Introduction to Collaborative Filtering recommendations

Disease prediction?

João Gama Oliveira
October 9, 2006
Example of Collaborative Filtering recommendations

<table>
<thead>
<tr>
<th>Name</th>
<th>Amélie</th>
<th>Star Wars</th>
<th>Hiver</th>
<th>Batman</th>
<th>Rambo</th>
<th>Whispers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brian</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ellen</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fred</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dean</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jason</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example of Collaborative Filtering recommendations

<table>
<thead>
<tr>
<th>Name</th>
<th>Movie 1</th>
<th>Movie 2</th>
<th>Movie 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brian</td>
<td>Amélie</td>
<td>Star Wars</td>
<td></td>
</tr>
<tr>
<td>Ellen</td>
<td>Amélie</td>
<td>Star Wars</td>
<td>Hiver</td>
</tr>
<tr>
<td>Fred</td>
<td>Star Wars</td>
<td>Batman</td>
<td></td>
</tr>
<tr>
<td>Dean</td>
<td>Star Wars</td>
<td>Batman</td>
<td>Rambo</td>
</tr>
<tr>
<td>Jason</td>
<td>Hiver</td>
<td></td>
<td>Whispers</td>
</tr>
</tbody>
</table>

Karen likes Amélie. What else might she like?
Example of Collaborative Filtering recommendations

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Brian</td>
<td>Amélie</td>
<td>Star Wars</td>
<td></td>
</tr>
<tr>
<td>Ellen</td>
<td>Amélie</td>
<td>Star Wars</td>
<td>Hiver</td>
</tr>
<tr>
<td>Fred</td>
<td></td>
<td>Star Wars</td>
<td>Batman</td>
</tr>
<tr>
<td>Dean</td>
<td></td>
<td>Star Wars</td>
<td>Batman</td>
</tr>
<tr>
<td>Jason</td>
<td></td>
<td>Hiver</td>
<td>Whispers</td>
</tr>
</tbody>
</table>

Karen likes Amélie. What else might she like?

Star Wars... but almost everyone likes Star Wars!
Example of Collaborative Filtering recommendations

<table>
<thead>
<tr>
<th>Name</th>
<th>Movie 1</th>
<th>Movie 2</th>
<th>Movie 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brian</td>
<td>Amélie</td>
<td>Star Wars</td>
<td></td>
</tr>
<tr>
<td>Ellen</td>
<td>Amélie</td>
<td>Star Wars</td>
<td>Hiver</td>
</tr>
<tr>
<td>Fred</td>
<td>Star Wars</td>
<td>Batman</td>
<td></td>
</tr>
<tr>
<td>Dean</td>
<td>Star Wars</td>
<td>Batman</td>
<td>Rambo</td>
</tr>
<tr>
<td>Jason</td>
<td>Hiver</td>
<td>Whispers</td>
<td></td>
</tr>
</tbody>
</table>

Karen likes Amélie. What else might she like?

Star Wars... but almost everyone likes Star Wars!

Hiver would be a good recommendation.
Example of Collaborative Filtering recommendations

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Brian</td>
<td></td>
<td>Amélie</td>
<td>Star Wars</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ellen</td>
<td></td>
<td>Amélie</td>
<td>Star Wars</td>
<td>Hiver</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fred</td>
<td></td>
<td>Star Wars</td>
<td>Batman</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dean</td>
<td></td>
<td>Star Wars</td>
<td>Batman</td>
<td>Rambo</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jason</td>
<td></td>
<td>Hiver</td>
<td>Whispers</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Karen likes Amélie. What else might she like?

Star Wars... but almost everyone likes Star Wars!

Hiver would be a good recommendation

One might even recommend Whispers...
Principle of Collaborative Filtering or Recommender Systems

- Typically these systems do not use any information regarding the actual content of the items (as opposed to content filtering, which is used, e.g., by search engines).

- They are based on usage or preference patterns of other users.

- People who agreed in the past will probably agree again.

