MINING DATA FROM MOBILE DEVICES

Dimitrios Gunopulos
University of Athens
dg@di.uoa.gr

Smart-phones: Novel applications

Smart-phones as a platform for sense and share information applications.

- Personalized traffic, weather information
- Social networking
- Smart city applications
- Similar trajectories (e.g. for popular tourist routes)

Data come from GPS, also other sources (e.g. GPS tagged images, crowdsourcing mechanisms)

Users are becoming producers and consumers of data

Interesting Applications

- Intelligent Transportation Systems: “Find new possible bus routes which would help at least K users.”
- Social Networks: “Find new cycling routes from MOMA to the Julliard”
- GeoLife, GPS-Waypoints, Sharemyroutes, etc. offer centralized counterparts.
- Habitant Monitoring: Find zebras that moved more similarly to zebra X before it got injured.

Analyzing GPS trajectory data

- App deployment allows location data collection

- How to use trajectory data
  - Integrate with other (e.g. twitter/sensor) data
  - Supplement flow based traffic data
  - Where is the improvement?
  - Building better/more general traffic models?
  - Better and more accurate maps?
  - Demand modeling?
    - Finding patterns, traffic modes
  - Building modeling tools for communities?
  - Real-time situation understanding and reaction

A Word on GPS Trace Collection

- Popular Smartphones are already collect positional information. Same applies to Social Networking Applications (e.g., Latitude, Gowalla, Twitter, etc.)
- iPhone User Position Logging:
  - iPhone collects coarse-grain positional information (i.e., triangulated Cell tower) locally on your smartphone (and iTunes backup).
  - The unencrypted log file is even migrated between devices!
  - Displaying your iPhone trace history on a map:
    http://petewarden.github.com/iPhoneTracker/
Problems

- Learning from trajectories
  - Distance measures
  - Clustering
  - Indexing, Distributed Indexing
  - Pattern Queries
  - Applications
    - Map Creation
    - Mode of Transportation
    - Social networks applications
  - Crowdsourcing Applications
  - Privacy issues

Trajectory Similarity

- Given a query trajectory $Q$ and a trajectory database $D(t)$ search among them for the most similar trajectory.
- Determine the most similar by using not only the spatial shape of the trajectories but also their evolution in time.

Similarity applications:
- Clustering
- Classification
- Frequent subtrajectories
- Indexing

Requirements for Similarity Model

- Different Lengths
- Outliers
  - Random Peaks
  - Noise Everywhere
  - Non Recoverable Part

Similarity Functions

- Distance Function: Hausdorff distance has NO “Temporal Awareness”

A man walks the dog...

What is the needed length of the leash?
Fréchet distance

- Fréchet distance (length of the leash)

Let $C_1: [a_1, b_1] \to \mathbb{R}^2$ and $C_2: [a_2, b_2] \to \mathbb{R}^2$ denote two curves. Their Fréchet distance is defined as

$$\delta(C_1, C_2) = \inf_{\alpha, \beta} \max_{t \in [0,1]} || C_1(\alpha(t)) - C_2(\beta(t)) ||$$

$\alpha$ (respectively $\beta$) range over all the continuous and monotonically increasing functions $[0,1] \to [a_1, b_1]$ (respectively $[0,1] \to [a_2, b_2]$)

Clearly, a lot more general – but also harder to compute than the Hausdorff distance.

Computation: $O(mn \log(mn))$ [AG95]

Dynamic time warping (DTW)

- Time Warping
  - Allows stretching in time axis
  - Matches all elements
  - Extensively used in speech recognition domain

Dynamic time warping algorithm

- Euclidean (L2) distance is
  $$D_2(x, y) := \sum_{i=1}^{N} (x_i - y_i)^2$$
  or recursively,
  $$D_2(x_{1:i}, y_{1:j}) := (x_{1:i} - y_{1:j})^2 + D_2(x_{1:i-1}, y_{1:j-1})$$

- Dynamic time warping distance is
  $$D_{w}(x_{1:i}, y_{1:j}) := (x_i - y_j)^2 + \min \begin{cases} 
D_{w}(x_{1:i-1}, y_{1:j-1}) \\
D_{w}(x_{1:i}, y_{1:j-1}) \\
D_{w}(x_{1:i-1}, y_{1:j}) 
\end{cases}$$

where $x_{1:i}$ is the subsequence $(x_1, \ldots, x_i)$

Longest Common Subsequence (LCSS)

