# Recommender Systems

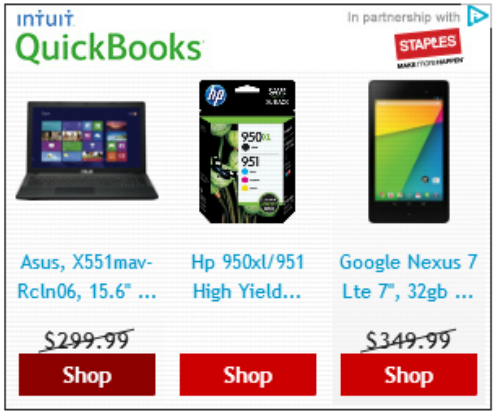
**Amine Benhalloum**

Senior Machine Learning Engineer

# Agenda

- Who is Criteo ?
  - What's our business ?
  - How do recommender systems help us ?
- Recommender Systems are everywhere
- Collaborative Filtering
  - Neighborhood based: User-User, Item-Item
  - Model based: Matrix Factorization
- Content-Based Recommender Systems
- Recommender systems: the good, the bad and the ugly
- Summary


tripadvisor
Planen und vergleichen Sie und buchen Sie Ihre perfekte Reise.
Ab
56 €
Axel Hotel Berlin
Berlin
Hotels anzeigen


intuit
QuickBooks
In partnership with
STAPLES
Asus, X551mav-Rcln06, 15.6" ...
$299.99
Shop
Hp 950xl/951 High Yield...
Shop
Google Nexus 7 Lte 7", 32gb ...
$349.99
Shop


AVATAR
Rund vier Jahre von der Erde entfernt...
» KAUFEN
Star Wars Trilogie...
SATURN
SOO! MUSS TECHNIK


ESPRIT
Piqué-Polo aus 100% Baumwolle
29,99 €
Shop



criteo

# Our business

## We buy

Ad space

## We sell

Clicks

# Our business

## We buy

Ad space

## We sell

Clicks

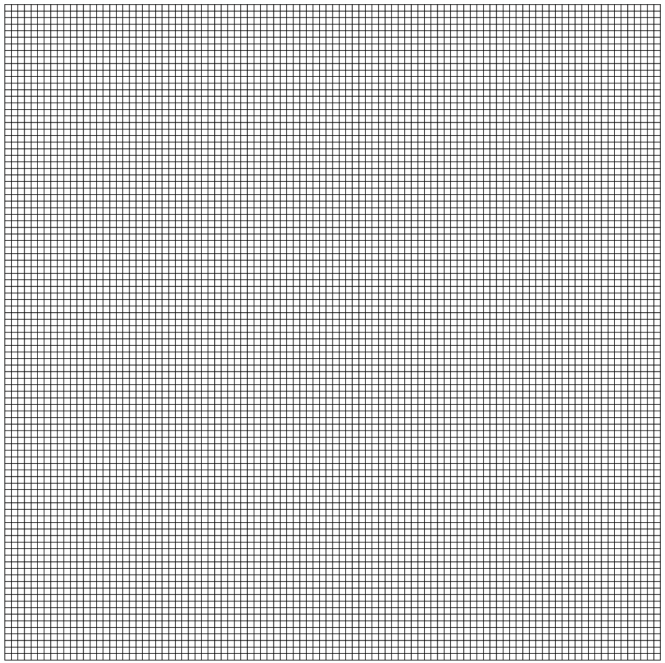
that convert !



