The Recommender Problem *Revisited*

Xavier Amatriain
Research/Engineering Director @ Netflix

Bamshad Mobasher
Professor @ DePaul University

NETFLIX

Index

PART I - Xavier Amatriain (2h)

PART 2 - Bamshad Mobasher (1h)

Recent/Upcoming Publications

- The Recommender Problem Revisited. KDD and Recsys 2014 Tutorial
- KDD: Big & Personal: data and models behind Netflix recommendations. 2013
- SIGKDD Explorations: Mining large streams of user data for personalized recommendations. 2012
- Recsys: Building industrial-scale real-world recommender systems. 2012
- Recys - Walk the Talk: Analyzing the relation between implicit and explicit feedback for preference elicitation. 2011
- SIGIR – Temporal behavior of CF. 2010
- Web Intelligence – Expert-based CF for music. 2010
- Recsys – Tensor Factorization. 2010
- Mobile HCI – Tourist Recommendation. 2010
- Recsys Handbook (book) – Data mining for recsys. 2010 & Recommender Systems in Industry. 2014
- SIGIR – Wisdom of the Few. 2009
- Recsys – Denoising by re-rating. 2009
- CARS – Implicit context-aware recommendations. 2009
- UMAP – I like it I like it not. 2009

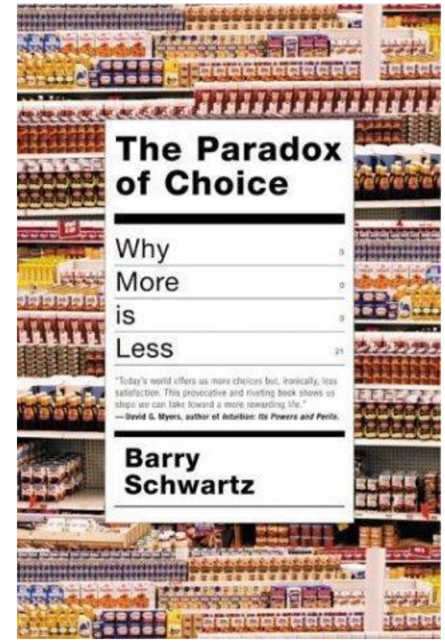
Index

1. The Recommender Problem

The Age of Search has come to an end

- **... long live the Age of Recommendation!**
- **Chris Anderson in “The Long Tail”**
  - *“We are leaving the age of information and entering the age of recommendation”*
- **CNN Money, “The race to create a 'smart' Google”:**
  - *“The Web, they say, is leaving the era of search and entering one of discovery. What's the difference? Search is what you do when you're looking for something. Discovery is when something wonderful that you didn't know existed, or didn't know how to ask for, finds you.”*

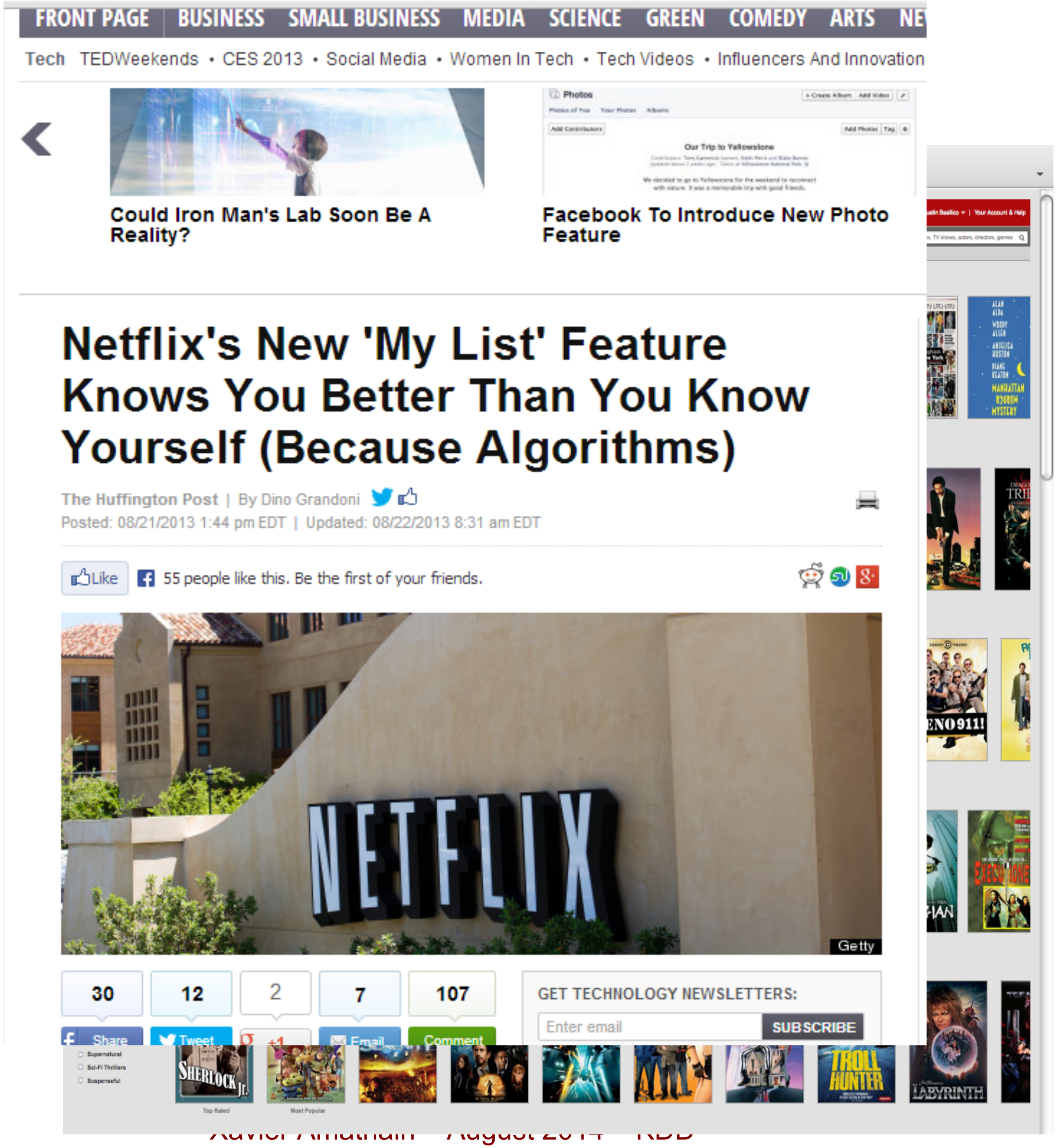
Information overload



"People read around 10 MB worth of material a day, hear 400 MB a day, and see 1 MB of information every second" - The Economist, November 2006

In 2015, consumption will raise to 74 GB a day - UCSD Study 2014

Everything is personalized



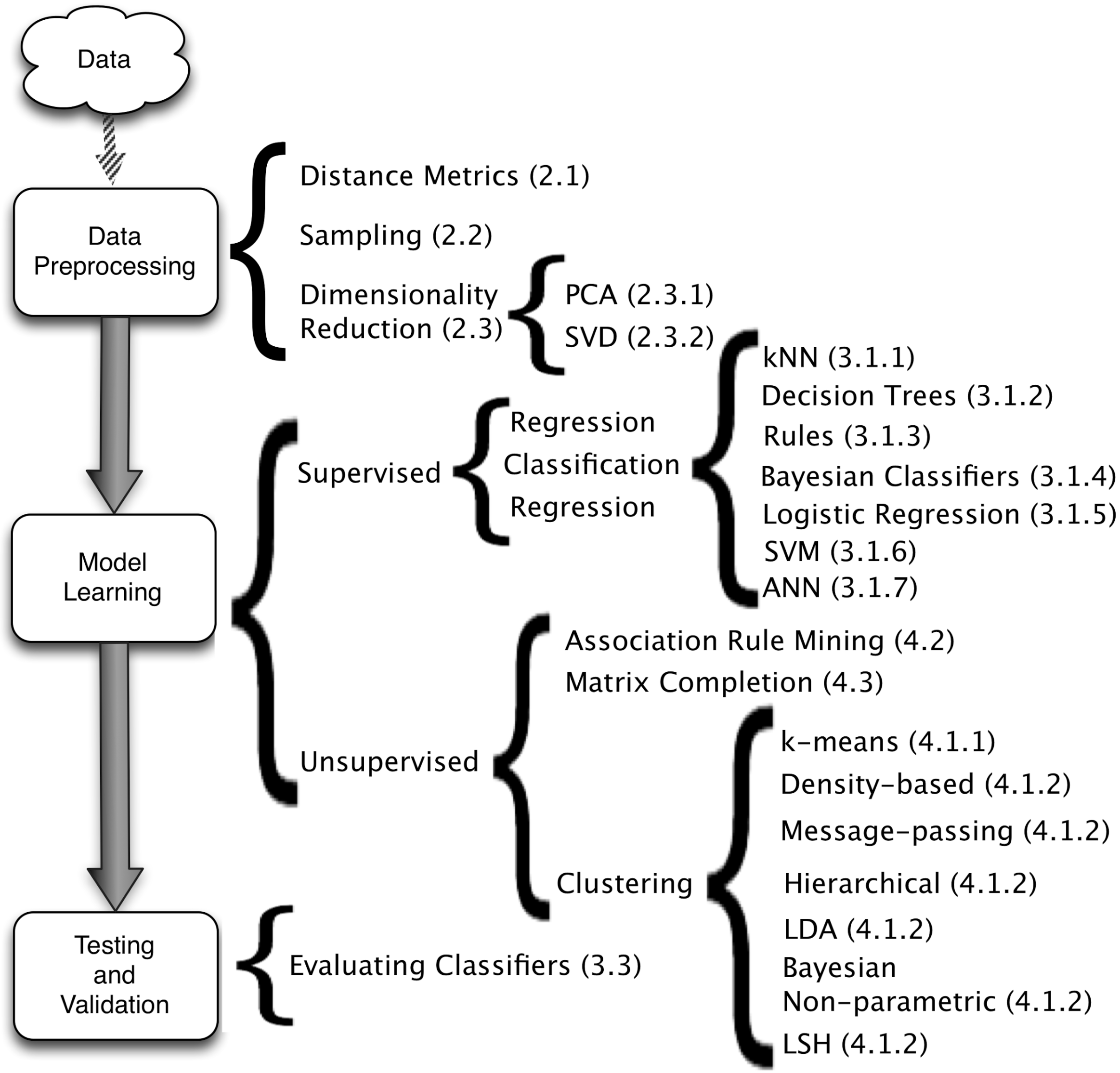
The “Recommender problem”

- *Traditional* definition: Estimate a utility function that automatically predicts how a user will like an item.
- Based on:
  - Past behavior
  - Relations to other users
  - Item similarity
  - Context
  - …

Recommendation as data mining

The core of the Recommendation Engine can be assimilated to a general data mining problem

(Amatriain et al. Data Mining Methods for Recommender Systems in Recommender Systems Handbook)



NETFLIX

Machine Learning + all those other things

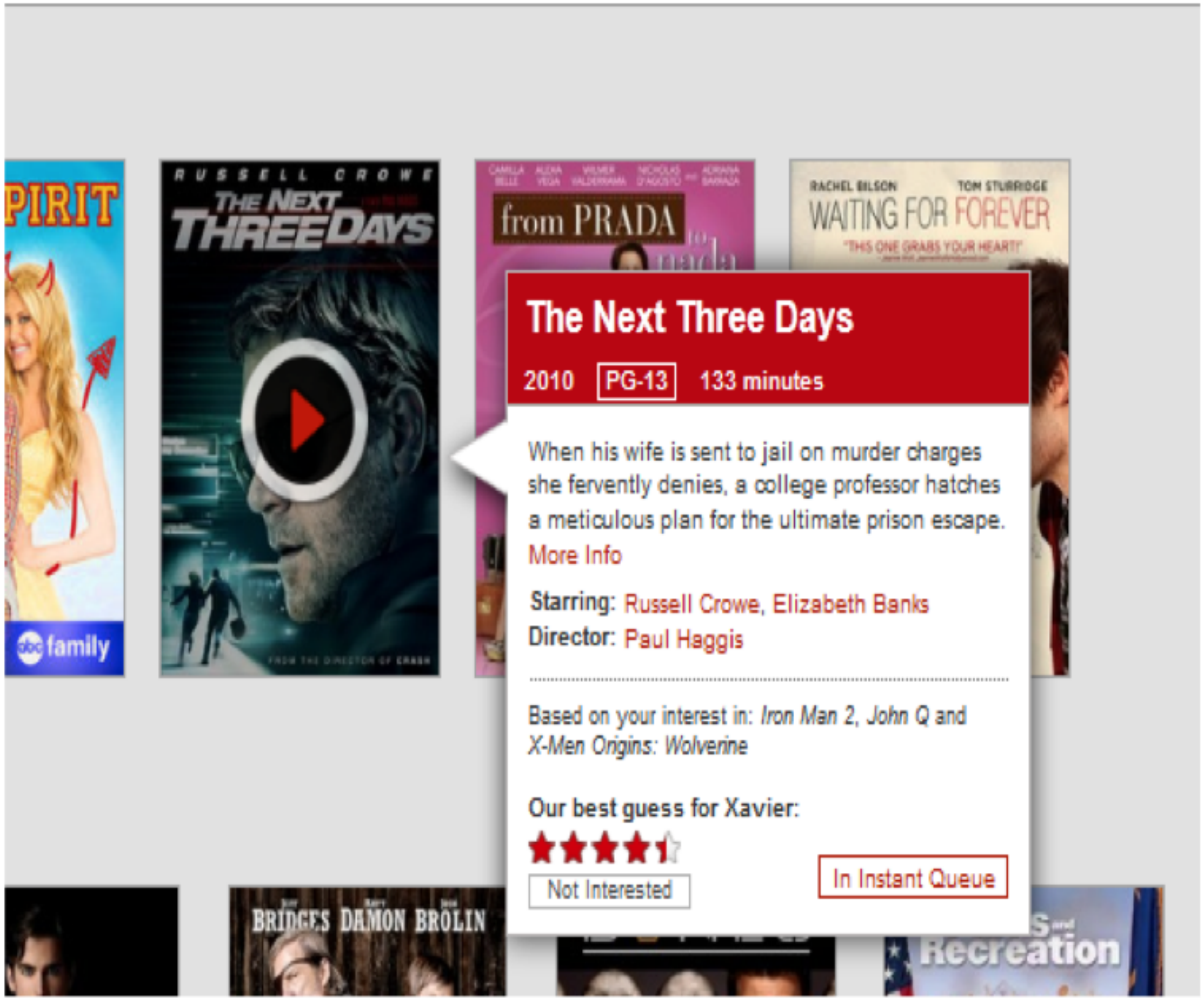
- User Interface
- System requirements (efficiency, scalability, privacy....)
- Serendipity
- Diversity
- Awareness
- Explanations
- …

NETFLIX

Serendipity

- Unsought finding
- Don't recommend items the user already knows or **would have found anyway**.
- Expand the user's taste into neighboring areas by improving the obvious
- Collaborative filtering can offer controllable serendipity (e.g. controlling how many neighbors to use in the recommendation)