- Definition
  $$LCCS(A, B) := \max \{ j \mid \text{length of a common subsequence of } A[1:j] 	ext{ and } B[1:j] \}$$

- Dynamic Programming Solution
  $$LCCS[i][j] := \begin{cases} 
0 & \text{if } i = 0 \text{ or } j = 0 \\
LCCS[i-1][j-1] + 1 & \text{if } A[i] = B[j] \\
\max\{LCCS[i-1][j], LCCS[i][j-1]\} & \text{otherwise} 
\end{cases}$$

Differences between Fréchet, DTW & LCSS

- DTW has to much element, so is more sensitive to outliers
- LCSS matches only the similar parts


[DTSWK08] provides experimental comparison of 8 representation methods; 9 similarity measures (+variants); 38 datasets. [WMDSK12] provides further extensions.
LCSS for real values

LCSS can be computed in $O(\delta (l_1 + l_2))$ by dynamic programming algorithm.

LCSS Dynamic Programming algorithm

![LCSS Dynamic Programming algorithm](image)

Space Transformations: Rotation

Method:
- Transform trajectory to rotation invariant space
  - Positional information -> Angle/ Arc length information
- Proper normalization ensures also:
  - Translation invariance
  - Scale Invariance
- Perform matching in new space
  - Use Warped Matching to compensate for shape variations

Mapping Example

Angles to utilize:
- Exact angles (as in figure)
- Relative Angles (to previous segment)
- Angles from center of mass

Parallel Movements: Normalization

![Parallel Movements: Normalization](image)
Extending LCSS: Translation Invariant Similarity

- Standard techniques cannot detect parallel movements.
- So, we define \( S_2 \):
  \[
  S_2 = \max_{f, c} \text{LCSS}(\tilde{A}, \tilde{B}, f, c)
  \]
- \( S_2 \) can detect parallel movements
- Better accuracy than simple normalization
- Distance: \( D_1 \approx 1 - S_1 \) and \( D_2 \approx 1 - S_2 \)

**Example:**
- Similarity: \( \text{LCSS}(A, B, f, c) \)
- \( f \) and \( c \) are translations in 1D:
  \[
  f = a \quad \text{and} \quad c = b
  \]
- \( D_1 = 1 - \text{LCSS}(A, B, f, c) \)
- \( D_2 = 1 - \text{LCSS}(A, B, f, c) \)

**Classification Accuracy**

- The 2D time series of a human tracking feature
- Example using Video Tracking data
- The data we used correspond to 'athens', 'berlin', 'frankfurt', 'london', 'paris'

**Similarity and Uncertainty**

1. Uncertainty in AI and Databases
   - Model-Theoretic issues (Possible Models)
   - NULLs (open vs. closed world assumption)
   - Probabilistic databases (and possible worlds)
2. Time Geography and Inexact Geometries
   - Uncertain location over time
   - Re-evaluation of Euclidian postulates

**Classification Accuracy**

- Test classification accuracy using Hierarchical Clustering

**Vector representation of approximate/uncertain trajectories (Pelekis et al, 09)**

- Assume a regular grid \( G(m \times n) \) consisting of cells \( c_{ij} \), a trajectory \( T = (x_1, y_1, t_1, \ldots, x_n, y_n, t_n) \) and a target dimension \( p << n \)
- The \textit{approximate trajectory} \( \hat{T} = \langle \hat{x}_1, \ldots, \hat{x}_p \rangle \) consists of \( p \) regions (i.e., sets of cells) crossed by \( T \) during period \( p \)
- The \textit{Uncertain Trajectory} is the \( c \)-buffer of \( \hat{T} \)