criteo

# The scale

10000 Displays





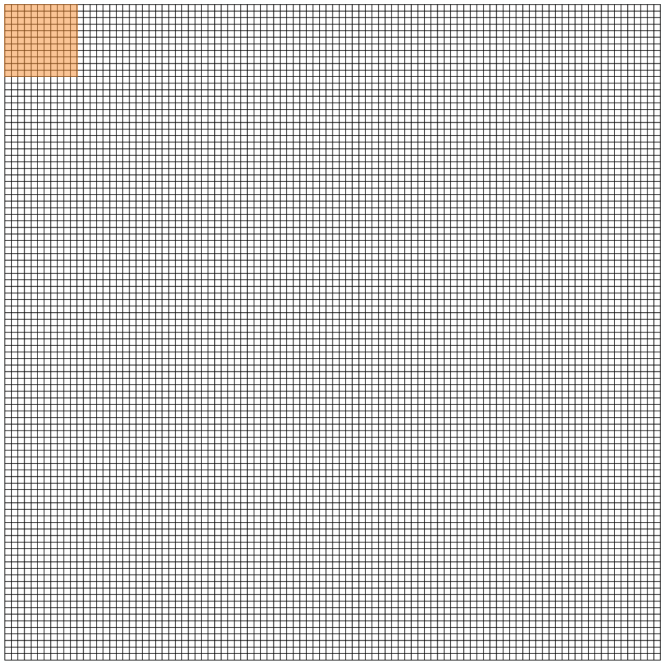
criteo

# The scale

10000 Displays

lead to

100 Clicks





criteo

# The scale

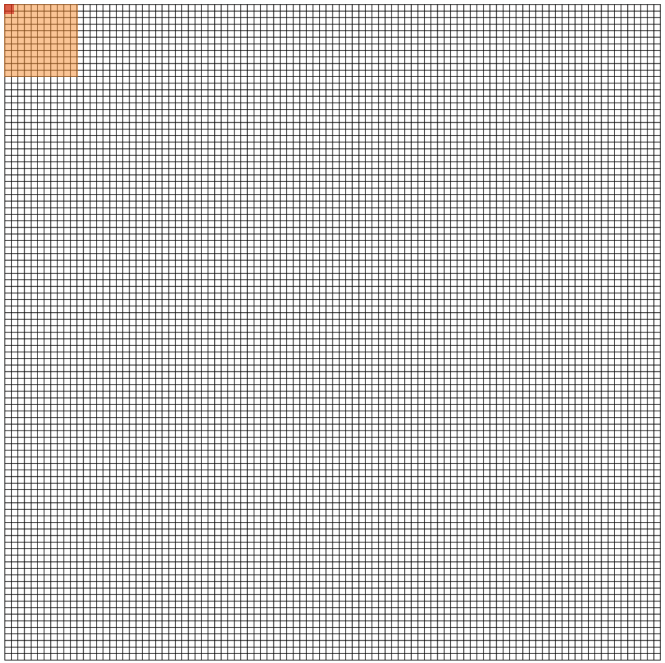
10000 Displays

lead to

100 Clicks

lead to

1 Sale





criteo

# The scale

**166M**

Comments/Day

**500M**

Tweets/Day

**3.5 B**

Searches/Day

**200 B**

Bid Requests/Day

< 10ms

**3.5 B**

Personalized Banners /Day

~ 100ms



criteo

# The scale

**3.5 B**

Personalized banners /day

**4.5 B**

Active catalog size

**~100 ms**

Time to assemble the banner

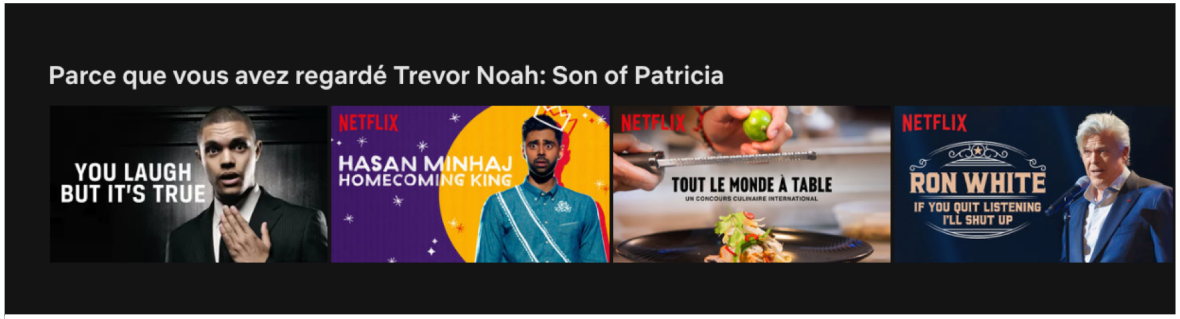
**18, 000**

Advertisers across 98 countries

# Recommender systems: motivation

- Information overload
  - Many choices available
  - « The paradox of choice » (jam experiment, choice overload)
- Recommender system
  - Provide aid
  - Given a user and his « context » and a set of items => selection of items predicted to be « good » for the user

# Recommender systems: they are everywhere







criteo

# Recommender systems: they are everywhere

- Movies (Netflix)
- Video (YouTube)
- Articles (Google news)
- Restaurants (La Fourchette, Tripadvisor)
- Think of a use case !

# Recommender systems: they are valuable

- Netflix: 2 / 3 of the movies watched
- Amazon: 35% of sales
- Google news: recommendations => 38% more clickthrough

# Recommender systems: they are valuable for you !

- It’s one way your data actually serves you and your interests
- Discovering new (and maybe unexpected) items you like
- Leaving the era of search and entering the era of discovery



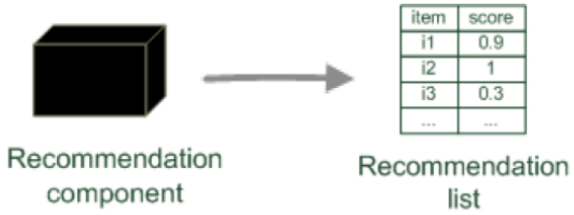
criteo

# Recommender systems: the problem statement

- Estimate a utility function that automatically predicts how a user will like an item or chooses a subset of items that a user will like
- Based on:
  - Past behavior
  - Relations to other ysers
  - Item similarity
  - Context
  - …

# Types of recommender systems

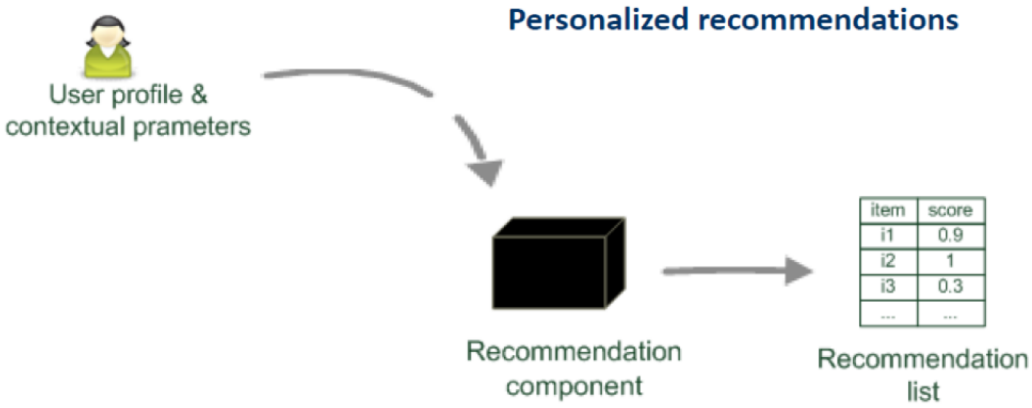
**Recommender systems reduce information overload by estimating relevance**