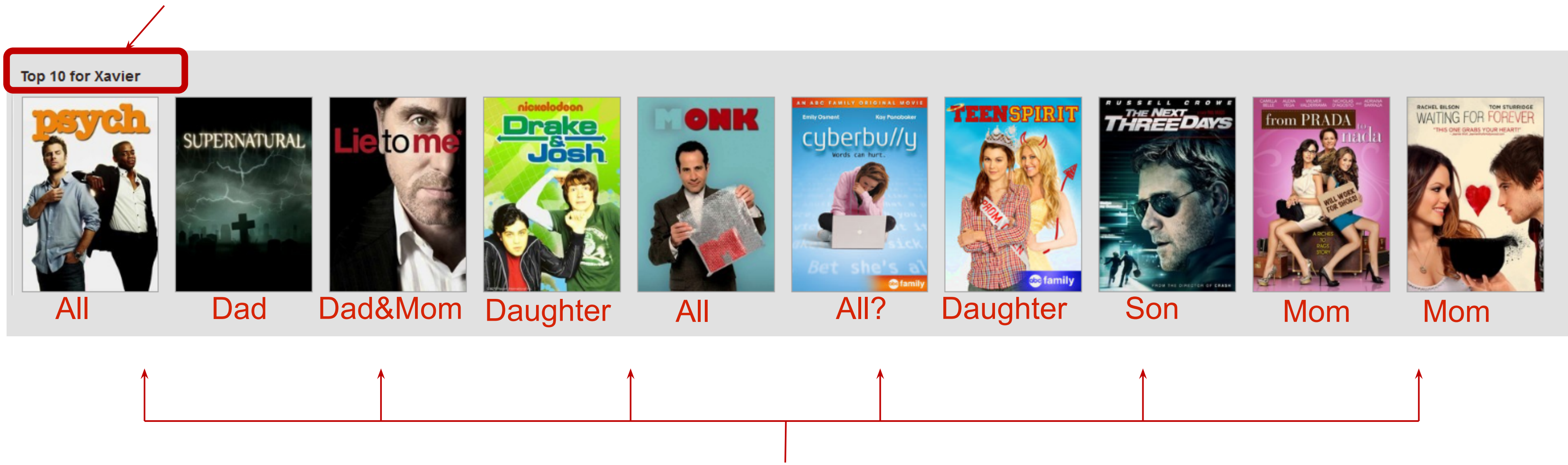
Explanation/Support for Recommendations



Social Support

Diversity & Awareness

Personalization awareness



Diversity

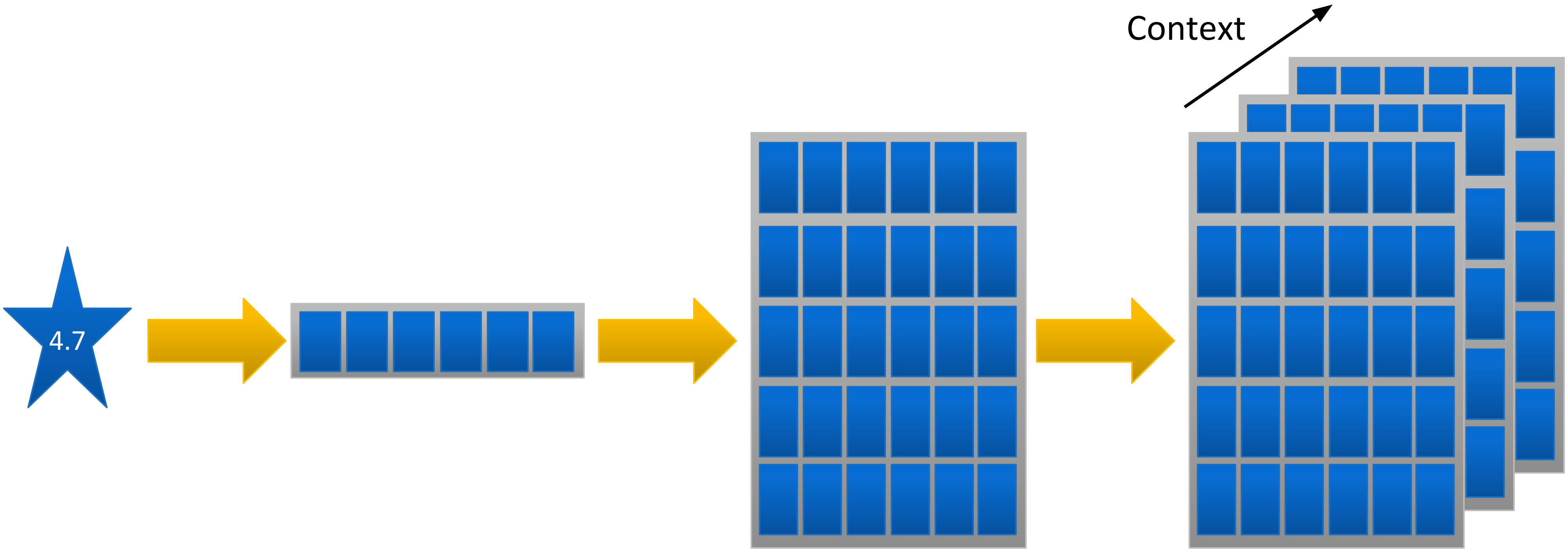
NETFLIX

What works

- Depends on the **domain** and particular **problem**
- However, in the general case it has been demonstrated that the best isolated approach is CF.
  - Other approaches can be hybridized to improve results in specific cases (cold-start problem...)
- What matters:
  - Data preprocessing: outlier removal, denoising, removal of global effects (e.g. individual user's average)
  - “Smart” dimensionality reduction using MF/SVD
  - Combining methods through ensembles

NETFLIX

Evolution of the Recommender Problem



Rating

Ranking

Page Optimization

Context-aware Recommendations

NETFLIX

Index

2. Traditional Approaches

2.1. Collaborative Filtering

The CF Ingredients

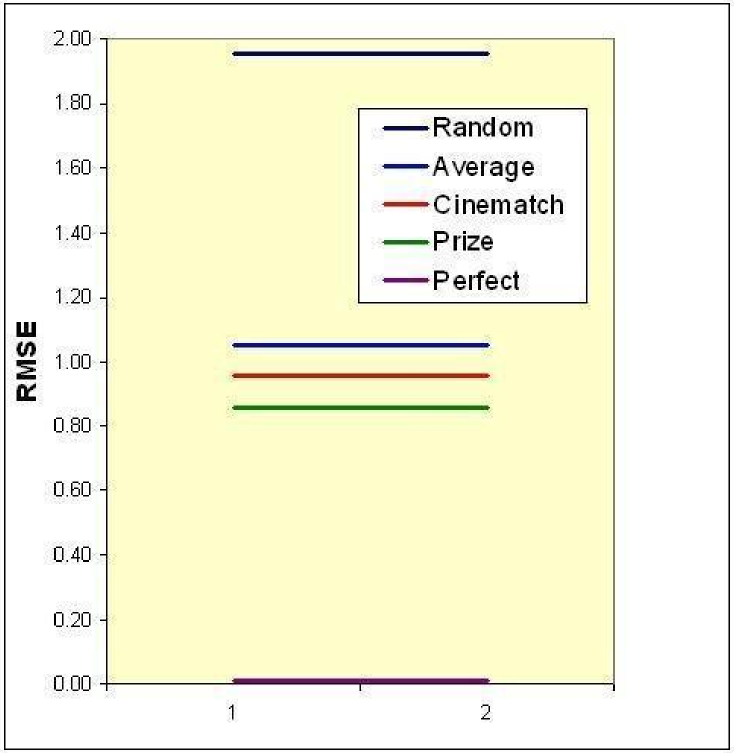
- List of **m Users** and a list of **n Items**
- Each user has a **list of items** with associated **opinion**
  - **Explicit opinion** - a rating score
  - Sometime the rating is **implicitly** – purchase records or listen to tracks
- **Active user** for whom the CF prediction task is performed
- **Metric** for measuring **similarity between users**
- Method for selecting a subset of **neighbors**
- Method for **predicting a rating** for items not currently rated by the active user.

Personalized vs Non-Personalised CF

- CF recommendations are **personalized:** prediction based only on **similar users**
- **Non-personalized** collaborative-based recommendation: averagge the recommendations of **ALL** the users
- How would the two approaches compare?

Personalized vs. Not Personalized

- Netflix Prize: it is very simple to produce "reasonable" recommendations and extremely difficult to improve them to become "great"
- But there is a huge difference in business value between reasonable and great



Data Set	users	items	total	density	MAE Non Pers	MAE Pers
Jester	48483	100	3519449	0,725	0,220	0,152
MovieLens	6040	3952	1000209	0,041	0,233	0,179
EachMovie	74424	1649	2811718	0,022	0,223	0,151

NETFLIX

User-based Collaborative Filtering

Collaborative Filtering

The basic steps:

1. Identify set of ratings for the **target/active user**
2. Identify set of users most similar to the target/active user according to a similarity function (**neighborhood** formation)
3. Identify the products these similar users liked
4. **Generate a prediction** - rating that would be given by the target user to the product - for each one of these products
5. Based on this predicted rating recommend a set of top N products

UB Collaborative Filtering

- A collection of user u_i, $i=1, \ldots n$ and a collection of products p_j, $j=1, \ldots, m$
- An $n \times m$ matrix of ratings v_{ij} , with $v_{ij} = ?$ if user i did not rate product j
- Prediction for user i and product j is computed

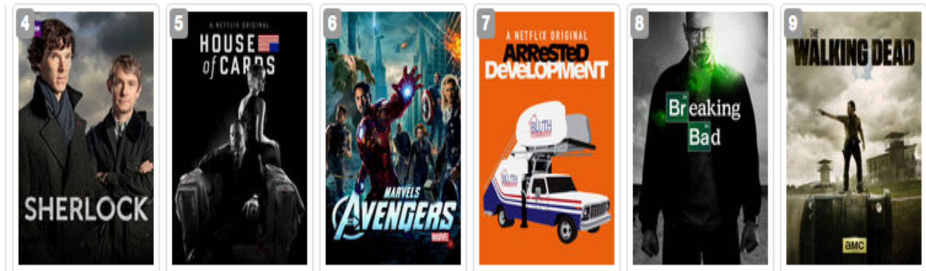
$$v_{ij}^{*} = K \sum_{v_{kj} \neq ?} u_{jk} v_{kj} \quad \text{or} \quad v_{ij}^{*} = v_i + K \sum_{v_{kj} \neq ?} u_{jk} (v_{kj} - v_k)$$

- Similarity can be computed by Pearson correlation

$$u_{ik} = \frac{\sum_j (v_{ij} - v_i)(v_{kj} - v_k)}{\sqrt{\sum_j (v_{ij} - v_i)^2 \sum_j (v_{kj} - v_k)^2}} \quad \text{or} \quad \cos(u_i, u_j) = \frac{\sum_{k=1}^{m} v_{ik} v_{jk}}{\sqrt{\sum_{k=1}^{m} v_{ik}^2 \sum_{k=1}^{m} v_{jk}^2}}$$

NETFLIX

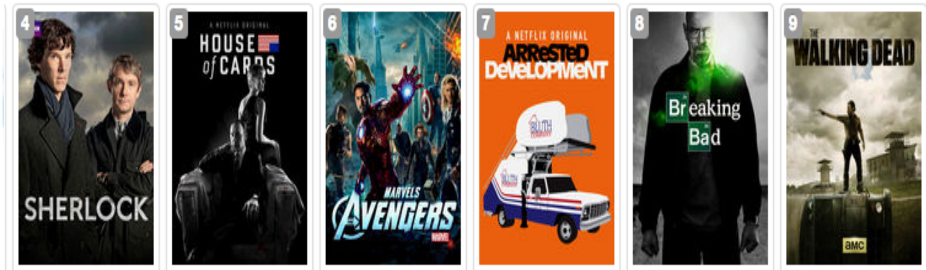
User-based CF Example



						sim(u,v)
2		2	4	5		NA
5		4			1	
		5		2		
	1		5		4	
		4			2	
4	5		1			NA

NETFLIX

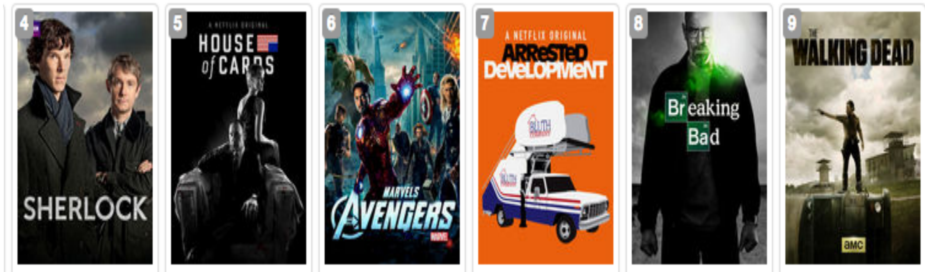
User-based CF Example



						sim(u,v)
2		2	4	5		NA
5		4			1	0.87
		5		2		
	1		5		4	
		4			2	
4	5		1			NA

NETFLIX

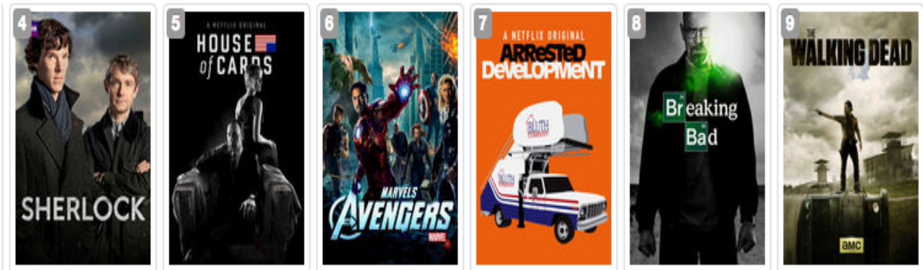
User-based CF Example



						sim(u,v)
2		2	4	5		NA
5		4			1	0.87
		5		2		1
	1		5		4	
		4			2	
4	5		1			NA

NETFLIX

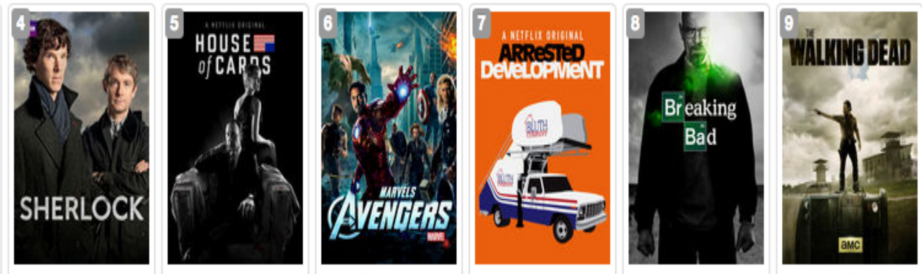
User-based CF Example



						sim(u,v)
2		2	4	5		NA
5		4			1	0.87
		5		2		1
	1		5		4	-1
		4			2	
4	5		1			NA

NETFLIX

User-based CF Example



						sim(u,v)
2		2	4	5		NA
5		4			1	0.87
		5		2		1
	1		5		4	-1
3.51*	3.81*	4	2.42*	2.48*	2	
4	5		1			NA

NETFLIX

Item-based Collaborative Filtering

Item Based CF Algorithm

- Look into the items the target user has rated
- Compute how similar they are to the target item
  - Similarity **only using** past **ratings** from other users!
- Select k most similar items.
- Compute Prediction by taking weighted average on the target user's ratings on the most similar items.