**Example:**

- Assume a regular grid \( G(m \times n) \) consisting of cells \( c_{ij} \), a trajectory \( T = (x_1, y_1, t_1, \ldots, x_n, y_n, t_n) \) and a target dimension \( p << n \)
- The \textit{approximate trajectory} \( \hat{T} = \langle \hat{x}_1, \ldots, \hat{x}_p \rangle \) consists of \( p \) regions (i.e., sets of cells) crossed by \( T \) during period \( p \)
- The \textit{Uncertain Trajectory} is the \( c \)-buffer of \( \hat{T} \)
Problems
- Learning from trajectories
- Distance measures
- Clustering
- Indexing, Distributed Indexing
- Pattern Queries
- Applications
  - Map Creation
  - Mode of Transportation
  - Social networks applications
- Crowdsourcing Applications
- Privacy issues

Similarity in Parts of Trajectories
- (a) Trajectory Segmentation
  - segments of homogeneous movement characteristics
  - to convert a trajectory into a simplified and yet revealing structure
  - trajectory is transformed to a movement parameter class sequence
- (b) Similarity Computation
  - Normalized Weighted Edit Distance (NWED)
  - Based on edit distance
  - applies weighted conversion costs (insertion/deletion, substitution)
  \[ DLW12 \]

Similarity of Parts of Trajectories
- Finding longest similar part (time-shifts) \[ BBvK+09 \]

Similar shapes (routes), but different speeds at different times

\[
\int_{t_0}^{t_1 + T} d(t_1(t), t_2(t + t_{shift})) \, dt
\]

Average distance in a time-interval T, with time-shift...

Trajectories Clustering
- TraClass \[ LHL+08 \]
  - Partition trajectories, and then perform hierarchical -Region-based; -Trajectory-based clustering.

Flocks
- Trajectories that “stay together for a while”

\[
flock(m, k, r) = meet(m, k, r)
\]

Maximize \( k \) for
- Fixed
- Varying
  - “versions” of \( m \)
Flocks

- Trajectories that “stay together for a while”

Approximate variants of both flock/meet and fixed/varying:
for the same duration, make sure that there exists
a disk of radius $c \cdot r$ where $c = 1 + e$

Complexity Results

<table>
<thead>
<tr>
<th>Subset</th>
<th>Partials</th>
<th>Meeting</th>
<th>Flock</th>
</tr>
</thead>
<tbody>
<tr>
<td>Varying subset</td>
<td>$O((\alpha^2 + log n + \alpha)^2)$ (Exact)</td>
<td>$O(\alpha^2 \log n)$ (Exact)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$O((\alpha^2 + log n + \alpha)^2)$ (Approx. $\alpha$)</td>
<td>$O(\alpha^2 \log n)$ (Approx. $\alpha$)</td>
<td></td>
</tr>
<tr>
<td>Fixed subset</td>
<td>$O((\alpha^2 + log n + \alpha)^2)$ (Exact)</td>
<td>$O(\alpha^2 \log n)$ (Exact)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$O((\alpha^2 + log n + \alpha)^2)$ (Approx. $\alpha$)</td>
<td>$O(\alpha^2 \log n)$ (Approx. $\alpha$)</td>
<td></td>
</tr>
</tbody>
</table>

Other predicates (patterns): flock, leadership, convergence, encounter

[KvK04]

Convoys

- Moving Clusters and “lossy flocks”…

Triangle inequality does not hold for DTW or LCSS
(but holds for Edit Distance or DTW with gaps).

Example 1: $A = 1,1,5$
$B = 2,2,5$
$C = 2,2,2,2,2,5$
Then:
$DTW(A,B) = 2$, $DTW(B,C) = 0$, $DTW(A,C) = 5$

Example 2:
$D_1(\delta, \epsilon, A,C) > D_2(\delta, \epsilon, A,B) + D_2(\delta, \epsilon, B,C)$, where $D_2(\delta, \epsilon, A,B) = 1$

Learning from trajectories
- Distance measures
- Clustering
- Indexing, Distributed Indexing
- Pattern Queries
- Applications
  - Map Creation
  - Mode of Transportation
  - Social networks applications
- Crowdsourcing Applications
- Privacy issues

Pivotal properties of Similarity Measures

Sequential Scan

- Compare the Query to all the trajectories in the database
- Use Lower/Upper Bounds or Early Termination to speed up the search:

$$\text{If } \min(\text{row}) > \text{(best So Far Match)} \text{ then exit}$$

Fully distributed, privacy preserving trajectory search application using mobile phones

