Urban Computing

Dr. Mitra Baratchi

Leiden Institute of Advanced Computer Science - Leiden University

March 21, 2019
Third Session: Urban Computing - Machine learning
Agenda for this session

- Part 1: Machine learning for spatio-temporal data
- Part 2: Modeling spaces
  - Spatial profiling
- Part 3: Modeling individual trajectories
  - Trajectory clustering
  - Trajectory forecasting
- Part 4: Modeling social trajectories
  - Memory-based POI recommendation
  - Model-based POI recommendation
Part 1: Machine learning for spatio-temporal data
Machine learning for spatio-temporal data

How can we use machine learning algorithms to deal with data of spatio-temporal nature?
How can we use machine learning algorithms to deal with data of spatio-temporal nature?

- High dimensional (in time and space)
- Inherent patterns (spatial and temporal auto-correlation)
- Many types of imperfections (noise, missing data, inconsistent sampling rate)
Machine learning for spatio-temporal data

- Do we know any algorithm that is suited for high-dimensional data?
- Do you know any machine learning algorithm that is inherently aware of space (areas, distances, neighborhoods) and time (periodicity, durations, intervals, etc.)?
- Do you know any machine learning algorithm that is inherently robust to noise, missing data, etc.?
Questions we often need to answer

- How to define the machine learning task for a given problem?
  - Representation
  - Similarity measure?
  - Objective function?
    - Unsupervised learning ← More popular approach
    - Requires thinking about a means for evaluating the performance
- How to find algorithms that are both aware of space and time?
- How to deal with data imperfections algorithmically?
Data look

<table>
<thead>
<tr>
<th>Id</th>
<th>Timestamp</th>
<th>Location-long</th>
<th>Location-lat</th>
</tr>
</thead>
<tbody>
<tr>
<td>2635</td>
<td>1997-07-24 20:50:00</td>
<td>-149.007</td>
<td>63.809</td>
</tr>
<tr>
<td>2635</td>
<td>1997-07-31 02:52:48</td>
<td>-149.026</td>
<td>63.803</td>
</tr>
<tr>
<td>2635</td>
<td>1997-08-03 01:47:04</td>
<td>-149.046</td>
<td>63.795</td>
</tr>
</tbody>
</table>

- **Time**
- **Location**
- **MobileEntity ID**, or anything else

GPS

<table>
<thead>
<tr>
<th>ScannerId</th>
<th>Time</th>
<th>DeviceAddress</th>
</tr>
</thead>
<tbody>
<tr>
<td>1297</td>
<td>1434550018</td>
<td>58884273fc450e86</td>
</tr>
<tr>
<td>1297</td>
<td>1434550018</td>
<td>58884273fc450e86</td>
</tr>
<tr>
<td>1297</td>
<td>1434550018</td>
<td>58884273fc450e86</td>
</tr>
<tr>
<td>1297</td>
<td>1434550018</td>
<td>58884273fc450e86</td>
</tr>
<tr>
<td>1297</td>
<td>1434550018</td>
<td>58884273fc450e86</td>
</tr>
<tr>
<td>1297</td>
<td>1434550018</td>
<td>58884273fc450e86</td>
</tr>
</tbody>
</table>
What are different ways we can look at trajectory data?

<table>
<thead>
<tr>
<th>Query type</th>
<th>Location</th>
<th>EntityID</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Fixed</td>
<td>Fixed</td>
<td>Variable</td>
</tr>
<tr>
<td>2</td>
<td>Fixed</td>
<td>Variable</td>
<td>Variable</td>
</tr>
<tr>
<td>3</td>
<td>Variable</td>
<td>Fixed</td>
<td>Variable</td>
</tr>
<tr>
<td>4</td>
<td>Variable</td>
<td>Variable</td>
<td>Variable</td>
</tr>
</tbody>
</table>

**Table:** Different ways of looking at trajectory data
Part 2: Modeling spaces
What are different ways we can look at trajectory data?