NETFLIX

Item Similarity Computation

- Similarity: find users who have rated items and apply a similarity function to their ratings.
- Cosine-based Similarity (difference in rating scale between users is not taken into account)

$$S(i,j)=\cos(\vec{i},\vec{j})=\frac{\vec{i}\cdot\vec{j}}{\|\vec{i}\|_2 * \|\vec{j}\|_2}$$

- Adjusted Cosine Similarity (takes care of difference in rating scale)

$$S(i,j)=\frac{\sum_{u\in U}(R_{u,i}-\bar{R}_u)(R_{u,j}-\bar{R}_u)}{\sqrt{\sum_{u\in U}(R_{u,i}-\bar{R}_u)^2}\sqrt{\sum_{u\in U}(R_{u,j}-\bar{R}_u)^2}}$$

NETFLIX

Challenges of memory-based CF

CF: Pros/Cons

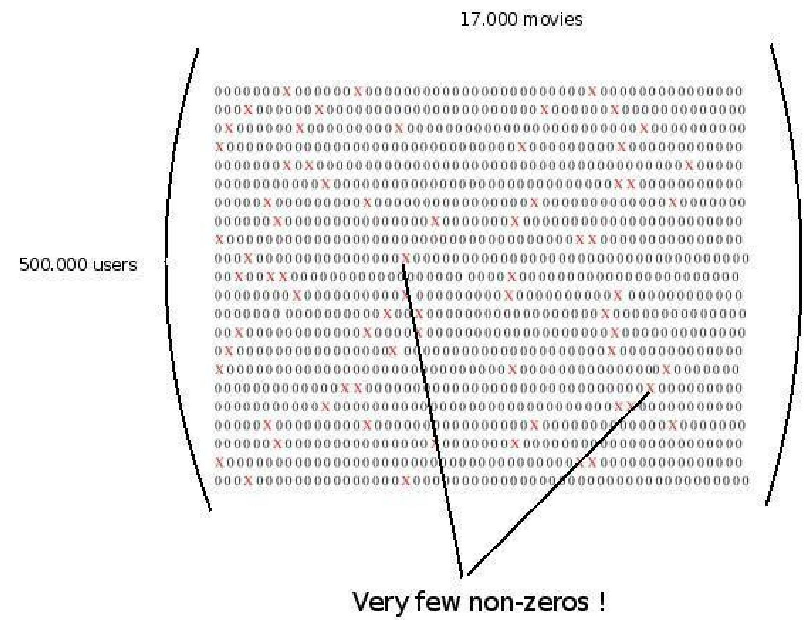
- Requires **minimal knowledge** engineering efforts
- Users and products are symbols without any internal structure or characteristics
- Produces good-enough results in most cases

Challenges:

- **Sparsity** – evaluation of large itemsets where user/item interactions are under 1%.
- **Scalability** - Nearest neighbor require computation that grows with both the number of users and the number of items.

The Sparsity Problem

- Typically: large product sets, user ratings for a small percentage of them
  (e.g. Amazon: millions of books and a user may have bought hundreds of books)
- If you represent the Netflix Prize rating data in a User/Movie matrix you get…
  - 500,000 x 17,000 = 8.5 B positions
  - Out of which only 100M are non-zero
- Number of users needs to be ~ 0.1 x size of the catalog



Model-based Collaborative Filtering

Model Based CF Algorithms

- Memory based
  - Use the entire user-item database to generate a prediction.
  - Usage of statistical techniques to find the neighbors – e.g. nearest-neighbor.
- Model-based
  - First develop a model of user
  - Type of model:
    - Probabilistic (e.g. Bayesian Network)
    - Clustering
    - Rule-based approaches (e.g. Association Rules)
    - Classification
    - Regression
    - LDA
    - ...

Model-based CF: What we learned from the Netflix Prize

NETFLIX

Netflix Prize

COMPLETED

What we were interested in:

- High quality *recommendations*

Proxy question:

- Accuracy in predicted rating
- Improve by 10% = $1million!

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$

NETFLIX

2007 Progress Prize

- Top 2 algorithms
  - SVD - Prize RMSE: 0.8914
  - RBM - Prize RMSE: 0.8990
- Linear blend Prize RMSE: 0.88
- Currently in use as part of Netflix' rating prediction component
- Limitations
  - Designed for 100M ratings, we have 5B ratings
  - Not adaptable as users add ratings
  - Performance issues

SVD/MF

$$X\,[n \times m] = U\,[n \times r]\,S\,[r \times r]\,(V\,[m \times r])^T$$

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

- **X**: *m x n* matrix (e.g., m users, n videos)
- **U**: *m x r* matrix (m users, r factors)
- **S**: *r x r* diagonal matrix (strength of each ‘factor’) (r: rank of the matrix)
- **V**: r x n matrix (n videos, r factor)

Simon Funk's SVD

- One of the most interesting findings during the Netflix Prize came out of a blog post
- Incremental, iterative, and approximate way to compute the SVD using gradient descent

NETFLIX

SVD for Rating Prediction

- User factor vectors $p_u \in \Re^f$ and item-factors vector $q_v \in \Re^f$
- Baseline (bias) $b_{uv} = \mu + b_u + b_v$ (user & item deviation from average)
- Predict rating as $r'_{uv} = b_{uv} + p_u^T q_v$
- SVD++ (Koren et. Al) asymmetric variation w. implicit feedback

$$r'_{uv} = b_{uv} + q_v^T \left(|R(u)|^{\frac{-1}{2}} \sum_{j \in R(u)} (r_{uj} - b_{uj}) x_j + |N(u)|^{\frac{-1}{2}} \sum_{j \in N(u)} y_j \right)$$

- Where
  - $q_v, x_v, y_v \in \Re^f$ are three item factor vectors
  - Users are not parametrized, but rather represented by:
  - R(u): items rated by user u
  - N(u): items for which the user has given implicit preference (e.g. rated vs. not rated)

NETFLIX

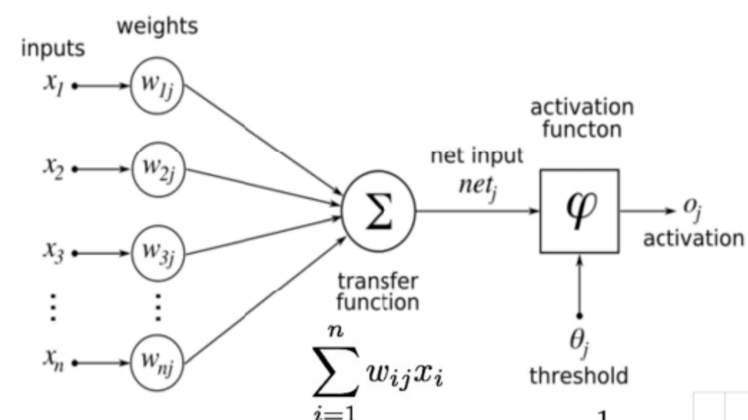
Restricted Boltzmann Machines

- Each unit is a state that can be active or not active
- Each input to a unit is associated to a weight
- The transfer function Σ calculates a score for every unit based on the weighted sum of inputs
- Score is passed to the activation function ϕ that calculates the probability of the unit to be active
- Restrict the connectivity to make learning easier.

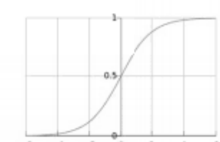
  Only one layer of hidden units.

  No connections between hidden units.

  Hidden units are independent given visible states



$$\phi = \frac{1}{1 + e^{-\Sigma}}$$



$$P(o_j = 1 | x) = \phi\left(\sum_{i=1}^{n} w_{ij} x_i + \theta_j\right)$$

NETFLIX

RBM for Recommendations

- Each visible unit = an item
- Num. of hidden units a is parameter
- In training phase, for each user:
  - If user rated item, v_i is activated
  - Activation states of v_i = inputs to h_j
  - Based on activation, h_j is computed
  - Activation state of h_j becomes input to v_i
  - Activation state of v_i is recalculated
  - Difference between current and past activation state for v_i used to update weights w_{ij} and thresholds
- In prediction phase:
  - For the items of the user the v_i are activated
  - Based on this the state of the h_j is computed
  - The activation of h_j is used as input to recompute the state of v_i
  - Activation probabilities are used to recommend items

Restricted Boltzmann Machines
for Collaborative Filtering

Ruslan Salakhutdinov RSALAKHU@CS.TORONTO.EDU
Andriy Mnih AMNIH@CS.TORONTO.EDU
Geoffrey Hinton HINTON@CS.TORONTO.EDU
University of Toronto, 6 King's College Rd., Toronto, Ontario M5S 3G4, Canada

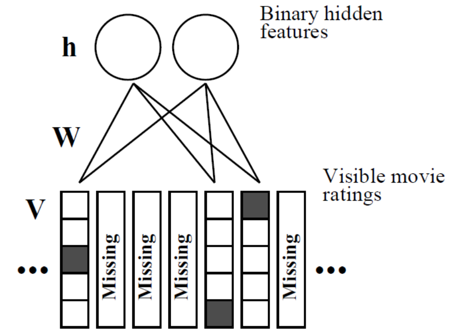
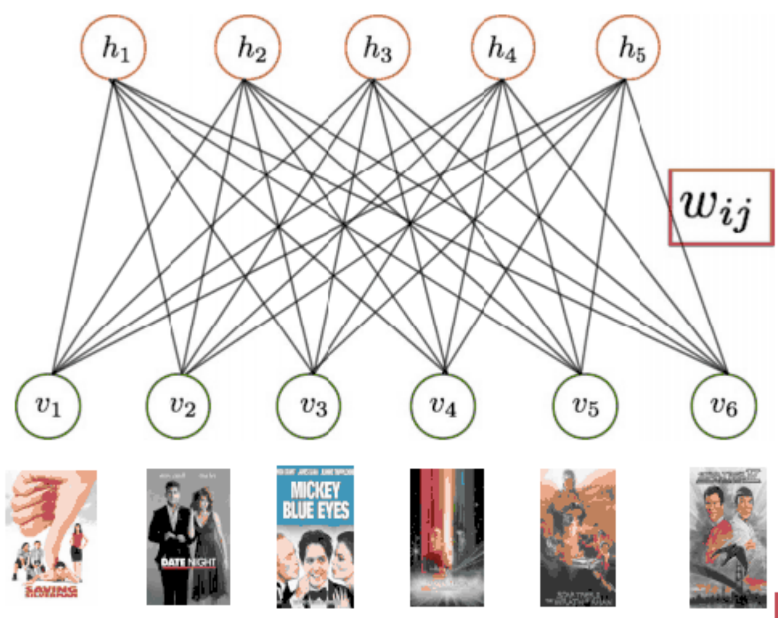


Figure 1. A restricted Boltzmann machine with binary hidden units and softmax visible units. For each user, the RBM only includes softmax units for the movies that user has rated. In addition to the symmetric weights between each hidden unit and each of the $K = 5$ values of a softmax unit, there are 5 biases for each softmax unit and one for each hidden unit. When modeling user ratings with an RBM that has Gaussian hidden units, the top layer is composed of linear units with Gaussian noise.



NETFLIX

Putting all together

- Remember that current production model includes an **ensemble** of both SVD++ and RBMs

What about the final prize ensembles?

- Our offline studies showed they were too computationally intensive to scale
- Expected improvement not worth the engineering effort
- Plus…. Focus had already shifted to other issues that had more impact than rating prediction.

Clustering

Clustering

- Goal: **cluster** users and compute per-cluster "typical" preferences
- Users receive recommendations computed at the cluster level

Locality-sensitive Hashing (LSH)

- Method for grouping similar items in highly dimensional spaces
- Find a hashing function s.t. similar items are grouped in the same buckets
- Main application is Nearest-neighbors
  - Hashing function is found iteratively by concatenating random hashing functions
  - Addresses one of NN main concerns: performance

Other "interesting" clustering techniques

- k-means and all its variations
- Affinity Propagation
- Spectral Clustering
- Non-parametric Bayesian Clustering (e.g. HDPs)

Association Rules

Association rules

- Past purchases are interpreted as transactions of "associated" items
- If a visitor has some interest in Book 5, she will be recommended to buy Book 3 as well
- Recommendations are constrained to some minimum levels of confidence
- Fast to implement and execute (e.g. A Priori algorithm)

		Also bought...					
		Book1	Book2	Book3	Book4	Book5	Book6
Customers who bought...	Book1				1	1	
	Book2			2		1	1
	Book3		2			2	
	Book4	1					
	Book5	1	1	2			
	Book6		1				

Classifiers

Classifiers

- **Classifiers** are general computational models trained using positive and negative examples
- They may take in inputs:
  - Vector of item features (action / adventure, Bruce Willis)
  - Preferences of customers (like action / adventure)
  - Relations among item
- E.g. Logistic Regression, Bayesian Networks, Support Vector Machines, Decision Trees, etc...

Classifiers

- Classifiers can be used in CF and CB Recommenders
- Pros:
  - Versatile
  - Can be combined with other methods to improve accuracy of recommendations
- Cons:
  - Need a relevant training set
  - May overfit (Regularization)
- E.g. Logistic Regression, Bayesian Networks, Support Vector Machines, Decision Trees, etc...

Limitations of Collaborative Filtering

Limitations of Collaborative Filtering

- **Cold Start**: There needs to be enough other users already in the system to find a match. New items need to get enough ratings.
- **Popularity Bias**: Hard to recommend items to someone with unique tastes.
  - Tends to recommend popular items (items from the tail do not get so much data)

Index

2.2 Content-based Recommenders

Content-Based Recommendation

- Recommendations based on content of items rather than on other users' opinions/interactions
- Goal: recommend items **similar** to those the user liked
- Common for recommending **text-based products** (web pages, usenet news messages,)
- Items to recommend are "described" by their associated **features** (e.g. keywords)
- **User Model** structured in a "similar" way as the content: features/keywords more likely to occur in the preferred documents (lazy approach)
- The user model can be a **classifier** based on whatever technique (Neural Networks, Naïve Bayes...)

Pros/cons of CB Approach

Pros

- No need for data on other users: No cold-start or sparsity
- Able to recommend to users with unique tastes.
- Able to recommend new and unpopular items
- Can provide explanations by listing content-features

Cons

- Requires content that can be encoded as meaningful features (difficult in some domains/catalogs)
- Users represented as learnable function of content features.
- Difficult to implement serendipity
- Easy to overfit (e.g. for a user with few data points)

A word of caution

Recommending New Movies: Even a Few Ratings Are More Valuable Than Metadata

István Pilászy [*]
Dept. of Measurement and Information Systems
Budapest University of Technology and Economics
Magyar Tudósok krt. 2.
Budapest, Hungary
pila@mit.bme.hu

Domonkos Tikk [*,†]
Dept. of Telecom. and Media Informatics
Budapest University of Technology and Economics
Magyar Tudósok krt. 2.
Budapest, Hungary
tikk@tmit.bme.hu

ABSTRACT

The Netflix Prize (NP) competition gave much attention to collaborative filtering (CF) approaches. Matrix factorization (MF) based CF approaches assign low dimensional feature vectors to users and items. We link CF and content-based filtering (CBF) by finding a linear transformation that transforms user or item descriptions so that they are as close as possible to the feature vectors generated by MF for CF.

We propose methods for explicit feedback that are able to handle 140 000 features when feature vectors are very sparse. With movie metadata collected for the NP movies we show that the prediction performance of the methods is comparable to that of CF, and can be used to predict user preferences on new movies.

We also investigate the value of movie metadata compared to movie ratings in regards of predictive power. We compare

1. INTRODUCTION

The goal of recommender systems is to give personalized recommendation on items to users. Typically the recommendation is based on the former and current activity of the users, and metadata about users and items, if available.

There are two basic strategies that can be applied when generating recommendations. Collaborative filtering (CF) methods are based only on the activity of users, while content-based filtering (CBF) methods use only metadata. In this paper we propose hybrid methods, which try to benefit from both information sources.