Find the K most similar trajectories to Q without pulling together all traces at QN
Constraints and Objectives

Don’t Disclose the User’s Trajectories (privacy constraints)
- Social sites are already undergoing significant privacy restructuring
- Trajectories are large (270MB/year with 2s samples)

Minimize Net Traffic and Optimize Local Processing (algorithmic constraints)
- 3G/4G and WiFi traffic: i) depletes smartphone battery and ii) degrades network health

<table>
<thead>
<tr>
<th>Basic Operation on Smartphone</th>
<th>Power (mW = mJ/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU Idle (OS running)</td>
<td>175 mW</td>
</tr>
<tr>
<td>CPU Busy (Processing)</td>
<td>309 mW</td>
</tr>
<tr>
<td>WiFi Idle (Connected)</td>
<td>38 mW</td>
</tr>
<tr>
<td>WiFi Busy (Uplink 123Kbps, -5dBm)</td>
<td>600 mW</td>
</tr>
<tr>
<td>LCD Brightness (economy mode)</td>
<td>300 mW</td>
</tr>
</tbody>
</table>

Constraints and Objectives

Quick and efficient deployment (Programmability constraints)
MapReduce programming model
- Flexible distributed computing framework for processing large data sets
- Consists of 2 major steps, map and reduce

Misco (DEBS’10)
- Distributed MapReduce implementation for a cloud of mobile smart-phones
- Allows efficient programming of sophisticated tasks
- Simplifies development of applications which fit into this framework
- Hides difficult low-level implementation details
- Code distribution, failures, heterogeneity

Clustering in a distributed MapReduce setting

Dowser - a Connectivity Discovery system
- Identify areas of high WiFi connectivity
- Processing computed in a online fashion
- Data from multiple cooperating users
- Preserves data privacy between users

Clustering is performed on individual devices without the need to distribute data between devices

Each Worker:
- selects K points or uses centroids from server
- performs local clustering
- sends centroids to server

Server:
- performs weighted clustering on centroids
- new centroids are sent back to workers
- Repeats until centroids no longer move.

Trajectory Similarity Search in a Distributed Environment

SMARTTRACE: Server implemented in JAVA (4,500 LOC)
Client implemented in JAVA on Android (2,500 LOC + XML files)

Sensor Networks Setting

- Monitoring area G, m objects moving inside
- n cells each has a camera sensor:C1,C2,C3,C4
- Each record of the trajectory is stored locally at the closest sensor
- eat view
Distributed LCSS Lower Bound

- For each trajectory \( T_i \), cell \( c \) finds the time region \( T_d = \{t(s)p | p \in A_i \} \) when \( A \) stays in cell \( c \). Filter \( Q \) into \( Q^c \) such that \( Q^c \) is in the same time intervals as \( A_i \), \( Q^c = \{p | p \in Q \} \) and \( t(s)p \) in \( T_d \).
- Each cell performs a local computation of \( \text{LCSS}_{\delta, \lambda}(Q^c, A_i) \)
- The lower bound \( \text{DLB}_{\text{LCSS}}(Q, A) \) is computed by adding the \( n \) local results

\[ \sum_{i=1}^{n} \text{LCSS}_{\delta, \lambda}(Q^c, A_i) \leq \text{LCSS}_{\delta, \lambda}(Q, A) \]

Distributed top-K computation with bounds

- Now we have the Lower and Upper Bounds rather than Exact scores.
- e.g. instead of \( \text{sim}(A_0, Q) = 20 \) it gives us \( [A_0, 15, 25] \)

UBL-K Algorithm

Query: Find the K=2 highest ranked answers

<table>
<thead>
<tr>
<th>METADATA id,ub</th>
<th>DATA id,ub</th>
</tr>
</thead>
<tbody>
<tr>
<td>A4,30</td>
<td>A4,23</td>
</tr>
<tr>
<td>A2,27</td>
<td>A2,22</td>
</tr>
<tr>
<td>A0,16</td>
<td>A0,15</td>
</tr>
<tr>
<td>A3,20</td>
<td>A3,18</td>
</tr>
<tr>
<td>A9,18</td>
<td>A9,17</td>
</tr>
<tr>
<td>A7,12</td>
<td>A7,10</td>
</tr>
</tbody>
</table>