- Selection (or filtering) of items is done in a method similar to individuals collaborating to make recommendations for each other.
## Ratings matrix

<table>
<thead>
<tr>
<th></th>
<th>Amélie</th>
<th>Star Wars</th>
<th>Batman</th>
<th>Rambo</th>
<th>Hiver</th>
<th>Whispers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brian</td>
<td>y</td>
<td>y</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ellen</td>
<td>y</td>
<td>y</td>
<td></td>
<td>y</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fred</td>
<td></td>
<td>y</td>
<td>y</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dean</td>
<td></td>
<td>y</td>
<td>y</td>
<td>y</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jason</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>y</td>
<td>y</td>
</tr>
</tbody>
</table>

## Ratings matrix

### Missing data

Missing and sparse data are two inherent factors of CF computation. If the data were complete, there would be no need for CF systems to predict the missing data points.

### Explicit voting:

Refers to a user consciously expressing his or her preference for a title. (E.g. rating score on a numerical scale)

### Implicit voting:

Refers to interpreting user behavior or selections to impute a vote or preference. (E.g. purchase records, item views in an online store)
Collaborative filtering algorithms
Collaborative filtering algorithms

Memory-based algorithms:
- Operate over the entire user-item database to make predictions.
- Statistical techniques are employed to find the *neighbors* of the active user and then combine their preferences to produce a prediction.
- Dynamic structure. More popular and widely used in practice.
Collaborative filtering algorithms

Memory-based algorithms:
- Operate over the entire user-item database to make predictions.
- Statistical techniques are employed to find the neighbors of the active user and then combine their preferences to produce a prediction.
- Dynamic structure. More popular and widely used in practice.

Model-based algorithms:
- Input the user database to estimate or learn a model of user ratings, then run new data through the model to get a predicted output.
- A prediction is computed through the expected value of a user rating, given his/her ratings on other items.
- Static structure. In dynamic domains the model could soon become inaccurate.
Memory-based CF algorithms: Correlation, Vector Similarity

Vote of user $i$ on item $j$ $v_{i,j}$

Mean vote for user $i$ $\bar{v}_i = \frac{1}{|I_i|} \sum_{j \in I_i} v_{i,j}$
Memory-based CF algorithms: Correlation, Vector Similarity

Vote of user $i$ on item $j$ 

\[ v_{i,j} \]

Mean vote for user $i$

\[ \bar{v}_i = \frac{1}{|I_i|} \sum_{j \in I_i} v_{i,j} \]

Predicted vote of the active user

\[ p_{a,j} = \bar{v}_a + \kappa \sum_{i=1}^{n} w(a,i)(v_{i,j} - \bar{v}_i) \]
Memory-based CF algorithms: Correlation, Vector Similarity

Vote of user $i$ on item $j$ \( v_{i,j} \)

Mean vote for user $i$ \( \overline{v}_i = \frac{1}{|I_i|} \sum_{j \in I_i} v_{i,j} \)

Predicted vote of the active user \( p_{a,j} = \overline{v}_a + \kappa \sum_{i=1}^{n} w(a, i)(v_{i,j} - \overline{v}_i) \)

How to find the weights \( w(a,i) \), i.e. the similarity between $a$ and $i$?
Memory-based CF algorithms:
Correlation, Vector Similarity

Vote of user $i$ on item $j$ 

Mean vote for user $i$ 

Predicted vote of the active user 

How to find the weights $w(a,i)$, i.e. the similarity between $a$ and $i$?

- Correlation
- Vector similarity
Memory-based CF algorithms: Correlation, Vector Similarity

Vote of user $i$ on item $j$  
$$v_{i,j}$$

Mean vote for user $i$  
$$\bar{v}_i = \frac{1}{|I_i|} \sum_{j \in I_i} v_{i,j}$$

Predicted vote of the active user  
$$p_{a,j} = \bar{v}_a + \kappa \sum_{i=1}^{n} w(a,i) (v_{i,j} - \bar{v}_i)$$

How to find the weights $w(a,i)$, i.e. the similarity between $a$ and $i$?

