Google Your Life: Learning Big (Sensors) Data

Dan Feldman,
Andrew Sugaya, Cynthia Sung, Gary Wang, Cathy Wu, Katherine Fung,
Daniela Rus

Distributed Robotics Lab of MIT
- You left home at 9:17 AM.
- You arrived at Maiden Center Station at 9:26 AM, after traveling by foot for 9 minutes.
- You arrived at Kendall Station at 9:52 AM, after traveling by public transportation for 26 minutes.
- You arrived at work at 9:57 AM, after traveling by foot for 5 minutes.
- You stayed at work for 3 hours, leaving at 1:03 PM.
- You arrived at Quiznos for lunch at 1:09 PM, after traveling by foot for 6 minutes.
- You stayed at Quiznos for 27 minutes, leaving at 1:36 PM.
- You arrived at work at 1:43 PM, after traveling by foot for 7 minutes.
- You stayed at work for 5 hours, leaving at...
Restaurants you visited on July 11th, 2012

1. **Anna’s Taqueria**
   You were here on July 11th from 7:03 PM to 7:31 PM, with John Smith, Foo Bar, and 3 OTHERS.
   You have been here 142 OTHER TIMES.
   [VIEW SIMILAR RESTAURANTS](#)

2. **Toscanini’s Ice Cream**
   You were here on July 11th from 7:44 PM to 7:58 PM, with Tim Yang, John Smith, and 4 OTHERS.
   You have been here 17 OTHER TIMES.
   [VIEW SIMILAR RESTAURANTS](#)
You and Tim Yang

Recent encounters:

1. Toscanini's Ice Cream – July 11th
2. Home – July 7th
3. Home – July 1st
4. MIT Student Center – June 25th
5. 123 Main Street – June 24th
6. Home – June 23rd

See more...

Time analysis

1. Home – 5 hours/week
   - Weekend evenings
2. Leisure – 3 hours/week
   - Weekday afternoons
How much data?

- 1 GPS packet (latitude, longitude, time) ~ 100 bytes
- ~ 144 million smart phones sold in 2011 (Gartner)
- 1 GPS packet sampled every 10 seconds *100 millions devices = Terabytes per day
Challenges

- Storing data on iPhone is expensive
- Transmission data is expensive
- Hard to analyze raw data
- Dynamic real-time streaming data
GPS-points Data

- iPhones can collect high-frequency GPS traces
- GPS-point = (latitude, longitude, time)

<table>
<thead>
<tr>
<th>latitude</th>
<th>longitude</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.295783</td>
<td>103.7816</td>
<td>8:44:57</td>
</tr>
<tr>
<td>1.295785</td>
<td>103.7816</td>
<td>8:44:59</td>
</tr>
<tr>
<td>1.295782</td>
<td>103.7816</td>
<td>8:45:00</td>
</tr>
<tr>
<td>1.295782</td>
<td>103.7816</td>
<td>8:45:01</td>
</tr>
<tr>
<td>1.29579</td>
<td>103.7817</td>
<td>8:45:04</td>
</tr>
<tr>
<td>1.295802</td>
<td>103.7817</td>
<td>8:45:05</td>
</tr>
<tr>
<td>1.295915</td>
<td>103.7818</td>
<td>8:45:08</td>
</tr>
<tr>
<td>1.29598</td>
<td>103.7819</td>
<td>8:45:09</td>
</tr>
<tr>
<td>1.296015</td>
<td>103.7819</td>
<td>8:45:10</td>
</tr>
<tr>
<td>1.296057</td>
<td>103.782</td>
<td>8:45:11</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
From GPS Signal to Semantic Database

Input: \( n \) points

Output: \( m \) locations, \( k \) trajectories

---

**Input:**

<table>
<thead>
<tr>
<th>Location ID</th>
<th>Begin Point</th>
<th>End Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>103.7816</td>
<td>103.7816</td>
</tr>
<tr>
<td>b</td>
<td>103.7819</td>
<td>103.7819</td>
</tr>
<tr>
<td>c</td>
<td>105.7816</td>
<td>106.6316</td>
</tr>
<tr>
<td>d</td>
<td>103.6816</td>
<td>108.3416</td>
</tr>
<tr>
<td>e</td>
<td>103.7519</td>
<td>105.7816</td>
</tr>
<tr>
<td>f</td>
<td>104.7816</td>
<td>107.4516</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

