  - DBSCAN: Number of Regions p is not controllable
Time Efficiency

- Opensense Lausanne
- Start time of time window is constant
- Normal Percentage error is constant
- Increase of $H \rightarrow$ number of raw tuples

Observations
- Complex methods are slow
- Grid based modeling is the fastest

Experiments

![Model cover creation time](chart)

- Time (milliseconds)
- No. of raw values

![Query time](chart)

- Time (milliseconds)
- No. of raw values
Model Cover Maintenance

- Training period of 6 hours
- \( H = 30 \) minutes, \( W_0, W_1, \ldots, W_{88} \) streamed into Condense
- Updating only region with high normal percentage error
- Flops: rough estimate of update cost

Observations

- Adaptive K-Means is able to adapt to data