criteo

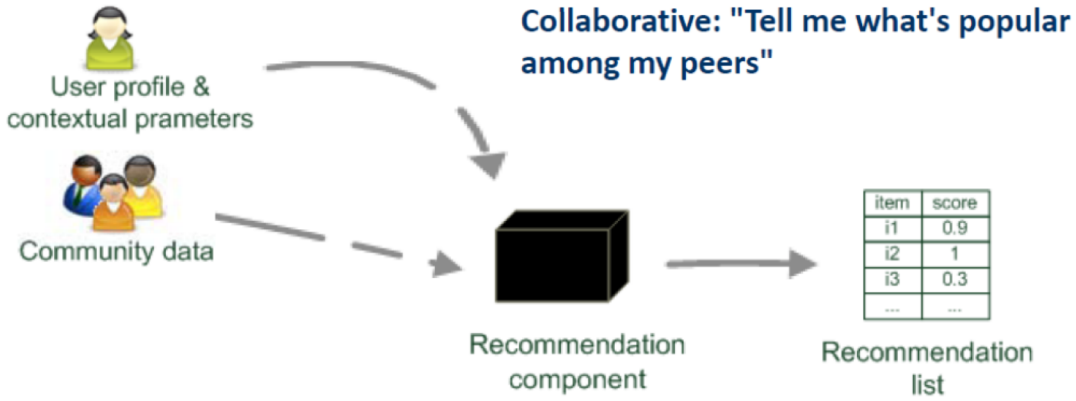
# Types of recommender systems



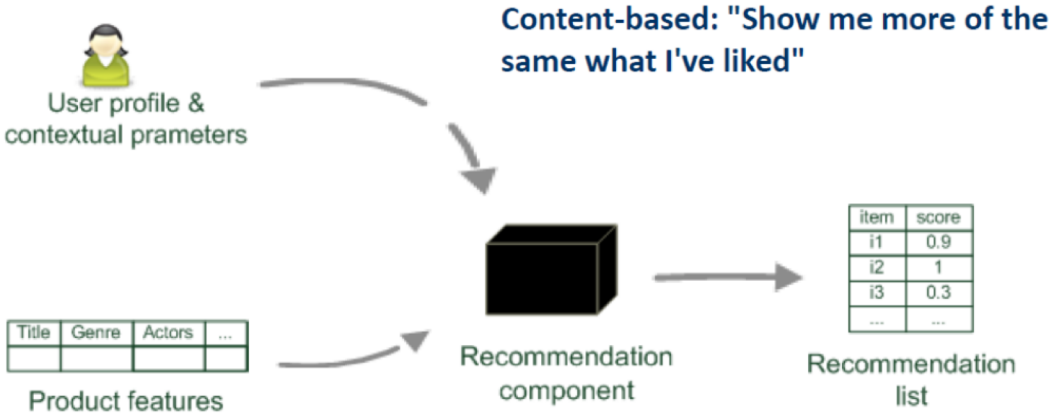



criteo

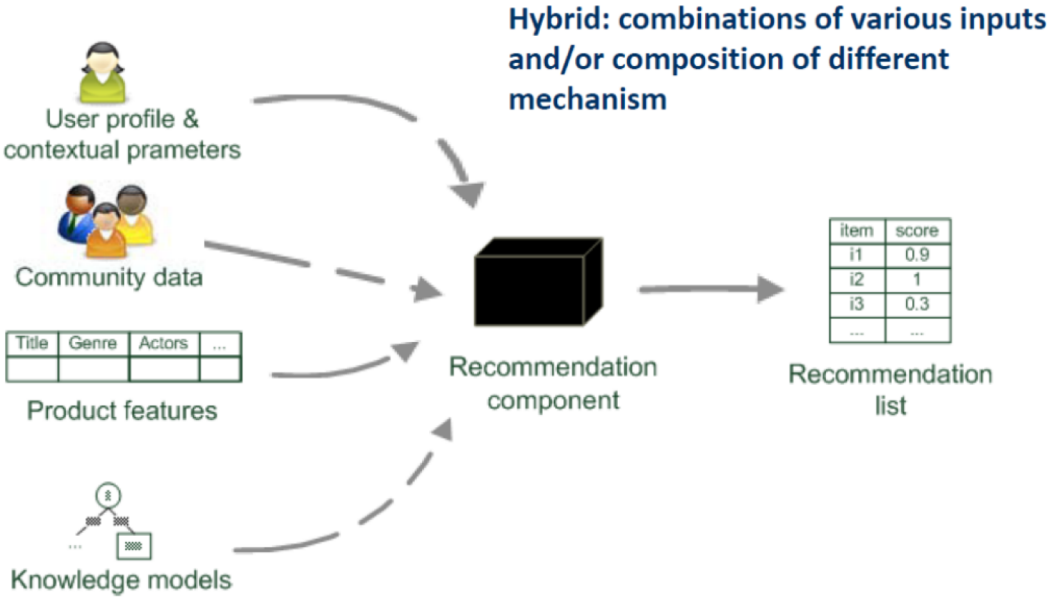
# Types of recommender systems



# Types of recommender systems



# Types of recommender systems






criteo

# Types of recommender systems

- Non personalized (Most viewed items, trending items)
- User profiling (based on age, language, …)
- Collaborative filtering (what people like you liked)
- Content based (items similar to what you liked)
- Or all of the above !

# Brief history

- **1990s** – first systems (e.g. GroupLens), basic algorithms
- **1995-2000** – Rapid commercialization, challenges of scale
- **2000-2005** – Research explosion, mainstream applications
- **2006** – Netflix prize !
- **2007** – First Edition of the RecSys conference
- **2009** – End of the Netflix prize
- **Now** – very active research, many applications

# Netflix prize

- **Netflix:** you all know it, but at that time they were delivering DVDs (and the service still exists under the name dvd.com)
- Contest: 10% improvement of the quality of recommendations
- Prize: **1 Million dollars**
- Data: user ID, movie ID, time, rating (need to predict ratings of test set of user ID, movie ID)
- The contest gave a big boost to Collaborative Filtering algorithms research

# Collaborative Filtering

- *« tell me what's popular among my peers »*
- One of the most often and successfully used techniques
- Widely applicable, does not need a lot of domain knowledge
- **Hypothesis**: Users who shared similar tastes in the past will continue to do so in the futur



criteo

# Collaborative Filtering: Recipe

- **Input**: matrix of user-item feedback or ratings (with missing values of course, this matrix very sparse)

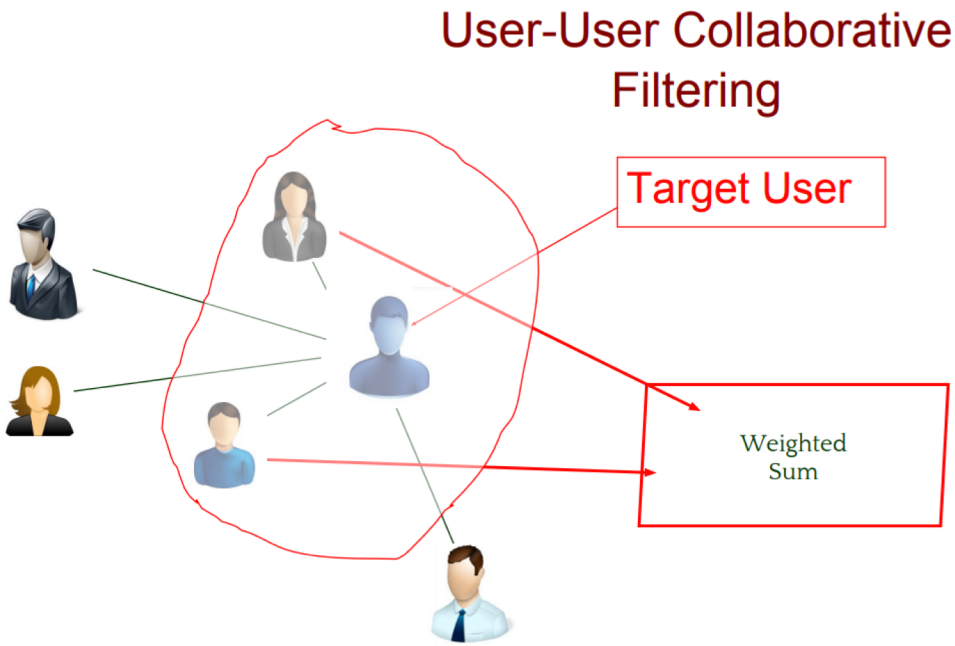
- **Output**: Predictions for missing values

