Trajectory Regression on Road Networks

Tsuyoshi Idé (IBM Research – Tokyo)
Masashi Sugiyama (Tokyo Institute of Technology)
Agenda

- Problem setting
- Formulation
- Relationship with Gaussian process regression
- Experiments
Problem: Predict the “cost” of an arbitrary path on networks

- **Input:** arbitrary path on a network
  - A sequence of adjacent link IDs
    
    $x^{(3)} = (25020201, 24021102, 222020101, 258020001, ...)$
  - link ID defined in digital maps

- **Output:** total cost of the path
  - Scalar (e.g. travel time)

- **Training data:**
  
  $\mathcal{D} \equiv \{(x^{(n)}, y^{(n)})| n = 1, 2, ..., N\}$
  
  - $x^{(n)}$: $n$-th trajectory (or path)
  - $y^{(n)}$: $n$-th cost

Trajectory regression

$y = f(\quad)$

(“trajectory”)
Why do we focus on the total cost of a path?

- GPS produces a sequence of sparse coordinate points
  - Must be related with link IDs
    - Map-matching is not easy
  - Very hard to measure the cost of individual links precisely

- Total cost can be precisely measured even in that case
  - Simply computed as the time difference between O and D: $t_{destination} - t_{origin}$

---

Brakatsoulas et al., "On map-matching vehicle tracking data," Proc. VLDB '05, pp.853--864
This work proposes a feature-based approach to trajectory regression.

\[ y = f(\ldots) \]

**Kernel-based** [Idé-Kato 09]
- Use kernel matrix
- No need for data matrix

**Feature-based**
- Use data matrix
- No need for kernels
Agenda

☐ Problem setting

☐ Formulation

☐ Relationship with Gaussian process regression

☐ Experiments
Representing the total cost using latent variables \( \{f_e\} \)

\[
y(x) = \sum_{e \in x} C_e
\]

for all links included in input path \( x \)

Our goal is to find cost deviation \( \{f_e\} \) from the baseline

\[
C_e \equiv l_e (\phi_e^0 + f_e)
\]

Link length (known)

Baseline unit cost (known from e.g. legal speed limit)

Cost deviation per unit length from the baseline
Setting two criteria to identify the value of \( \{f_e\} \)

- **The observed total cost must be reproduced well**

  - Minimize:
  
    \[
    \sum_{n=1}^{N} \left(y^{(n)} - \sum_{e \in x^{(n)}} C_e(f_e)\right)^2
    \]

    - Observed cost for \( x^{(n)} \)
    - Estimated cost for a path \( x^{(n)} \)

- **Neighboring links should have similar values of cost deviation**

  - While keeping this constant

    \[
    \sum_{e=1}^{M} \sum_{e'=1}^{M} S_{e,e'} |f_e - f_{e'}|^2
    \]

    - Similarity between link \( e \) and \( e' \)
Example of definition of link similarities

\[ S_{e,e'} \equiv \begin{cases} 
\omega^{1 + d(e,e')} , & d(e, e') \leq d_0 \\
0 , & \text{otherwise}
\end{cases} \]

\[ d = (\text{# of hops between edges}) \]

In this case, \(d(e,e')=2\)
Final objective function to be minimized

\[
\Psi(f|\lambda) = \sum_{n=1}^{N} \left( y^{(n)} - \sum_{e \in x^{(n)}} c_e(f_e) \right)^2 + \frac{\lambda}{2} \sum_{e=1}^{M} \sum_{e'=1}^{M} S_{e,e'} |f_e - f_{e'}|^2
\]

“Predicted cost should be close to observed values”

- “Neighboring links should take similar values”
- \(S\) is the similarity matrix between links
This optimization problem can be analytically solved

Matrix representation of the objective

$$\Psi(f | \lambda) = \left\| y_N - Q^T f \right\|^2 + \lambda f^T L f$$

Graph Laplacian induced by the link-similarity Matrix

$$L_{i,j} \equiv \delta_{i,j} \sum_{k=1}^{M} S_{i,k} - S_{i,j}$$

“Data matrix” of the trajectories

$$Q \equiv [q^{(1)}, \ldots, q^{(N)}] \in \mathbb{R}^{M \times N}$$

Vector of the trajectory costs

$$q_e^{(n)} = \begin{cases} 
    l_e, & \text{for } e \in x^{(n)} \\
    0, & \text{otherwise}
\end{cases}$$

Analytic solution

$$f = \left[ QQ^T + \lambda L \right]^{-1} Q y_N$$

Lambda is determined by cross-validation
Agenda

- Problem setting
- Formulation
- Relationship with Gaussian process regression
- Experiments
The link-Laplacian matrix bridges the two approaches

$$y = f(\cdot)$$

**Kernel-based** [Idé-Kato 09]

**Proposition:**
These two approaches are equivalent if the kernel matrix is chosen as

$$K_{n,n'} = q(n)^\top L^\dagger q(n')$$

**Feature-based** (this work)
Practical implications of the link-Laplacian kernel

- The proposition suggests a natural choice of kernel
  - In GPR, the choice of kernel is based on just an intuition
  - Since a link-similarity is much easier to define, our approach is more practical than the GPR method
Agenda

- Problem setting
- Formulation
- Relationship with Gaussian process regression
- Experiments
Traffic simulation data on real road networks

- **Network data**
  - Synthetic 25x25 square grid
  - Downtown Kyoto

- **Trajectory-cost data**
  - Generated with IBM Mega Traffic Simulator
    - Agent-based simulator
  - Data available on the Web

- **Method compared**
  - Legal: perfectly complies with the legal speed limit
  - GPR: GPR with a string kernel
  - RETRACE: proposed method

![Downtown Kyoto City map](image)

**Table 1: Summary of data.**

<table>
<thead>
<tr>
<th></th>
<th>Grid25x25</th>
<th>Kyoto</th>
</tr>
</thead>
<tbody>
<tr>
<td># nodes</td>
<td>625</td>
<td>1518</td>
</tr>
<tr>
<td># links</td>
<td>2400</td>
<td>3478</td>
</tr>
<tr>
<td># generated paths</td>
<td>1200</td>
<td>1739</td>
</tr>
</tbody>
</table>
Proposed methods gave the best performance

- **Legal**
  - Does not reproduce traffic congestion in Kyoto

- **GPR**
  - Worst
  - Due to the choice of kernel that can be inconsistent to the network structure

- **RETRACE (proposed)**
  - Clearly the best
  - Outliers still exist: cannot handle dynamic changes of traffic
Concluding remarks

☐ **Summary**
- Proposed the RETRACE algorithm for trajectory regression
- RETRACE has an analytic solution that is easily implemented
- Gave an interesting insight to the relationship with the GPR approach

☐ **Future work**
- To test the algorithm using probe-car data
- To study how to improve the scalability