ML Week

Recommendation

Jeff Abrahamson

23–24 novembre 2016

Definition

Given data about a user, his environment, and some items of interest (*training data*), determine items to recommend.

Definition

Given data about a user, his environment, and some items of interest (*training data*), determine items to recommend.

We don't have to find the max k.

It's enough to find k within some max n.

Examples

- Amazon
- Google News (or Le Monde)
- Facebook
- Medical testing
- App Store / Google Play
- Youtube
- Advertising
- Netflix, last.fm, Spotify, Pandora, ...
- Browser (URL recommendations)
- Search

Client Value Proposition

- Find opportunities
- Reduce choice
- Explore options
- Discover long tails
- Recreation

Provider Value Proposition

- Offer a unique or additional service (beyond competitors)
- Customer trust and loyalty
- Increase sales, CTR, conversions
- Better understand customers

Recommendation

Content-based filtering *(filtrage basée sur le continu)*	More things similar to what I like
Collaborative filtering *(filtrage collaboratif)*	More of what other peo-ple who like what I like like
Knowledge-based filtering *(filtrage basée sur connaissance)*	More of what I need.

Content-based filtering

More things similar to what I like
Plus de ce qui ressemble à ce que j'aime

Advantages

yes! No need for community

yes! Possible to compare items

Disadvantages

no Understand content

yes Cold start problem

no Serendipity

Collaborative filtering

More of what other people who like what I like like
Plus de ce que d'autres qui aiment ce que j'aime aiment

Advantages

yes! No need to understand content

yes! Serendipity

yes! Learn market

Disadvantages

no User feedback

yes Cold start problem (users)

yes Cold start problem (items)

Knowledge-based filtering

More of what I need
Plus de ce qu'il faut

Advantages

yes! Deterministic

yes! Certainty

no! Cold start problem

yes! Market knowledge

Disadvantages

yes Studies to bootstrap

yes Static model, doesn't learn from trends

Utility Matrix

- Users (utilisateurs)
- Items (objets)

Utility Matrix

- Users (utilisateurs)
- Items (objets)

The goal is to fill in the blanks.

	I_1	I_2	I_3	I_4	I_5
U_1	1				
U_2			1	1	1
U_3		1		1	1

Example: books sales at Amazon.

But thousands or millions of columns and rows.

Utility Matrix

- Users (utilisateurs)
- Items (objets)

The goal is to fill in the blanks.

	I_1	I_2	I_3	I_4	I_5
U_1	3				
U_2			5	1	4
U_3		2		5	1

Example: film advice at Netflix.

But thousands or millions of columns and rows.

Utility Matrix

How do we make the matrix?

- Ask users
- Observe users

That's usually expensive...

Item Profiles

Examples:

- Films $\Rightarrow$?
- Books $\Rightarrow$?
- News $\Rightarrow$?
- Images $\Rightarrow$?

Item Profiles

Examples:

- Films $\Rightarrow$?
- Books $\Rightarrow$?
- News $\Rightarrow$?
- Images $\Rightarrow$?

Films :

Content: acters, directors, year (decade, etc.), length

Collaborative: seen, opinion (1–5), when seen relative to release

Item Profiles

Examples:

- Films $\Rightarrow$?
- Books $\Rightarrow$?
- News $\Rightarrow$?
- Images $\Rightarrow$?

Books:

Content : authers, genre, year (decade, etc.), number of pages, content (very difficult)

Collaborative: read, opinion (1–5), how read

Item Profiles

Examples:

- Films $\Rightarrow$?
- Books $\Rightarrow$?
- News $\Rightarrow$?
- Images $\Rightarrow$?

News:

Content : source, section, TF-IDF word vectors

Collaborative:

Item Profiles

Examples:

- Films $\Rightarrow$?
- Books $\Rightarrow$?
- News $\Rightarrow$?
- Images $\Rightarrow$?

Images :

Content:

Collaborative:

Item Profiles

Examples:

- Films $\Rightarrow$?
- Books $\Rightarrow$?
- News $\Rightarrow$?
- Images $\Rightarrow$?

