Learning to Rank for Recommender Systems

Alexandros Karatzoglou[a] , Linas Baltrunas[a], Yue Shi[b]

[a]Telefonica Research, Spain
[b]Delft University of Technology, Netherlands

Who are we?

Alexandros,
Yue
Linas

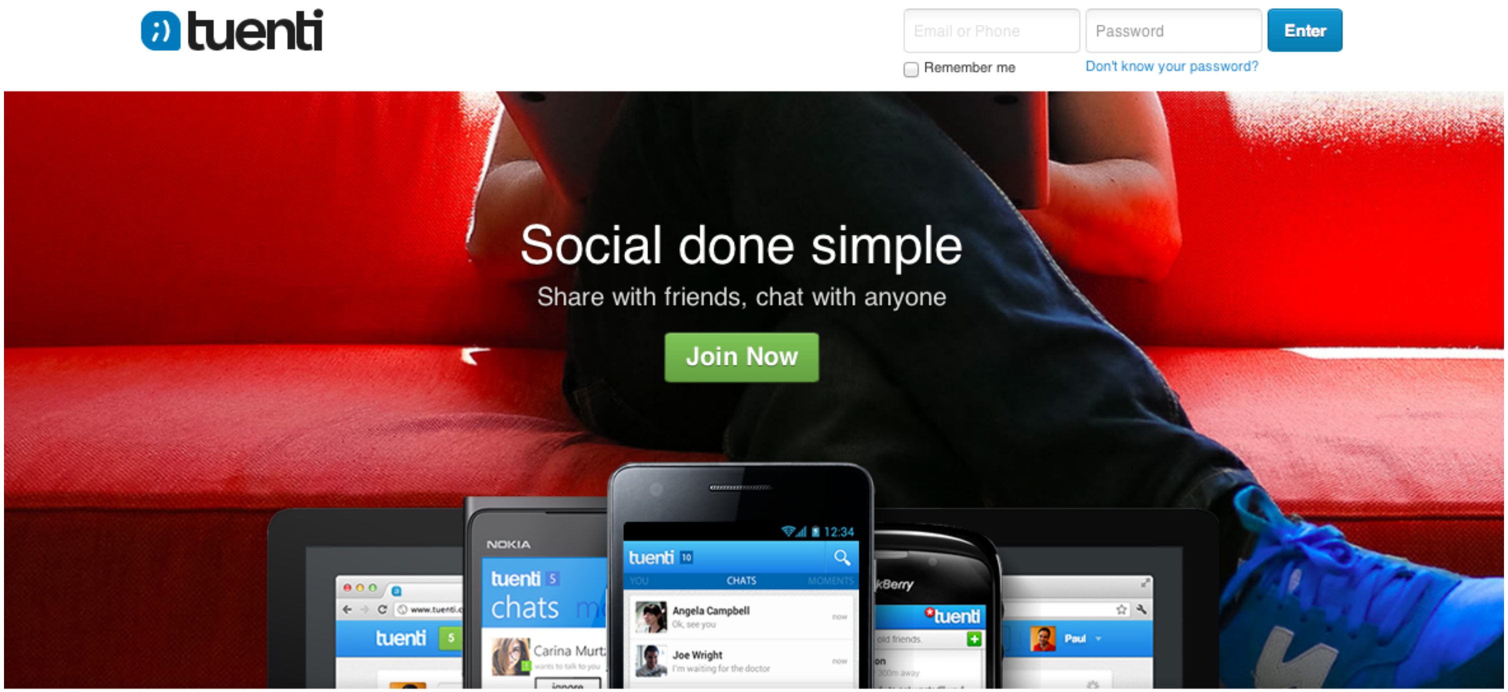
- Multimedia
- Recommender Systems
- Data Mining, Social Networks
- System & Networking

Telefónica
Telefónica Investigación y Desarrollo

- Machine Learning
- Recommender Systems
- Data Mining, Social Networks
- Multimedia Indexing & Analysis
- HCI
- System & Networking

- *We are looking for interns!*

Recommendations in Telefonica



Recommendations in Telefonica

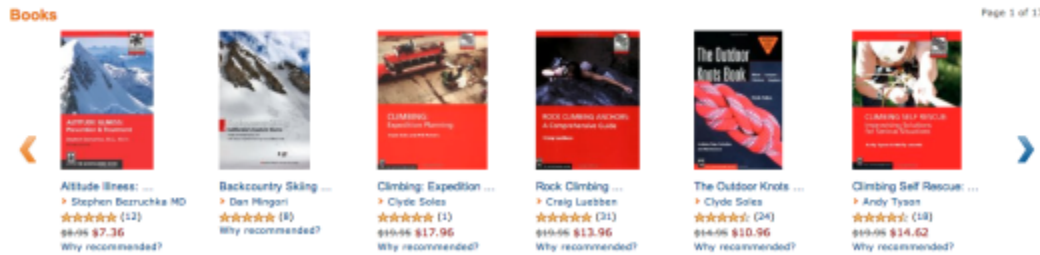
- 1 Million mobile apps NOW!
  - 48B downloads by 2015
- $10B market and growing!

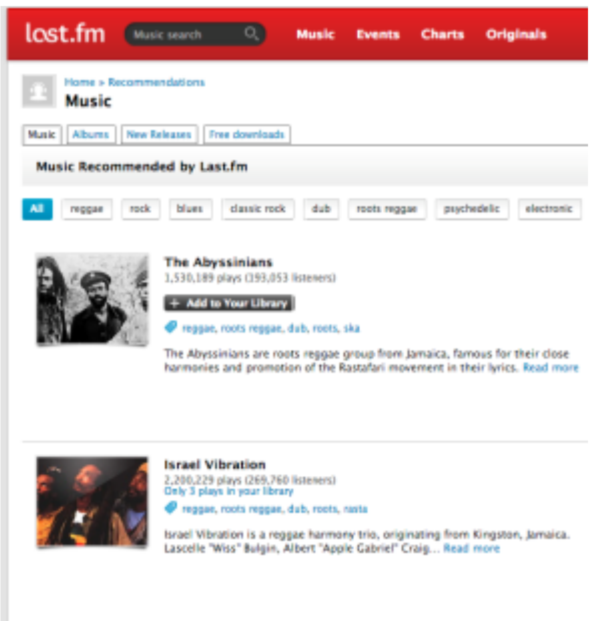
Recommendations in Telefonica

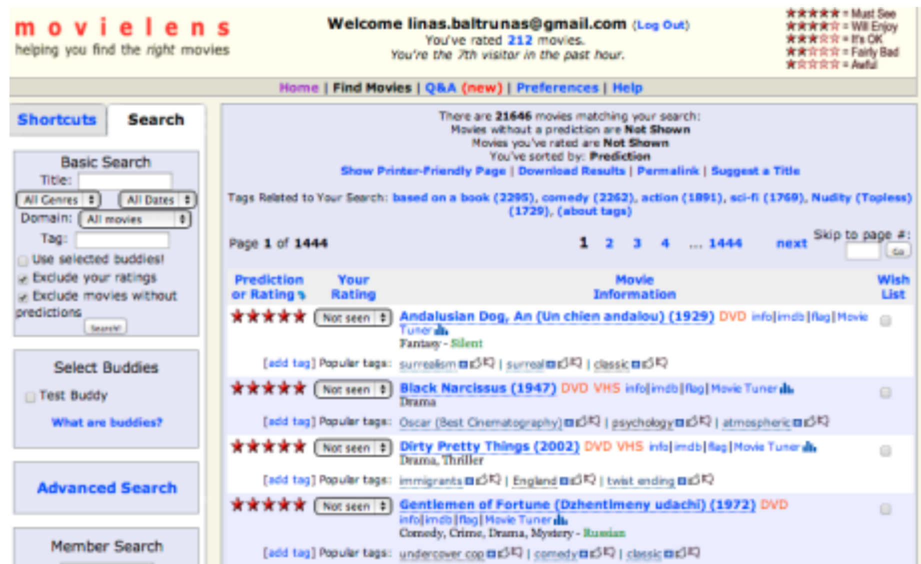
Previous Contributions in Ranking

CIKM 2013: *GAPfm: Optimal Top-N Recommendations for Graded Relevance Domains*
RecSys 2013: *xCLiMF: Optimizing Expected Reciprocal Rank for Data with Multiple Levels of Relevance*
*RecSys 2012: CLiMF: Learning to Maximize Reciprocal Rank with Collaborative Less-is-More Filtering * Best Paper Award*
SIGIR 2012: TFMAP: Optimizing MAP for Top-N Context-aware Recommendation
*Machine Learning Journal, 2008: Improving Maximum Margin Matrix Factorization * Best Machine Learning Paper Award at ECML PKDD 2008*
RecSys 2010: List-wise Learning to Rank with Matrix Factorization for Collaborative Filtering
NIPS 2007: CoFiRank - Maximum Margin Matrix Factorization for Collaborative Ranking

Recommendations are ranked lists







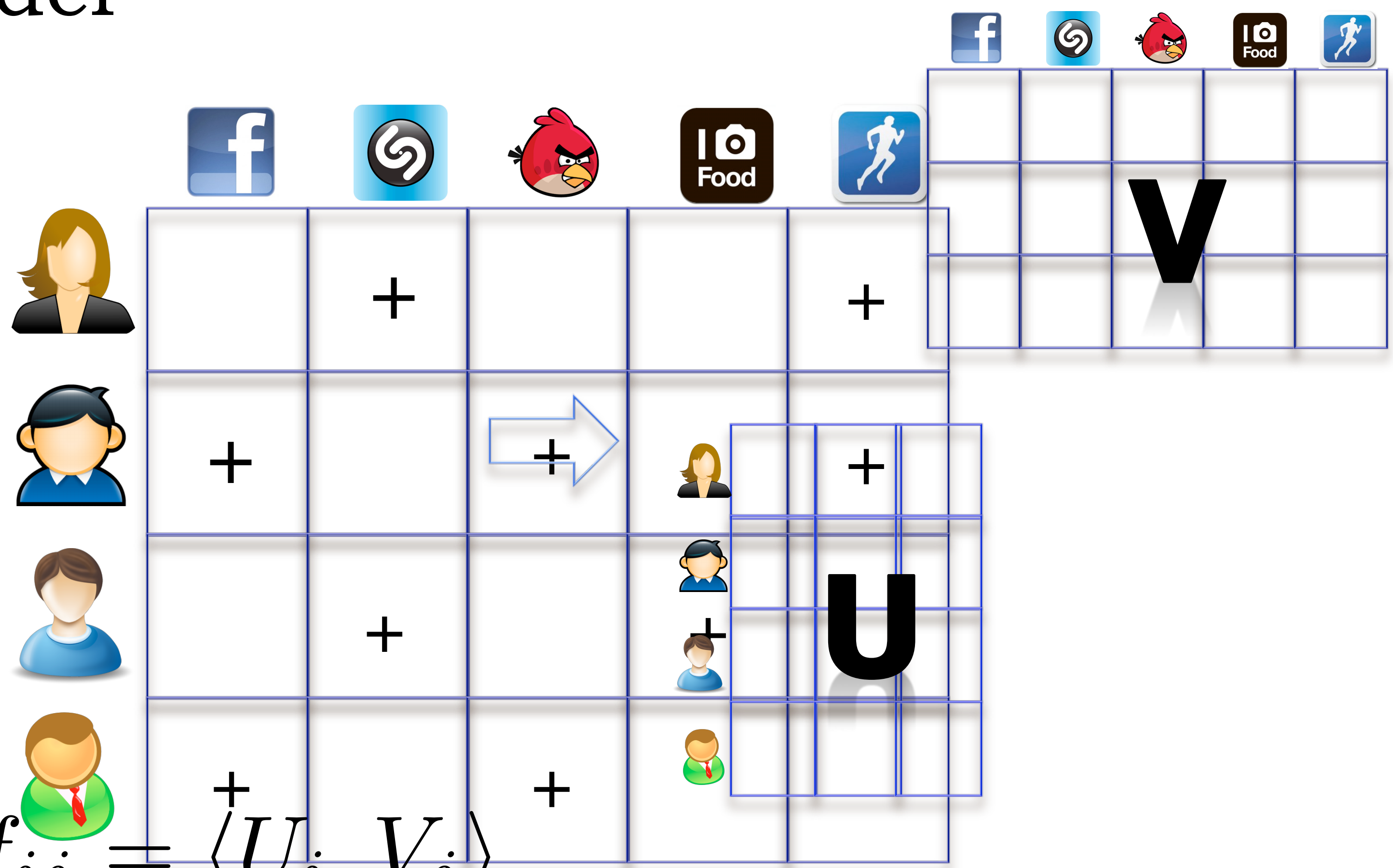
- **Intro to in Ranking**
  - Ranking measures
- Learning to Rank for Recommender Systems
  - Classification of approaches
- Trends and Challenges

