Attacks on collaborative recommender systems
Agenda

- Introduction
- Characterization of Attacks
- Attack models
- Effectiveness analysis
- Countermeasures
- Privacy aspects
- Discussion
Introduction / Background

- **(Monetary) value of being in recommendation lists**
  - Individuals may be interested to push some items by manipulating the recommender system
  - Individuals might be interested to decrease the rank of other items
  - Some simply might want to sabotage the system..

- **Manipulation of the "Internet opinion"**
  - Malevolent users try to influence behavior of recommender systems
    - System should include a certain item very often/seldom in its recommendation list

- **A simple strategy?**
  - (Automatically) create numerous fake accounts / profiles
  - Issue high or low ratings to the "target item"
    - Will not work for neighbor-based recommenders
    - More elaborate attack models required
    - Goal is to insert profiles that will appear in neighborhood of many
Example profile injection

- Assume that a memory-based collaborative filtering is used with:
  - Pearson correlation as similarity measure
  - Neighborhood size of 1
    - Only opinion of most similar user will be used to make prediction

<table>
<thead>
<tr>
<th>User</th>
<th>Item1</th>
<th>Item2</th>
<th>Item3</th>
<th>Item4</th>
<th>...</th>
<th>Target</th>
<th>Pearson</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>...</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>User1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>...</td>
<td>5</td>
<td>-0.54</td>
</tr>
<tr>
<td>User2</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>...</td>
<td>2</td>
<td>0.68</td>
</tr>
<tr>
<td>User3</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>...</td>
<td>4</td>
<td>-0.72</td>
</tr>
<tr>
<td>User4</td>
<td>1</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>...</td>
<td>1</td>
<td>-0.02</td>
</tr>
</tbody>
</table>
Example profile injection

- Assume that a memory-based collaborative filtering is used with:
  - Pearson correlation as similarity measure
  - Neighborhood size of 1
    - Only opinion of most similar user will be used to make prediction

<table>
<thead>
<tr>
<th></th>
<th>Item1</th>
<th>Item2</th>
<th>Item3</th>
<th>Item4</th>
<th>...</th>
<th>Target</th>
<th>Pearson</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>...</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>User1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>...</td>
<td>5</td>
<td>-0.54</td>
</tr>
<tr>
<td>User2</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>...</td>
<td>2</td>
<td>0.68</td>
</tr>
<tr>
<td>User3</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>...</td>
<td>4</td>
<td>-0.72</td>
</tr>
<tr>
<td>User4</td>
<td>1</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>...</td>
<td>1</td>
<td>-0.02</td>
</tr>
</tbody>
</table>

User2 most similar to Alice
Example profile injection

- Assume that a memory-based collaborative filtering is used with:
  - Pearson correlation as similarity measure
  - Neighborhood size of 1
    - Only opinion of most similar user will be used to make prediction

<table>
<thead>
<tr>
<th></th>
<th>Item1</th>
<th>Item2</th>
<th>Item3</th>
<th>Item4</th>
<th>...</th>
<th>Target</th>
<th>Pearson</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>...</td>
<td>?</td>
<td>-0.54</td>
</tr>
<tr>
<td>User1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>...</td>
<td>5</td>
<td>-0.54</td>
</tr>
<tr>
<td>User2</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>...</td>
<td>2</td>
<td>0.68</td>
</tr>
<tr>
<td>User3</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>...</td>
<td>4</td>
<td>-0.72</td>
</tr>
<tr>
<td>User4</td>
<td>1</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>...</td>
<td>1</td>
<td>-0.02</td>
</tr>
<tr>
<td>Attack</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>...</td>
<td>5</td>
<td>0.87</td>
</tr>
</tbody>
</table>

← User2 most similar to Alice

Attack
Example profile injection

- Assume that a memory-based collaborative filtering is used with:
  - Pearson correlation as similarity measure
  - Neighborhood size of 1
    - Only opinion of most similar user will be used to make prediction

<table>
<thead>
<tr>
<th></th>
<th>Item1</th>
<th>Item2</th>
<th>Item3</th>
<th>Item4</th>
<th>...</th>
<th>Target</th>
<th>Pearson</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>...</td>
<td>?</td>
<td></td>
</tr>
<tr>
<td>User1</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>...</td>
<td>5</td>
<td>-0.54</td>
</tr>
<tr>
<td>User2</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>...</td>
<td>2</td>
<td>!</td>
</tr>
<tr>
<td>User3</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>...</td>
<td>4</td>
<td>-0.72</td>
</tr>
<tr>
<td>User4</td>
<td>1</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>...</td>
<td>1</td>
<td>-0.02</td>
</tr>
<tr>
<td>Attack</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>...</td>
<td>5</td>
<td>0.87</td>
</tr>
</tbody>
</table>