<table>
<thead>
<tr>
<th>Query type</th>
<th>Location</th>
<th>Entity</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Fixed</td>
<td>Fixed</td>
<td>Variable</td>
</tr>
<tr>
<td>Fixed</td>
<td></td>
<td>Variable</td>
<td>Variable</td>
</tr>
<tr>
<td>Variable</td>
<td>Fixed</td>
<td>Fixed</td>
<td>Variable</td>
</tr>
<tr>
<td>Variable</td>
<td>Variable</td>
<td>Variable</td>
<td>Variable</td>
</tr>
</tbody>
</table>

**Table:** Different ways of looking at trajectory data
Research directions:

- Spatial patterns
- Point of interest labeling
Example: Spatial profiles, spatial fingerprints (Spaceprints)
Profiling locations

- **Given:**
  - Data in form of \( \{ (s_i, e_j, t) | i \in 1...N, j \in 1...M, t \in 1...T \} \)

- **Objective:**
  - Creating profiles for each space \( s_i \)
  - Each space should have a unique profile
  - Profiles reflect functions of spaces
    - Restaurant
    - Cafe
    - Classroom
    - ...

How does the data look like?

- Detections of entities with unique identifiers in a space
- How do we compare spaces to each other based on this form of data?

- How to represent data? Instances, attributes?
How does the data look like?

- Detections of entities with unique identifiers in a space
- How do we compare spaces to each other based on this form of data?

How to represent data? Instances, attributes?
Creating instances and attributes

Option 1:

- **Instances**: Each day in a space
- **Attributes**: Hourly densities

**Figure**: Density-based features
Example: What does the profile of cafes look like?

If we collect data from different cafes, how can we use such data to create profiles for them so that we see their similarities?
What features define a space?

What people do in cafes?

- Meeting
- Take away coffee
- Work
- Watching sport matches (a cafes next to a sport center)

How can we capture these activities in form of features?
What features define a space?

What people do in cafes?

- Meeting
- Take away coffee
- Work
- Watching sport matches (a cafés next to a sport center)

How can we capture these activities in form of features? possibly people being present synchronously in different windows over time?

- Density based features do not represent them
Windows over time

Where can presences over time happen?
Windows over time
Windows over time
Windows over time
Windows over time
Windows over time
Many possible windows
Example: Windows over time
Example: Windows over time

Looking at these windows and see who is present in them
Example: Windows over time

- Looking at these windows and see who is present in them
- We need to determine how to count within a window
Example: Windows over time

Presence in a window is considered together with a resolution of counting
Example: Windows over time
Example: Windows over time
Example: Windows over time

Many groups are possibly formed → in real world each group can be formed following a common activity.

If the activity is recurring, it can be part of the profile or fingerprint of the space.
Resolution of windows

- We are not sure about the frequency with which devices are being detected. This is device dependent.
- Number of entities in the same window can be considered using different resolutions.
Creating instances and attributes

Option 2: Spaceprints feature vector [BHvS17]

- **Instances:** Each day in a space
- **Attributes:** The number of devices being present in windows \( w \) with variable:
  - Starting time \( t_{\text{start}} \)
  - Duration \( \tau \)
  - Sampling resolution \( t_s \)
Feature vector

If we calculate all possible features according to same template, we will have a feature vector.

Figure: 1

Image sources: [BHvS17]
Space profiles

Figure: (Left) Option 2: feature vectors acquired from Spaceprint (right) Option 1: feature vectors acquired from density based counting.  

\[^2\] Image sources: [BHvS17]
Part 3: Modeling individual trajectories
What are different ways we can look at trajectory data?

Table: Different ways of looking at trajectory data

<table>
<thead>
<tr>
<th>Query type</th>
<th>Location</th>
<th>Entity</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Fixed</td>
<td>Fixed</td>
<td>Variable</td>
</tr>
<tr>
<td>2</td>
<td>Fixed</td>
<td>Variable</td>
<td>Variable</td>
</tr>
<tr>
<td>3</td>
<td>Variable</td>
<td>Fixed</td>
<td>Variable</td>
</tr>
<tr>
<td>4</td>
<td>Variable</td>
<td>Variable</td>
<td>Variable</td>
</tr>
</tbody>
</table>
Research directions

- Trajectory clustering
- Trajectory prediction
Example 1: clustering trajectories
Objective

- **Given:**
  - A set of trajectories presented in form of multi-dimensional points $Tr = p_1, p_2, p_3, p_n$.
  - A point $p_i$ is a 2-dimensional entity $(x, y)$.
  - Trajectories segmented to day level

- **Objective:**
  - We look for clusters representing frequent patterns
  - Clusters represent the most visited path
    - Road segment
What clustering algorithms exist? Which ones can be useful?
Density-based clustering

Very popular in trajectory data mining

- Clustering based on **density** (local cluster criterion), such as density connected points
- Each cluster has a considerably higher density of points
- **Advantage**: easier parameter setting compared to algorithms such as K-means:
  - You do not need to define $K$. 
DBSCAN