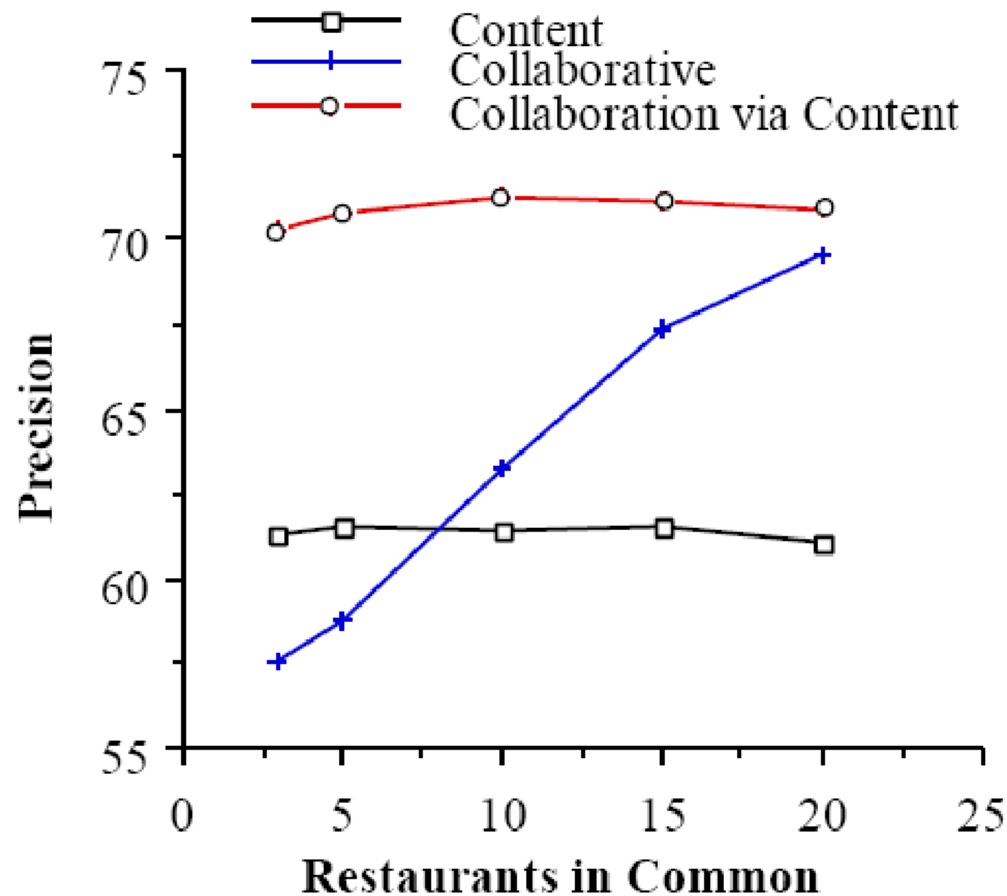
The two most important families of CF methods are matrix factorization (MF) and neighbor-based approaches. Usually, the goal of MF is to find a low dimensional representation for both users and movies, i.e. each user and movie is associated with a feature vector. Movie metadata (which

Index

2.3 Hybrid Approaches

Comparison of methods (FAB system)

- Content–based recommendation with Bayesian classifier
- Collaborative is standard using Pearson correlation
- Collaboration via content uses the content-based user profiles



Averaged on 44 users

Precision computed in top 3 recommendations

NETFLIX

Hybridization Methods

Hybridization Method	Description
Weighted	Outputs from several techniques (in the form of scores or votes) are combined with different degrees of importance to offer final recommendations
Switching	Depending on situation, the system changes from one technique to another
Mixed	Recommendations from several techniques are presented at the same time
Feature combination	Features from different recommendation sources are combined as input to a single technique
Cascade	The output from one technique is used as input of another that refines the result
Feature augmentation	The output from one technique is used as input features to another
Meta-level	The model learned by one recommender is used as input to another

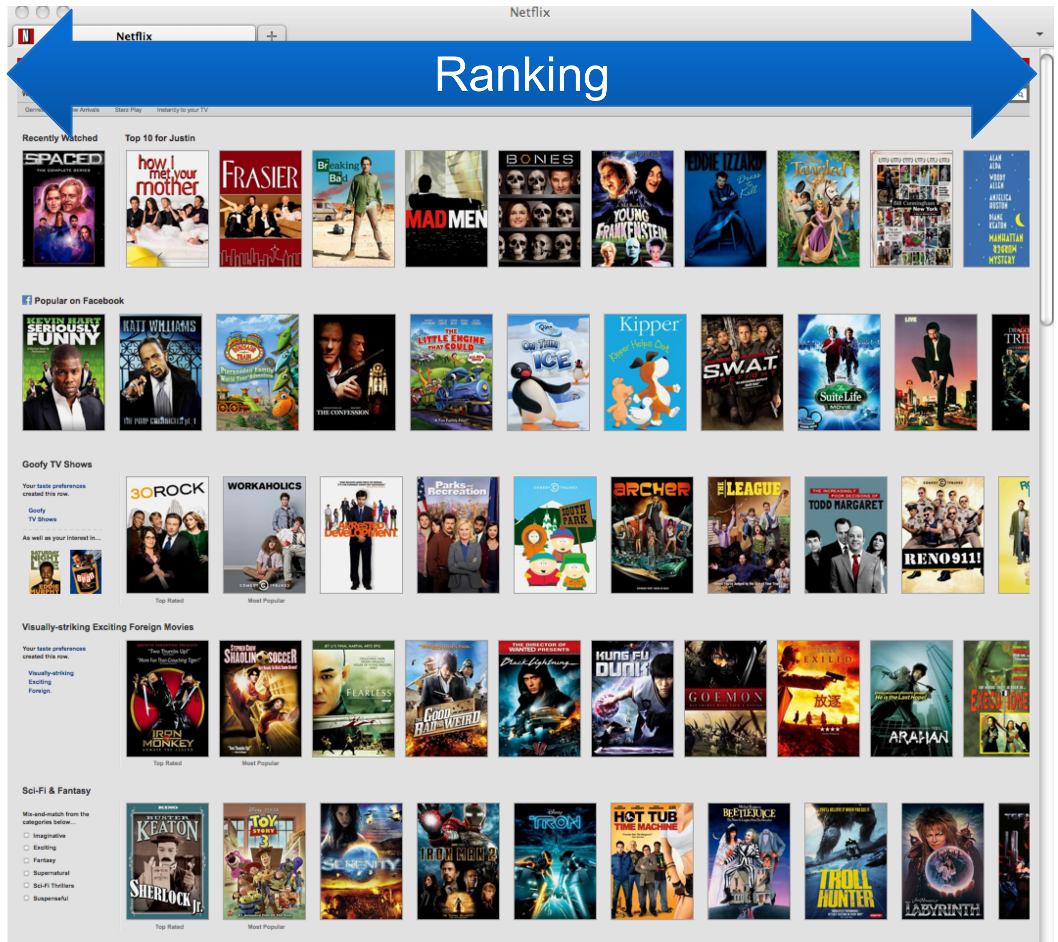
Index

3. Beyond traditional approaches to Recommendation

3.1 Ranking

Ranking

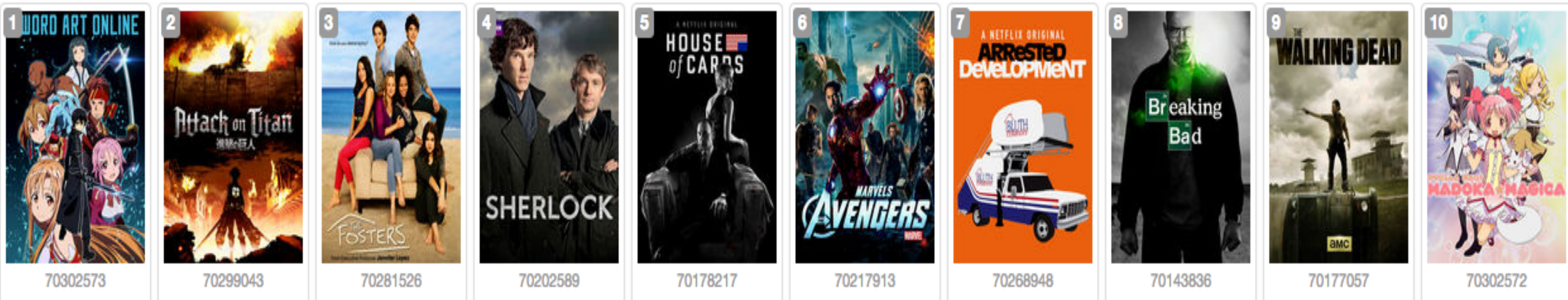
Key algorithm, sorts titles in most contexts



Ranking

- Most recommendations are presented in a sorted list
- Recommendation can be understood as a ranking problem
- Popularity is the obvious baseline
- Ratings prediction is a clear secondary data input that allows for personalization
- Many other features can be added

Ranking by ratings



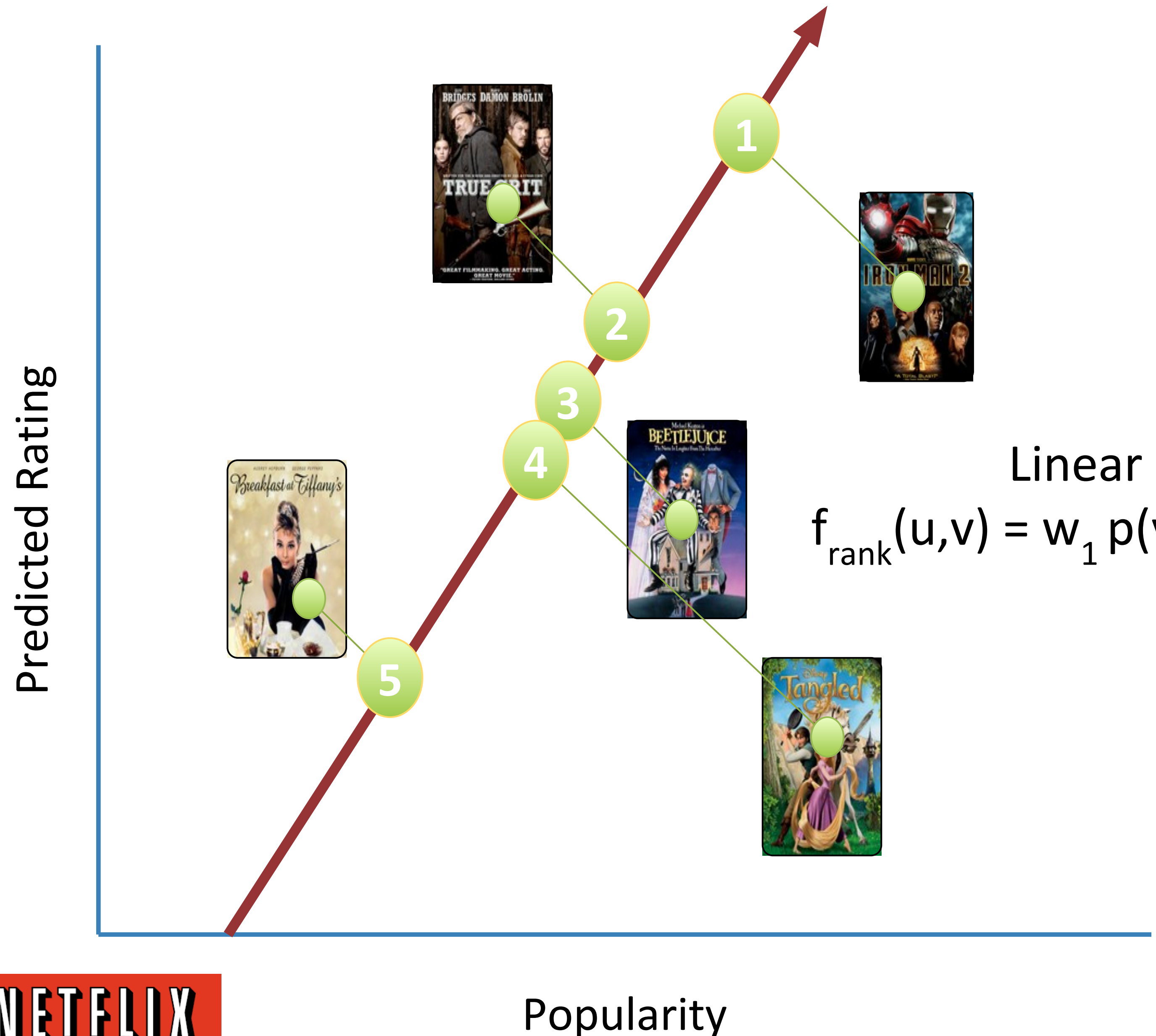
4.7 4.6 4.5 4.5 4.5 4.5 4.5 4.5 4.5 4.5

Niche titles

High average ratings… by those who would watch it

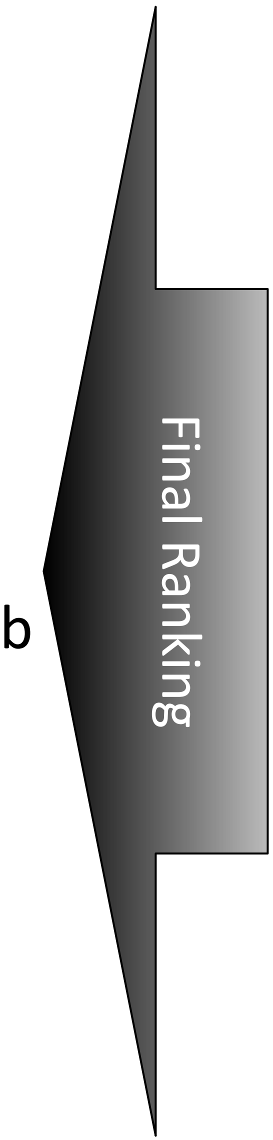

RMSE

Example: Two features, linear model



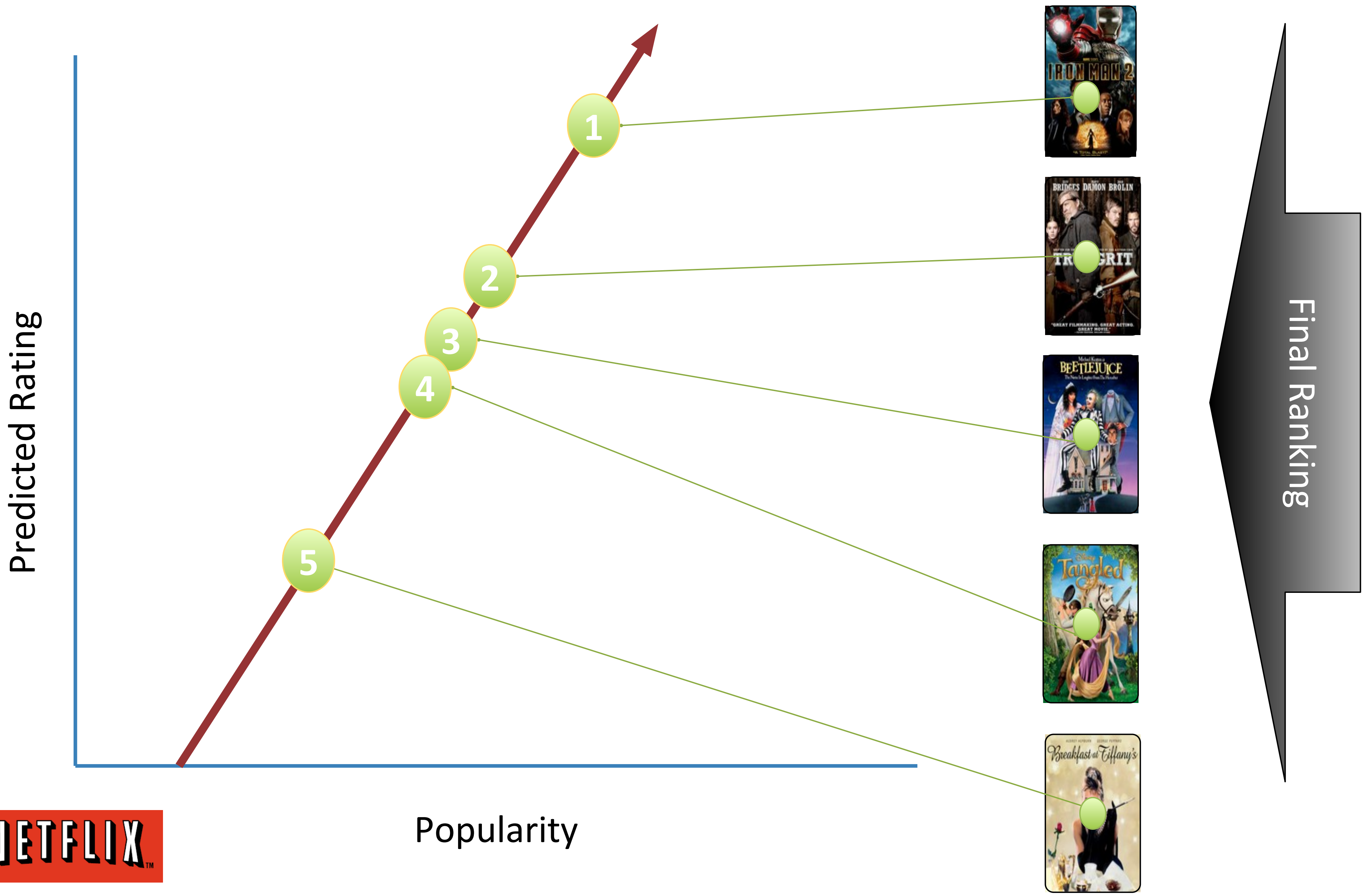
Linear Model:

$$f_{rank}(u,v) = w_1 \, p(v) + w_2 \, r(u,v) + b$$

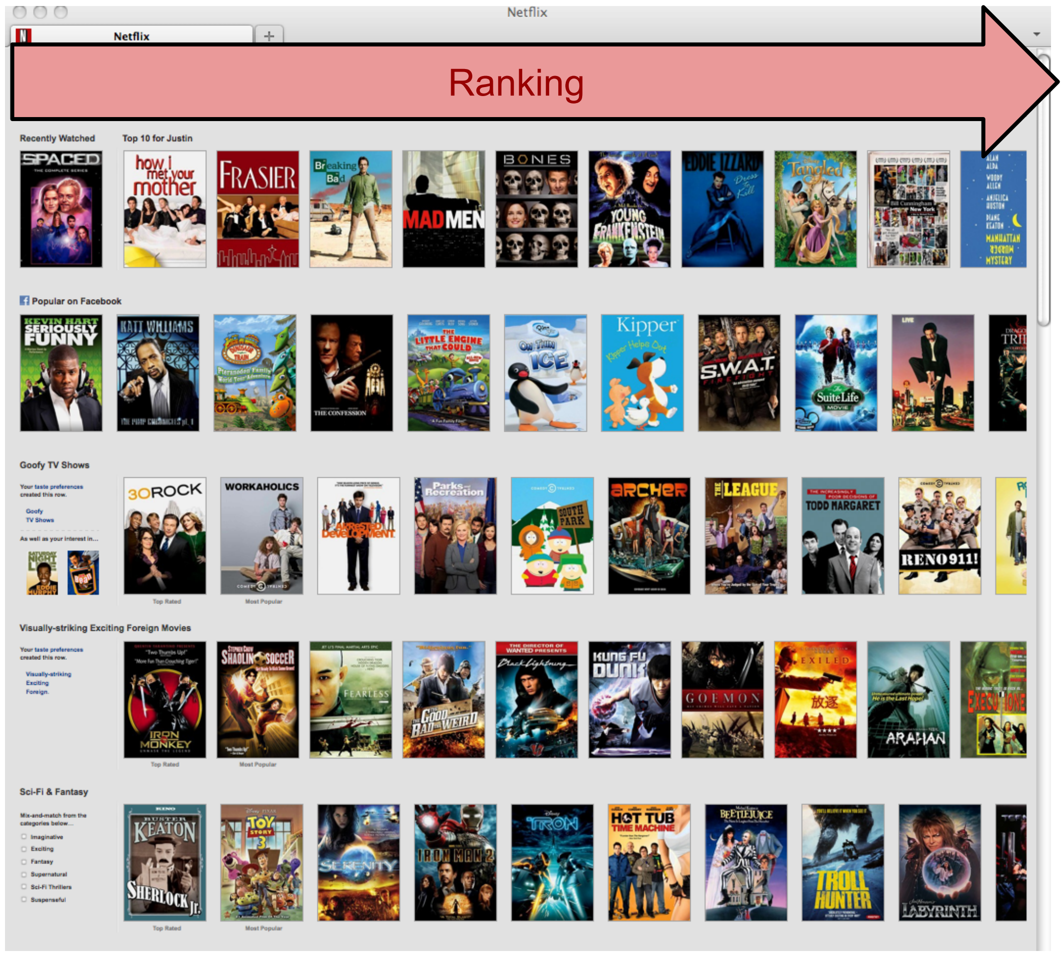


NETFLIX

Example: Two features, linear model



Ranking


Netflix
Ranking
Recently Watched
Top 10 for Justin
Popular on Facebook
Goofy TV Shows
Visually-striking Exciting Foreign Movies
Sci-Fi & Fantasy

Learning to rank

- Machine learning problem: goal is to construct ranking model from training data
- Training data can be a partial order or binary judgments (relevant/not relevant).
- Resulting order of the items typically induced from a numerical score
- Learning to rank is a key element for personalization
- You can treat the problem as a standard supervised classification problem

NETFLIX

Learning to rank - Metrics

- Quality of ranking measured using metrics as
  - Normalized Discounted Cumulative Gain
  - Mean Reciprocal Rank (MRR)
  - Fraction of Concordant Pairs (FCP)
  - Others…
- But, it is hard to optimize machine-learned models directly on these measures (e.g. non-differentiable)
- Recent research on models that directly optimize ranking measures

NETFLIX

Learning to rank - Approaches

1. Pointwise
   - Ranking function minimizes loss function defined on individual relevance judgment
   - Ranking score based on regression or classification
   - Ordinal regression, Logistic regression, SVM, GBDT, …
2. Pairwise
   - Loss function is defined on pair-wise preferences
   - Goal: minimize number of inversions in ranking
   - Ranking problem is then transformed into the binary classification problem
   - LambdaMart, RankSVM, RankBoost, RankNet, FRank…

Learning to rank - Approaches

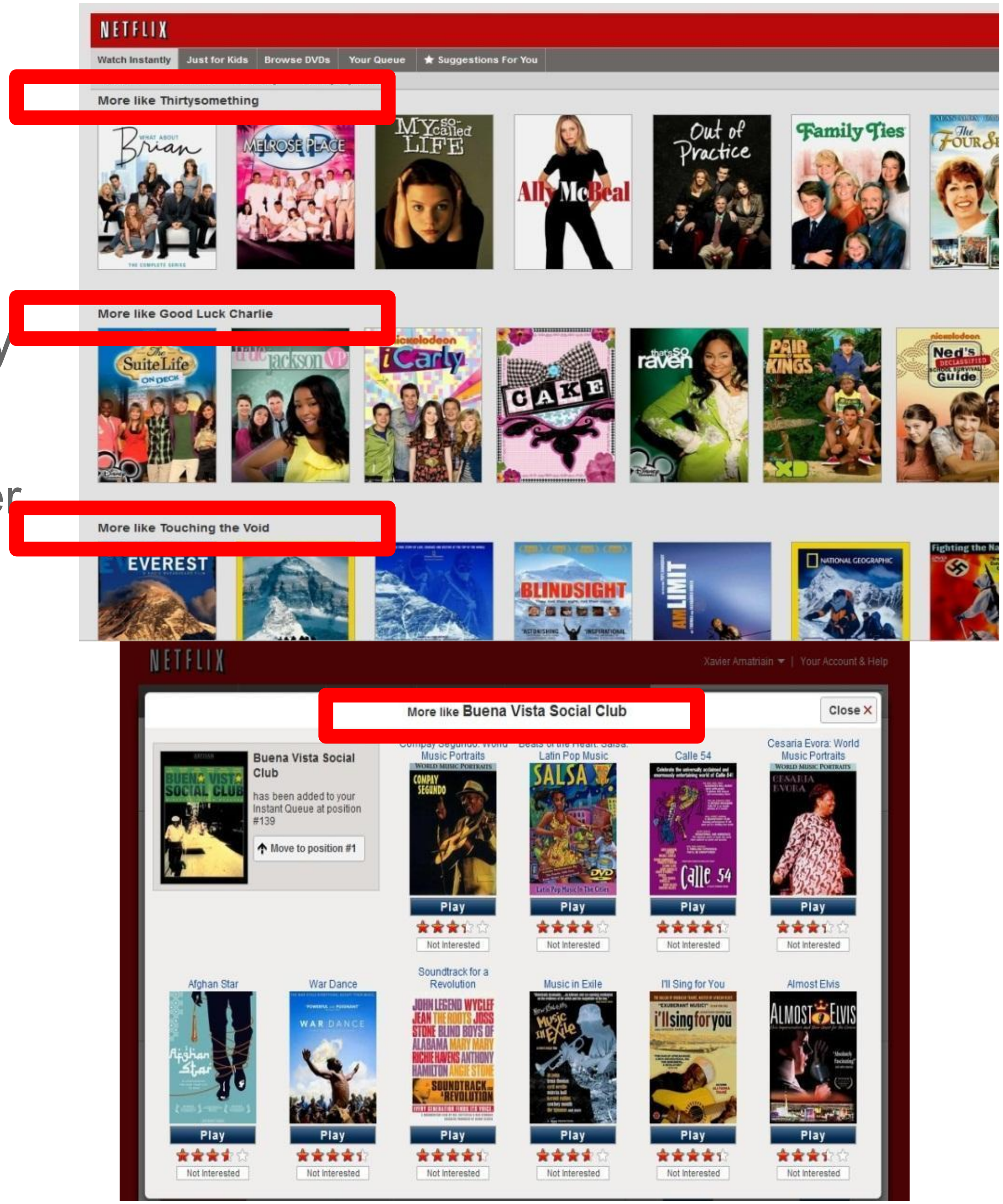
3. Listwise
   - Indirect Loss Function
     - RankCosine: similarity between ranking list and ground truth as loss function
     - ListNet: KL-divergence as loss function by defining a probability distribution
     - Problem: optimization of listwise loss function may not optimize IR metrics
   - Directly optimizing IR measures (difficult since they are not differentiable)
     - Genetic Programming or Simulated Annealing
     - Gradient descent on smoothed version of objective function (e.g. CLiMF or TFMAP)
     - SVM-MAP relaxes MAP metric by adding to SVM constraints
     - AdaRank uses boosting to optimize NDCG

Index

3.2 Similarity as Recommendation

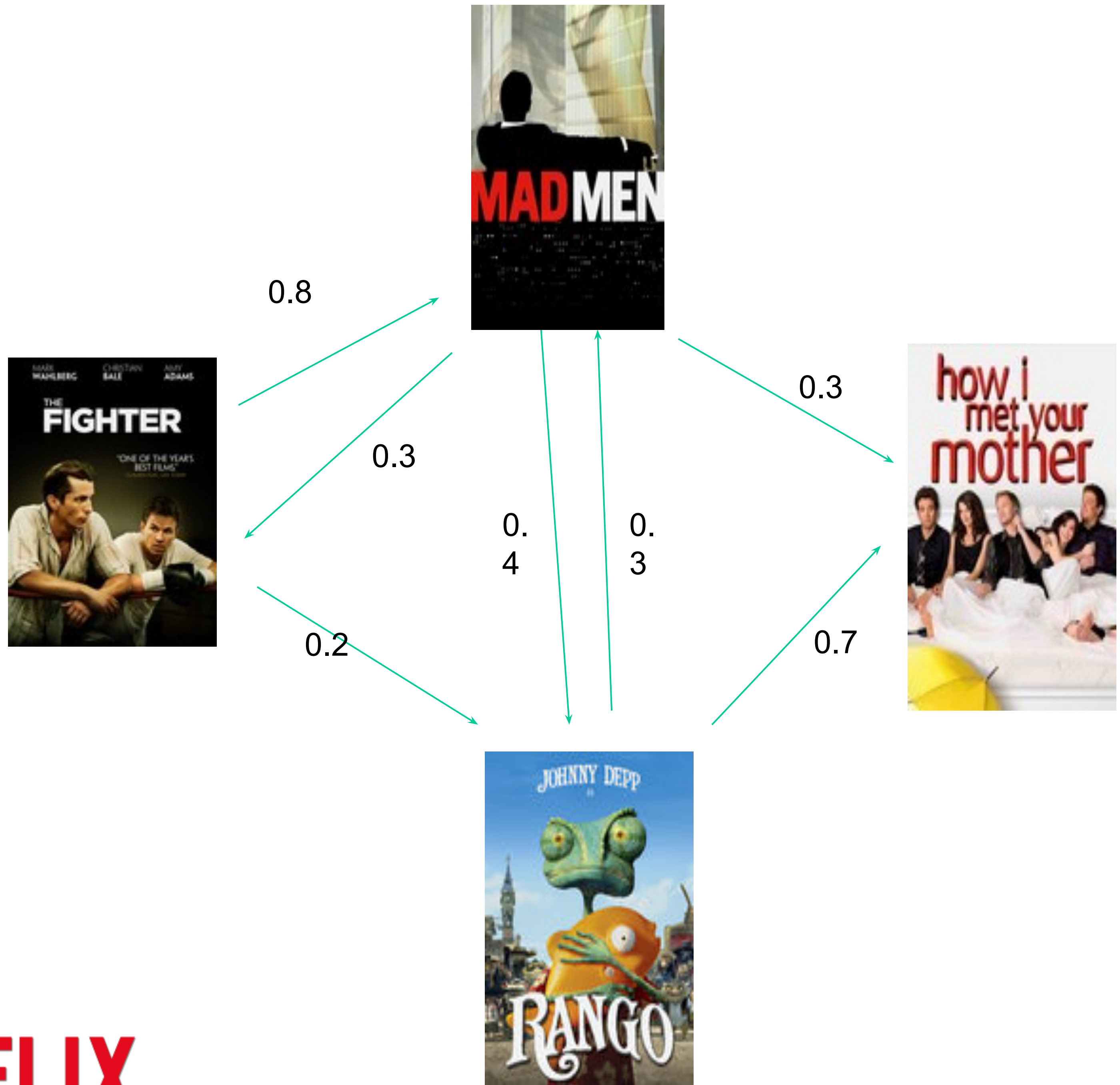
Similars

- Displayed in many different contexts
  - In response to user actions/context (search, queue add…)
  - More like… rows



NETFLIX

Graph-based similarities



NETFLIX

Example of graph-based similarity: SimRank

- SimRank (Jeh & Widom, 02): "two objects are similar if they are referenced by similar objects."

$$s(a,b) = \frac{C}{|I(a)||I(b)|} \sum_{i=1}^{|I(a)|} \sum_{j=1}^{|I(b)|} s(I_i(a), I_j(b))$$

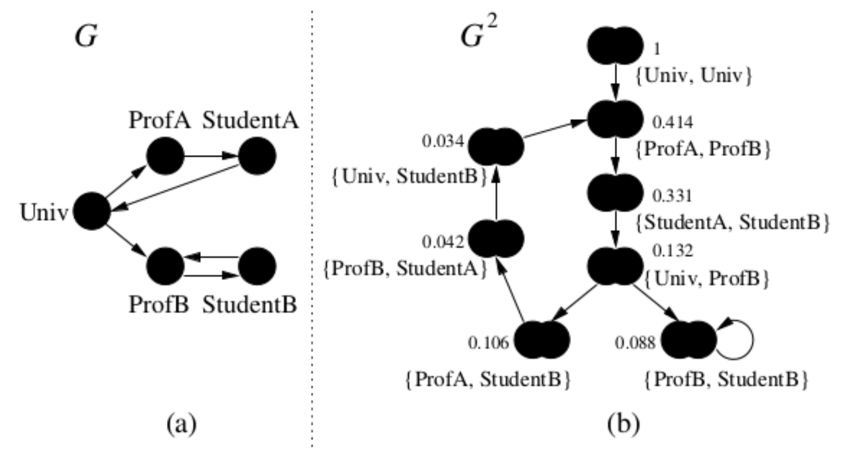


Figure 1: A small Web graph G and simplified node-pairs graph G^2. SimRank scores using parameter $C = 0.8$ are shown for nodes in G^2.