Also: user profile, user behavior

Mathematics

Vectors

Similarity

Similarity : Jaccard Index

or: *Indice de Jaccard, Jaccard similarity coefficient*

Similarity:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

Similarity : Jaccard Index

Similarity:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

Distance:

$$J_\delta(A, B) = 1 - J(A, B)$$

cosine similarity

or: *mesure cosinus, Similarité cosinus*

Similarity:

$$\cos\theta = \frac{A \cdot B}{\| A \| \| B \|}$$

cosine similarity

Similarity:

$$S_C(A, B) = \frac{A \cdot B}{\| A \| \| B \|}$$

cosine similarity

Similarity:

$$S_C(A, B) = \frac{A \cdot B}{\| A \| \| B \|}$$

Distance:

$$D_C(A, B) = 1 - S_C(A, B)$$

Similarity:

$$S_C(A, B) = \frac{A \cdot B}{\| A \| \| B \|}$$

Distance:

$$D_C(A, B) = 1 - S_C(A, B)$$

We only consider non-empty components in the vector.

Texts: TF-IDF

- Vectors of word frequencies
- Frequency $\not\Longrightarrow$ significance

Texts: TF-IDF

- Vectors of word frequencies
- Frequency $\not\Rightarrow$ significance
- Term Frequency - Inverse Document Frequency

Texts: TF-IDF

$$TF_{ij} = \frac{f_{ij}}{\max_k f_{kj}} \qquad\qquad IDF_l = \log_2\left(\frac{N}{n_i}\right)$$

$$TF\text{-}IDF_{ij} = TF_{ij} \cdot IDF_i$$

with :

$$\begin{aligned} f_{ij} &= \text{frequency of word } i \text{ in document } j \\ N &= \text{number of documents} \\ n_i &= \text{number of documents in which we find word } i \end{aligned}$$

Texts: TF-IDF

$$TF_{ij} = \frac{f_{ij}}{\max_k f_{kj}} \qquad\qquad IDF_I = \log_2\left(\frac{N}{n_i}\right)$$

$$TF\text{-}IDF_{ij} = TF_{ij} \cdot IDF_i$$

with :

$$\begin{aligned} f_{ij} &= \text{frequency of word } i \text{ in document } j \\ N &= \text{number of documents} \\ n_i &= \text{number of documents in which we find word } i \end{aligned}$$

IDF is a mesure of how much information a word carries

TF-IDF tells us which words best characterise a document

Texts: TF-IDF

$$TF_{ij} = \frac{f_{ij}}{\max_k f_{kj}} \qquad IDF_I = \log_2\left(\frac{N}{n_i}\right)$$

$$TF\text{-}IDF_{ij} = TF_{ij} \cdot IDF_i$$

with :

$$\begin{aligned} f_{ij} &= \text{frequency of word } i \text{ in document } j \\ N &= \text{number of documents} \\ n_i &= \text{number of documents in which we find word } i \end{aligned}$$

IDF is a mesure of how much information a word carries

TF-IDF tells us which words best characterise a document

Variation: boolean, log, stop word filtering

Content-Based Filtering

	I_1	I_2	I_3	I_4	I_5
U_1	3				
U_2			5	1	4
U_3		2		5	1

More things similar to what I like

Plus de ce qui ressemble à ce que j'aime

Content-Based Filtering

	I_1	I_2	I_3	I_4	I_5
U_1	3				
U_2			5	1	4
U_3		2		5	1

More things similar to what I like

Plus de ce qui ressemble à ce que j'aime

Then, we can cluster (*regroupement*, *partitionnement de données*), etc.

Content-Based Filtering

Based on item profiles

- More stable (in principle)
- $O(n^2)$ (but often less, items often aren't categorised together)
- Can reduce to threshold
- Can pre-calculate, queries become faster

Collaborative Filtering

	I_1	I_2	I_3	I_4	I_5
U_1	3		$\boxed{4}$	$\boxed{2}$	
$\boxed{U_2}$			5	1	4
U_3		2		5	1

More of what other people who like what I like like

Plus de ce que d'autres qui aiment ce que j'aime aiment

Collaborative Filtering

	I_1	I_2	I_3	I_4	I_5
U_1	3				
U_2			5	1	4
U_3		2		5	1

User profile

Collaborative Filtering

	I_1	I_2	I_3	I_4	I_5
U_1	3		4	2	
U_2			5	1	4
U_3		2		5	1

Item profile

Utility Matrix Symmetry

- Propose items based on users
- Propose users based on items

Utility Matrix Symmetry

- Propose items based on users
- Propose users based on items

But remember: 2 items being similar $\not\Leftrightarrow$ 2 users similar.

Utility Matrix Symmetry

- Propose items based on users
- Propose users based on items

But remember: 2 items being similar $\nRightarrow$ 2 users similar.

Thought experiment: consider comparing people vs comparing objects.