- DBSCAN: Density-based spatial clustering of applications with noise
- Two parameters
  - Eps ($\epsilon$): Maximum radius of the neighborhood from a point
  - MinPts: Minimum number of points in an Eps-neighborhood of that point
DBSCAN: Core, Border and Noise Points

- $N_\epsilon(q) : \{ p | \text{dist}(p, q) \leq \epsilon \}$
- Directly density-reachable: A point $p$ is directly density-reachable from a point $q$ wrt. $\epsilon$, MinPts if
  - $p$ belongs to $N_\epsilon(q)$
  - core point condition $|N_\epsilon(q)| \geq \text{MinPts}$
Trajectory clustering

- DBSCAN for trajectory clustering
- Option 1:
  - Take trajectories as data instances
  - Modify DBSCAN to cluster trajectories
Issues with option 1

- **Trajectory partitions:** If we consider only complete trajectories, we miss valuable information on common Sub-trajectories.
  - Finding the characteristic point of trajectories

- **Similarity measure:** How to measure the distance between trajectories
Option 2: Traclus: An example of using DBSCAN for trajectory clustering [LHW07]
Figure: How to find common sub-trajectories?

- Data instances for DBSCAN should represent sub-trajectory candidates
- Partition trajectories to simple line segments first
Distance function

Now we need a way to measure the distance between line segments?
Distance measure

\[ \text{Dist}(L_i, L_j) = w_\perp \cdot d_\perp(L_i, L_j) + w_\parallel \cdot d_\parallel(L_i, L_j) + d_\theta(L_i, L_j) \]

- **Perpendicular distance:** \( d_\perp = \frac{l_\perp^2 + l_\perp^2}{l_\perp + l_\perp} \)
- **Parallel distance:** \( d_\parallel = \text{Min}(l_\parallel, l_\parallel) \)
- **Angle distance:** \( d_\theta = ||L_k|| \cdot \sin(\theta) \)
Final solution:

Partition and group framework:

- Partition trajectories
- Cluster line segments using DBSCAN modified based on the new similarity measure
Example 2: trajectory forecasting
Objective

▶ Given:
  ▶ A set of trajectories presented in form of multidimensional points $Tr = p_1, p_2, p_3, p_n$.
  ▶ A point $p_i$ is 2-dimensional entity $(x, y)$.

▶ Objective:
  ▶ We want to forecast future points of the trajectory $Tr = p_{n+1}, p_{n+2}, \ldots$
What algorithms do we know that can capture temporal aspects? Which ones can be used for forecasting?
Some algorithms that are aware of time (sequential order)

- Dynamic Bayesian networks: a Bayesian network which relates variables to each other over adjacent time steps.
- Hidden markov model
Markovian process

- A Markov process can be thought of as memory-less
  - The future of the process is solely based on its present state just as well as one could know the process’s full history.
- The $n$ th observation in a chain is influenced only by the $n - 1$ observation

$$p(x_n|x_1, \ldots, x_{n-1}) = p(x_n|x_{n-1})$$
Hidden Markov model

- $X$ - States
- $Y$ - Observations
- $A$ - State transition probabilities
  - $a_{ij}$ is probability of transition from state $i$ to $j$
- $B$ - Output probabilities
  - $b_{ij}$ is probability emission state $i$ to observation $j$
- $\pi$ Initial state
Hidden Markov Model

- Option 1: using Hidden Markov Model to model trajectories → instances are points on trajectories

- Issue with Option 1:
  - Trajectories are composed of movements with high speed and almost zero speed
  - Staying at home for 5 hours, being at work for 8 hours, ...
  - States are meaningful if the durations is considered → Hidden semi Markov model considers an extra duration distribution for states
Hidden semi Markov Model

Give instances as ordered trajectory points in time the following model parameters should be calculated:

- $A$ (transitions matrix)
- $B$ (emission matrix)
- $\Pi$ (initial state vector)
- $D$ (State duration distribution)
How to estimate model parameters?