- Correlation
- Vector similarity
  - Extensions:
    - Default voting
    - Inverse user frequency
    - Case amplification
Memory-based CF algorithms: Correlation

Correlation

\[ w(a, i) = \frac{\sum_j (v_{a,j} - \bar{v}_a)(v_{i,j} - \bar{v}_i)}{\sqrt{\sum_j (v_{a,j} - \bar{v}_a)^2 \sum_j (v_{i,j} - \bar{v}_i)^2}} = \frac{\langle v_a v_i \rangle - \langle v_a \rangle \langle v_i \rangle}{\sigma_a \sigma_v}, \quad \text{where } j \in I_a \cap I_i \]
Memory-based CF algorithms: Correlation

Correlation

\[ w(a, i) = \frac{\sum_j (v_{a,j} - \bar{v}_a)(v_{i,j} - \bar{v}_i)}{\sqrt{\sum_j (v_{a,j} - \bar{v}_a)^2 \sum_j (v_{i,j} - \bar{v}_i)^2}} = \frac{\langle v_a v_i \rangle - \langle v_a \rangle \langle v_i \rangle}{\sigma_v a \sigma_v i}, \text{ where } j \in I_a \cap I_i \]

Summations are over the items \( j \) for which both users \( a \) and \( i \) have recorded votes.
I. MEMORY-BASED CF ALGORITHMS

Correlation

\[ w(a, i) = \frac{\sum_j (v_{a, j} - \bar{v}_a)(v_{i, j} - \bar{v}_i)}{\sqrt{\sum_j (v_{a, j} - \bar{v}_a)^2 \sum_j (v_{i, j} - \bar{v}_i)^2}} = \frac{\langle v_a v_i \rangle - \langle v_a \rangle \langle v_i \rangle}{\sigma_v a \sigma_v i} \]

where \( j \in I_a \cap I_i \)

Summations are over the items \( j \) for which both users \( a \) and \( i \) have recorded votes.

The weights \( w(a, i) \) generate a \textit{weighted network} where nodes are users and links have weights in the interval \([-1, 1]\), where negative weight corresponds to anti-correlated nodes.
Memory-based CF algorithms: Vector Similarity
Memory-based CF algorithms: Vector Similarity

- Document: Vector of word frequencies
- Similarity between two documents: cosine of the angle formed by the two frequency vectors.
Memory-based CF algorithms: Vector Similarity

- Document: Vector of word frequencies
- Similarity between two documents: cosine of the angle formed by the two frequency vectors.
- Analogously in CF:
  
  \[
  \text{users} \leftrightarrow \text{documents} \mid \text{items} \leftrightarrow \text{words} \mid \text{votes} \leftrightarrow \text{word freq.}
  \]
Memory-based CF algorithms: Vector Similarity

- Document: Vector of word frequencies
- Similarity between two documents: cosine of the angle formed by the two frequency vectors.

- Analogously in CF:
  
  users ↔ documents | items ↔ words | votes ↔ word freq.

- Observed votes indicate a positive preference, there is no role for negative votes, and unobserved items receive a zero vote.

- More appropriate for implicit voting.
Memory-based CF algorithms: Vector Similarity

- Document: Vector of word frequencies
- Similarity between two documents: cosine of the angle formed by the two frequency vectors.
- Analogously in CF:
  
  \[ w(a, i) = \sum_j \frac{v_{a,j}}{\sqrt{\sum_{k \in I_a} v_{a,k}^2}} \frac{v_{i,j}}{\sqrt{\sum_{k \in I_i} v_{i,k}^2}} = \cos(\vec{v}_a, \vec{v}_i) \]

  \[
  \text{users} \leftrightarrow \text{documents} \quad | \quad \text{items} \leftrightarrow \text{words} \quad | \quad \text{votes} \leftrightarrow \text{word freq.}
  \]

- Observed votes indicate a positive preference, there is no role for negative votes, and unobserved items receive a zero vote.
- More appropriate for implicit voting.

Vector similarity
Extensions: Default Voting
Extensions: Default Voting

When relatively few items are evaluated the correlation algorithm won’t do well since it uses only votes on $I_a \cap I_i$. 
Extensions: Default Voting

When relatively few items are evaluated the correlation algorithm won’t do well since it uses only votes on \( I_a \cap I_i \)

Assume some default value as a vote for items for which we do not have explicit votes.

Then we can form the match over the union \( I_a \cup I_i \).
Extensions: Default Voting

When relatively few items are evaluated the correlation algorithm won’t do well since it uses only votes on $I_a \cap I_i$.

Assume some default value as a vote for items for which we do not have explicit votes.