INTRO TO RANKING

Popular Ranking Methods

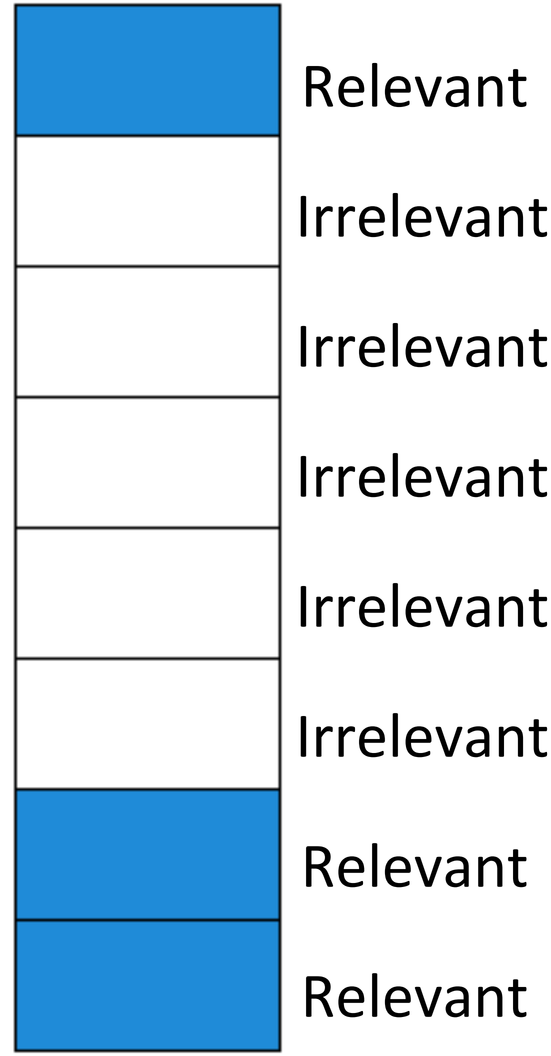
- In order to generate the ranked item list, we need some relative utility score for each item
  - Popularity is the obvious baseline
  - Score could depend on the user (personalized)
  - Score could also depend on the other items in the list (list-wise)
- One popular way to rank the items in RS is to sort the items according to the rating prediction
  - Works for the domains with ratings
  - Wastes the modeling power for the irrelevant items

Model



$$f_{ij} = \langle U_i, V_j \rangle$$

Graphical Notation



Ranking using latent representation

- If user = [-100, -100]
  - 2d latent factor
  - We get the corresponding ranking

5	[0.180,0.019]
4	[0.013,0.487]
1	[0.165,0.632]
4	[0.730,0.109]
2	[0.420,0.485]
1	[0.463,0.725]
2	[0.894,0.857]
1	[0.942,0.851]

Matrix Factorization (for ranking)

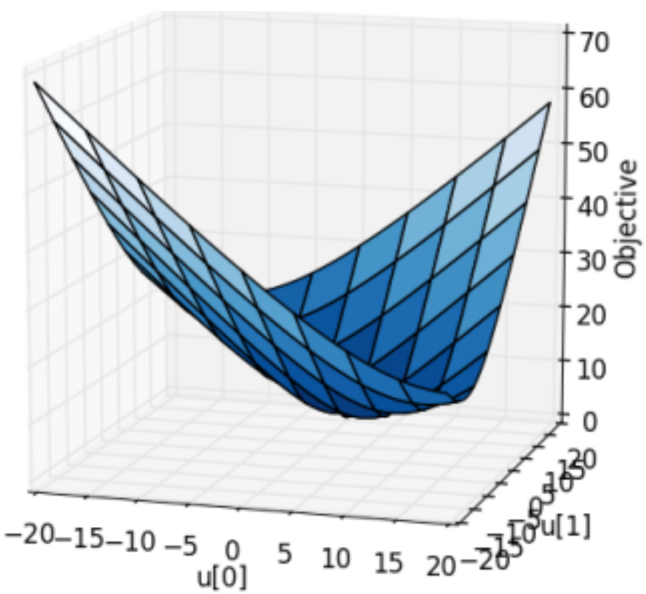
- Randomly initialize item vectors
- Randomly initialize user vectors
- While not converged
  - Compute rating prediction error
  - Update user factors
  - Update item factors

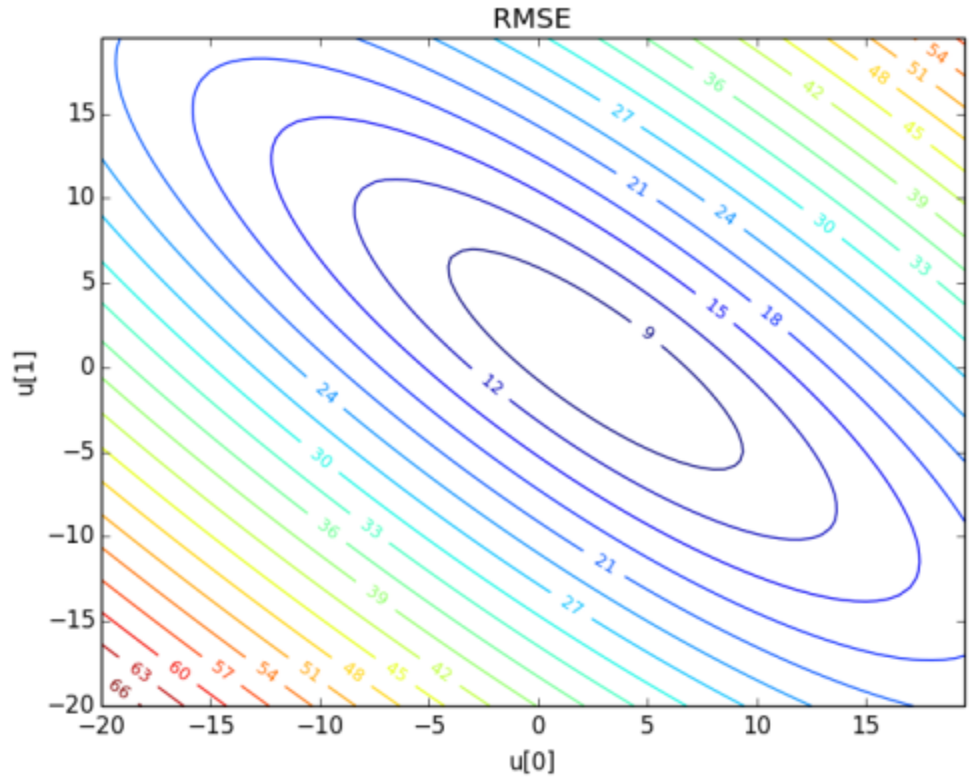
- Lets say user is [-100, -100]
  - Compute the square error
    - $(5-<[-100, -100], [0.180, 0.19]>)^2=1764$
  - Update the user and item to the direction where the error is reduced (according to the gradient of the loss)

8 items with ratings and random factors

2	[0.894,0.857]
1	[0.463,0.725]
1	[0.942,0.851]
4	[0.730,0.109]
4	[0.013,0.487]
1	[0.165,0.632]
$\boxed{5}$	[0.180,0.019]
2	[0.420,0.485]

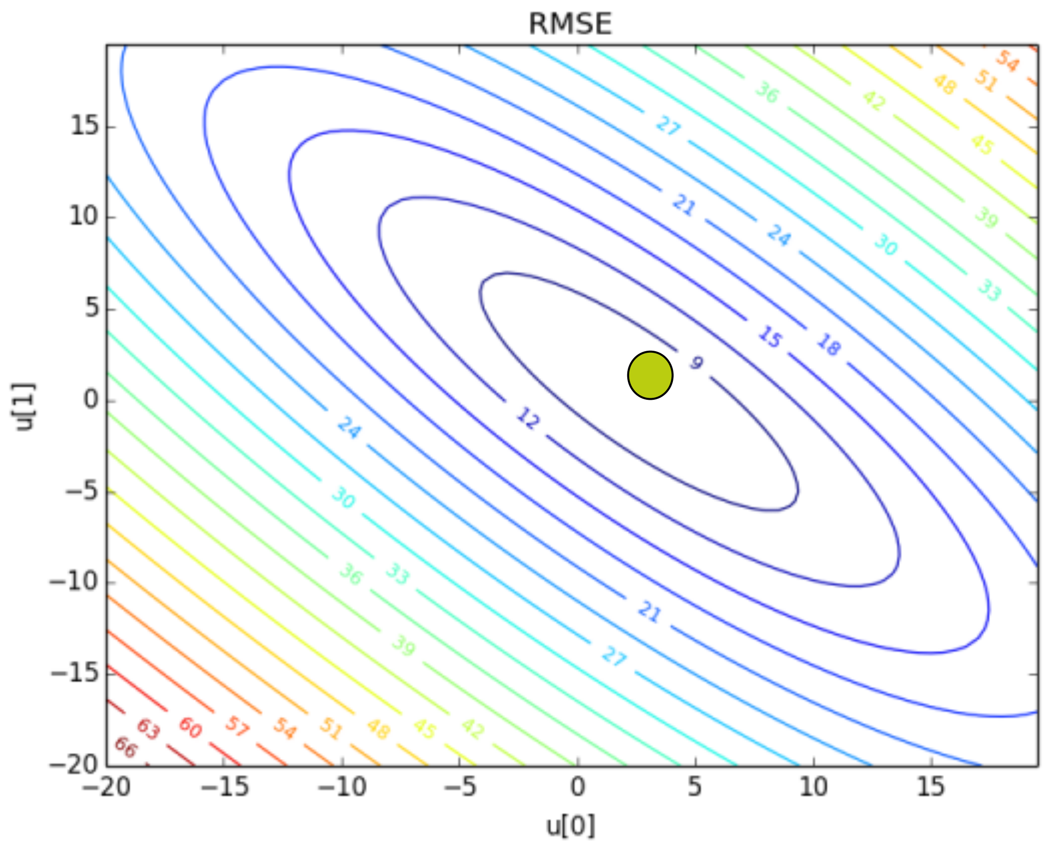
Learning: Stochastic Gradient Descent with Square Loss