\( \geq? \)

Why not stop at 25?
Because we might have another object X [UB:24, Real:23]
Problems
• Learning from trajectories
• Distance measures
• Clustering
• Indexing, Distributed Indexing
• Pattern Queries
• Applications
• Map Creation
• Mode of Transportation
• Social networks applications
• Crowdsourcing Applications
• Privacy issues

Querying Full-Trajectories (Past)
• Examples:
  - Querying Along the Temporal Dimension: What was the location of a certain object from 7:00 AM to 10:00 AM yesterday?
  - Querying Along the Spatial Dimension: Find all objects that were in a certain area at 7:00 AM yesterday
  - Querying Along the Spatio-temporal Dimension: Find all objects that were close to each other from 7:00 AM to 8:00 AM yesterday

• Features:
  - Large number of historical trajectories
  - Persistent read-only data
  - The ability to query the spatial and/or temporal dimensions

Querying Full-Trajectories (Future)

Continuous queries over trajectories
• A framework was proposed for managing trajectory streams
• Continuous spatiotemporal queries could be posed over trajectories
• Real time responses of the given queries

A sql-like interface was proposed to declare the spatiotemporal queries
A graphical interface was proposed, where users could specify queries

Motion Pattern Queries

Example: “find trajectories that were in downtown LA, then sometime later went as close as possible to the Hollywood sign, then later ended up at LAX”
Motion Pattern Queries

Example: “find trajectories that were in downtown LA, then sometime later went as close as possible to the Hollywood sign, then later ended up at LAX”

Example: “find trajectories that were in downtown LA, then sometime later went as close as possible to the Hollywood sign, then later ended up at LAX”

Problems

- Learning from trajectories
  - Distance measures
  - Clustering
  - Indexing, Distributed Indexing
  - Pattern Queries
  - Applications
    - Map Creation
    - Mode of Transportation
    - Social networks applications
  - Crowdsourcing Applications
  - Privacy issues
Maps are incomplete and often out of date, so improve their quality using collections of sparse trajectories (GPS cloud).

The Map creation problem

Combine road segments into a network

(a) T-junction

(b) X-junction

Y-junction

How to model street networks

Comparison with OpenStreetMap

Problems

- Learning from trajectories
  - Distance measures
  - Clustering
  - Indexing, Distributed Indexing
  - Pattern Queries
  - Applications
    - Maps
    - Mode of Transportation
    - Social networks applications
  - Crowdsourcing Applications
  - Privacy issues
Accelerometer-based transportation mode detection on smartphones

**Segment-based features**
- The segment-based features we consider are the frequency of acceleration and breaking periods, the frequency and duration of the intermittent stationary periods, and the variance of individual peak-based features.

<table>
<thead>
<tr>
<th>Domain</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Segment</td>
<td>Variance of peak features (10 features), Peak frequency of features, Stationary duration, Stationary frequency</td>
</tr>
</tbody>
</table>

---

Using mobile phones to determine transportation mode

Classifier able to identify the transportation mode of an individual using:
- the phone’s GPS receiver sensor
- accelerometer sensor

The target classes:
- (stationary, walking, running, biking, motorized transport).

Features Used:
- GPS speed
- Accelerometer Variance
- Accelerometer DFT(3Hz)
- Accelerometer DFT(2Hz)
- Accelerometer DFT(1Hz)

Classifies used:
- C4.5 Decision Trees
- two stage system with a Decision Tree combined with an HMM

Checked the accuracy of the model when the user had the phone at a different position (Arm, Bag, Chest, Hand, Pocket, Waist).