---

**Output:**

<table>
<thead>
<tr>
<th>Begin time</th>
<th>End time</th>
<th>Location ID</th>
<th>Speed</th>
</tr>
</thead>
<tbody>
<tr>
<td>8:44:57</td>
<td>8:48:57</td>
<td>c</td>
<td>30</td>
</tr>
<tr>
<td>8:44:59</td>
<td>8:45:00</td>
<td>d</td>
<td>24</td>
</tr>
<tr>
<td>8:52:00</td>
<td>8:54:00</td>
<td>g</td>
<td>24</td>
</tr>
<tr>
<td>8:54:01</td>
<td>8:55:01</td>
<td>q</td>
<td>11</td>
</tr>
<tr>
<td>8:56:57</td>
<td>8:57:57</td>
<td>r</td>
<td>120</td>
</tr>
<tr>
<td>8:58:57</td>
<td>8:59:57</td>
<td>m</td>
<td>55</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>65</td>
</tr>
</tbody>
</table>

---

Input: From GPS Signal to Semantic Database

Output: \( m \) locations, \( k \) trajectories
Applications

• Signal is now a string
  - Run PKZIP to compress it
  - Hidden Markov models for semantic maps

• Prediction of Locations:
  Save battery life

• Correlations in users/locations matrix
  Who behaves like me in London?

• Clustering of users and locations
• Denoising for Road Matching
From Semantic Database Text Mining

\( m \) locations

<table>
<thead>
<tr>
<th>Location</th>
<th>ID</th>
<th>Begin Point</th>
<th>End Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>103.7816</td>
<td>103.7816</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>103.7819</td>
<td>103.7819</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>105.7816</td>
<td>106.6316</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>103.6816</td>
<td>108.3416</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>103.7519</td>
<td>105.7816</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>104.7816</td>
<td>107.4516</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Reverse Geo-coding

Google maps, mapQuest...

<table>
<thead>
<tr>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>220 Walden St., Cambridge</td>
</tr>
<tr>
<td>15 Main St., Cambridge</td>
</tr>
<tr>
<td>125 Vassar Aven., Cambridge</td>
</tr>
<tr>
<td>33 Elm Street, Sumerville</td>
</tr>
<tr>
<td>1 St Andrew's Rd, Singapore 178957</td>
</tr>
<tr>
<td>390A Havelock Rd, Boston 169664</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>
Latent Semantic Analysis (PCA) of Yelp Reviews
Overview Of Solution

(a) GPS Signal  (b) Coreset  (c) Clustering
System Overview

GPS Device (Phone) → Raw Data → Django (Python) → Raw Data → ØMQ (Python) → Raw Data → Coreset Creation (MATLAB) → Compressed Data

Query Interface (Website) → Search Terms → Latent Semantic Analysis for Search (Python) → Trajectory Clusters → Trajectory Clustering (MATLAB) → Trajectory Clusters

History Interface (Website) → Diary Text → Latent Semantic Analysis for Diary (Python) → Database (PostGIS) → Trajectory Clusters
Definition: $k$-Segment

A $k$-segment $f : \mathbb{R} \rightarrow \mathbb{R}^d$ is a $k$-piecewise linear function

$k = 5$
$d = 1$
Definition: Fitting Cost

Sum of squared distances: \( \text{cost}(P, f) = \sum_t \| p_t - f(t) \|^2 \)
Definition: $k$-Segment mean

The $k$-segment $f^*$ that minimizes the fitting cost to $P$

\[
\sum_t \| p_t - f(t) \|^2
\]
Problem Statement

• **Input:**
  $d$-dimensional signal $P$, and integer $k > 0$

• **Output:**
  Optimal $k$-segment $f^*$ for $P$ (that approximates $P$)
Related Work

Provable Guarantee:

Exact solution in $O(n^2 k)$ time and $O(n^2)$ space using Dynamic Programming [Bellman’68]

Numerous heuristics:

• for the off-line problem
• for maximum distances (instead of sum)
• Matlab heuristic for connected segments
Main Tool: Coreset

A weighted set $C \subseteq P$ such that for every $k$-segment $f$

$$\text{cost}(P, f) \sim \text{cost}_w(C, f)$$

$$\sum_t \|f(t) - p_t\|^2 \quad (1 \pm \epsilon) \quad \sum_{p_t \in C} w(p_t) \cdot \|f(t) - p_t\|^2$$
Streaming and Parallel Computation

- The \( n \) GPS-points of \( P \) arrive one by one
- Size of data >> memory (RAM)
- Take advantage of network of servers and GPUs (Graphical Processing Units)

• We need a data structure that supports the following methods in \( \sim \log n \) time and space:
  - \texttt{Insert(p)}
  - \texttt{GetCurrentOptimalSpline()}
• ..and can run in parallel simple processors
Streaming Compression using merge & reduce

(From Piotr Indyk)
Parallel computation
(a) Coreset tree

(b) Construction of C₃
Bad News: No small coreset exists for $k$-segment queries!
Input $P$: $n$ points on the $x$-axis
Input $P$: $n$ points on the $x$-axis