Utility Matrix Symmetry

- Propose items based on users
- Propose users based on items

To estimate $m_{u,i}$,

- Find k users like U_u
- Find k items like I_i

Utility Matrix : Estimate $m_{u,i}$

	I_1	I_2	I_3	I_4	I_5
U_1	3		4	2	$\boxed{}$
U_2			5	1	4
U_3		2		2	3

- Find k users like U_u, take $\frac{1}{k}\sum_{j=1}^{k} m_{u_j,i}$
- Find k items like I_i, take $\frac{1}{k}\sum_{j=1}^{k} m_{u,i_j}$

Utility Matrix : Estimate $m_{u,i}$

	I_1	I_2	I_3	I_4	I_5
U_1	3		4	2	$\boxed{}$
U_2			5	1	4
U_3		2		2	3

- Find k users like U_u, take $\frac{1}{k}\sum_{j=1}^{k} m_{u_j,i}$
- Find k items like I_i, take $\frac{1}{k}\sum_{j=1}^{k} m_{u,i_j}$

We have to compute the entire line (or the part which is likely to be important)

Utility Matrix : Estimate $m_{u,i}$

	I_1	I_2	I_3	I_4	I_5
U_1	3		4	2	$\boxed{}$
U_2			5	1	4
U_3		2		2	3

- Find k users like U_u, take $\frac{1}{k}\sum_{j=1}^{k} m_{u_j,i}$
- Find k items like I_i, take $\frac{1}{k}\sum_{j=1}^{k} m_{u,i_j}$

Once we've computed U_u, the other k users lets us take a shortcut.

Utility Matrix : Estimate $m_{u,i}$

	I_1	I_2	I_3	I_4	I_5
U_1	3		4	2	$\boxed{}$
U_2			5	1	4
U_3		2		2	3

- Find k users like U_u, take $\frac{1}{k}\sum_{j=1}^{k} m_{u_j,i}$
- Find k items like I_i, take $\frac{1}{k}\sum_{j=1}^{k} m_{u,i_j}$

For I_i, we have to compute most of the I_j before we can fill in a single line. But item-item filters are often more reliable.

Utility Matrix : Estimate $m_{u,i}$

	I_1	I_2	I_3	I_4	I_5
U_1	3		4	2	
U_2			5	1	4
U_3		2		2	3

- Find k users like U_u, take $\frac{1}{k}\sum_{j=1}^{k} m_{u_j,i}$
- Find k items like I_i, take $\frac{1}{k}\sum_{j=1}^{k} m_{u,i_j}$

In any case, we can mostly precompute in advance.

Utility Matrix

The matrix is sparse.

$\implies$ clustering $\implies$ reduced matrix

Utility Matrix

The matrix is sparse.

$\Longrightarrow$ clustering $\Longrightarrow$ reduced matrix

Estimate on the reduced matrix, then take items and users as representative for the cluster.

Amazon : Item-to-Item Collaborative Filtering

Observations :

Clustering is expensive, reduces quality

Amazon : Item-to-Item Collaborative Filtering

Observations :

Dimension reduction reduces quality

Amazon : Item-to-Item Collaborative Filtering

Observations :

Users interact with very few items

Amazon : Item-to-Item Collaborative Filtering

Observations :

Rapid response desirable

Amazon : Item-to-Item Collaborative Filtering

Scales independent of the number of users or of items

- Online
- Offline

G. Linden, B. Smith, J. York, *Amazon.com Recommendations: Item-to-Item Collaborative Filtering*, Internet Computing (7, 1), 22 Jan 2003.

Offline (Precomputation)

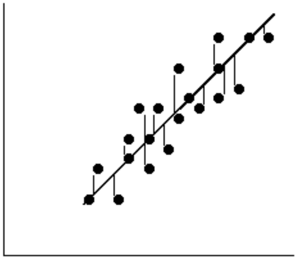
for *each item* I_1 *to sell* **do**
    for *each user C who has purchased* I_1 **do**
        for *each item* I_2 *bought by C* **do**
            (I_1, I_2)++
        end
    end
    for *each item* I_2 **do**
        $S_{I_1, I_2} \leftarrow S(I_1, I_2)$
    end
end

Slope One

Linear regression on user opinions (ratings)

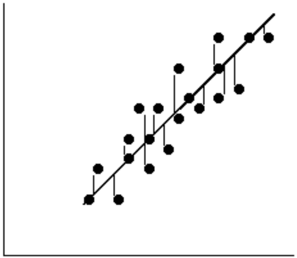
Daniel Lemire and Anna Maclachlan, *Slope One Predictors for Online Rating-Based Collaborative Filtering*,Proceedings of SIAM Data Mining (SDM) 2005.