Square Loss

User: [3, 1], RMSE=6.7



1	[0.942,0.851]
2	[0.894,0.857]
4	[0.730,0.109]
1	[0.463,0.725]
2	[0.420,0.485]
1	[0.165,0.632]
5	[0.180,0.019]
4	[0.013,0.487]

Learning to Rank for Top-k RecSys

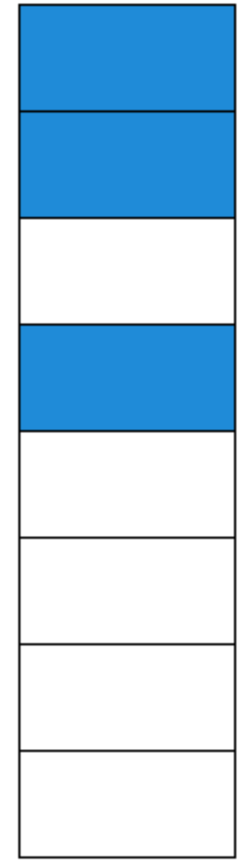
- Usually we care to make accurate ranking and not rating prediction
  - Square loss optimizes to accurately predict 1s and 5s.
- RS should get the top items right -> Ranking problem
- Why not to learn how to rank directly?
  - Learning to Rank methods provide up to 30% performance improvements in off-line evaluations
  - It is possible, but a more complex task

Example: average precision (AP)

- AP: we compute the precision at each relevant position and average them

$$\mathrm{AP} = \frac{\sum_{k=1}^{|S|} P(k)}{|S|}$$

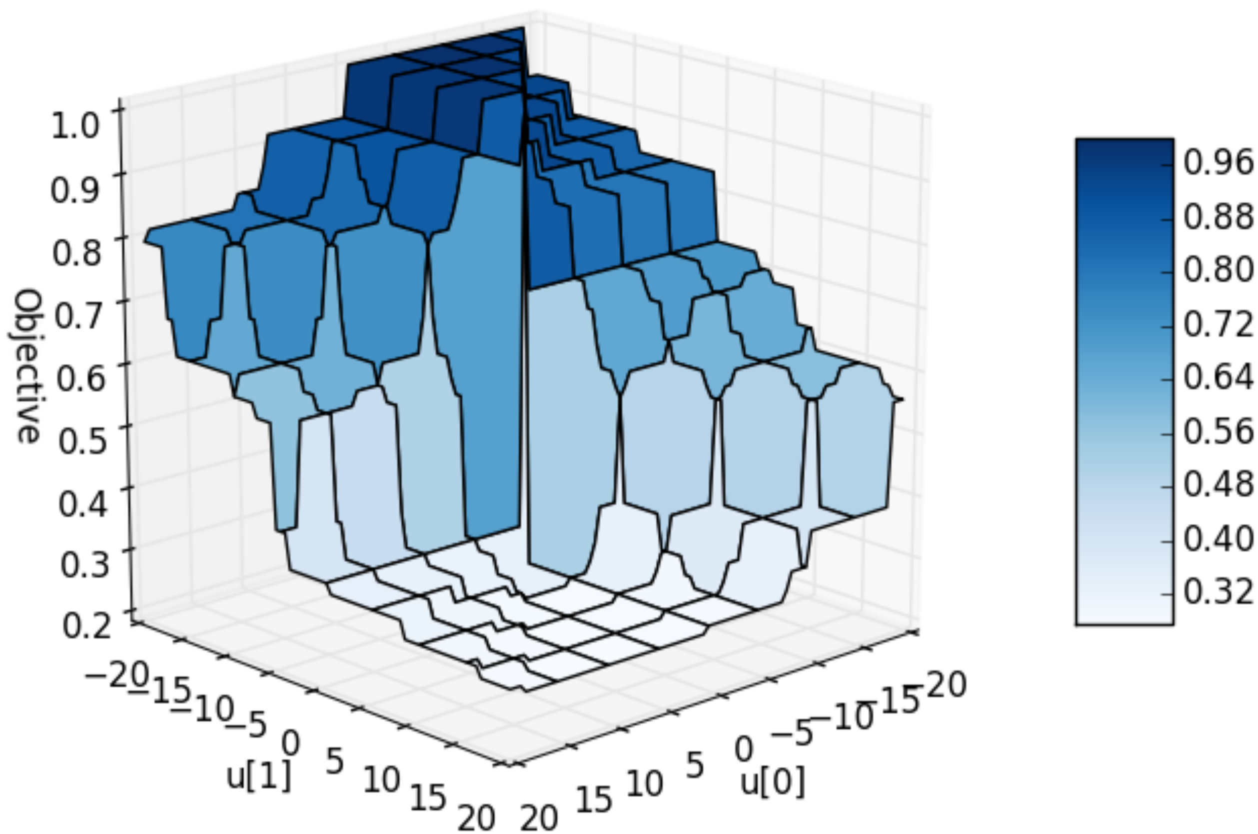
$$\frac{P@1 + P@2 + P@4}{3} = \frac{1/1 + 2/2 + 3/4}{3} = 0.92$$



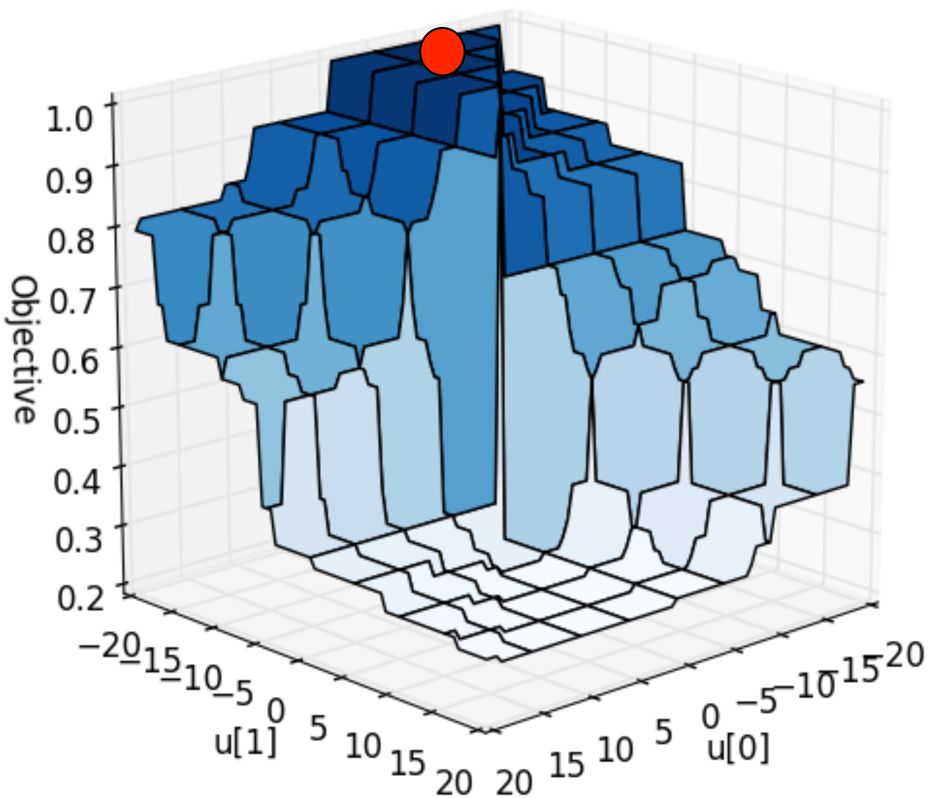
Why is hard? Non Smoothness
Example: AP

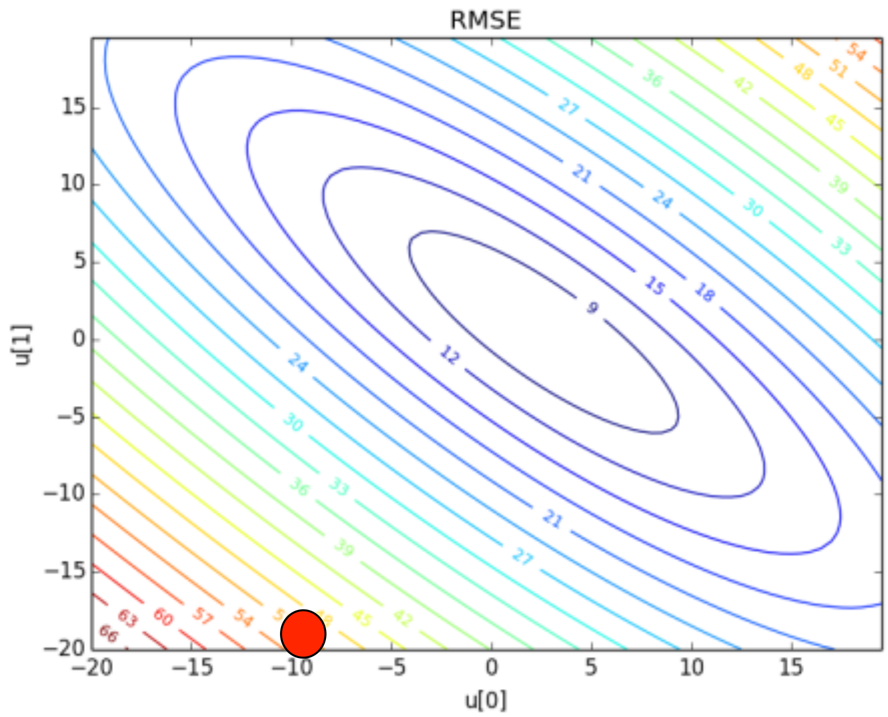
u:[-20,-20]

u:[20,20]



AP vs RMSE





The Non-smoothness of Average Precision

$$\mathrm{AP} = \frac{\sum_{k=1}^{|S|} P(k)}{|S|}$$

$$AP_m = \frac{1}{\sum_{i=1}^{N} y_{mi}} \sum_{i=1}^{N} \frac{y_{mi}}{r_{mi}} \sum_{j=1}^{N} y_{mj} \mathbb{I}(r_{mj} \leq r_{mi})$$

y_{mi} is 1 if item i is relevant for user *m* and 0 otherwise

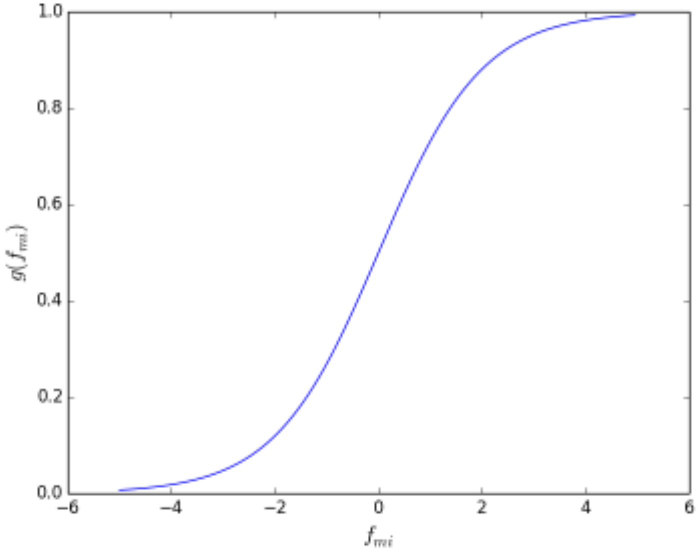
$\mathbb{I}(\cdot)$ indicator function (1 if it is true, 0 otherwise)

r_{mi} Rank of item *i* for user *m*

How can we get a smooth-AP?