Then we can form the match over the union $I_a \cup I_i$.

In applications with implicit voting, an observed vote is typically an indication of a positive preference (e.g. a visit to a web page is assigned a vote value of ‘1’). In this case an unobserved item can take on the default vote ‘0’.
Extensions:
Inverse User Frequency, Case Amplification

Inverse user frequency

\[ f_j = \log \frac{n}{n_j} \]

Universally liked items are not as useful in capturing similarity as less common items.

Each item is assigned a weight \( f_j \) ... where \( n_j \) is the total number of users that have voted on item \( j \) and \( n \) is the total no. of users.
Extensions:
Inverse User Frequency, Case Amplification

Inverse user frequency
\[ f_j = \log \frac{n}{n_j} \]

Universally liked items are not as useful in capturing similarity as less common items.

Each item is assigned a weight \( f_j \) ... where \( n_j \) is the total number of users that have voted on item \( j \) and \( n \) is the total no. of users.

\[
w(a, i) = \frac{\sum_j f_j v_{a,j} v_{i,j} - (\sum_j f_j v_{a,j})(\sum_j f_j v_{i,j})}{\sqrt{[\sum_j f_j v_{a,j}^2 - (\sum_j f_j v_{a,j})^2][\sum_j f_j v_{i,j}^2 - (\sum_j f_j v_{i,j})^2]}}
\]
Extensions:
Inverse User Frequency, Case Amplification

Inverse user frequency \[ f_j = \log \frac{n}{n_j} \]

Universally liked items are not as useful in capturing similarity as less common items.

Each item is assigned a weight \( f_j \) ... where \( n_j \) is the total number of users that have voted on item \( j \) and \( n \) is the total no. of users.

\[
\begin{align*}
  w(a, i) &= \frac{\sum_j f_j v_{a,j} v_{i,j} - (\sum_j f_j v_{a,j})(\sum_j f_j v_{i,j})}{\sqrt{[\sum_j f_j v_{a,j}^2 - (\sum_j f_j v_{a,j})^2][\sum_j f_j v_{i,j}^2 - (\sum_j f_j v_{i,j})^2]}}
\end{align*}
\]

Case amplification

\[
w'_{a,i} = \begin{cases} 
  w_{a,i}^\rho & \text{if } w_{a,i} \geq 0 \\
  -(-w_{a,i})^\rho & \text{if } w_{a,i} < 0
\end{cases}
\]
Model-based CF algorithms:
Cluster algorithms, Bayesian Networks
Model-based CF algorithms: Cluster algorithms, Bayesian Networks

The prediction for the active user’s vote on item $j$ is obtained by calculating the vote’s expected value, given what we know about him/her.
Model-based CF algorithms: Cluster algorithms, Bayesian Networks

The prediction for the active user’s vote on item $j$ is obtained by calculating the vote’s expected value, given what we know about him/her.

Predicted vote for the active user

$$p_{a,j} = E(v_{a,j}) = \sum_{i=0}^{m} \mathcal{P}(v_{a,j} = i | v_{a,k}, k \in I_a) i$$
Model-based CF algorithms:
Cluster algorithms, Bayesian Networks

The prediction for the active user’s vote on item \( j \) is obtained by calculating the vote’s expected value, given what we know about him/her.

Predicted vote for the active user
\[
p_{a,j} = E(v_{a,j}) = \sum_{i=0}^{m} \mathcal{P}(v_{a,j} = i|v_{a,k}, k \in I_a) i
\]

Once the model is learned, the predictions for the active user only involve his/her previous votes \( v_{a,k} \).

Unlike memory-based algorithms, where each prediction is made by resorting to the database, in model-based algorithms the database is used only to build the model.
### Model-based algorithms: Cluster algorithms

<table>
<thead>
<tr>
<th></th>
<th>Amélie</th>
<th>Star Wars</th>
<th>Batman</th>
<th>Rambo</th>
<th>Hiver</th>
<th>Whispers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brian</td>
<td>y</td>
<td>y</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ellen</td>
<td>y</td>
<td>y</td>
<td></td>
<td></td>
<td></td>
<td>y</td>
</tr>
<tr>
<td>Fred</td>
<td>y</td>
<td>y</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dean</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td>y</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jason</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>y</td>
<td>y</td>
</tr>
</tbody>
</table>
## Model-based algorithms: Cluster algorithms