# Collaborative Filtering: Feedback

- Explicit:
  - The user rated explicitely the item (like/dislike, star rating …)
  - It requires effort from the user (friction)
  - Somewhat clear signal of what the user feels about the item
- Implicit:
  - Click/Non Click, buying an item, visiting a page, viewing a video
  - Easier to collect, minimal friction
  - More « honest » (Neftlix example: highly rated vs watched)

Recommended reading: https://www.wired.com/2013/08/qq_netflix-algorithm

# Collaborative Filtering: User-User






criteo

# Collaborative Filtering: User-User

We want to recommend item to Alice

1. Identify the set of ratings that Alice gave
2. Identify the set of users most similar to Alice according to a similarity function (computing neighbours)
3. Identify products these similar users liked (candidate products)
4. Generate predictions – predict the rating that the target user would give to these candidate products
5. Based on these predicted ratings, generate a set of top N recommendations for Alice

# Collaborative Filtering: User-User

First questions:

- How do we measure similarity between users ?
- How many neighbours should we consider ?
- How do we generate predictions from the neighbors ratings ?

| | Item1 | Item2 | Item3 | Item4 | Item5 |
|---|---|---|---|---|---|
| Alice | 5 | 3 | 4 | 4 | ? |
| User1 | 3 | 1 | 2 | 3 | 3 |
| User2 | 4 | 3 | 4 | 3 | 5 |
| User3 | 3 | 3 | 1 | 5 | 4 |
| User4 | 1 | 5 | 5 | 2 | 1 |

# Collaborative Filtering: User-User

- Collection of users $u_i$ , $i \in \{1, 2, \dots, n\}$
- Collection of items $p_j, j \in \{1, 2, \dots, m\}$
- Ratings $r_{ij}$ for user $i$ and item $j$, where $r_{ij} = ?$ If user $i$ didn't rate item $j$
- $ru_i$ average rating given by user $i$
- $rp_j$ average rating given to product $j$

# Collaborative Filtering: User-User

- Similarity between users a, b can be computed through their ratings
- Pearson correlation coefficient

$$corr(a,b) = \frac{\sum_{j \in items} (r_{aj} - ru_a) * (r_{bj} - ru_b)}{\sqrt{\sum_{j \in items} (r_{aj} - ru_a)^2 * \sum_{j \in items} (r_{bj} - ru_b)^2}}$$

- Cosine similarity

$$cos(a,b) = \frac{\sum_{j \in items} r_{aj} * r_{bj}}{\sqrt{\sum_{j \in items} r_{aj}^2 * \sum_{j \in items} r_{bj}^2}}$$

# Collaborative Filtering: User-User

| | Item1 | Item2 | Item3 | Item4 | Item5 |
|---|---|---|---|---|---|
| Alice | 5 | 3 | 4 | 4 | ? |
| User1 | 3 | 1 | 2 | 3 | 3 |
| User2 | 4 | 3 | 4 | 3 | 5 |
| User3 | 3 | 3 | 1 | 5 | 4 |
| User4 | 1 | 5 | 5 | 2 | 1 |

sim = 0,85

sim = 0,00

sim = 0,70

sim = -0,79



criteo

# Collaborative Filtering: User-User

- **Naive approach**: the rating for a new item is just the average rating among neighbours
- Item to rate $i$
- Neighbours set $N = \{\, k\ most\ similar\ users\ to\ user\ a \,\}$

$$r_{predicted}(a, i) = \frac{\sum_{b \in N} r_{bi}}{k}$$

# Collaborative Filtering: User-User

- How to improve ?
- User bias: some users are generous with their ratings, some or not, consider deviation from average rating

$$(r_{bi} - ru_b)$$

- Weight more similar users more

$$r_{predicted}(a,i) = ru_a + \frac{\sum_{b \in N} sim(a,b) * (r_{bi} - ru_b)}{\sum_{b \in N} sim(a,b)}$$

# Collaborative Filtering: User-User

- More improvements:
  - Number of co-rated items
  - Agreement on more « exotic » items more important
  - Neighbour selection

# Collaborative Filtering: Item-Item




criteo

# Collaborative Filtering: Item-Item

Similar to user-user but « reversed »

We want to recommend again an item to Alice

1. Look into the items Alice has rated
2. Compute how similar they are to the target items
    - Similarity of two items only using past ratings from other users
    - Two items are similar if they were rated similarly by the same users, or in the case of implicit feedback, they cooccur often
3. Select k most similar items
4. Predicted rating is the average rating of the similar items

# Collaborative Filtering: Item-Item

| | Item1 | Item2 | Item3 | Item4 | Item5 |
|---|---|---|---|---|---|
| Alice | 5 | 3 | 4 | 4 | ? |
| User1 | 3 | 1 | 2 | 3 | 3 |
| User2 | 4 | 3 | 4 | 3 | 5 |
| User3 | 3 | 3 | 1 | 5 | 4 |
| User4 | 1 | 5 | 5 | 2 | 1 |

# Collaborative Filtering: Item-Item

- Similarity between items a, b can computed through their ratings vectors
- We can use correlation coefficient, cosine similarity, …
- Adjusted cosine similarity
- $U$ = set of users that rated both items a and b

$$\cos_{adjusted}(a,b) = \frac{\sum_{u \in U}(r_{ua} - rp_a) * (r_{ub} - rp_b)}{\sqrt{\sum_{u \in U}(r_{ua} - rp_a)^2 \ * \ \sum_{u \in U}(r_{ub} - rp_b)^2}}$$

# Collaborative Filtering: Item-Item

- Predicted rating is then just the weighted average of the similar items
- We consider typically a small neighborhood
- $R$ = set of items that the user rated