- Different algorithms:
  - Baum welch
  - Viterbi,
  - etc
Hierarchical HSMM on human mobility data [BMH+14]

We will be able to find:

- **Super states** with duration of weekdays and week ends
- **States** with the duration of hours of stay in different locations
Example of Hierarchical HSMM on Geolife data

4 image source: [BMH+14]
Part 3: Modeling social trajectories
What are different ways we can look at trajectory data

Table: Different ways of looking at trajectory data

<table>
<thead>
<tr>
<th>Query type</th>
<th>Location</th>
<th>Entity</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Fixed</td>
<td>Fixed</td>
<td>Variable</td>
</tr>
<tr>
<td>2</td>
<td>Fixed</td>
<td>Variable</td>
<td>Variable</td>
</tr>
<tr>
<td>3</td>
<td>Variable</td>
<td><strong>Fixed</strong></td>
<td>Variable</td>
</tr>
<tr>
<td>4</td>
<td>Variable</td>
<td>Variable</td>
<td>Variable</td>
</tr>
</tbody>
</table>
Research directions:

- Understanding users interests based upon their locations.
- Understanding locations functions via user mobility.
- Point of interest (POI) recommendation
POI recommendation

- **Given:**
  - Given data $U = \{u_1, u_2, .. u_n\}$ a set of users, and $L = \{l_1, l_2, ... l_m\}$ a set of POIs, and $C = \{c_{1,1}, ..., c_{i,j}\}$ a set of check-in of users in POIs where $c_{i,j}$ denotes the number of times user $u_i$ checked in $l_j$.

- **Objective:**
  - Recommending a location to a user through inferring the preference of the user to check-in to a location they have not checked-in before.
  - Predicting will this user ever check-in to a POI (time is not that important).
  - Performance is typically measured through precision and recall of top K recommended locations.
POI recommendation

- Recommender systems are information filtering systems which attempt to predict the rating or preference that a user would give, based on ratings that similar users gave and ratings that the user gave on previous occasions.
- Many different types of Location-Based Social Networks (LBSN) (Foursquare, Brightkite, Gowalla)
- These are the context information we want the recommender system for LBSN to be aware of
Challenges of POI recommendation

- **Implicit feedback**: check-ins, visits rather than explicit feedback in form of ratings

- **Data sparsity**: A lot of places do not have visit data. For example, the sparsity of Netflix data set is around 99%, while the sparsity of Gowalla is about $2.08 \times 10^{-4}\%$

- **Cold start**:
  - New locations have no ratings
  - New users have no history

- **Context**: we want the algorithms to be aware of:
  - Spatial influence
  - Social influence
  - Temporal influence
What recommendation algorithms exist? Which ones can be useful?
Collaborative filtering

- Memory-based
  - User-based
  - Item-based
- Model-based
  - Matrix factorization
  - SVD
Example 1: Memory-based POI recommendation
Memory-based

- **Memory-based**: Uses memory of past ratings
- **K-nearest neighbor**: Using data of nearest neighbors
- Predicting ratings by getting an average of ratings:
  - **User-based**: ratings based on a user’s most similar neighbors
  - **Item-based**: ratings of a user based on an item’s most similar neighbors
We need to measure the similarity between users based on their check-in history.

- The first component of user-based POI recommendation algorithm is determining how to compute the similarity weight $sim(u, v)$ between user $u$ and $v$. 
Collaborative filtering, similarity

<table>
<thead>
<tr>
<th></th>
<th>item₁</th>
<th>item₂</th>
<th>item₃</th>
<th>item₄</th>
<th>item₅</th>
<th>item₆</th>
<th>item₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>u₁</td>
<td>4</td>
<td></td>
<td></td>
<td>5</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>u₂</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>u₃</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>u₄</td>
<td>3</td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

- Consider $u_i$ and $u_j$ with rating vectors $r_i$ and $r_j$
- Intuitively capture this: $sim(u₁, u₂) > sim(u₁, u₃)$
Cosine similarity

\[
\begin{array}{c|cccccc}
 & \text{item}_1 & \text{item}_2 & \text{item}_3 & \text{item}_4 & \text{item}_5 & \text{item}_6 \\
\hline
\text{u}_1 & 4 &  &  & 5 & 1 &  \\
\text{u}_2 & 5 & 5 & 4 &  &  &  \\
\text{u}_3 &  & 2 & 4 & 5 &  &  \\
\text{u}_4 & 3 &  &  &  &  & 3
\end{array}
\]

\[
\text{sim}(u_i, u_j) = \frac{r_i \cdot r_j}{||r_i|| \ ||r_j||} = \frac{r_i \cdot r_j}{\sqrt{\sum_i r_i^2} \sqrt{\sum_j r_j^2}}
\]
Cosine similarity

\[
sim(u_i, u_j) = \frac{r_i \cdot r_j}{\|r_i\| \|r_j\|} = \frac{r_i \cdot r_j}{\sqrt{\sum_i r_i^2} \sqrt{\sum_j r_j^2}}
\]