- We replace non smooth places of MAP with smooth approximation

$$\frac{1}{r_{mi}} \approx g(f_{mi}) = g(\langle U_m, V_i \rangle)$$

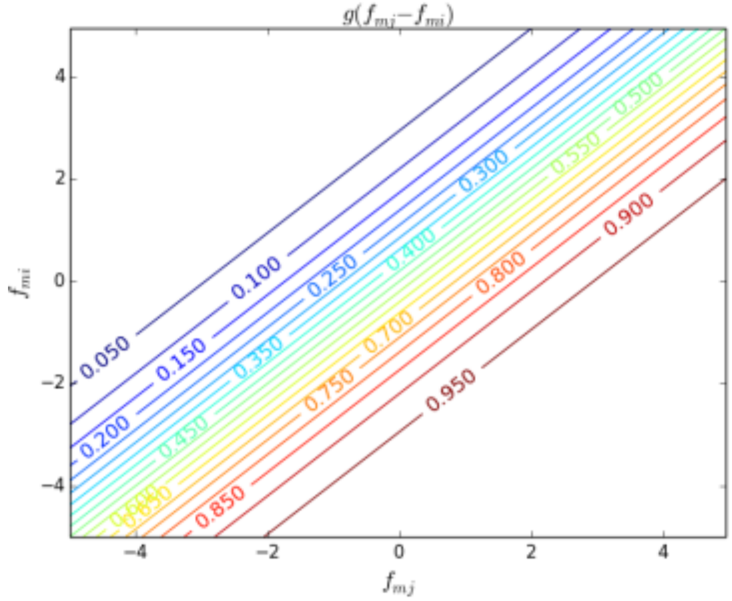


$$g(x) = 1/(1 + e^{-x})$$

How can we get a smooth-MAP?

- We replace non smooth places of MAP with smooth approximation

$$\mathbb{I}(r_{mj} \leq r_{mi}) \approx g(f_{mj} - f_{mi}) = g(\langle U_m, V_j - V_i \rangle)$$

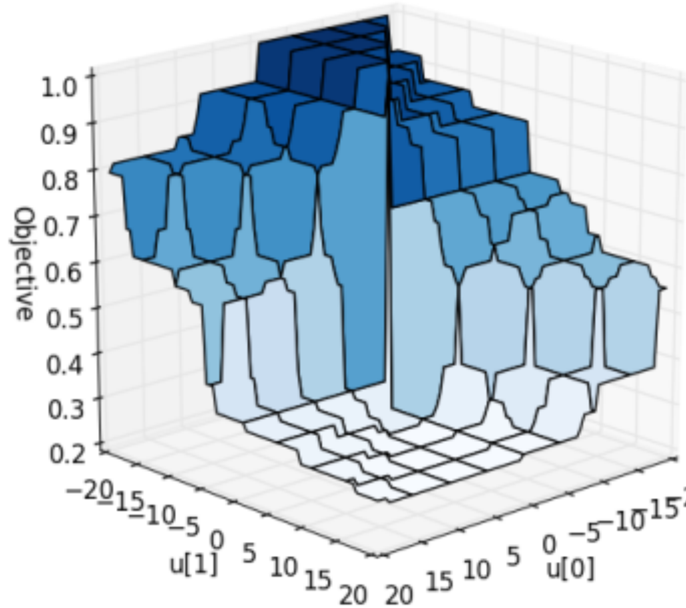


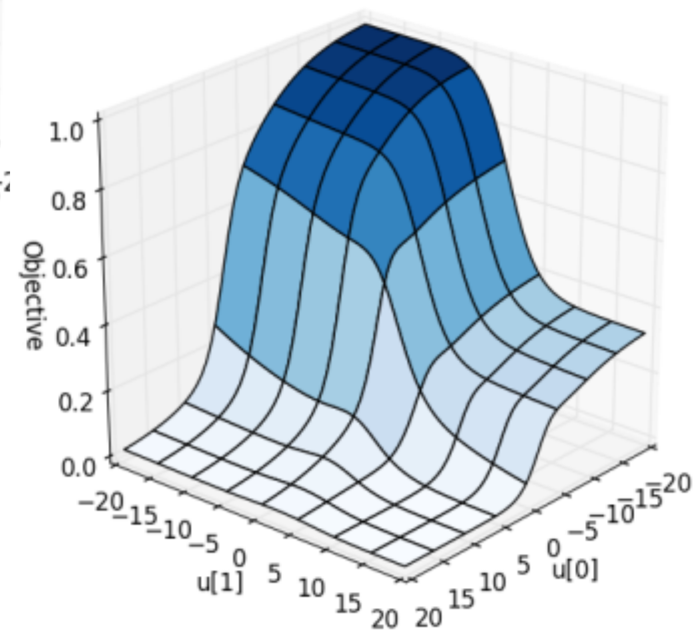
$$g(x) = 1/(1 + e^{-x})$$

Smooth version of MAP

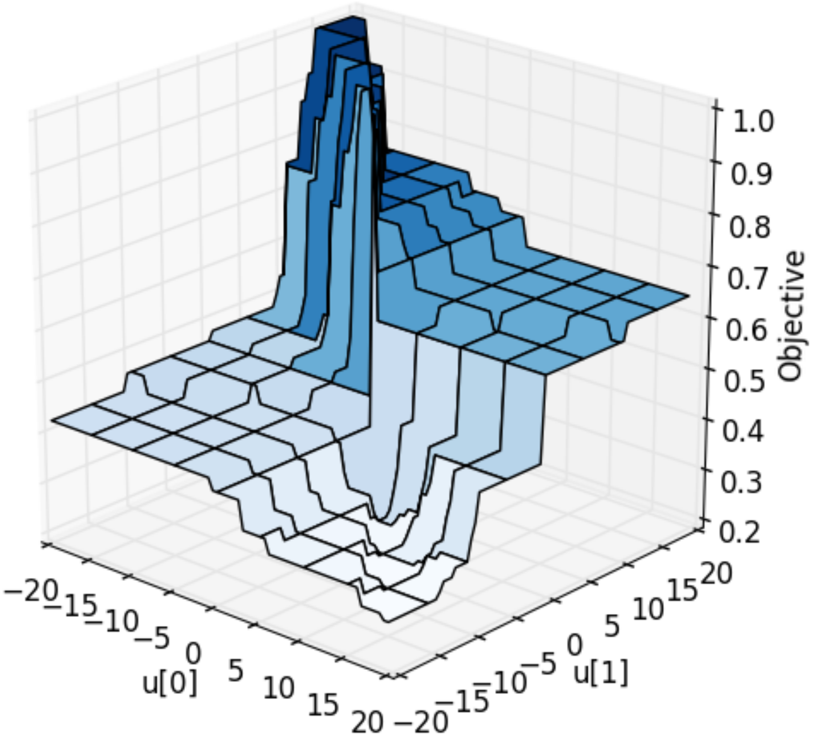
u:[-20,-20]

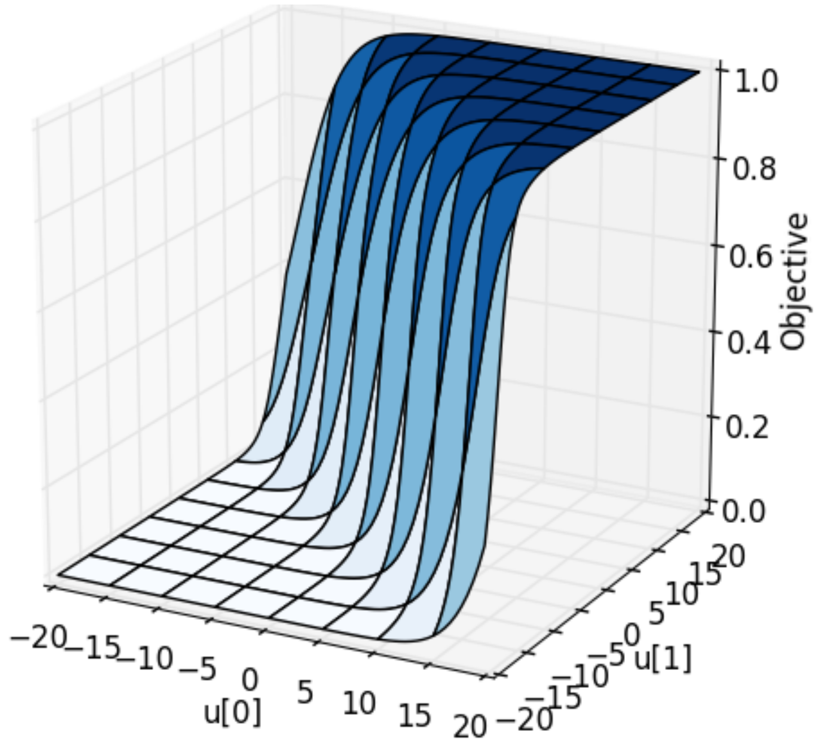
u:[20,20]





Not always the case:
only sometimes approximation is very good…

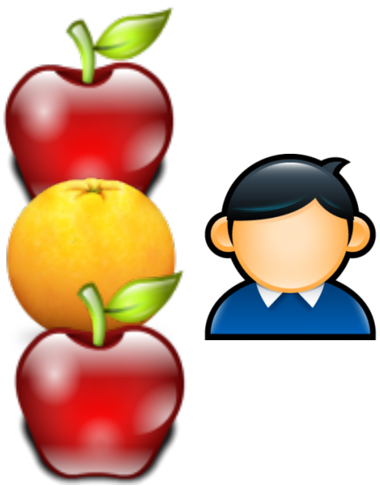


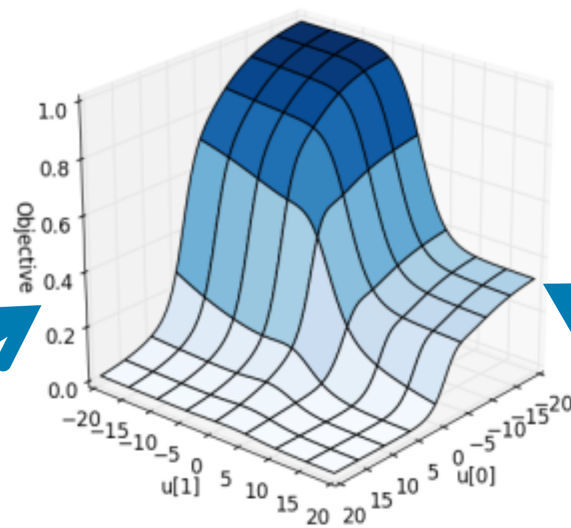


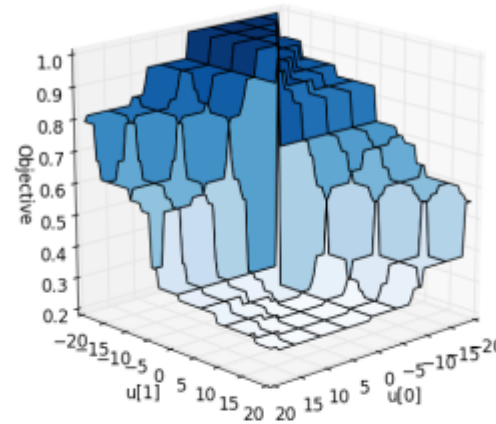
Ranking Inconsistencies

- Achieving a perfect ranking for all users is not possible
- Two Sources of Inconsistencies:
- 1) Factor Models (all models) have limited expressive power and cannot learn the perfect ranking for all users
- 2) Ranking functions approximations are inconsistent e.g.
  A >B & B>C but C > A