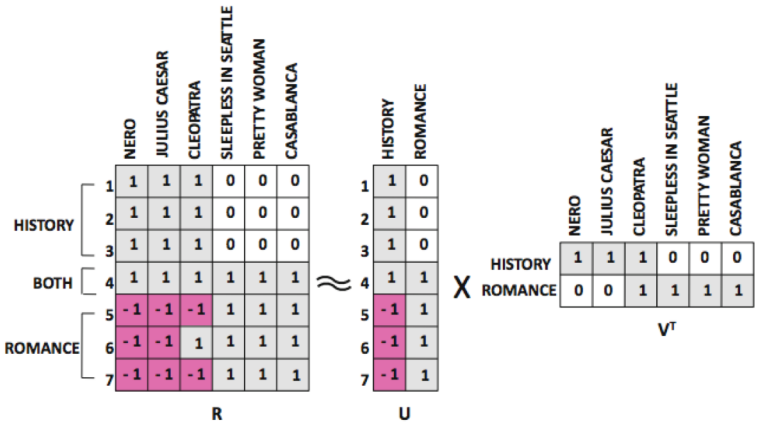
$$r_{predicted}(a,i) = \frac{\sum_{j \in R} sim(i,j) * r_{aj}}{\sum_{j \in R} sim(i,j)}$$

# Collaborative Filtering: Item-Item

- Item-item CF is supposed to be stable (similar items will stay similar, similar users will maybe diverge)
- Original article: *Item-item recommendations by Amazon (2003)*
- Criteo uses item-item CF

# Matrix factorization

- Main idea: latent factors of users/items
  - Some users like action movies, romance, … at different proportions, same can be said about movies
- Use these to predict ratings




(a) Example of rank-2 matrix factorization



criteo

# Matrix factorization

- Model based dimensionality reduction technique (do you know PCA ?)
- One of the main methods: SVD (Singular Value Decomposition)
- There are several decomposition theorems in Linear Algebra (QR, LU …)

# Singular Value Decomposition

- Given a matrix $X$ with m rows and n matrices and a rank r
- SVD is the decomposition of $X$ into $U, V$ orthogonal matrices and $S$ diagonal matrix (singular values)

$$X = USV^T$$

- We call this a low rank approximation of X

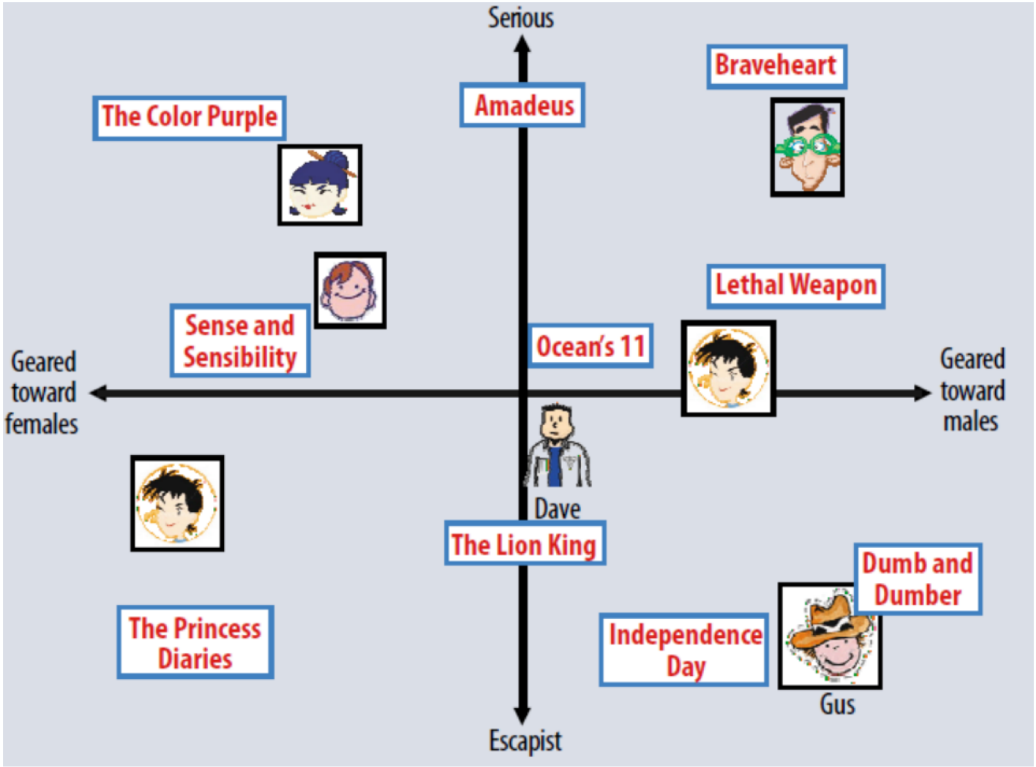
$$\underset{m \times n}{\begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & \\ \vdots & \vdots & \ddots & \\ x_{m1} & & & x_{mn} \end{pmatrix}} = \underset{m \times r}{\begin{pmatrix} u_{11} & \cdots & u_{1r} \\ \vdots & \ddots & \\ u_{m1} & & u_{mr} \end{pmatrix}} \underset{r \times r}{\begin{pmatrix} s_{11} & 0 & \cdots \\ 0 & \ddots & \\ \vdots & & s_{rr} \end{pmatrix}} \underset{r \times n}{\begin{pmatrix} v_{11} & \cdots & v_{1n} \\ \vdots & \ddots & \\ v_{r1} & & v_{rn} \end{pmatrix}}$$

# Singular Value Decomposition interpretation

$$X = USV^T$$

- $X$ ratings matrix ($m$ users and $n$ items)
- $U$ – user-factor strengths (for each user we have a $r$ dimensional vector)
- $V$ – item-factor strengths (for each item we have a $r$ dimensional vector)
- $S$ – importance of factors