Coreset $C$: all points except one
Input $P$: $n$ points on the $x$-axis

Coreset $C$: all points except one

Query $f$: covers all except this one
Input $P$: $n$ points on the $x$-axis

Coreset $C$: all points except one

Query $f$: covers all except this one

$\text{Cost}(P, f) > 0$

$\text{Cost}(C, f) = 0$
Input \( P \): \( n \) points on the \( x \)-axis

Coreset \( C \): all points except one

Query \( f \): covers all except this one

\[
\text{Cost}(P, f) > 0 \quad \text{No \( \varepsilon \)-coreset for any \( \varepsilon > 0 \)}
\]

\[
\text{Cost}(C, f) = 0
\]
Observation: *End-points+frequency gives an exact representation of the signal.*
Observation: *End-points+frequency gives an exact representation of the signal.*

Holds for any set of consecutive points on a segment
Definition: Coreset

A weighted set $C \subseteq P$ such that for every $k$-segment $f$:

$$\text{cost}(P,f) \sim \text{cost}_w(C,f)$$
Our Main Compression Theorem

For every discrete signal of $n$ points in $\mathbb{R}^d$, there is a coreset of size $O\left(\frac{dk}{\epsilon^2}\right)$ that can be computed in $O(n)$ expected time.

Via merge-and-reduce, using:

- $O(\log (n))$ memory
- $O(\log n)$ time per point update
- $O\left(\frac{n}{M}\right)$ time on $m$ processors
**$k$-segment mean using coreset**

<table>
<thead>
<tr>
<th>time</th>
<th>latitude</th>
<th>longitude</th>
</tr>
</thead>
<tbody>
<tr>
<td>8:44:57</td>
<td>1.295783</td>
<td>103.7816</td>
</tr>
<tr>
<td>8:44:59</td>
<td>1.295785</td>
<td>103.7816</td>
</tr>
<tr>
<td>8:45:00</td>
<td>1.295782</td>
<td>103.7816</td>
</tr>
<tr>
<td>8:45:01</td>
<td>1.295782</td>
<td>103.7816</td>
</tr>
<tr>
<td>8:45:04</td>
<td>1.29579</td>
<td>103.7817</td>
</tr>
<tr>
<td>8:45:05</td>
<td>1.295802</td>
<td>103.7817</td>
</tr>
<tr>
<td>8:45:08</td>
<td>1.295915</td>
<td>103.7818</td>
</tr>
<tr>
<td>8:45:09</td>
<td>1.29598</td>
<td>103.7819</td>
</tr>
<tr>
<td>8:45:10</td>
<td>1.296015</td>
<td>103.7819</td>
</tr>
<tr>
<td>8:45:11</td>
<td>1.296057</td>
<td>103.782</td>
</tr>
</tbody>
</table>

$|P| = n$

$k$-segment mean

$|C| = \frac{2k}{\epsilon^2}$
Example Implication

By running $O(n^3)$ time optimal algorithm on the coreset:

A $(1 + \varepsilon)$ approximation to the $k$-segment mean of $P$ can be computed in $O(n)$ expected time

Via merge-and-reduce, using:

$O(\log (n))$ memory

$O(\log n)$ time per point update

$O\left(\frac{n}{M}\right)$ time on $m$ processors
Coreset Construction

Input: signal of $n$ points, constants $k, \varepsilon$
Coreset Construction

**Input:** signal of $n$ points, constants $k$, $\varepsilon$

Compute $k$-segment mean
Coreset Construction

Input: signal of $n$ points, constants $k, \varepsilon$

Compute $k$-segment mean

Project points onto segments
Coreset Construction

Input: signal of $n$ points, constants $k$, $\varepsilon$

Compute $k$-segment mean

Project points onto segments

Sample $|S| = \frac{k}{\varepsilon}$ input points

Assign probability $\sigma_p = \frac{d_p}{\sum_p d_p}$
Coreset Construction

**Input:** signal of $n$ points, constants $k$, $\varepsilon$

Compute $k$-segment mean

Project points onto segments

Sample $|S| \sim \frac{k}{\varepsilon^2}$ points

with probability $\sigma_p = \frac{d_p}{\sum_p d_p}$
**Coreset Construction**

**Input:** signal of $n$ points, constants $k$, $\varepsilon$

Compute $k$-segment mean

Project points onto segments

Sample $|S| \sim \frac{k}{\varepsilon^2}$ points

with probability $\sigma_p = \frac{d_p}{\sum_p d_p}$

Assign weights $\pm \frac{1}{|S| \sigma_p}$
Coreset Construction

Input: signal of $n$ points, constants $k$, $\varepsilon$, $\delta$

Compute $k$-segment mean

Project points onto segments

Sample $|S| \sim \frac{k}{\varepsilon^2}$ points

with probability $\sigma_p = \frac{d_p}{\sum_p d_p}$

Assign weights $\pm \frac{1}{|S|\sigma_p}$

Output: $k$ segments and $|S|$ weighted points pairs
longitude
time
latitude

original data $P$
(input)