User2 most similar to Alice
Attack most similar to Alice
Characterization of profile insertion attacks

- **Attack dimensions**
  - Push attack:
    - Increase the prediction value of a target item
  - Nuke attack:
    - Decrease the prediction value of a target item
    - Make the recommender system unusable as a whole

- No technical difference between push and nuke attacks

- Nevertheless Push and Nuke attacks are not always equally effective

- **Another differentiation factor between attacks:**
  - Where is the focus of an attack? Only on particular users and items?
  - Targeting a subset of items or users might be less suspicious
  - More focused attacks may be more effective (attack profile more precisely defined)
Characterization of profile insertion attacks

- **Classification criteria for recommender system attacks include:**
  - **Cost**
    - How costly is it to make an attack?
    - How many profiles have to be inserted?
    - Is knowledge about the ratings matrix required?
      - Usually it is not public, but estimates can be made
  - **Algorithm dependability**
    - Is the attack designed for a particular recommendation algorithm?
  - **Detectability**
    - How easy is it to detect the attack
The Random Attack

General scheme of an attack profile

<table>
<thead>
<tr>
<th>Item1</th>
<th>...</th>
<th>ItemK</th>
<th>...</th>
<th>ItemL</th>
<th>...</th>
<th>ItemN</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>r_1</td>
<td>...</td>
<td>r_k</td>
<td>...</td>
<td>r_l</td>
<td>...</td>
<td>r_n</td>
<td>X</td>
</tr>
</tbody>
</table>

- selected items  filler items  unrated items

- Attack models mainly differ in the way the profile sections are filled

Random attack model

- Take random values for filler items
  - Typical distribution of ratings is known, e.g., for the movie domain
    (Average 3.6, standard deviation around 1.1)
  - Idea:
    - generate profiles with "typical" ratings so they are considered as neighbors to many other real profiles
- High/low ratings for target items
- Limited effect compared with more advanced models
The Average Attack

- use the individual item's rating average for the filler items
- intuitively, there should be more neighbors
- additional cost involved: find out the average rating of an item
- more effective than Random Attack in user-based CF
  - But additional knowledge is required
- Quite easy to determine average rating values per item
  - Values explicitly provided when item is displayed
Effectiveness

- By the way: what does effective mean?
- Possible metrics to measure the introduced bias
  - Robustness
    - deviation in general accuracy of algorithm
  - Stability
    - change in prediction for a target item (before/after attack)
- In addition: rank metrics
  - How often does an item appear in Top-N lists (before/after)
Bandwagon Attack

- Exploits additional information about the community ratings

- Simple idea:
  - Add profiles that contain high ratings for "blockbusters" (in the selected items); use random values for the filler items
  - Will intuitively lead to more neighbors because
    - popular items will have many ratings and
    - rating values are similar to many other user-profiles

- Example: Injecting a profile with high rating values for the *Harry Potter* series

- Low-cost attack
  - Set of top-selling items/blockbusters can be easily determined

- Does not require additional knowledge about mean item ratings
Segment Attack

- Designing an attack that aims to push item A
- Find items that are similar to target item,
  - These items probably liked by the same group of people
  - Identify subset of user community that is interested in items similar to A
- Inject profiles that have
  - high ratings for fantasy novels and
  - random or low ratings for other genres
- Thus, item will be pushed within the relevant community
- For example: Push the new Harry Potter book
  - Attacker will inject profile with positive ratings for other popular fantasy books
  - Harry Potter book will be recommended to typical fantasy book reader
- Additional knowledge (e.g. genre of a book) is required
Special nuke attacks

- **Love/hate attack**
  - Target item is given the minimum value
  - Filler items are given the highest possible rating value
  - Serious effect on system’s recommendations when goal is to nuke an item
  - Other way around (push an item) it is not effective

- **Reverse bandwagon**
  - Associate target item with other items that are disliked by many people.
  - Selected item set is filled with minimum ratings
Effectiveness analysis

- Effect depends mainly on the attack size (number of fake profiles inserted)

- User-based recommenders:
  - Bandwagon / Average Attack:
    - Bias shift of 1.5 points on a 5-point scale at 3% attack size
  - Average Attack slightly better but requires more knowledge
  - 1.5 points shift is significant; 3% attack size means inserting e.g., 30,000 profiles into one-million rating database ...