# Singular Value Decomposition interpretation

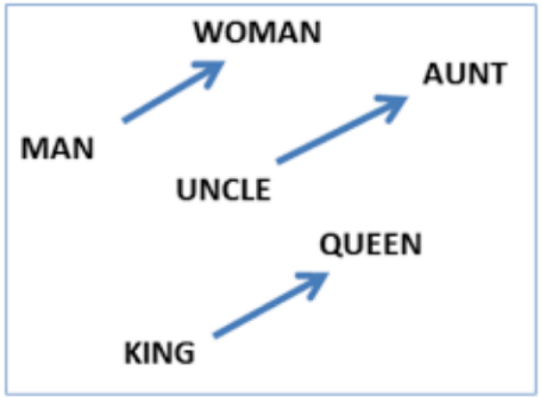


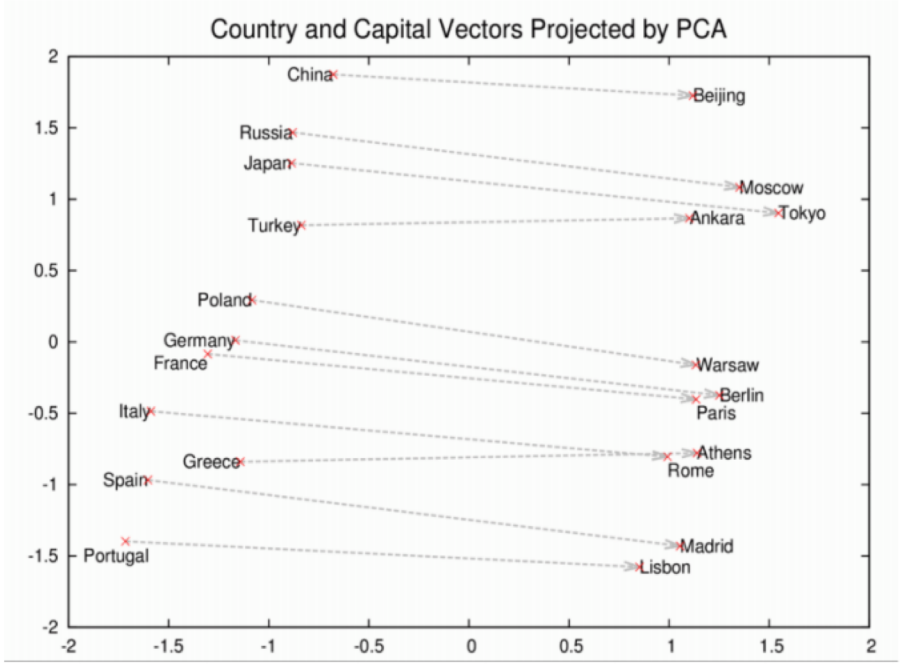



criteo

# Sidenote: embeddings, Word2Vec

- Maybe you saw this at your Textmining course ?






criteo

# Singular Value Decomposition interpretation

- **Problem**: our ratings matrix is sparse
- most (user, item) ratings are missing
- If we replace missing ratings by 0, we will be making the hypothesis that users actually dislike the items they didn't rate
- There are other ways to do matrix factorization: Gradient descent, Alternating least squares …

# Matrix factorization: Some notation

- User $u$ – item $i$
- $r_{ui}$ rating of user u for item i
- $\widehat{r_{ui}}$ predicted rating
- $b, b_u, b_i$ global bias, user bias, item bias
- $q_i, p_u$ item latent factors, user latent factors (r-dimensional vectors)

# Matrix factorization: Baselines

- Naive $\widehat{r_{ui}} = b$
- Biases $\widehat{r_{ui}} = b + b_u + b_i$
- $b_u, b_i$ - biases, average deviations
- Some users/items give/get systematically larger or lower ratings

# Matrix factorization: Latent factors

- Let's forget the biases for a moment and consider only the latent factors

$$\widehat{r_{ui}} = q_i^T p_u$$

- Scalar product between user and item factors to capture interaction
- We need to find $q_i, p_u$ - Learn them from data

# Matrix factorization: Latent factors

- In case of explicit ratings, for instance from 1 to 5
- We want to minimize the « squared errors » between predictions and ground truth

$$min_{q,p} \sum_{(u,i)\in T} \left(r_{ui} - q_i^T p_u\right)^2$$

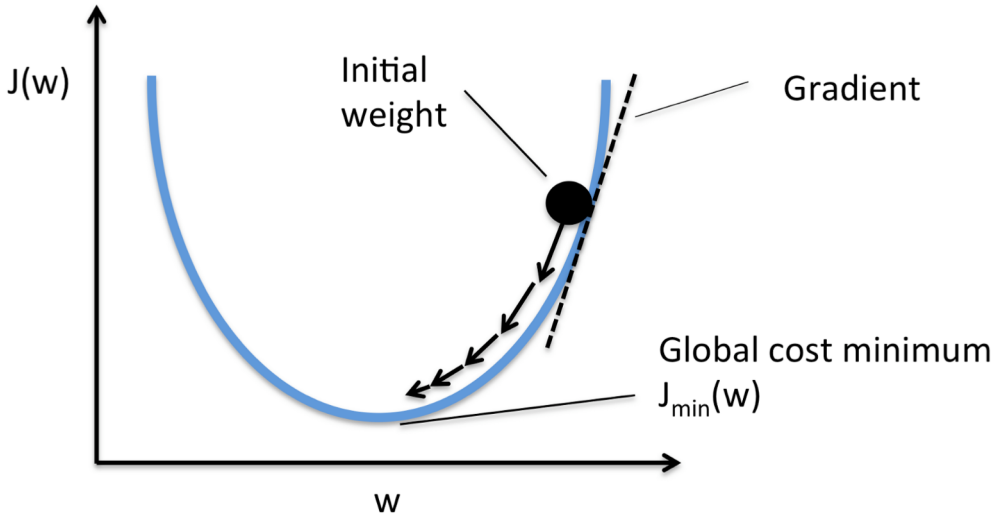
- Some regularization to avoid overfitting

$$min_{q,p} \sum_{(u,i)\in T} \left(r_{ui} - q_i^T p_u\right)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2)$$

- How to find the minimum ?