NETFLIX

Similarity ensembles

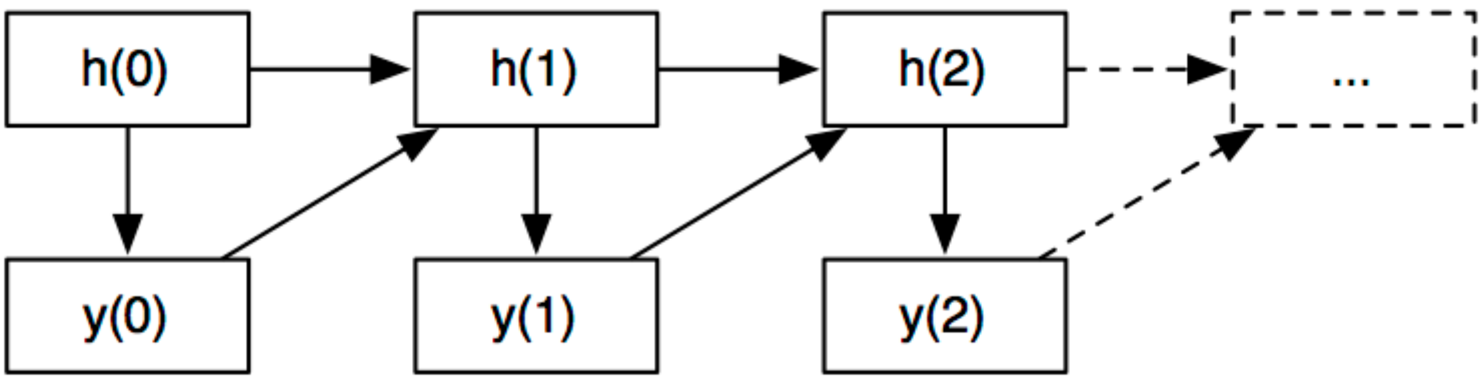
- Similarity can refer to different dimensions
  - Similar in metadata/tags
  - Similar in user play behavior
  - Similar in user rating behavior
  - …
- Combine them using an ensemble
  - Weights are learned using regression over existing response
  - Or… some MAB explore/exploit approach
- The final concept of "similarity" responds to what users vote as similar

Index

3.3 Deep Learning for Recommendation

Deep Learning for Collaborative Filtering

- Let's look at how Spotify uses Recurrent Networks for Playlist Prediction (http://erikbern.com/?p=589)



Deep Learning for Collaborative Filtering

- We assume $P(y_i|h_i)$ is a normal distribution, log-
  loss is just the (negative) L2 loss: $-(y_t - h_t)$
- We can specify that $h_{i+1} = \tanh(Uy_i + Vh_i)$ and that $h_0 = 0$
  - Model is now completely specified and we have $3k^2$ unknown parameters
  - Find U, V, and W to maximize log likelihood over all examples using backpropagation

$$\log L = \sum_{\text{all examples}} \left(\sum_{i=0}^{t-1} -(y_i - h_i)^2 \right)$$

Deep Learning for Collaborative Filtering

- In order to predict the next track or movie a user is going to watch, we need to define a distribution $P(y_i|h_i)$
  - If we choose Softmax as it is common practice, we get:

$$P(y_i|h_i) = \frac{\exp(h_t^T a_j)}{\sum_k \exp(h_t^T a_k)}$$

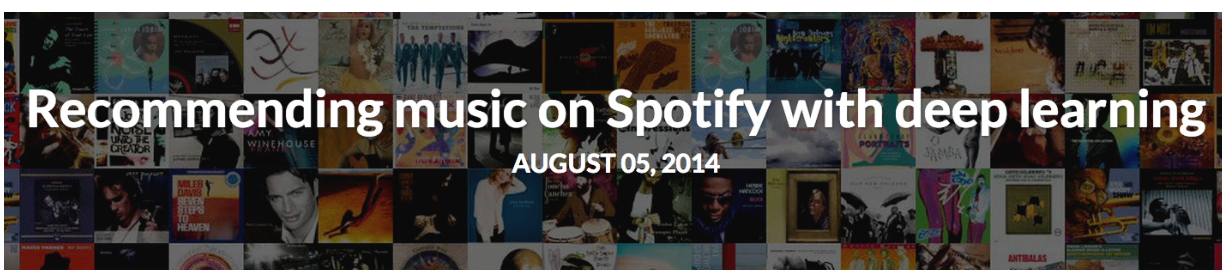
    - Problem: denominator (over all examples is very expensive to compute)
    - Solution: build a tree that implements a hierarchical softmax

- More details on the blogpost

NETFLIX

Deep Learning for Content-based Recommendations

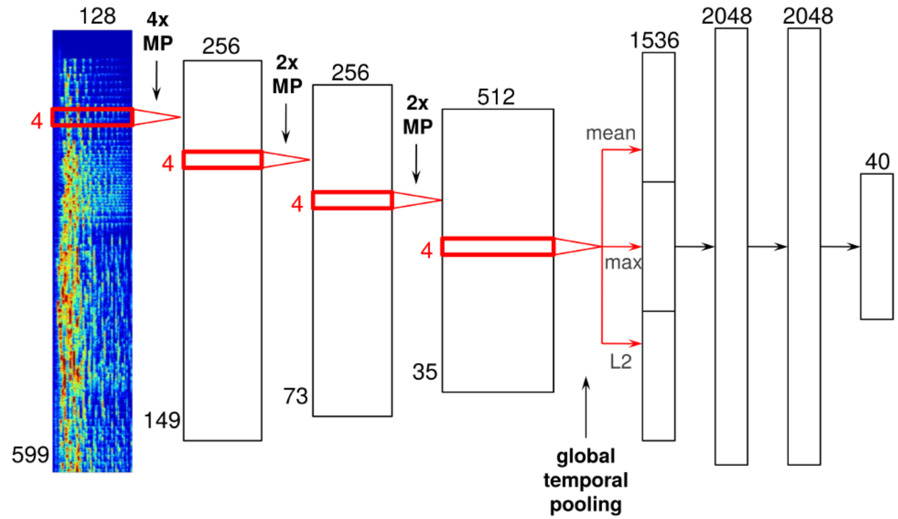
- Another application of Deep Learning to recommendations also from Spotify
  - http://benanne.github.io/2014/08/05/spotify-cnns.html also *Deep content-based music recommendation, Äaron van den Oord, Sander Dieleman and Benjamin Schrauwen, NIPS 2013*
- Application to coldstart new titles when very little CF information is available
- Using mel-spectrograms from the audio signal as input
- Training the deep neural network to predict 40 latent factors coming from Spotify's CF solution



NETFLIX

Deep Learning for Content-based Recommendations

- Network architecture made of 4 convolutional layers + 4 fully connected dense layers

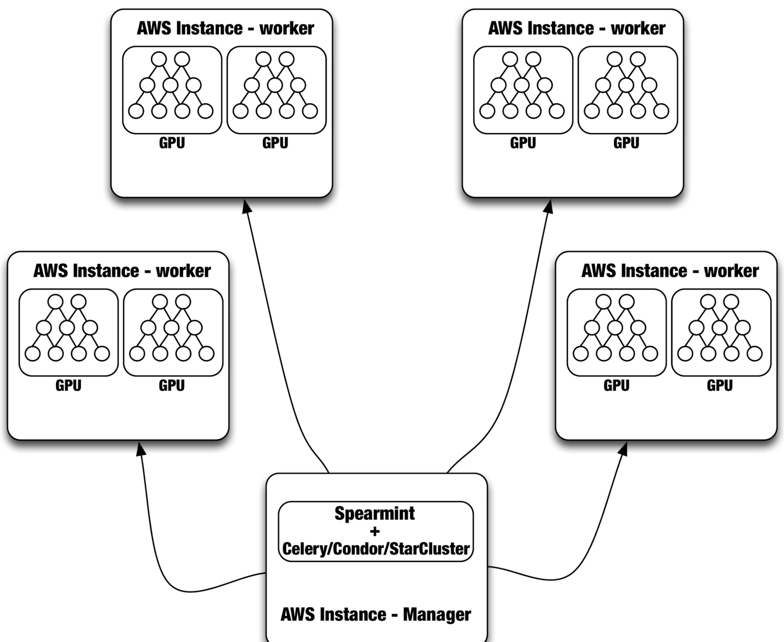


- One dimensional convolutional layers using RELUs (Rectified Linear Units) with activation max(0,x)
- **Max-pooling operations** between convolutional layers to downsample intermediate representations in time, and add time invariance
- **Global temporal pooling layer** after last convolutional layer: pools across entire time axis, computing statistics of the learned features across time:: mean, maximum and L2-norm
- Globally pooled features are fed into a series of **fully-connected layers** with 2048 RELUs

NETFLIX

ANN Training over GPUS and AWS

- How did we implement our ANN solution at Netflix?
  - Level 1 distribution: machines over different AWS regions
  - Level 2 distribution: machines in AWS and same AWS region
    - Use coordination tools
      - Spearmint or similar for parameter optimization
      - Condor, StarCluster, Mesos… for distributed cluster coordination
  - Level 3 parallelization: highly optimized parallel CUDA code on GPUs



http://techblog.netflix.com/2014/02/distributed-neural-networks-with-gpus.html

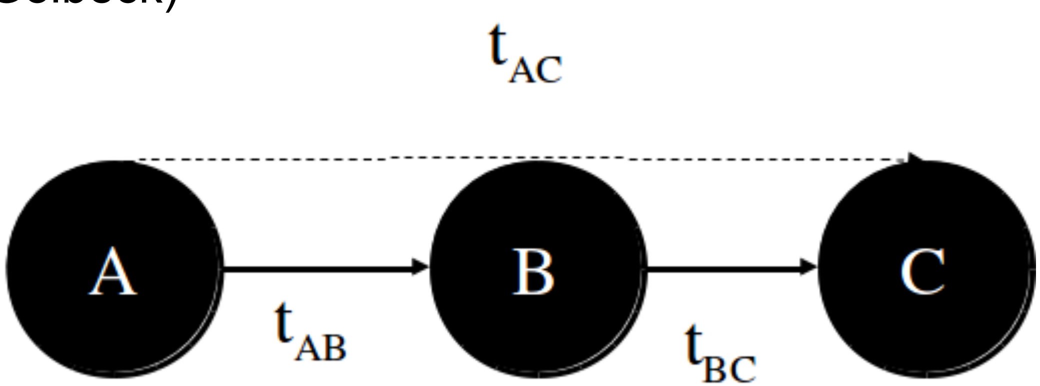
NETFLIX

Index

3.4 Social Recommendations

Social and Trust-based recommenders

- A social recommender system recommends items that are "popular" in the social proximity of the user.
- Social proximity = trust (can also be topic-specific)
- Given two individuals - the *source* (node A) and *sink* (node C) - derive how much the source should trust the sink.
- Algorithms
  - Advogato (Levien)
  - Appleseed (Ziegler and Lausen)
  - MoleTrust (Massa and Avesani)
  - TidalTrust (Golbeck)



NETFLIX

Other ways to use Social

- Social connections can be used in combination with other approaches
- In particular, “friendships” can be fed into collaborative filtering methods in different ways
  - replace or modify user-user “similarity” by using social network information
  - use social connection as a part of the ML objective function as regularizer
  - ...

Demographic Methods

- Aim to categorize the user based on personal attributes and make recommendation based on demographic classes
- Demographic groups can come from marketing research – hence experts decided how to model the users
- Demographic techniques form people-to-people correlations

NETFLIX

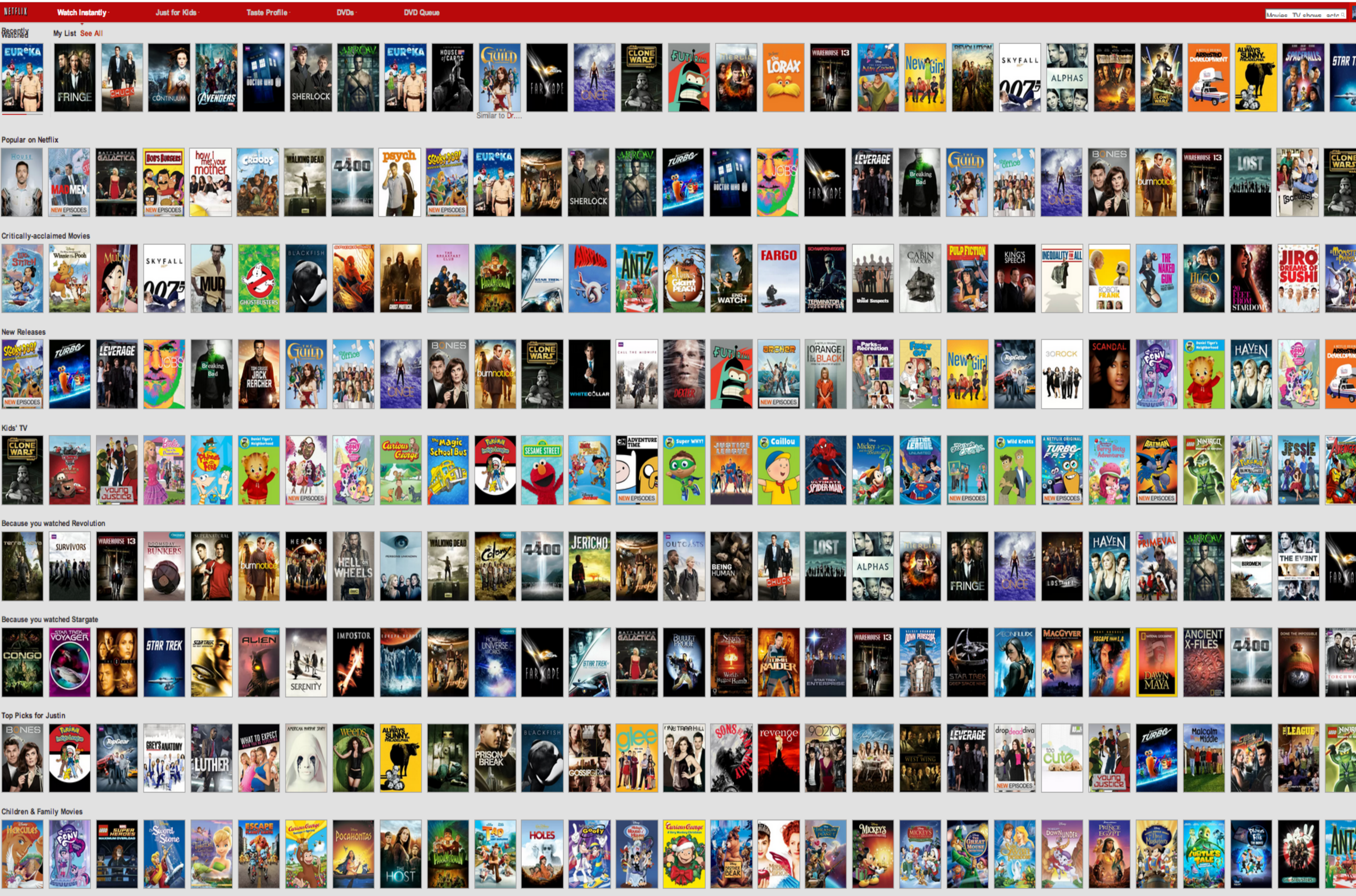
Index

1. The Recommender Problem
2. "Traditional" Methods
   2.1. Collaborative Filtering
   2.2. Content-based Recommendations
   2.3. Hybrid Approaches
3. Beyond Traditional Methods
   3.1. Learning to Rank
   3.2. Similarity
   3.3. Deep Learning
   3.4. Social Recommendations
   3.5. Page Optimization
   3.6. Tensor Factorization and Factorization Machines
   3.7. MAB Explore/Exploit
4. References

NETFLIX

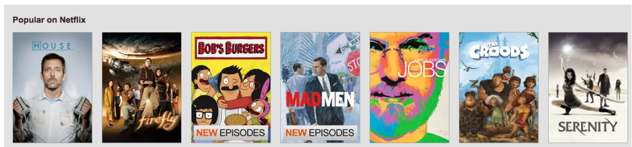
3.5. Page Optimization

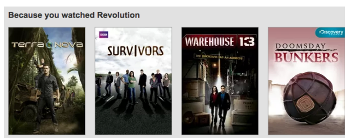
Page Composition



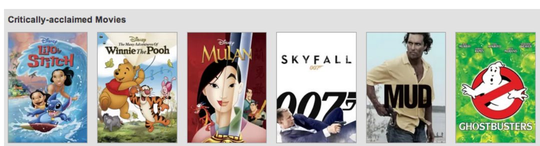
Page Composition

10,000s of possible rows





. . .