Summary on Ranking 101



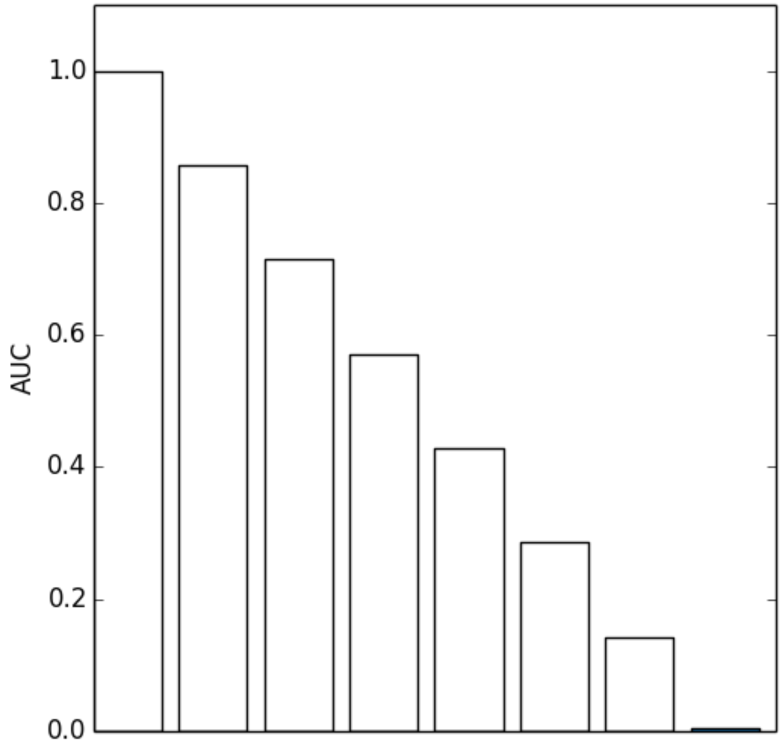




- Intro to in Ranking
  - **Ranking measures**
- Learning to Rank for Recommender Systems
  - Classification of approaches
- Trends and Challenges

RANKING METRICS

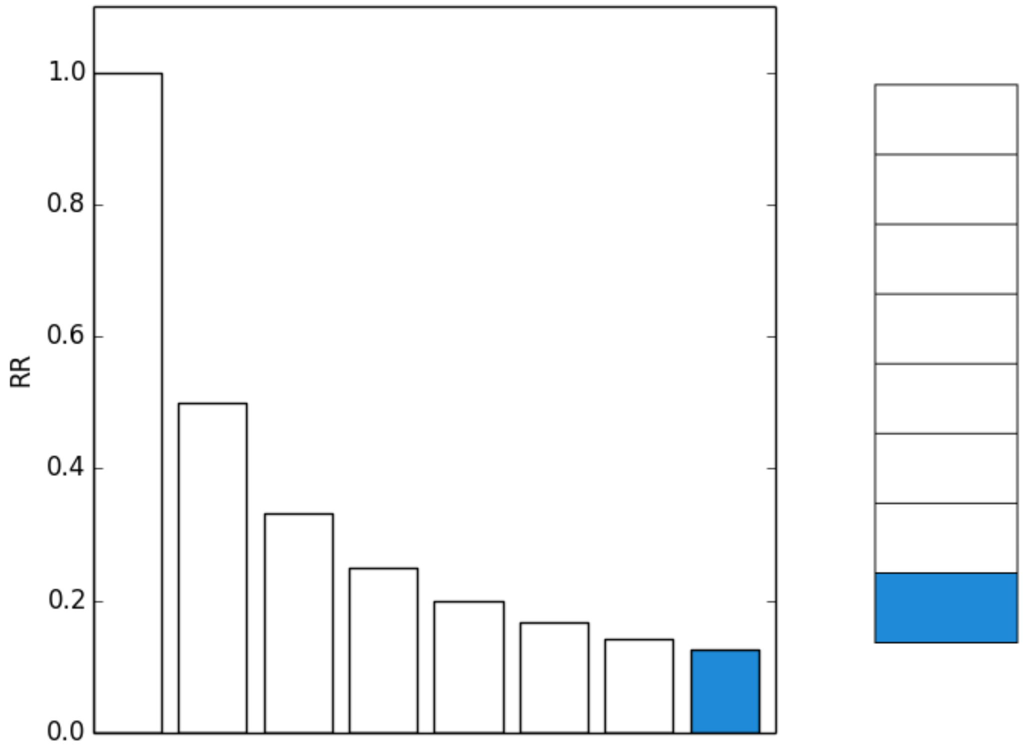
Area Under the ROC Curve (AUC)



$$AUC := \frac{1}{|S^+||S^-|} \sum_{i}^{S^+} \sum_{j}^{S^-} \mathcal{I}(R_i < R_j)$$

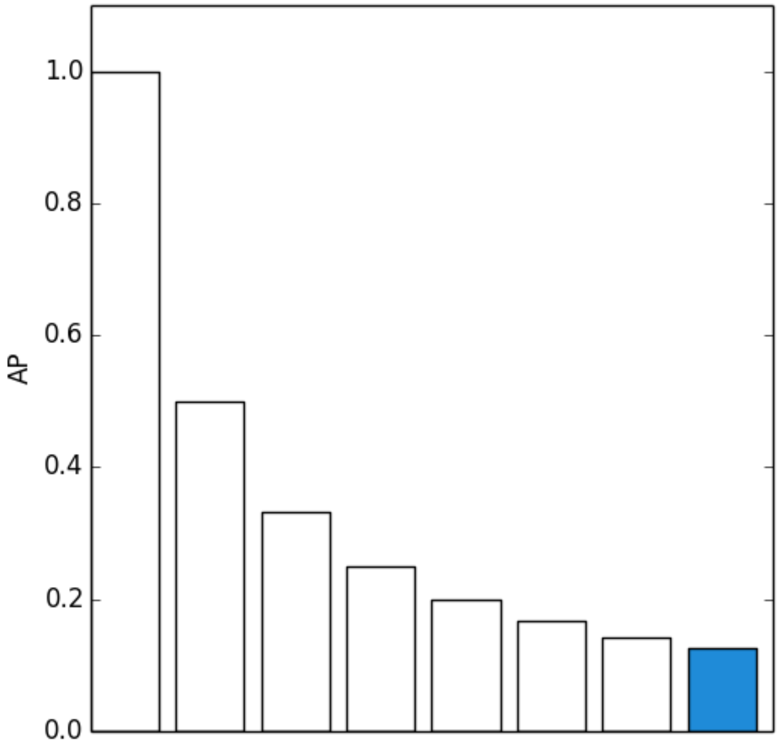
Reciprocal Rank (RR)

$$RR := \frac{1}{R_i}$$

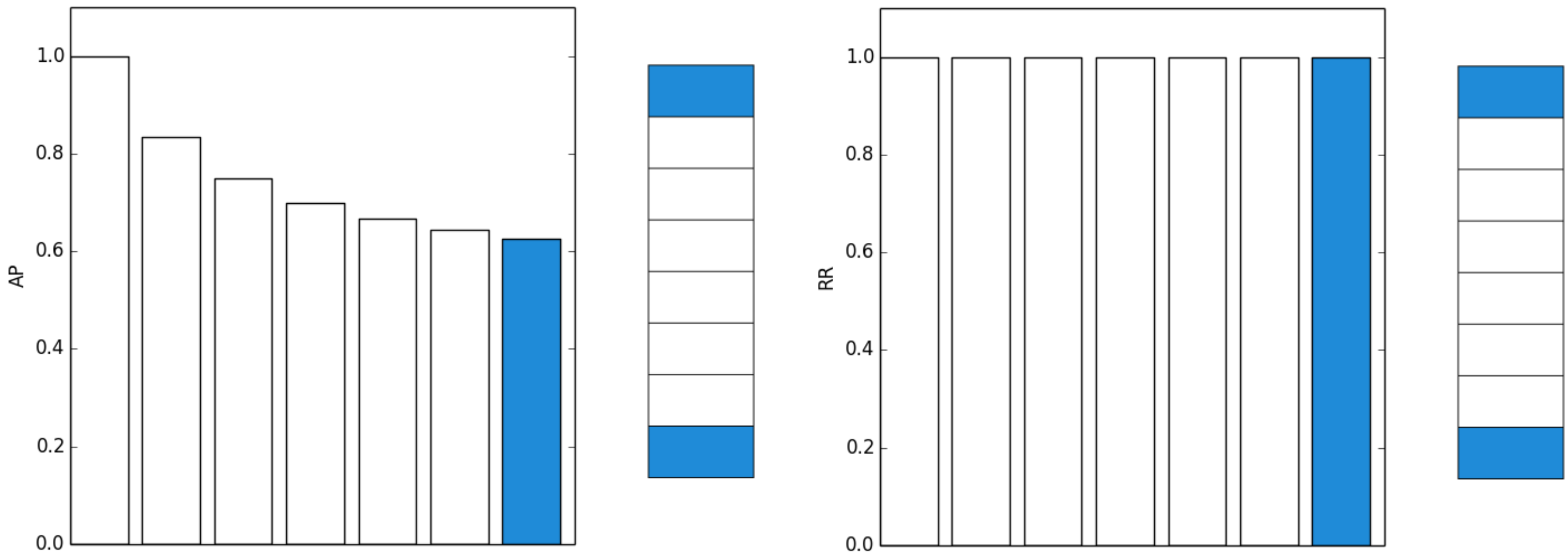


Average Precision (AP)

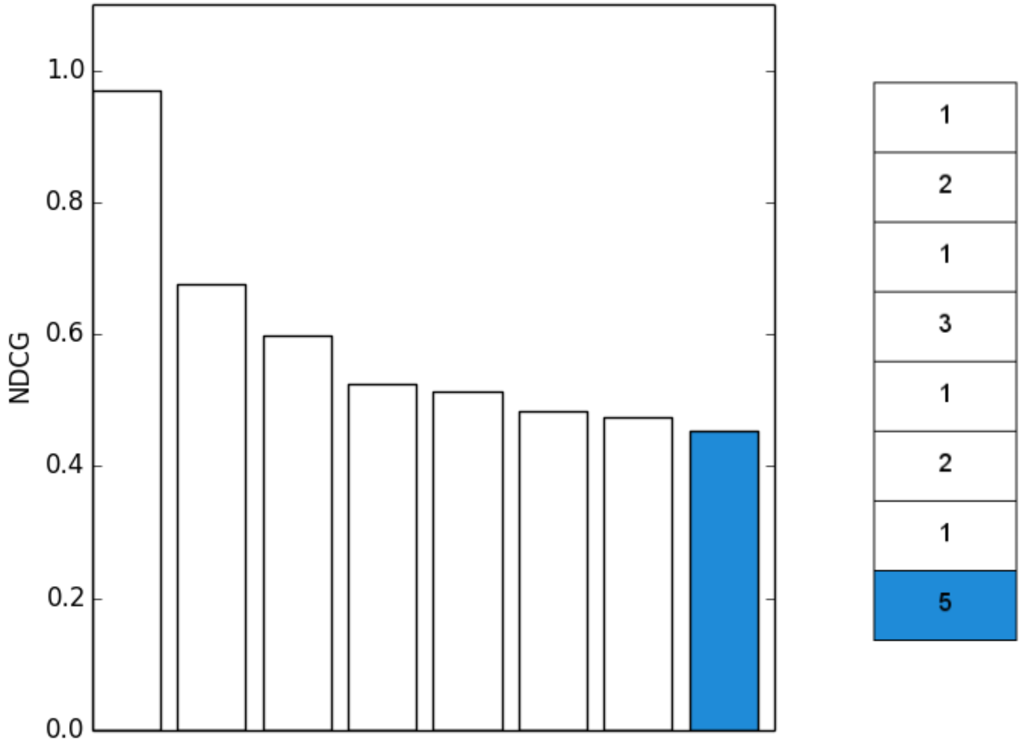
$$\mathrm{AP} = \frac{\sum_{k=1}^{|S|} P(k)}{|S|}$$