# Stochastic gradient descent

- We are trying to minimize a function with respect to some pareameters
- Start from some initial point and move in the direction of the gradient






criteo

# Matrix factorization: SGD

- For each $(user\ u, item\ i, rating\ r_{ui})$
  - Update equations:
  - Let's denote $e_{ui} = (r_{ui} - q_i^T p_u)$

$$q_i \coloneqq q_i + \alpha\ (e_{ui} * p_u - \lambda q_i)$$

$$p_u \coloneqq p_u + \alpha\ (e_{ui} * q_i - \lambda p_u)$$

    - These equations are just by taking the partial derivative of the cost function with respect to $p_u$ and $q_i$
    - $\alpha$ learning rate
    - $\lambda$ regularization parameter

# Matrix factorization: Full cost function

- If we add back the biases

$$\widehat{r_{ui}} = q_i^T p_u + b_u + b_i + b$$

- Function to minimize

$$min_{q,p} \sum_{(u,i)\in T} \left(r_{ui} - b - b_u - b_i - q_i^T p_u\right)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2 + b_u^2 + b_i^2)$$

# Matrix factorization: Alternating least squares

- Another way to optimize the cost function
  - Let's group user factors into matrix $P$ (number of user x r)
  - Group item factors into matrix $Q$ (number of items x r)
  - Recall the cost function (without biases)

$$min_{q,p} \sum_{(u,i)\in T} \left(r_{ui} - q_i^T p_u\right)^2$$

  - Assume item factors Q are fixed, solve for $p_u$

$$p_u = \left(Q^T Q + \lambda I\right)^{-1} Q^T r_u$$

  - Then assume user factors P are fixed, solve for Q

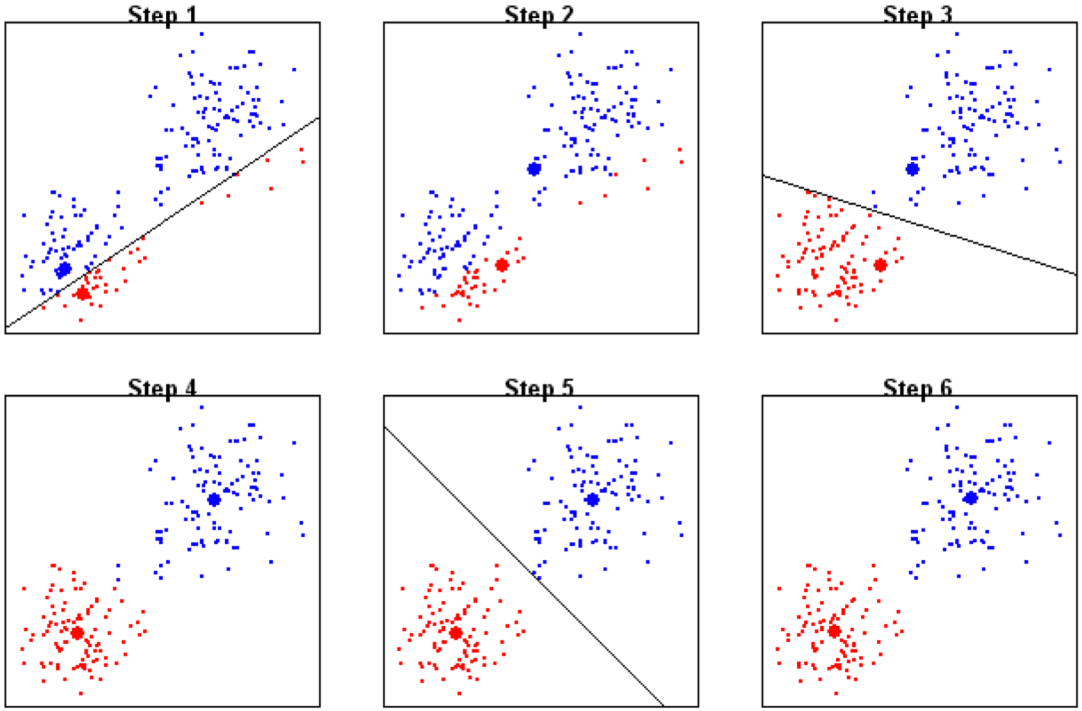
$$q_i = \left(P^T P + \lambda I\right)^{-1} P^T r_i$$

  - Alternate both steps for a few iterations

# Other approaches: Clustering

- Group users into groups by considering what items they interacted with and what were their rating
- We can then recommend for user in a cluster, the items that were highly rated by other users in the cluster (this is very similar to user-user CF but more scalable)

# Other approaches: Clustering – K-means






criteo

# Other approaches: Association rules

- Find groups of items commonly seen or purchased together
- Famous example: beers and diapers (google it !)
- We want to find groups that appear often enough, but not due to chance
- Finding groups efficiently is a challenge
- Most common algorithm: Apriori

# Limitations of Collaborative filtering

- Cold start problems (new items, new users)
- Popularity bias – difficult to recommend items from the long tail
- Noise can have an impact (one account used by different people, hard to disentangle tastes)
- Possibility of attacks

# Collaborative filtering: summary

- Requires only ratings, widely applicable
- Neighborhood methods, latent factors
- Use of machine learning techniques

# Content based recommendation

- Collaborative filtering: “recommend items that similar users liked”
- Content based: “recommend items that are similar to those the user liked in the past”

# Content based recommendation

- We need explicit
  - Item metadata (descriptions, images)
  - And/or user profile (preferences in terms of keywords, …)

# Content based recommendation

- Most content based recommendations techniques were applied to recommending text documents
  - Web pages, news articles
  - Content of items like products or books can be represented as text documents (descriptions, summary, …)
- We can also have structured data as well (categories, price, …)
- At Criteo we use this !

# Content based similarity: keywords

- General similarity approach based on keywords
- two sets of keywords A, B (description of two items or description of item and user)
- how to measure similarity of A and B
- You probably saw this at your text mining course as well

# Content based similarity: keywords

- Sets of keywords A, B
- Dice coefficient

$$\frac{2 * |A \cap B|}{|A| + |B|}$$

- Jaccard coefficient

$$\frac{|A \cap B|}{|A \cup B|}$$

And many others !

# Content based similarity: keywords

- To recommend, just compute content based similarities between user and items or user history and items and look for k nearest neighbours
- Predicting rating is similar to item-item CF case (weighted average)
- Good for modeling short term interest (e.g. follow-up articles)

# Content based similarity: text descriptions

- Examples: product description, recipe instructions, movie plot
- basic approach: bag-of-words representation (words + counts of occurrences)
- limitations?