<table>
<thead>
<tr>
<th></th>
<th>Batman</th>
<th>Rambo</th>
<th>Amélie</th>
<th>Hiver</th>
<th>Whispers</th>
<th>Star Wars</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brian</td>
<td></td>
<td></td>
<td>y</td>
<td></td>
<td></td>
<td>y</td>
</tr>
<tr>
<td>Ellen</td>
<td></td>
<td></td>
<td>y</td>
<td>y</td>
<td></td>
<td>y</td>
</tr>
<tr>
<td>Jason</td>
<td></td>
<td></td>
<td></td>
<td>y</td>
<td>y</td>
<td></td>
</tr>
<tr>
<td>Fred</td>
<td>y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>y</td>
</tr>
<tr>
<td>Dean</td>
<td>y</td>
<td>y</td>
<td></td>
<td></td>
<td></td>
<td>y</td>
</tr>
</tbody>
</table>
## Model-based algorithms: Cluster algorithms

<table>
<thead>
<tr>
<th></th>
<th>Batman</th>
<th>Rambo</th>
<th>Amélie</th>
<th>Hiver</th>
<th>Whispers</th>
<th>Star Wars</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brian</td>
<td></td>
<td></td>
<td>y</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ellen</td>
<td></td>
<td></td>
<td>y</td>
<td>y</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Jason</td>
<td></td>
<td></td>
<td></td>
<td>y</td>
<td>y</td>
<td></td>
</tr>
<tr>
<td>Fred</td>
<td>y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>y</td>
</tr>
<tr>
<td>Dean</td>
<td>y</td>
<td>y</td>
<td></td>
<td></td>
<td></td>
<td>y</td>
</tr>
</tbody>
</table>

### Genre Counts

<table>
<thead>
<tr>
<th>Intellectual</th>
<th>Action</th>
<th>Foreign</th>
<th>Classic</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0/6</td>
<td>5/9</td>
<td>2/3</td>
</tr>
<tr>
<td>Fun</td>
<td>3/4</td>
<td>0/6</td>
<td>2/2</td>
</tr>
</tbody>
</table>
## Model-based algorithms: Cluster algorithms

In this case, for example, the model would infer from the database the **people classes** and the **movies classes**. Furthermore it would extract the probabilities:

\[
P_k = \text{probability that a person is in class } k  \\
P_l = \text{probability that a movie is in class } l  \\
P_{kl} = \text{probability that a person in class } k \text{ is linked to a movie in class } l
\]

<table>
<thead>
<tr>
<th></th>
<th>action</th>
<th>foreign</th>
<th>classic</th>
</tr>
</thead>
<tbody>
<tr>
<td>intellectual</td>
<td>0/6</td>
<td>5/9</td>
<td>2/3</td>
</tr>
<tr>
<td>fun</td>
<td>3/4</td>
<td>0/6</td>
<td>2/2</td>
</tr>
</tbody>
</table>
Model-based algorithms: Bayesian networks
Model-based algorithms: Bayesian networks

A Bayesian Network is a directed acyclic graph of
• nodes representing variables
• arcs representing probabilistic dependency relations among the variables
• local conditional probability distributions for each variable given values of its parents.
Model-based algorithms: Bayesian networks

A Bayesian Network is a directed acyclic graph of:
- nodes representing variables
- arcs representing probabilistic dependency relations among the variables
- local conditional probability distributions for each variable given values of its parents.

→ E.g.: 3 variables: Rain, Sprinkler, WetGrass | 2 states: True, False
Model-based algorithms: Bayesian networks

- In CF each node represents an item.
- The states of each node correspond to the possible vote values.
- By looking at co-rated items the learning algorithm establishes a model for the dependencies between them.

→ E.g.: 3 variables: Movie1, Movie2, Movie3 | 2 states: Like (T), Not Like (F)
Disease predictions

Users ↔ Patients

Items ↔ Diseases

Given the diseases from which a set of patients have suffered...

→ Predict probable future disease

Similarity between Patients can be extracted not only through common Diseases but also through common attributes such as age, gender, race, ...
References