k-segment mean $f$
(line 1)

sampled points $T$
(line 5)

projection of $T$ on $f$

final coreset
(output)
### Tested Data sets

<table>
<thead>
<tr>
<th>Name</th>
<th>No. of Users</th>
<th>Signal Length</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>DRL</td>
<td>1</td>
<td>5811 points</td>
<td>MIT, Distributed Robotics Lab</td>
</tr>
<tr>
<td>SIGSPATIAL</td>
<td>1</td>
<td>14,436 points</td>
<td>ACM-SIGSPATIAL CUP 2012</td>
</tr>
<tr>
<td>CABS</td>
<td>500</td>
<td>2,688,000 points</td>
<td>Taxi-Cabs in San-Francisco over few months (”Crowdad”)</td>
</tr>
</tbody>
</table>

### Tested Algorithms

<table>
<thead>
<tr>
<th>Name</th>
<th>Author</th>
<th>Streaming</th>
<th>Bounded Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bellman</td>
<td>Bellman’68</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>DPSegs (Douglas-Peucker)</td>
<td>Oracle</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>SPAP2</td>
<td>MATLAB</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>DeadRec</td>
<td>J. Muckel</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
$k=91$-segments, SIGSPATIAL
43-segments, DRL
Definition: \((k, m)\)-Segment

A \(k\)-segment \(f: \mathbb{R} \to \mathbb{R}^d\) whose projection on \(\mathbb{R}^d\) is only \(m\) segments

\[k = 5\]
\[m = 4\]
Definition: \((k, m)\)-Segment mean

Minimizes \(\text{cost}(P, f)\) over every \((k, m)\)-segment \(f\)
Previous Work

• NP-hard for non constant $m$
  – Reduction to $k$-means

• NP-hard for approximation for non constant $m$
  – Reduction to $k$-line-means

• Few Heuristic for small off-line data sets

• No problem definition for GPS signal clustering
EM-($k, m$)-segment

Input: signal of $n$ points, constants $k \geq m \geq 1$
EM-\((k, m)\)-segment

Input: signal of \(n\) points, constants \(k \geq m \geq 1\)

Guess the partition
EM-(**k**, **m**)-segment

**Input:** signal of **n** points, constants **k** ≥ **m** ≥ 1

Guess the partition

*Maximization:* Find 1-segment mean of each cluster
EM-$(k, m)$-segment

**Input:** signal of $n$ points, constants $k \geq m \geq 1$

Guess the partition

**Maximization:**
Find 1-segment mean of each cluster

**Expectation:**
Find optimal partition
EM-(k, m)-segment

**Input:** signal of \( n \) points, constants \( k \geq m \geq 1 \)

Guess the partition

**Maximization:**
Find 1-segment mean of each cluster

**Expectation:**
Find optimal partition

Iterate
EM-\((k, m)\)-segment

Input: signal of \(n\) points, constants \(k \geq m \geq 1\)

Guess the partition

Maximization:
Find 1-segment mean of each cluster

Expectation:
Find optimal partition

Iterate
EM-$(k, m)$-segment

**Input:** signal of $n$ points, constants $k \geq m \geq 1$

Guess the partition

**Maximization:**
Find 1-segment mean of each cluster

**Expectation:**
Find optimal partition

Iterate
EM-\((k, m)\)-segment

**Input:** signal of \(n\) points, constants \(k \geq m \geq 1\)

Guess the partition

**Maximization:**
Find 1-segment mean of each cluster

**Expectation:**
Find optimal partition

Iterate
EM-\((k, m)\)-segment

Input: signal of \(n\) points, constants \(k \geq m \geq 1\)

Guess the partition

Maximization: Find 1-segment mean of each cluster

Expectation: Find optimal partition

Iterate

Output: \((k, m)\)-segments
(91,70)-segments, SIGSPATIAL
(43, 25)-segments, DRL
Open Problem

• Compute approximation to the \((k,m)\)-segment problem for massive signals

• Linear time algorithms for “natural” instances

• Learn relations between set of signals
Contribution

• Searchable Text Diary from Smartphones
• Reduction from GPS-learning to Machine learning
• Signal Simplification and Clustering
• Streaming Parallel GPS-data compression