# Content based similarity: text descriptions

- disadvantages of simple counts:
- importance of words (“course” vs “recommender”)
- length of documents
- **TF-IDF** – standard technique in information retrieval
- **Term Frequency** – how often term appears in a particular document (normalized)
- **Inverse Document Frequency** – how often term appears in all documents

# Content based similarity: text descriptions

- Keyword (term) $t$, document $d$
- $TF(t,d) = \ frequency\ of\ t\ in\ d / maximal\ frequency\ of\ a\ term\ in\ d$
- $IDF(t) = \log(N/n_t)$
- $N$ – number of all documents
- $n_t$ - number of documents containing $t$
- $TFIDF(t,d) = TF(t,d) * IDF(t)$

# Content based similarity: text descriptions

- Improvements
- Common words, stop words (e.g. « a », « the », « on » )
- Stemming ( recommender –> recommend - went –> go )
- Cut-offs ( keep n most informative words )
- Phrases (« United Nations », « New York » )

# Advantages of content based

- user independence – does not depend on other users
- transparency – explanations, understandable
- new items can be easily incorporated (no cold start)

# Limitation of content based

- limited content analysis
- content may not be automatically extractable (multimedia)
- missing domain knowledge
- keywords may not be sufficient
- overspecialization – “more of the same”, too similar items
- new user – ratings or information about user has to be collected

# How does recommendation work at Criteo

- We want to show the user in a banner, products that interest him
- We use almost all of the techniques above
- But we also want to recommend items that have a reasonable probability of being clicked and sold
- So we need to predict this (do you know Logistic Regression ?)

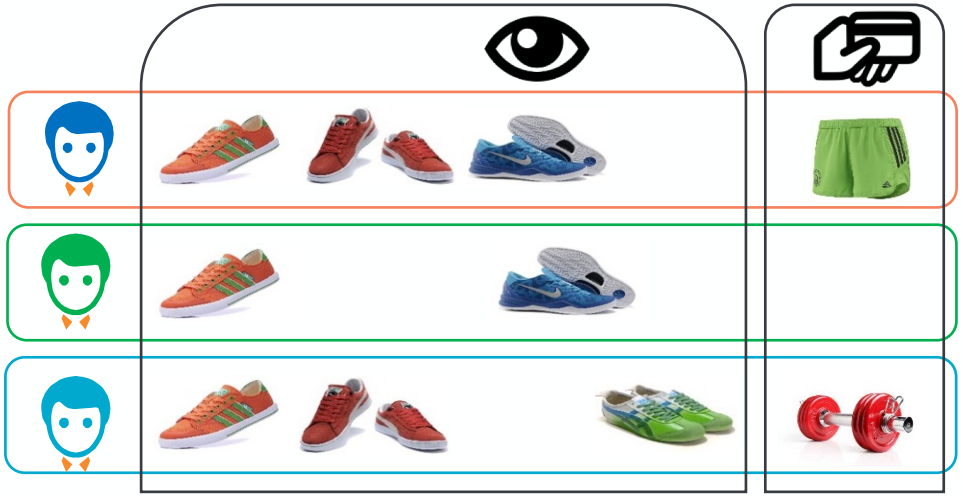
# The recommendation setting

- We are going to do a display for a give partner (fnac, la redoute, …)
- Now we have to choose the products that will appear in the banner

- Main issues:
  - A lot of products to choose from (millions !)
  - Different ways to choose them
  - Response latency < 100 ms

- Offline:
  - Compute different kind of candidate sources (most viewed, most bought, similarities,…)
- Online:
  - Get product candidates from sources
  - Rank the candidates
  - Pick the best ones (or sample ;) )

# Offline world






criteo

# Online world

saw orange shoes  we can then choose from


Historical




Most viewed




Similar to




Complementary to




criteo

# Online world

Similarities

Most viewed

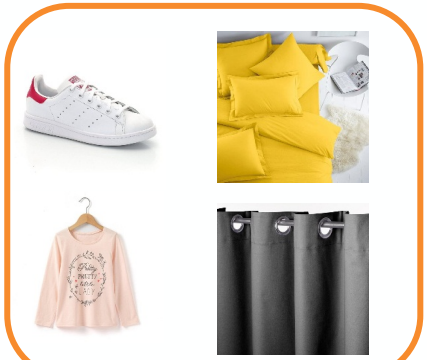
Most bought



criteo

# Online world

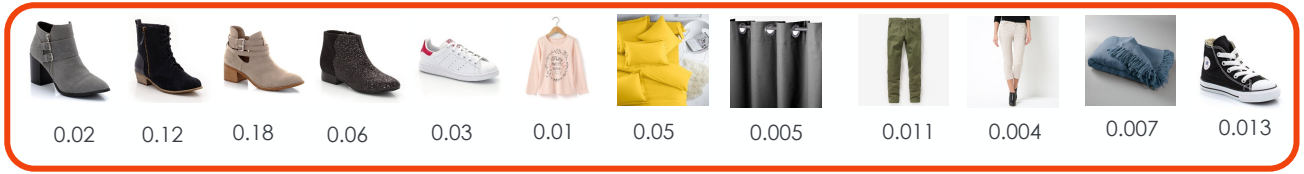
Similarities



Most viewed

Most bought

Compute Click Predictions

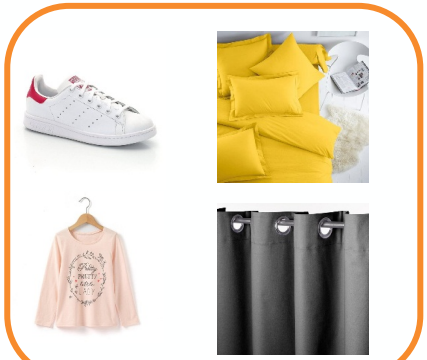





criteo

# Online world

Similarities



Most viewed

Most bought

Choose the winners to compose the banner




criteo

# Online world


Choose the winners to compose the banner




criteo

# Recommendation systems: The good, the bad and the ugly

# Recommendation systems: The good

- Helps you navigate through the abundance of content
- Save time, money and energy
- Active area of research, improves continuously
- Is a necessary feature for any e-commerce website

# Recommendation systems: The bad

- Still hard to do (right) at scale, with millions or billions of items and millions of users
- A working solution is a hybrid of several components, can be a complex system
- From a user perspective, sometimes hard to interpret recommendations, or worst recommendations can become too obvious (Harry Potter 1 -> Harry Potter 2)

# Recommendation systems: The ugly

- Only seing what you want to see – the bubble effect
- Hard to achieve serendipity i.e. finding surprising items by chance of coincidence that you like
- If done wrong, popular items will only get more popular, unpopular items will stay unpopular
- Biases everywhere

# Summary

- Several approaches to Recommender systems
  - Collaborative filtering
  - Neighborhood based: User-User or Item-Item
  - Matrix factorization: discovering latent factors
  - Content based: extract features from item metadata
- Still an a very active area of research !
- Think how you can apply it to your field

# Thank you, Questions ?

ma.benhalloum@criteo.com

linkedin.com/in/aminebenh/