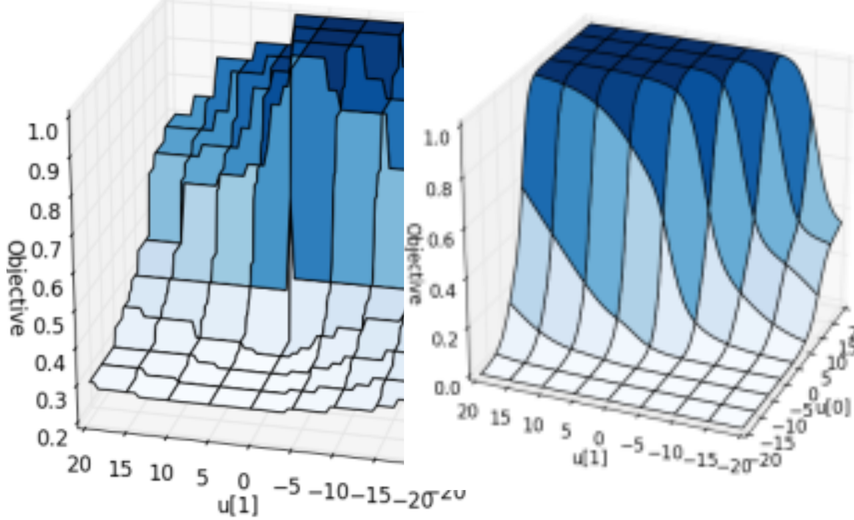
AP vs RR



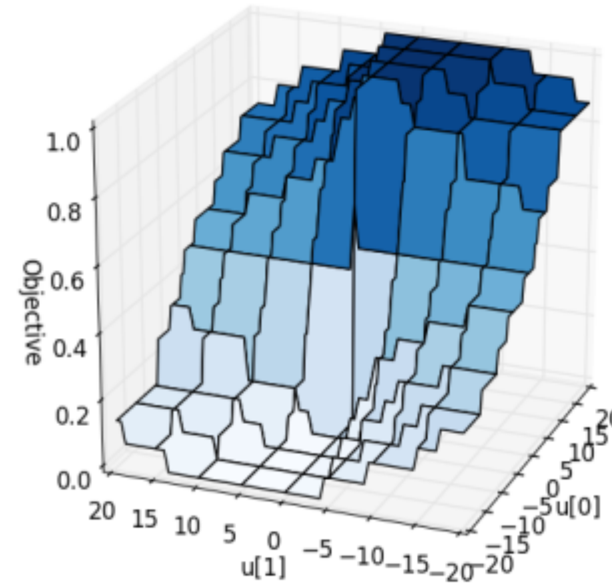
Normalized Discounted Cumulative Gain (nDCG)



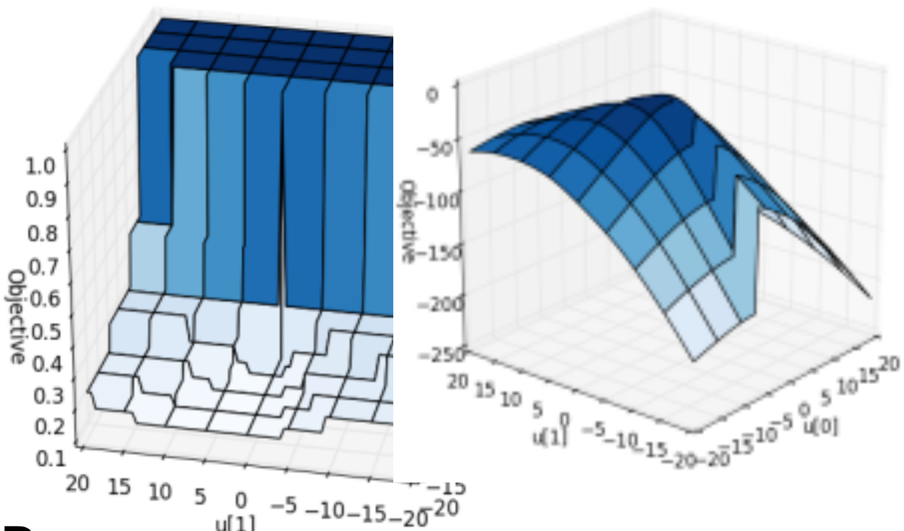
$$DCG = \sum_i \frac{2^{score(i)} - 1}{log_2(i+2)}$$



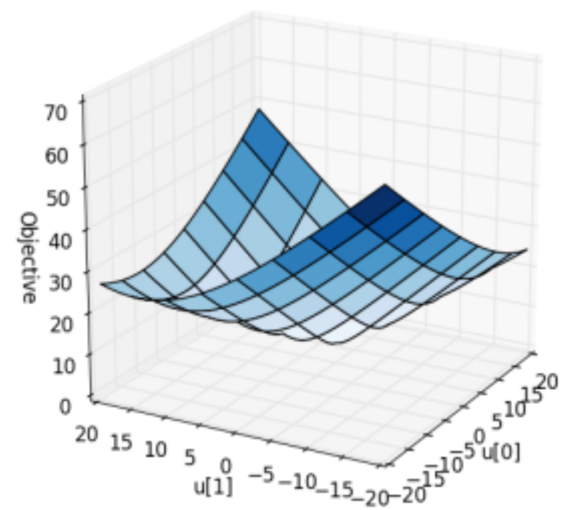
AP



AUC



RR



Square loss

Top-k Ranking

- Focus on the very top of the recommendation list since users pay attention only to the first k items
- Top heavy ranking measures put much more emphasis at the top of the list e.g. MRR, MAP, NDCG.
  - Drop in measure non-linear with the position in the list
- AUC gives the same emphasis to the top as to the bottom of the list
  - drop in measure linear to the position in the list

- Intro to in Ranking
  - Ranking measures
- **Learning to Rank for Recommender Systems**
  - Classification of approaches
- Trends and Challenges

LEARNING TO RANK FOR RECSYS

Learning to Rank for IR and ML

- Query-document (Q-D) retrieval: Given a query, rank documents.

(features of a Q-D pair, relevance label)

↓

Learn a ranking function

↓

Predict relevances of new Q-D pairs

RankSVM
LambdaRank
RankNet
RankBoost
AdaRank
ListNet
SVM-MAP
SoftRank
…

Analogy: Query-doc --- User-item

Learning to Rank in CF

- The Point-wise Approach $f(user, item) \rightarrow \mathbb{R}$
  - Reduce Ranking to Regression, Classification, or Ordinal Regression problem
- The Pairwise Approach $f(user, item_1, item_2) \rightarrow \mathbb{R}$
  - Reduce Ranking to pair-wise classification
- List-wise Approach $f(user, item_1, \ldots, item_n) \rightarrow \mathbb{R}$
  - Direct optimization of IR measures, List-wise loss minimization

Overview

Classification by *What to Model*

Pointwise	Pairwise	Listwise
Matrix factorization [Koren 2009]	BPR [Rendle 2009]	CofiRank [Weimer 2007]
SVD++ [Koren 2008]	EigenRank [Liu 2008]	ListRank [Shi 2010]
OrdRec [Koren 2011]	pLPA [Liu 2009]	WLT [Volkovs 2012]
Factorization machines [Rendle 2012]	CR [Balakrishnan 2012]	TFMAP [Shi 2012a]
(All rating prediction methods)		CLiMF [Shi 2012b]
		GAPfm [Shi 2013a]
		xCLiMF [Shi 2013b]

Overview

Classification by *How to Model*

Proxy of rankings	Structured estimation	Non-smooth optimization	Smoothing ranking measures
(All the rating prediction methods)	CofiRank [Weimer 2007]	WLT [Volkovs 2012]	BPR [Rendle 2009]
EigenRank [Liu 2008]			TFMAP [Shi 2012a]
pLPA [Liu 2009]			CLiMF [Shi 2012b]
ListRank [Shi 2010]			GAPfm [Shi 2013a]
CR [Balakrishnan 2012]			xCLiMF [Shi 2013b]

Pointwise

Standard Matrix Factorization [Koren 2009]

Learn latent factors of users (U) and latent factors of items (V), so that the product of U and V can fit the known data.

The ratings of unseen items predicted by corresponding inner products

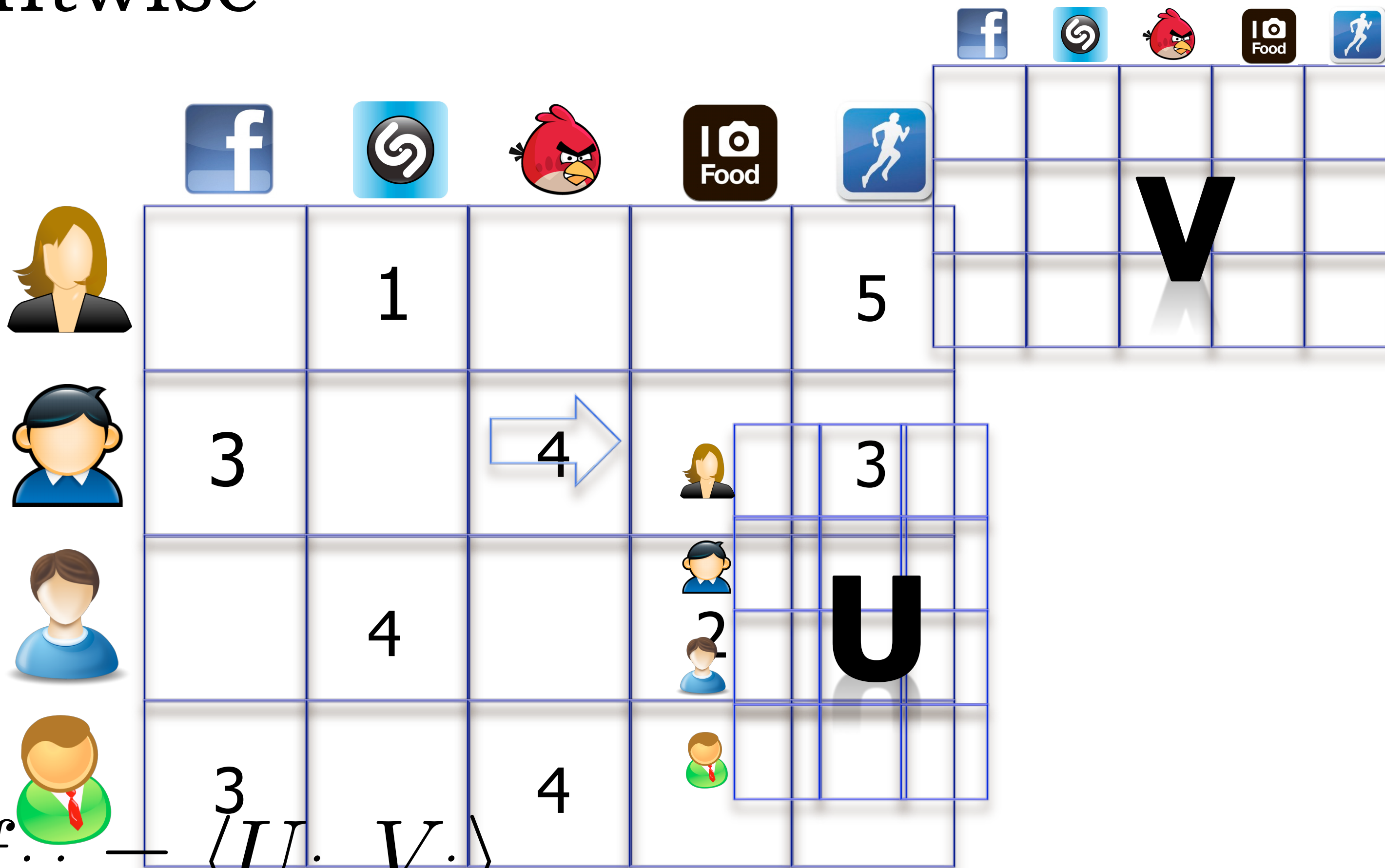
	i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8
u_1	1	1	1	2	3	5	?	2
u_2	4	?	3	?	5	1	1	4
u_3	3	3	?	4	?	3	3	3
u_4	?	1	?	5	5	3	?	1