Slope One : Regression



http://www.upa.pdx.edu/IOA/newsom/pa551/Image255.gif

Slope One : Regression



$$\min \sum (y_i - (ax_i + b))^2$$

http://www.upa.pdx.edu/IOA/newsom/pa551/Image255.gif

Slope One : algorithm

Offline :

for *chaque* I_i, I_j **do**

$\mathcal{U} \leftarrow \{\text{users who have expressed an opinion on } I_i, I_j\}$

$$\text{dev}_{i,j} \leftarrow \frac{1}{\| \mathcal{U} \|} \sum_{u \in \mathcal{U}} (r_u(i) - r_u(j))$$

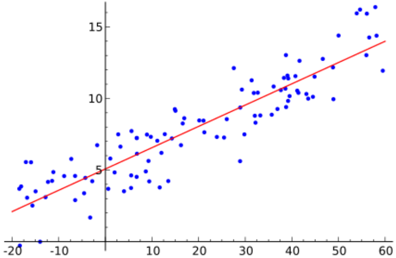
end

Online (for u) :

$\mathcal{V} \leftarrow \{j \mid u \text{ has expressed an opinion on } I_j\}$

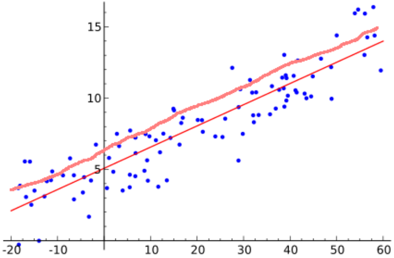
$$r_u(i) \leftarrow \frac{1}{\| \mathcal{V} \|} \sum_{u \in \mathcal{V}} (\text{dev}_{i,j} - r_u(j))$$

Slope One : Regression



"Linear regression" by Sewaqu - Own work. Licensed under Public domain via Wikimedia Commons - `http://commons.wikimedia.org/wiki/File:Linear_regression.svg#mediaviewer/File:Linear_regression.svg`

Slope One : Regression



Dimensionality reduction

SVD, typically $k = 20 \ldots 100$

$$M = U \Sigma V^*$$

Dimensionality reduction

SVD, typically $k = 20 \ldots 100$

$$\begin{pmatrix} a_1 & \cdots & a_m \end{pmatrix} \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix} = \text{scalar}$$

Dimensionality reduction

SVD, typically $k = 20 \ldots 100$

$$\begin{pmatrix} a_1 \\ \vdots \\ a_m \end{pmatrix} \begin{pmatrix} b_1 & \cdots & b_n \end{pmatrix} = \begin{pmatrix} c_{1,1} & \cdots & c_{1,n} \\ \vdots & & \vdots \\ c_{m,1} & \cdots & c_{m,n} \end{pmatrix}$$

Dimensionality reduction

SVD, typically $k = 20 \ldots 100$

$$\begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ \vdots & \vdots & \vdots \\ a_{m,1} & a_{m,2} & a_{m,3} \end{pmatrix} \begin{pmatrix} b_{1,1} & \cdots & b_{1,n} \\ b_{2,1} & \cdots & b_{2,n} \\ b_{3,1} & \cdots & b_{3,n} \end{pmatrix} = \begin{pmatrix} c_{1,1} & \cdots & c_{1,n} \\ \vdots & & \vdots \\ c_{m,1} & \cdots & c_{m,n} \end{pmatrix}$$

Dimensionality reduction

SVD, typically $k = 20 \ldots 100$

$$\begin{pmatrix} a_{1,1} & \cdots & a_{1,k} \\ \vdots & & \vdots \\ a_{m,1} & \cdots & a_{m,k} \end{pmatrix} \begin{pmatrix} c_{1,1} & \cdots & c_{1,n} \\ \vdots & & \vdots \\ c_{k,1} & \cdots & c_{k,n} \end{pmatrix} = \begin{pmatrix} c_{1,1} & \cdots & c_{1,n} \\ \vdots & & \vdots \\ c_{m,1} & \cdots & c_{m,n} \end{pmatrix}$$

Challenges

- How do we measure success?
- What are our features?

Clustering

- kNN
- Curse of Dimensionality
- Scalability

Clustering

- kNN *k*-Nearest Neighbor
- Curse of Dimensionality
- Scalability 10^7 clients, 10^6 objets

Questions?

`purple.com/talk-feedback`