- replace empty with 0
## Cosine similarity

<table>
<thead>
<tr>
<th></th>
<th>item(_1)</th>
<th>item(_2)</th>
<th>item(_3)</th>
<th>item(_4)</th>
<th>item(_5)</th>
<th>item(_6)</th>
<th>item(_3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(u_1)</td>
<td>4</td>
<td></td>
<td></td>
<td>5</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(u_2)</td>
<td>5</td>
<td>5</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(u_3)</td>
<td></td>
<td></td>
<td>2</td>
<td>4</td>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(u_4)</td>
<td></td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
</tr>
</tbody>
</table>

\[
\text{sim}(u_i, u_j) = \frac{r_i \cdot r_j}{\|r_i\| \|r_j\|} = \frac{r_i \cdot r_j}{\sqrt{\sum_i r_i^2} \sqrt{\sum_j r_j^2}}
\]

- replace empty with 0
- \(\text{sim}(u_1, u_2) = 0.38, \text{sim}(u_1, u_3) = 0.32\)
Cosine similarity for check-ins

If we replace the **rating vector** by the user’s **check-in vector** we can measure similarities

- Check-ins are often very sparse, we can consider binary check-in vectors
- \( c_{ij} = 1 \) if user \( u_i \) has checked in \( l_j \in L \) before
- The cosine similarity weight between users \( u_i \) and \( u_k \),
  \[
  w_{ik} = \frac{\sum_{l_j \in L} c_{ij} c_{kj}}{\sqrt{\sum_{l_j \in L} c_{ij}^2} \sqrt{\sum_{l_j \in L} c_{kj}^2}}
  \]
- Recommendation score based on \( k \) most similar users
  \[
  \hat{c}_{ij} = \frac{\sum_{u_k} w_{ik} \cdot c_{kj}}{\sum_{u_k} w_{ik}}
  \]
Context: Geographic influence

- How to include geographical influences?
- The Toblers First Law of Geography is also represented as geographical clustering phenomenon in users check-in activities.
Context: Geographic influence

- How to include geographical influences?
- The Toblers First Law of Geography is also represented as geographical clustering phenomenon in users check-in activities.
  - Users prefer to visit nearby POIs rather than distant ones; people tend to visit POIs close to their homes or offices
  - People may be interested in visiting POIs close to the POI they are in favor of even if it is far away from their home; users may be interested in POIs surrounded a POI that users prefer.
Power-law geographical model [YYLL11]

- Check-in probability follows power law distribution
  - \( y = a \times x^b \)
    - \( x \) and \( y \) refer to the distance between two POIs visited by the same user and its check-in probability
    - \( a \) and \( b \) are parameters of power law distribution
- For a given POI \( l_j \), user \( u_i \), and her visited POI set \( L_i \), the likelihood probability for \( u_i \) to check in \( l_j \) as follows
  - \( P(l_j|L_i) = \frac{P(l_j \cup L_i)}{P(L_i)} = \prod_{l_y \in L_i} P(d(l_j, l_y)) \)

Figure: Check-in probabilities may follow a power law distribution
Other ways of considering geographic influence [YC15]

- Power-law based model
- Distance based model
- Multi-center Gaussian model
Multi-center geographical influence

Geographical influence, multi-center

- Check-ins happen near a number of centers
  - Work area
  - Home area
  - etc.
Multi-center geographical influence

- Probability of check-in of user $u$ in location $l$
- Probability of $l$ belonging to any of those centers
- $P(l|C_u) = \sum_{c_u=1}^{\left|C_u\right|} P(l \in c_u) \frac{f_{cu}^\alpha}{\sum_{i \in C_u} f_{i}^\alpha} \frac{N(l|\mu_c, \Sigma_C)}{\sum_{i \in C_u} N(l|\mu_i, \Sigma_i)}$
  - Where $P(l \in c_u) = \frac{1}{d(l, c_u)}$ is the probability of POI $l$ belonging to the center $c_u$,
  - $\frac{f_{cu}^\alpha}{\sum_{i \in c_u} f_{i}^\alpha}$ is the normalized effect of check-in frequency on the center $c_u$ and parameter $\alpha$ maintains the frequency aversion property
  - $N(l|\mu_c, \Sigma_C)$ is the probability density function of Gaussian distribution
Social influence

- Depending on a source, social information may also be available which can be used to improve the recommendation performance
- The social influence weight between two friends $u_i$ and $u_k$ based on both of their social connections and similarity of their check-in activities

$$SI_{kj} = \nu \cdot \frac{|F_k \cap F_i|}{|F_k \cup F_i|} + (1 - \nu) \frac{|L_k \cap L_i|}{|L_k \cup L_i|}$$

- $\nu$ is a tuning parameter ranging within $[0, 1]$
- $F_k$ and $L_k$ denote the friend set and POI set of user $u_k$
How to put all information in one model?