$\sim$ U (4×D) × V (D×8)

Each user represented by a D-dim vector

Each item represented by a D-dim vector

Pointwise



$$f_{ij} = \langle U_i, V_j \rangle$$

$$l(f_{ij}, y_{ij}) = \frac{1}{2}(f_{ij} - y_{ij})^2$$

Ordrec [Koren 2011]

Ratings are not numeric, but ordinal.
Ordinal modeling by learning the thresholds

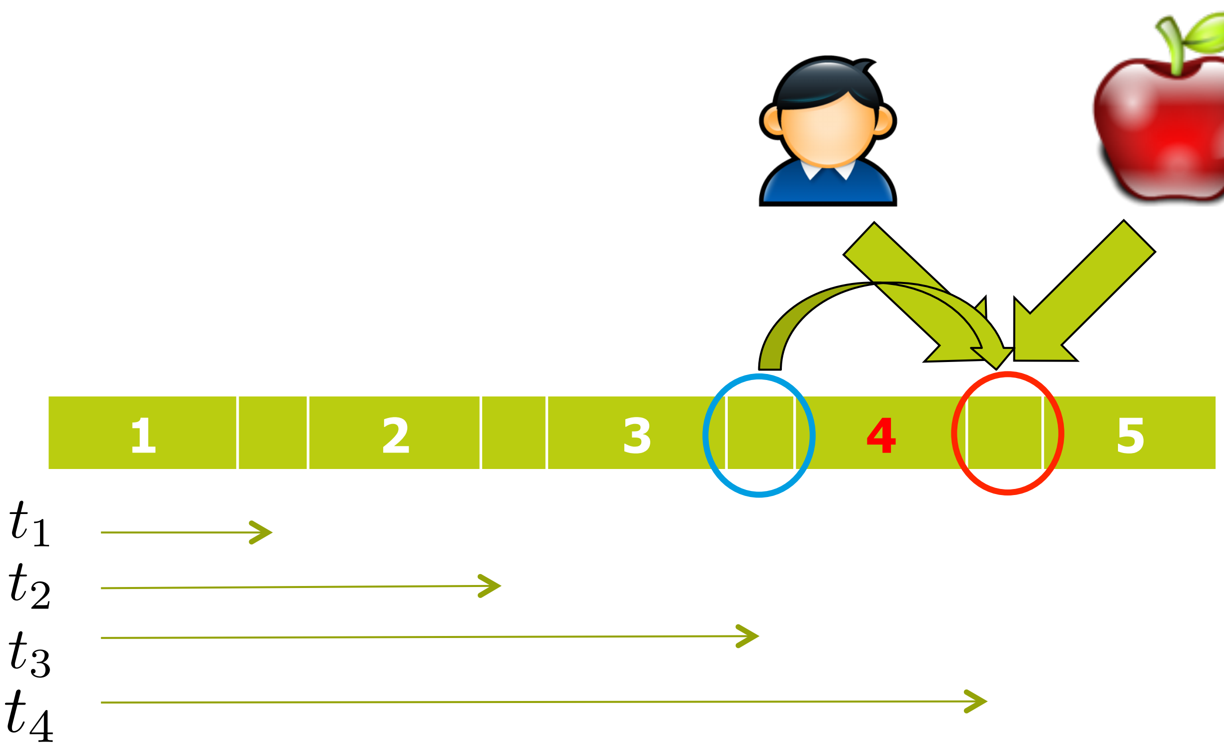
Rating
1
2
3
4
5

Item with rating 4 should be less similar to that with rating 1

Item with rating 4 should be more similar to that with rating 5

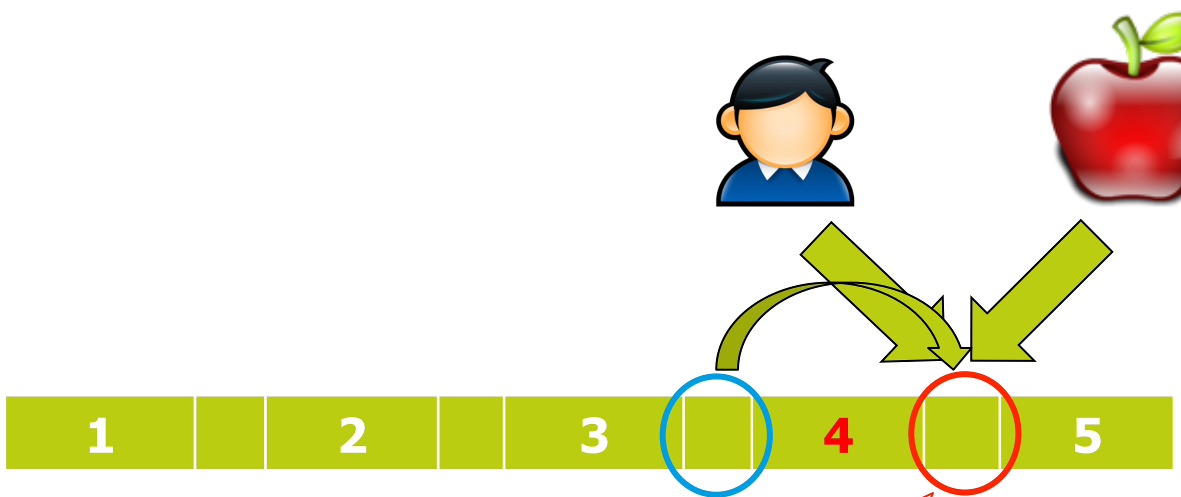
OrdRec [Koren 2011]

Pointwise, proxy



$$\log\left(\frac{\gamma_s}{1-\gamma_s}\right) = t_s - \langle U_i, M_j \rangle \qquad P(\leq s) = \gamma_s$$

OrdRec [Koren 2011]



Parameterize the threshold into:
User factor, item factor and
previous threshold

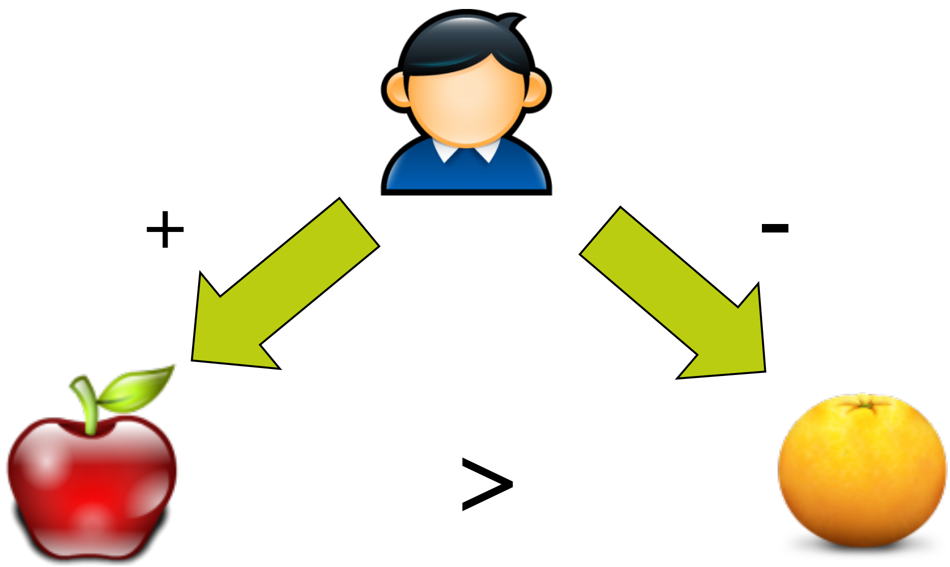
Full distribution estimation

BPR [Rendle 2009]
Pairwise, smoothing

Main idea: learn by comparing items that the user likes with items that he does not.

When dealing with implicit data sample the unseen data as negative

Optimize the AUC, use a smooth version for Optimization purposes



BPR [Rendle 2009]

Pairwise, smoothing

Representation of pairwise preference data

What Alice watched

Transformers (T)	?
Inception (I)	?
Scent of a woman (S)	1
Forrest Gump (F)	1
Zombieland (Z)	?

What Alice's pairwise preferences (Row to Column)

	T	I	S	F	Z
T		?	-	-	?
I	?		-	-	?
S	+	+		?	+
F	+	+	?		+
Z	?	?	-	-	

EigenRank [Liu 2008]

Pairwise, proxy

Ranking-based similarity instead of rating based similarity

	i1	i2	i3
u1	2	3	4
u2	4	2	5

User-user similarity based on *Kendall Rank Correlation*

Neighbour selection for a user and item preference estimation

Random walk with restart over pairwise item preference graph

Listwise

Learn to model individual ratings

Learn to model **ranked list**, or to model **evaluation metrics** that capture the quality of ranked list

CofiRank
ListRank
CLiMF
TFMAP
GAPfm
xCLiMF

CoFiRank [Weimer 2007]
Listwise, structured est.

- Based on Structured estimation i.e. Machine Learning methods for complex output domains (graphs, sequences, etc.)
- Cast the Ranking problem as a Structured estimation problem
- i.e. Learn a function that is maximized for the best possible ranking with respect to a ranking measure e.g. NDCG

CoFiRank [Weimer 2007]
Listwise, structured est.

- Step 1:
  - Instead of maximizing $$\mathrm{DCG@k}(\pi, y) = \sum_{i=1}^{k} \frac{(2^{y_i} - 1)}{\log(\pi_i + 1)}$$
  - Minimize $$\Delta(\pi, y) := 1 - \mathrm{NDCG}(\pi, y)$$

CoFiRank [Weimer 2007]

Listwise, structured est.

- Step 2:
  - Use the $\langle a, b \rangle \leq \langle \mathrm{sort}(a), \mathrm{sort}(b) \rangle \; \forall a, b \in R^n$
  - Create a linear mapping $\psi(\pi, f) := \langle c, f_\pi \rangle$
  - that is maximized for $\pi = \mathrm{argsort}(f)$
  - denote by a decreasing nonnegative sequence

CoFiRank [Weimer 2007]
Listwise, structured est.