Variable number of possible videos per row (up to thousands)

1 personalized page

10-40 rows

per device

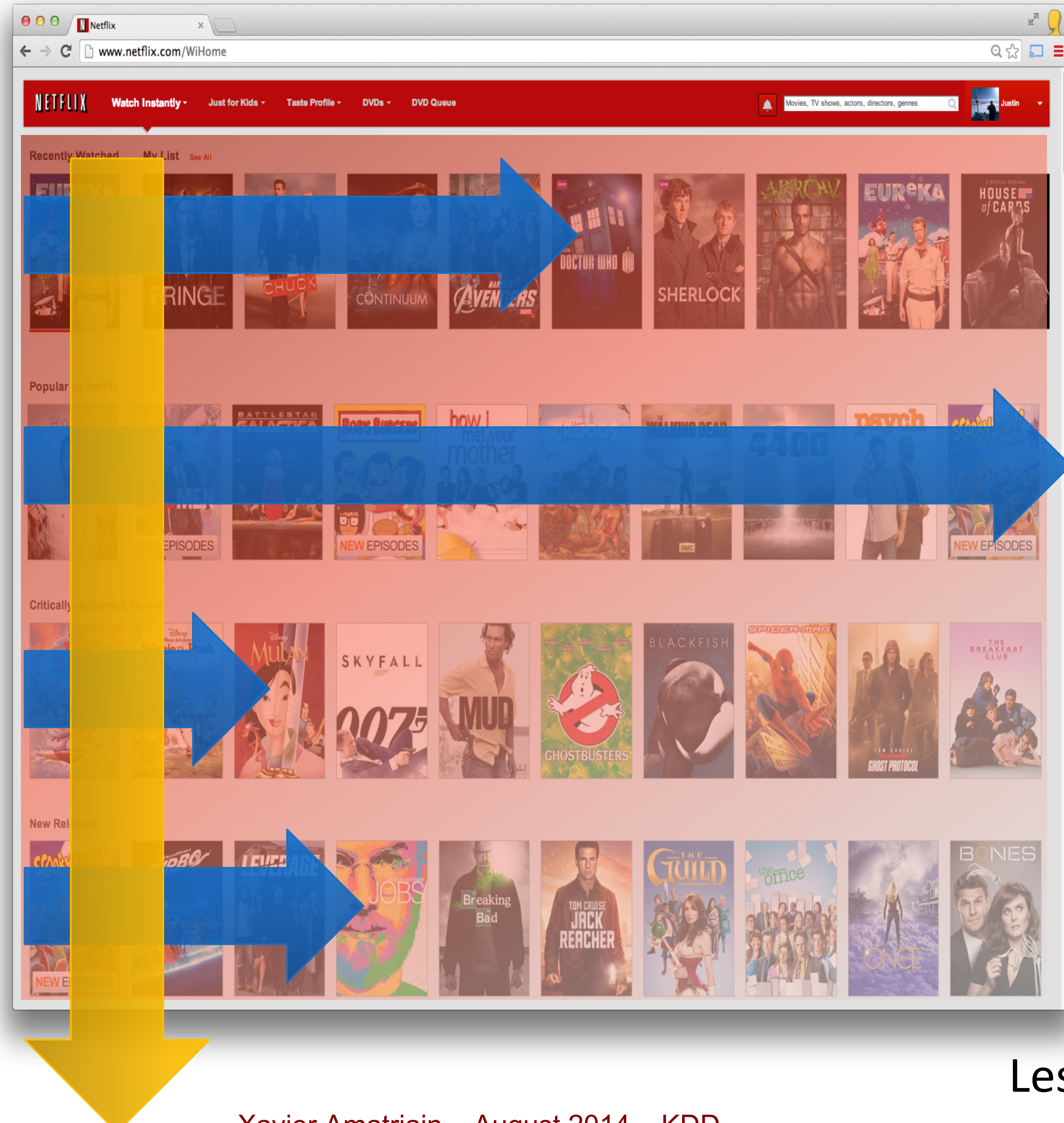
Page Composition

From “Modeling User Attention and Interaction on the Web” 2014 - PhD Thesis by Dmitry Lagun (Emory U.)

NETFLIX

User Attention Modeling

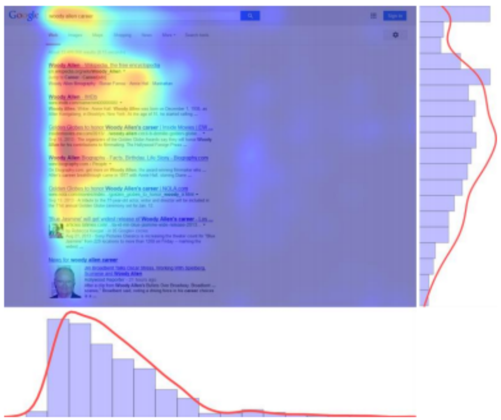
More likely to see



Less likely

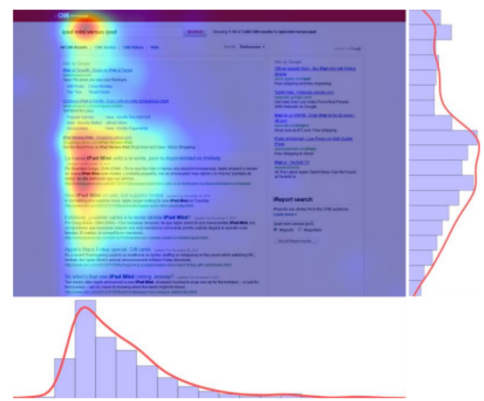
User Attention Modeling

Web Search (Google)



Social Network (Twitter)

News (CNN)



Shopping (Amazon)

From “Modeling User Attention and Interaction on the Web” 2014 - PhD Thesis by Dmitry Lagun (Emory U.)

Page Composition

Accurate vs. Diverse
Discovery vs. Continuation
Depth vs. Coverage
Freshness vs. Stability
Recommendations vs. Tasks

- To put things together we need to combine different elements
  - Navigational/Attention Model
  - Personalized Relevance Model
  - Diversity Model

Fair and Balanced: Learning to Present News Stories

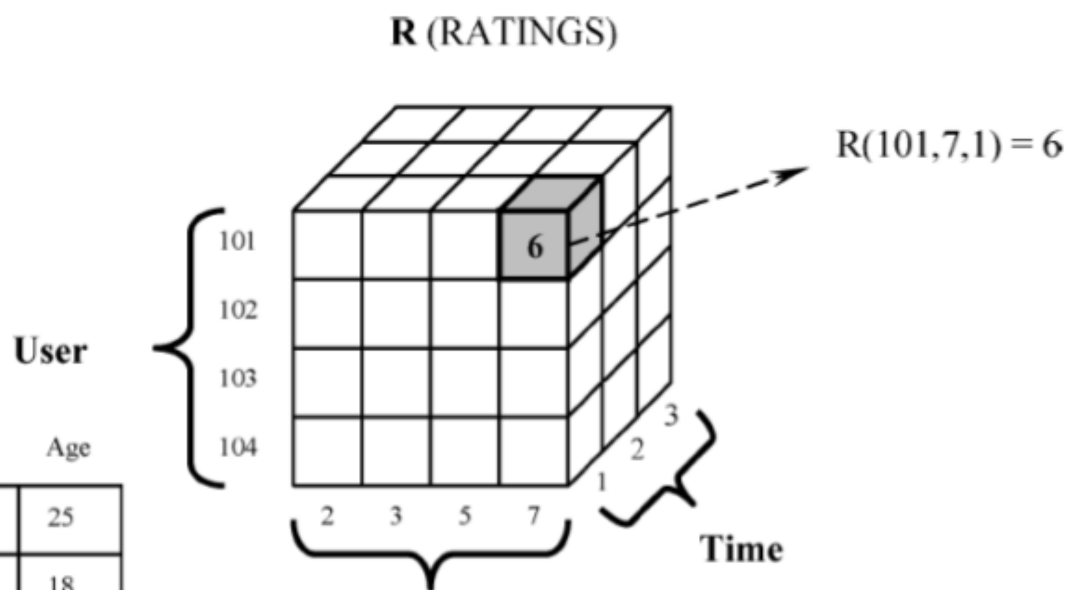
Amr Ahmed[*1], Choon Hui Teo[*1], S.V. N. Vishwanathan[2], Alex Smola[1]
[*] Co-first authors.
[1]Yahoo! Research, Santa Clara, CA 95053, USA
[2]Purdue University, West Lafayette, IN 47907, USA
{amahmed,choonhui,smola}@yahoo-inc.com, vishy@stat.purdue.edu

NETFLIX

Index

3.6 Tensor Factorization & Factorization Machines

N-dimensional model



User

Id	Name	Age
101	John	25
102	Bob	18
103	Alice	27
104	Mary	21

Item

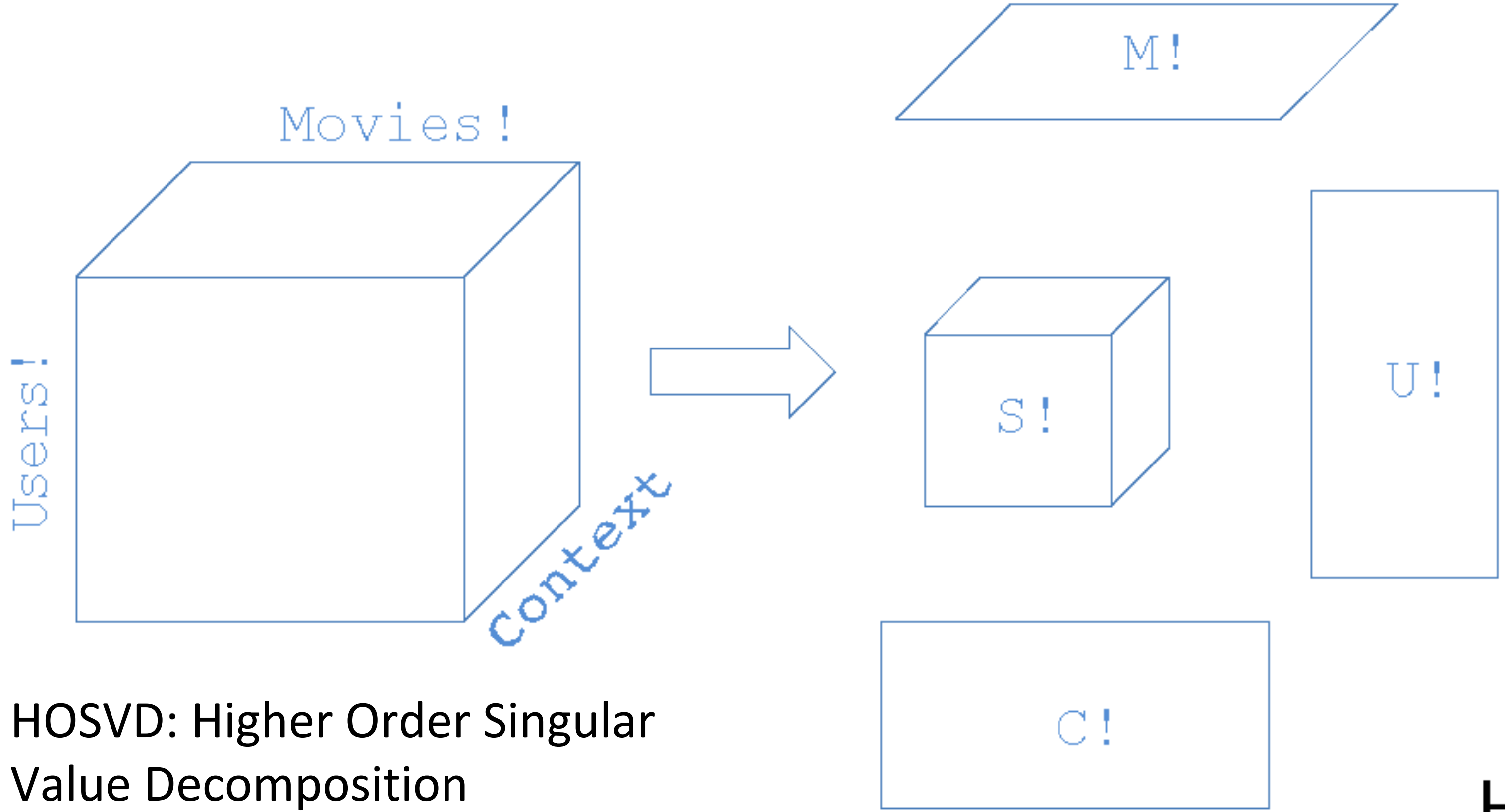
Id	Name	Cost
2	AB17	250.00
3	AB23	299.95
5	XY70	150.00
7	ZZ55	115.50

Time

Id	Name
1	Weekday
2	Weekend
3	Holiday

[Adomavicius et al., 2005]

Tensor Factorization



HOSVD: Higher Order Singular Value Decomposition

HOSVD Model

$U \in \mathbb{R}^{n \times d_U}$, $M \in \mathbb{R}^{m \times d_M}$ and $C \in \mathbb{R}^{c \times d_C}$

$S \in \mathbb{R}^{d_U \times d_M \times d_C}$

$$R[U, M, C, S] := L(F, Y) + \Omega[U, M, C] + \Omega[S]$$

Tensor Factorization

$$R[U, M, C, S] := L(F, Y) + \Omega[U, M, C] + \Omega[S]$$

Where:

$$\Omega[F] = \lambda_M \|M\|_F^2 + \lambda_U \|U\|_F^2 + \lambda_C \|C\|_F^2 \qquad \Omega[S] := \lambda_S \|S\|_{\mathrm{F}}^2$$

- We can use a simple squared error loss function:

$$l(f, y) = \frac{1}{2}(f - y)^2$$

- Or the absolute error loss

$$l(f, y) = |f - y|$$

- The loss function over all users becomes

$$L(F, Y) = \sum_i^n \sum_j^m l(f_{ij}, y_{ij})$$

Factorization Machines

- Generalization of regularized matrix (and tensor) factorization approaches combined with linear (or logistic) regression
- Problem: Each new adaptation of matrix or tensor factorization requires deriving new learning algorithms
  - Hard to adapt to new domains and add data sources
  - Hard to advance the learning algorithms across approaches
  - Hard to incorporate non-categorical variables