---

Understanding transportation modes based on GPS data for web applications

Three part approach:
1. A change point segmentation method
   - Distinguish segments as walking or not walking (based on the idea that walking should be the transmission between different transportation modes).
   - Distinguish segments using a threshold of velocity and acceleration
2. An inference model
   - Received as input extracted features:
     - Heading Change Rate
     - Stop Rate
     - Velocity Change Rate
   - Used supervised classification methods
     - Decision Tree
     - Support Vector Machine
     - Bayesian Net
     - Conditional Random Field
3. A graph based post processing algorithm
   - Collect start/end points and change points and applied a hierarchical clustering algorithm on them
   - Build a Graph connecting the different clusters
   - Use spatial indexing to identify quickly the clusters
   - Calculate the probabilities changing from one transportation mode to another using the labeled data

---

Problems

- Learning from trajectories
  - Distance measures
  - Clustering
  - Indexing, Distributed Indexing
  - Pattern Queries
  - Applications
    - Map Creation
    - Mode of Transportaton
    - Social networks applications
  - Crowdsourcing Applications
  - Privacy issues

---

Capturing the Aesthetic Capital

- Urban spaces are related to the *emotional perceptions*
- Automatically generate routes that are not only *short* but also *emotionally pleasant*
- Which urban elements make people happy?

- Walkable cell of 200x200 meters
- Urbangems value for each emotional dimension

---

[HNT13]
[RMEDS10]
[ZCLXM10]
What is the purpose of people motion

- Create an Individual Mobility Network (IMN) for each user
- Extract a set of features:
  - Network Features
    - Centrality (clustering coefficient, avg path length)
    - Predictability (entropy)
    - Hubbiness (degree betweenness)
    - Volume (edge weight, flow per location)
  - Trip Features
    - Length, duration, time interval
- Use the Activity-Based Cascade (ABC) classifier that at each step creates a binary classifier aiming to distinguish a particular class (i.e. Going home) among the others

Problems

- Learning from trajectories
  - Distance measures
  - Clustering
  - Indexing, Distributed Indexing
  - Pattern Queries
  - Applications
    - Map Creation
    - Mode of Transportation
    - Social networks applications
- Crowdsourcing Applications
  - Privacy issues

Mobile Crowdsourcing Systems

- Recent mobile systems “sense” and collect sensor data provided by humans
  - Google Maps, Moovit, Uber, etc.
- Current solutions focus on centralized databases to:
  - Maintain all user sensed data
  - Preserve user privacy
  - Provide events of interest
- Bringing all data to a central location
  - involves high cost
  - May not be necessarily beneficial
- In the distributed setting each user
  - Maintains an individual local database that contains frequently-sampled sensed data
  - Issues queries on data stored across the system to retrieve sensed data and identify events of interest

Athens Authentic Marathon

- Waze Crowdsourcing App
  - Allows users to report traffic incidents
- Athens Authentic Marathon
  - Identified closed and congested roads
  - Users can use alternative routes
  - Traffic Events update in real-time

Challenges in Mobile Crowdsourcing

- The selection of the most appropriate users to perform a task is fundamental to the quality of the result
- Human users have different abilities and characteristics
  - Objectiveness, Perspective, Response Delays, location
  - Spammers perform tasks randomly
  - Users might be unavailable
- User privacy should be protected when users contribute to crowdsourcing systems
  - Mobile Crowdsourcing assumes that the shared data are coupled with the user mobility traces
  - Sophisticated attacks can expose the user identity, her sensitive locations and the places that the user has visited
- Exploiting public information can lead to leak of private information
  - Tracking: Identify the user’s mobility patterns
  - Identification: Isolate her frequently visited locations
  - Profiling: Identify places that could profile the user
Privacy Preservation Problem

How to design a system that will allow users to use crowdsourcing apps and share geo-located content without revealing sensitive information?

- **Design Objectives**
  - Users post data coupled with their identifiers, location and timestamp
  - Users tune their privacy levels
  - User mobility should be kept locally

- **Goals**
  - Evaluate the user privacy exposure on resource-constrained devices
  - Analyze large amounts of spatiotemporal data in real-time
  - Identify critical points that may expose user privacy and alert the user

- **Privacy Definition:** The set of shared data \( R_i \) preserves user privacy when
  - An attacker cannot generate an approximation of any user trajectory from \( T_i \)
  - The user sensitive locations cannot be determined from the shared data \( R_i \) based on their amount and frequency

Deal with attacks on user trajectories

- **Problem**
  - Local data contain a lot of similar trajectories
  - An attacker can identify the user reveal frequent locations or track the user
  - Each smartphone periodically
    - Defines the \( k \) most frequent locations \( loc_{ij} \)