- Item-based recommenders
  - Far more stable; only 0.15 points prediction shift achieved
  - Exception: Segment attack successful (was designed for item-based method)
  - Hybrid recommenders and other model-based algorithms cannot be easily biased (with the described/known attack models)
Countermeasures

- **Use model-based or hybrid algorithms**
  - More robust against profile injection attacks
  - Accuracy comparable with accuracy of memory-based approaches
  - Less vulnerable

- **Increase profile injection costs**
  - Captchas
    - Low-cost manual insertion ...
Countermeasures II

- Use statistical attack detection methods
  - detect groups of users who collaborate to push/nuke items
  - monitor development of ratings for an item
    - changes in average rating
    - changes in rating entropy
    - time-dependent metrics (bulk ratings)
  - use machine-learning methods to discriminate real from fake profiles
Privacy aspects

- **Problem:**
  - Store and manage sensitive customer information

- **Detailed customer profiles are the basis for market intelligence**
  - Such as segmentation of consumers

- **Ensuring customer privacy**
  - Important for success of a recommender system
  - Users refrain from using the application if privacy leaks get publicly known
Privacy aspects II

- Main architectural assumption of CF-Recommender system is
  - One central server holding the database and
  - the plain (non-encrypted) ratings are stored in this database

- Once an attacker achieved access to that system, all information can be directly used

- Prevent such privacy breaches by
  - Distributing the information or
  - Avoiding the exchange, transfer or central storage of the raw user ratings.
**Data perturbation**

- **Main Idea:** obfuscate ratings by applying random data perturbation
- **Server although does not know the exact values of the customer ratings**
  - Accurate recommendation can still be made because:
    - The range of data is known
  - Computation based on aggregation of obfuscated data sets
- **Tradeoff between degree of obfuscation and accuracy of recommendation**
  - The more "noise" in the data,
    - the better users' privacy is preserved
    - the harder the approximation of real data for the server
Data perturbation II

- Vector of numbers $A = (a_1, ..., a_n)$ provided by client
- Disguise $A$ by adding vector $R = (r_1, ..., r_n)$
- $r_1, ..., r_n$ taken from uniform distribution $[-\alpha, \alpha]$
- Perturbed vector $A' = (a_1 + r_1, ..., a_n + r_n)$ sent to server
- Server does not know original ratings but
  - If range of distribution is known and
  - Enough data are available

**Good estimation can be made of the sum of the vectors:**

$$\sum_{i=1}^{n} (a_i + r_i) = \sum_{i=1}^{n} (a_i) + \sum_{i=1}^{n} (r_i) \approx \sum_{i=1}^{n} (a_1)$$
Distributed collaborative filtering

- Distribute knowledge and avoid storing the information in one central place
- Peer-to-peer (P2P) CF
  - Exchange rating information in a scalable P2P network
  - Active user broadcasts a query (vector of user’s item ratings)
  - Peers calculate similarity between received and other known vectors
    - If similarity > threshold, known ratings returned to requester
    - If not, query forwarded to the neighboring peers
  - Active user calculates prediction with received ratings
Distributed collaborative filtering with obfuscation

- Combines P2P data exchange and data obfuscation
- Instead of broadcasting the "raw" profile only obfuscated version is published
- Peers received this broadcast return a prediction for target item
- Active user
  - collects these answers and
  - calculates a prediction using standard nearest-neighbor-method
- Obfuscation will help to preserve privacy of participants
- Advisable to perturb only profiles of respondent agents
- Obfuscation of requester profile deteriorates recommendation accuracy
Distributed CF with estimated concordance measures

- Picks up tradeoff problem "privacy vs. accuracy"
- Main idea: Do not use standard similarity measure (like Pearson)
- Instead: concordance measure with comparable accuracy to Pearson etc.
  - Given set of items rated by user A and user B. Determine:
  - number of concordant
    - Items on which both users have the same opinion
  - number of discordant
    - Items on which their disagree
  - number of items for which their ratings are tied
    - Same opinion or not rated item
  - Association between A and B computed by Somers' d measure

\[
d_{A,B} = \frac{\text{NbConcordant} - \text{NbDiscordant}}{\text{NbItemRatingsUsed} - \text{NbTied}}
\]
Community-building and aggregates

- Participants of knowledge communities share information
  - inside the community or
  - with outsiders

- Active user can derive predictions from shared information

- Informations are aggregated based on e.g. SVD

- Individual user ratings are not visible to users outside the community

- Use of cryptographic schemes for secure communication between participants in the network
Discussion & summary

- **Research on attacks**
  - Vulnerability of some existing methods shown
  - Specially-designed attack models may also exist for up-to-now rather stable methods
  - Incorporation of more knowledge-sources /hybridization may help

- **Practical aspects**
  - No public information on large-scale real-world attack available
  - Attack sizes are still relatively high
  - More research and industry-collaboration required