A recommender system which has embedded all these influences?
A recommender system which has embedded all these influences?

- **Fused model**: The fused model fuses recommended results from collaborative filtering method and recommended results from models capturing geographical influence, social influence, and temporal influence.
Fused model

- Check-in probability of user $i$ in location $j$:
  - $S_{i,j} = (1 - \alpha - \beta)S_{i,j}^u + \alpha S_{i,j}^s + \beta S_{i,j}^g$
  - $S_{i,j}^u$, $S_{i,j}^s$, $S_{i,j}^g$ are user preference, social influence, and geographical influence
  - where ($\alpha$ and $\beta$) ($0 \leq \alpha + \beta \leq 1$) are relative importance of social influence and geographical influence
Example 2: Model based POI recommendation
Model-based recommendation

- **Latent variable models**: how to model users and items without having any features?
- **Build the hidden model of a user**: what a user looks for in an item?
- **Build the hidden model of an item**: what does it offer?
- **Methods**:
  - Matrix factorization
  - Singular value decomposition
Factorization: Latent factor models

Assume that we can approximate the rating matrix $R$ as a product of $U$ and $P^T$

$$R = U \times P^T$$

<table>
<thead>
<tr>
<th></th>
<th>$p_1$</th>
<th>$p_2$</th>
<th>$p_3$</th>
<th>$p_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u_1$</td>
<td>4.5</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$u_2$</td>
<td>4.0</td>
<td>3.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$u_3$</td>
<td>5.0</td>
<td>2.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$u_4$</td>
<td>3.5</td>
<td>4.0</td>
<td>1.0</td>
<td></td>
</tr>
</tbody>
</table>

$(k = 2)$ factors

<table>
<thead>
<tr>
<th></th>
<th>$u_1$</th>
<th>$u_2$</th>
<th>$u_3$</th>
<th>$u_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_1$</td>
<td>1.2</td>
<td>1.4</td>
<td>1.5</td>
<td>1.2</td>
</tr>
<tr>
<td>$p_2$</td>
<td>0.8</td>
<td>0.9</td>
<td>1.0</td>
<td>0.8</td>
</tr>
<tr>
<td>$p_3$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$p_4$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$$
\begin{array}{cccc}
p_1 & p_2 & p_3 & p_4 \\
1.5 & 1.2 & 1.0 & 0.8 \\
1.7 & 0.6 & 1.1 & 0.4
\end{array}
$$
How do we find $Q$ and $P$

- Singular value decomposition SVD
- ...
SVD (Singular value decomposition)

- $\Sigma$ is a diagonal where entries are positive and sorted in decreasing order
- $U$ and $V$ are column orthogonal: $U^T U = I$, $V^T V = I$
- This leads to a unique decomposition $U$, $V$, $\Sigma$
Optimizing by solving this problem

- Find matrices $U$ and $\Sigma$ and $V$ that minimize this expression
  
  $\min_{U,V,\Sigma} \sum_{i,j \in A} (A_{ij} - [U\Sigma V^T]_{ij})^2$

- In case of sparse matrices we have to makes sure that error is calculated on the non-zero elements
How to include other context in a matrix factorization model?

- **Joint model**: The joint model establishes a joint model to learn the user preference and the influential factors together.
Joint model

Two different types of joint models:

- Incorporating factors (e.g., geographical influence and temporal influence) into traditional collaborative filtering model like matrix factorization and tensor factorization

- Generating a graphical model according to the check-ins and extra influences like geographical information.
Joint geographical modeling and matrix factorization

Augment user’s and POI’s latent factors with geographical influence

- **Activity areas** of a user are determined by the grid area where the user may show up and a number indicating the possibility of appearing in that area

- **Influence areas** of a POI is a collection of pairs of a grid area to which the influence of this POI can be propagated and a number indicating the quantity of influence from this POI.
Joint geographical modeling and matrix factorization [LZX+14]

Figure: Geo matrix factorization

- **MF**: $R = UP^T$
- **GeoMF**: $R = UP^T + XY^T$
  - $X$ is users’ activity area matrix
  - $Y$ is POIs’ influence area matrix
Generating influence areas

Figure: Generating influence areas for POIs

Image source: [LZX+ 14]
End of theory!
References I


References II