  - We choose \( m \) trajectories that contain \( m \) from the \( (m+k) \) most frequent locations
  - Trajectory is selected when removing the trajectory increases the Entropy
  - Measures the quality of trajectories with respect to the anonymity it provides
  - We select \( n \) trajectories that do not contain frequent locations
  - These trajectories can be forwarded to a specific neighbor or to different neighbors
  - Both types of trajectories may contain sensitive data (e.g. political meetings)

Trajectory Anonymizing

- Given a database of trajectories representing user mobility for a number of users, generate an anonymized version of this database such that:
  - The identities of all individuals are protected in the anonymized database from adversaries who have certain knowledge at their disposal
  - The effect of sanitization to the original database is minimal, i.e. the original database is minimally distorted to produce its anonymized counterpart so that utility is maximized

Algorithms for trajectory anonymization

- Several algorithms have been proposed for anonymizing a database of user trajectories
  - **Existing approaches can be categorized as follows:**
    - Approaches that simplify user trajectories into sequences of POIs \( \rightarrow \) spatiotemporal data is simplified to sequential data (**suppression approaches**)
    - Approaches that take into consideration the spatiotemporal nature of user trajectories when anonymizing the database (**clustering & perturbation**)
  - The assumed knowledge of the attackers also varies from approach to approach
PLS: Protecting privacy when publishing location sequences (Terrovitis et al 08)

- User trajectories are simplified into sequences of POIs (a1, b1, …)
- Different adversaries own different parts of the POIs (e.g., all “a” can denote branches of the same bank, where the adversary is the bank owner)
- The portions of trajectories can be used for a linkage attack to reveal user identity (e.g., only user 5 goes through a1 and b3 so attacker “a” easily learns the identity of this user as well as that he also went to b1)

NWA: Protecting privacy when publishing complete user trajectories (Abul et al 08)

- User trajectories are represented as cylindrical volumes (of radius δ) that capture the location imprecision due to sampling and positioning systems
- All trajectories are assumed to have the same sampling time, to move along a straight line and at a constant speed between two consecutive observations
- Two objects that move within the same cylinder and follow almost the same route are considered to be indistinguishable
- The goal is to anonymize trajectories as a whole by assuming attackers that can identify each user in any location and at any time

TGA: Trajectory anonymization based on grouping and reconstruction (Nergiz et al 09)

- TGA assumes that space is discretized into ε×ε grids and time is discretized in buckets of size τ
- User trajectories are represented as temporarily ordered sets of spatio-temporal points
- Adversaries with the following types of knowledge (and malicious intents) are assumed:
  - Already know some portion of the trajectory of an individual in the database and want to learn the rest
  - Already know the whole trajectory of an individual and want to learn some sensitive information (e.g., requests done to LSBs)

NWA Anonymization strategy

- Co-localization: 2 trajectories defined in the same time interval are co-localized if for each (x, y) point (same t) it holds that Euclidean distance dist((x1, y1), (x2, y2)) ≤ δ
- (k, δ)-anonymity: Based on co-localization
  - A set of at least k trajectories that are co-localized with respect to a radius δ

TGA Anonymization strategy

- TGA offers trajectory anonymization by
  - Generalizing sets of k trajectories to group them together by enclosing their respective locations into MBRs
  - Representing each group of k trajectories as an anonymized trajectory
  - Releasing atomic trajectories sampled randomly from the area covered by the anonymized trajectories
CrowdAlert app deployed in Dublin City

Using the mobile crowdsourcing app to reach humans during an event:

- Categories
  - 4 Ideas
  - 3 Ideas
  - 2 Ideas
  - 1 Ideas

References – Trajectories; Moving Objects

References – Trajectories; Moving Objects

References


References – Similarity of Spatial Data/Similarity and Distance Functions


References – Application Domains


References


References – Application Domains


References


References – Application Domains


References


References – Application Domains


References


References – Application Domains

References

- [SPG07] Spera-Papadimitriou, Jiayun Sun, Christina Faloutsos: Streaming Pattern Discovery in Multiple Time-Series. VLDB 2007: 697-708
MINING DATA FROM MOBILE DEVICES

Dimitrios Gunopulos
University of Athens
dg@di.uoa.gr