Factorization Machines

- Approach: Treat input as a real-valued feature vector
  - Model both linear and pair-wise interaction of *k* features (i.e. polynomial regression)
  - Traditional machine learning will overfit
  - Factor pairwise interactions between features
  - Reduced dimensionality of interactions promote generalization
  - Different matrix factorizations become different feature representations
  - Tensors: Additional higher-order interactions
- Combines “generality of machine learning/regression with quality of factorization models”

Factorization Machines

- Each feature gets a weight value and a factor vector
  - O(*dk*) parameters

$$b \in \mathbb{R}, \mathbf{w} \in \mathbb{R}^d, \mathbf{V} \in \mathbb{R}^{d \times k}$$

- Model equation:

$$f(\mathbf{x}) = b + \sum_{i=1}^{d} w_i x_i + \sum_{i=1}^{d} \sum_{j=i+1}^{d} x_i x_j \mathbf{v}_i^T \mathbf{v}_j \qquad O(d^2)$$

$$= b + \sum_{i=1}^{d} w_i x_i + \frac{1}{2} \sum_{f=1}^{k} \left(\left(\sum_{i=1}^{d} x_i v_{i,f} \right)^2 - \sum_{i=1}^{d} x_i^2 v_{i,f}^2 \right) \qquad O(kd)$$

NETFLIX

Factorization Machines

- Two categorical variables (*u*, *i*) encoded as real values:

Feature vector **x**

	A	B	C	...	TI	NH	SW	ST	...
$\mathbf{x}^{(1)}$	1	0	0	...	1	0	0	0	...
$\mathbf{x}^{(2)}$	1	0	0	...	0	1	0	0	...
$\mathbf{x}^{(3)}$	1	0	0	...	0	0	1	0	...
$\mathbf{x}^{(4)}$	0	1	0	...	0	0	1	0	...
$\mathbf{x}^{(5)}$	0	1	0	...	0	0	0	1	...
$\mathbf{x}^{(6)}$	0	0	1	...	1	0	0	0	...
$\mathbf{x}^{(7)}$	0	0	1	...	0	0	1	0	...
	User				Movie				

- FM becomes identical to MF with biases:

$$f(\mathbf{x}) = b + w_u + w_i + \mathbf{v}_u^T \mathbf{v}_i$$

From Rendle (2012) KDD Tutorial

Factorization Machines

- Makes it easy to add a time signal

Feature vector $\mathbf{x}$

	User				Movie					Time
	A	B	C	...	TI	NH	SW	ST	...	
$\mathbf{x}^{(1)}$	1	0	0	...	1	0	0	0	...	0.2
$\mathbf{x}^{(2)}$	1	0	0	...	0	1	0	0	...	0.6
$\mathbf{x}^{(3)}$	1	0	0	...	0	0	1	0	...	0.61
$\mathbf{x}^{(4)}$	0	1	0	...	0	0	1	0	...	0.3
$\mathbf{x}^{(5)}$	0	1	0	...	0	0	0	1	...	0.5
$\mathbf{x}^{(6)}$	0	0	1	...	1	0	0	0	...	0.1
$\mathbf{x}^{(7)}$	0	0	1	...	0	0	1	0	...	0.8

- Equivalent equation:

$$f(\mathbf{x}) = b + w_u + w_i + x_t w_t + \mathbf{v}_u^T \mathbf{v}_i + x_t \mathbf{v}_u^T \mathbf{v}_t + x_t \mathbf{v}_i^T \mathbf{v}_t$$

From Rendle (2012) KDD Tutorial

Factorization Machines (Rendle, 2010)

- L2 regularized
  - Regression: Optimize RMSE
  - Classification: Optimize logistic log-likelihood
  - Ranking: Optimize scores
- Can be trained using:
  - SGD
  - Adaptive SGD
  - ALS
  - MCMC

Gradient:

$$\frac{\partial}{\partial\theta} f(\mathbf{x}) = \begin{cases} 1 & \text{if } \theta \text{ is } b \\ x_i & \text{if } \theta \text{ is } w_i \\ x_i \sum_{j=1}^{d} v_{j,f} x_j - v_{i,f} x_i^2 & \text{if } \theta \text{ is } v_{i,f} \end{cases}$$

Least squares SGD:

$$\theta' = \theta - \eta \left((f(\mathbf{x}) - y) \frac{\partial}{\partial\theta} f(\mathbf{x}) + \lambda_\theta \theta \right)$$

NETFLIX

Factorization Machines (Rendle, 2010)

- Learning parameters:
  - Number of factors
  - Iterations
  - Initialization scale
  - Regularization (SGD, ALS) – Multiple
  - Step size (SGD, A-SGD)
  - MCMC removes the need to set those hyperparameters

Index

3.7 MAB Explore/Exploit

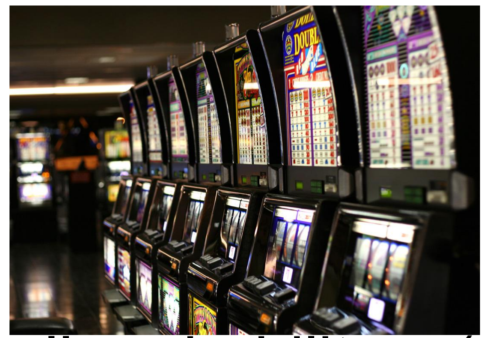
NETFLIX

Explore/Exploit

- One of the key issues when building any kind of personalization algorithm is how to trade off:
  - **Exploitation**: Cashing in on what we know about the user right now
  - **Exploration**: Using the interaction as an opportunity to learn more about the user
- We need to have informed and optimal strategies to drive that tradeoff
  - **Solution**: pick a reasonable set of candidates and show users only "enough" to gather information on them

Multi-armed Bandits

- Given possible strategies/candidates (slot machines) pick the arm that has the maximum potential of *being good* (minimize **regret**)



- Naive strategy: $\varepsilon - greedy$
  - Explore with a small probability ε (e.g. 5%) -> choose an arm at random
  - Exploit with a high probability ($1-\varepsilon$) (e.g. 95%) -> choose the best-known arm so far
- Translation to recommender systems
  - Choose an arm = choose an item/choose an algorithm (MAB testing)

NETFLIX

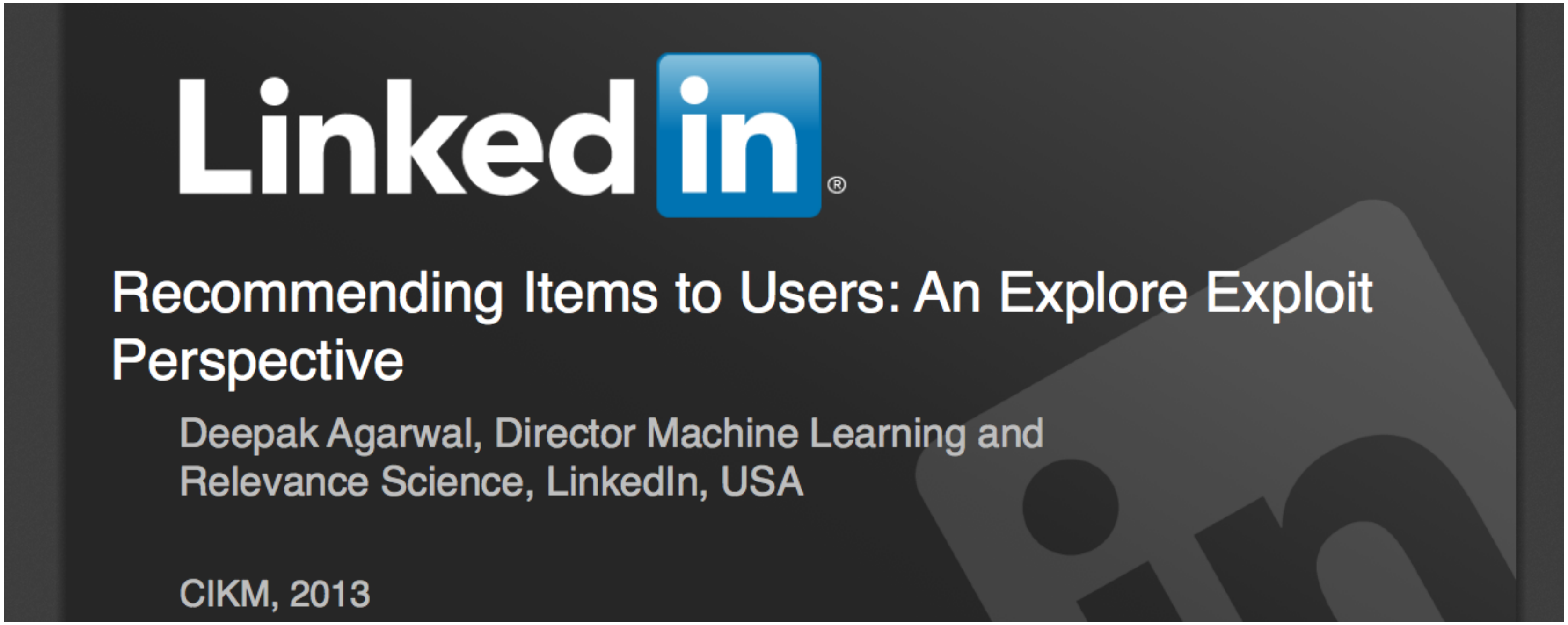
Multi-armed Bandits

- Better strategies not only take into account the mean of the posterior, but also the variance
- Upper Confidence Bound (UCB)
  - Show item with maximum score
  - Score = Posterior mean + Δ
  - Where Δ can be computed differently depending on the variant UCB (e.g. $\propto$ to the number of trials or the measured variance)
- Thompson Sampling
  - Given a posterior distribution
  - Sample on each iteration and choose the action that maximizes the expected reward

$$P(\theta|\mathcal{D}) \propto P(\mathcal{D}|\theta)P(\theta)$$

NETFLIX

Multi-armed Bandits



Index

1. The Recommender Problem
2. “Traditional” Methods
    2.1. Collaborative Filtering
    2.2. Content-based Recommendations
    2.3. Hybrid Approaches
3. Beyond Traditional Methods
    3.1. Learning to Rank
    3.2. Similarity
    3.3. Deep Learning
    3.4. Social Recommendations
    3.5. Page Optimization
    3.6. Tensor Factorization and Factorization Machines
    3.7. MAB Explore/Exploit
4. References

4. References

References

- *"Recommender Systems Handbook." Ricci, Francesco, Lior Rokach, Bracha Shapira, and Paul B. Kantor. (2010).*
- *"Recommender systems: an introduction". Jannach, Dietmar, et al. Cambridge University Press, 2010.*
- *"Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions"*. G. Adomavicious and A. Tuzhilin. 2005. IEEE Transactions on Knowledge and Data Engineering, 17 (6)
- *"Item-based Collaborative Filtering Recommendation Algorithms"*, B. Sarwar et al. 2001. Proceedings of World Wide Web Conference.
- *"Lessons from the Netflix Prize Challenge."*. R. M. Bell and Y. Koren. SIGKDD Explor. Newsl., 9(2):75–79, December 2007.
- *"Beyond algorithms: An HCI perspective on recommender systems"*. K. Swearingen and R. Sinha. In ACM SIGIR 2001 Workshop on Recommender Systems
- *"Recommender Systems in E-Commerce"*. J. Ben Schafer et al. ACM Conference on Electronic Commerce. 1999-
- *"Introduction to Data Mining"*, P. Tan et al. Addison Wesley. 2005

References

- *“Evaluating collaborative filtering recommender systems”*. J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl. ACM Trans. Inf. Syst., 22(1): 5–53, 2004.
- *“Trust in recommender systems”*. J. O’Donovan and B. Smyth. In Proc. of IUI ’05, 2005.
- *“Content-based recommendation systems”*. M. Pazzani and D. Billsus. In The Adaptive Web, volume 4321. 2007.
- *“Fast context-aware recommendations with factorization machines”*. S. Rendle, Z. Gantner, C. Freudenthaler, and L. Schmidt-Thieme. In Proc. of the 34th ACM SIGIR, 2011.
- *“Restricted Boltzmann machines for collaborative filtering”*. R. Salakhutdinov, A. Mnih, and G. E. Hinton.In Proc of ICML ’07, 2007
- “Learning to rank: From pairwise approach to listwise approach”*. Z. Cao and T. Liu. In In Proceedings of the 24th ICML, 2007.*
- *“Introduction to Data Mining”*, P. Tan et al. Addison Wesley. 2005

References

- D. H. Stern, R. Herbrich, and T. Graepel. “*Matchbox: large scale online bayesian recommendations*”. In Proc.of the 18th WWW, 2009.
- Koren Y and J. Sill. “*OrdRec: an ordinal model for predicting personalized item rating distributions*”. In Rec-Sys ’11.
- Y. Koren. “*Factorization meets the neighborhood: a multifaceted collaborative filtering model*”. In Proceedings of the 14th ACM SIGKDD, 2008.
- Yifan Hu, Y. Koren, and C. Volinsky. “*Collaborative Filtering for Implicit Feedback Datasets*”. In Proc. Of the 2008 Eighth ICDM
- Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson, N. Oliver, and A. Hanjalic. “*CLiMF: learning to maximize reciprocal rank with collaborative less-is-more filtering*”. In Proc. of the sixth Recsys, 2012.
- Y. Shi, A. Karatzoglou, L. Baltrunas, M. Larson,A. Hanjalic, and N. Oliver. “*TFMAP: optimizing MAP for top-n context-aware recommendation*”. In Proc. Of the 35th SIGIR, 2012.
- C. Burges. 2010. *From RankNet to LambdaRank to LambdaMART: An Overview*. MSFT Technical Report

References

- A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver. "*Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering*". In Proc. of the fourth ACM Recsys, 2010.
- S. Rendle, Z. Gantner, C. Freudenthaler, and L. Schmidt-Thieme. "*Fast context-aware recommendations with factorization machines*". In Proc. of the 34th ACM SIGIR, 2011.
- S.H. Yang, B. Long, A.J. Smola, H. Zha, and Z. Zheng. "*Collaborative competitive filtering: learning recommender using context of user choice*. In Proc. of the 34th ACM SIGIR, 2011.
- N. N. Liu, X. Meng, C. Liu, and Q. Yang. "*Wisdom of the better few: cold start recommendation via representative based rating elicitation*". In Proc. of RecSys'11, 2011.
- M. Jamali and M. Ester. "*Trustwalker: a random walk model for combining trust-based and item-based recommendation*". In Proc. of KDD '09, 2009.

References

- J. Noel, S. Sanner, K. Tran, P. Christen, L. Xie, E. V. Bonilla, E. Abbasnejad, and N. Della Penna. "*New objective functions for social collaborative filtering*". In Proc. of WWW '12, pages 859–868, 2012.
- X. Yang, H. Steck, Y. Guo, and Y. Liu. "*On top-k recommendation using social networks*". In Proc. of RecSys'12, 2012.
- Dmitry Lagun. "*Modeling User Attention and Interaction on the Web*" 2014 PhD Thesis (Emory Un.)
- *"Robust Models of Mouse Movement on Dynamic Web Search Results* Pages - F. Diaz et al. In Proc. of CIKM 2013
- Amr Ahmed et al. "*Fair and balanced".* In Proc. WSDM 2013
- Deepak Agarwal. 2013. *Recommending Items to Users: An Explore Exploit Perspective*. CIKM '13

Online resources

- Recsys Wiki: http://recsyswiki.com/
- Recsys conference Webpage: http://recsys.acm.org/
- Recommender Systems Books Webpage: http://www.recommenderbook.net/
- Mahout Project: http://mahout.apache.org/
- MyMediaLite Project: http://www.mymedialite.net/

Thanks!

Questions?

Xavier Amatriain

xavier@netflix.com

@xamat

NETFLIX