- Step 3:
  - Find convex upper on non-convex optimization problems: Let

$$l(f, y) := \max_{\pi} \left[\Delta(\pi, y) + \langle c, f_\pi - f \rangle \right]$$

Maximization by solving a Linear Assignment Problem to get an estimate for the worst case loss (maximum margin)

CoFiRank [Weimer 2007]

	★★	★★★★	★	★★★
f	1.6	2.6	2.9	1.9

$$l(f, y) := \max_{\pi} \left[\Delta(\pi, y) + \langle c, f_\pi - f \rangle \right]$$

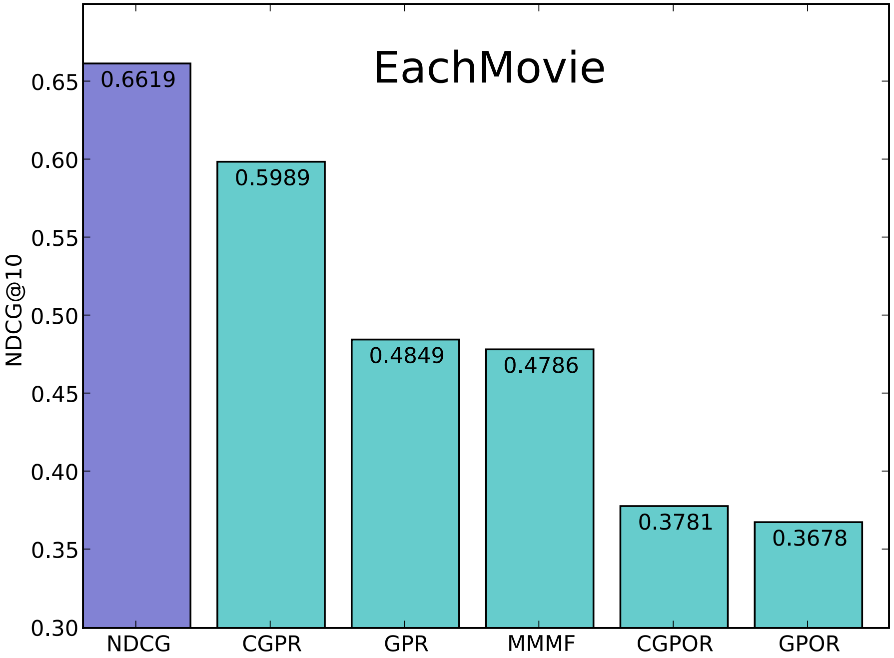
	★	★★★★	★★★	★★
f_π	2.9	2.6	1.9	1.6

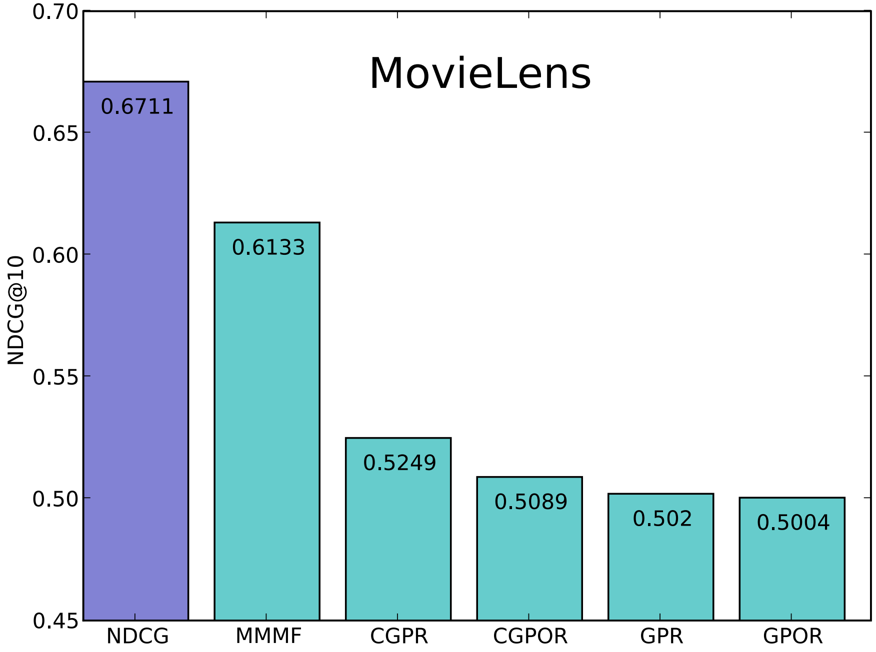
$\pi :=$ [3, 2 ,4 ,1] $\quad \pi^{-1} :=$ [4, 2 ,1 ,3]

$c := (i+1)^{-0.5} :=$ [1, 0.7 ,0.57 ,0.5]

$\partial_f l(f, y) = \left[c - c_{\pi^{-1}} \right] :=$ [-0.5, 0.0 ,0.42 ,0.1]

CoFiRank [Weimer 2007]





CLiMF [Shi 2012b]

Listwise, Smoothing Ranking Metrics

- Why optimize mean reciprocal rank (MRR)?
  - Focus at the very top of the list
  - Try to get at least one interesting item at the top of the list
- How to optimize MRR?
  - Find a smooth version of MRR
  - Find a lower bound of the smoothed MRR

The Non-smoothness of Reciprocal Rank

- Reciprocal Rank (RR): The inverse of the rank of the first relevant item in a given list.

RR$_i$

$$RR_i = \sum_{j=1}^{N} \frac{Y_{ij}}{R_{ij}} \prod_{k=1}^{N} (1 - Y_{ik}\mathbb{I}(R_{ik} < R_{ij}))$$

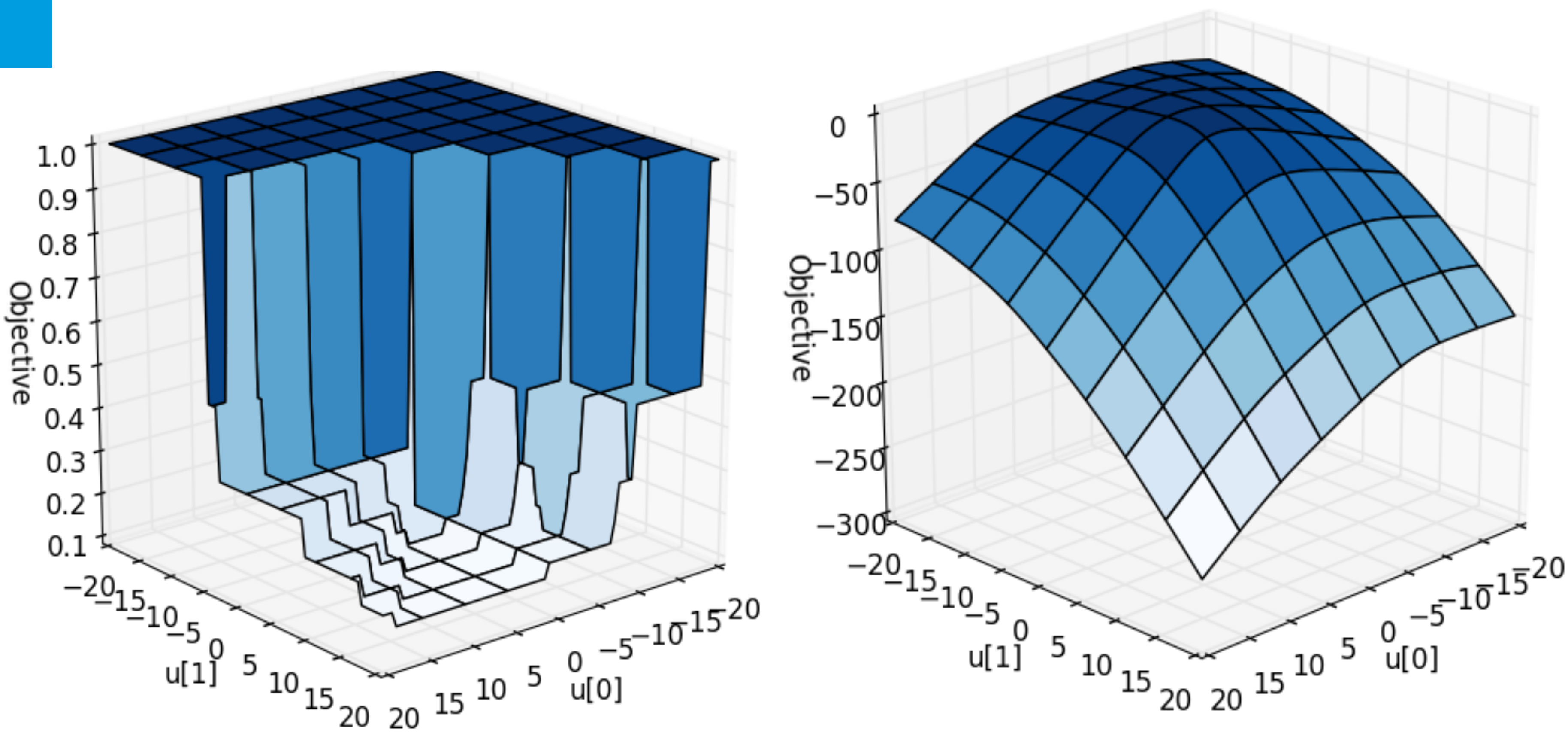
Rank-based component

$$\mathbb{I}(R_{ik} < R_{ij}) \approx g(f_{ik} - f_{ij})$$

$$\frac{1}{R_{ij}} \approx g(f_{ij})$$

Rank-based component

Reciprocal Rank



Lower Bound

$$RR_i \approx \sum_{j=1}^{N} Y_{ij} g(f_{ij}) \prod_{k=1}^{N} \left(1 - Y_{ik} g(f_{ik} - f_{ij})\right)$$

Jensen's inequality

Concavity of log function

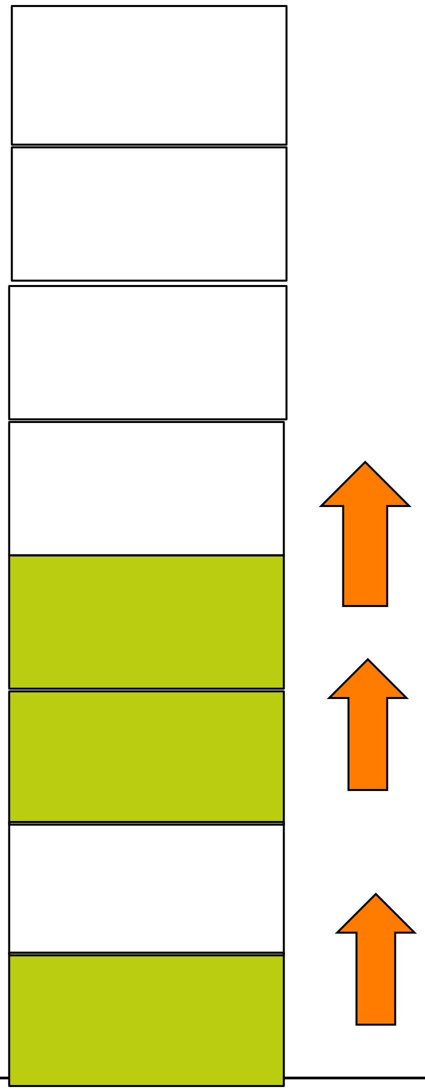
$$L(U_i, V) = \sum_{j=1}^{N} Y_{ij} \left[\ln g(f_{ij}) + \sum_{k=1}^{N} \ln \left(1 - Y_{ik} g(f_{ik} - f_{ij})\right)\right]$$

$$O(n^{+2})$$

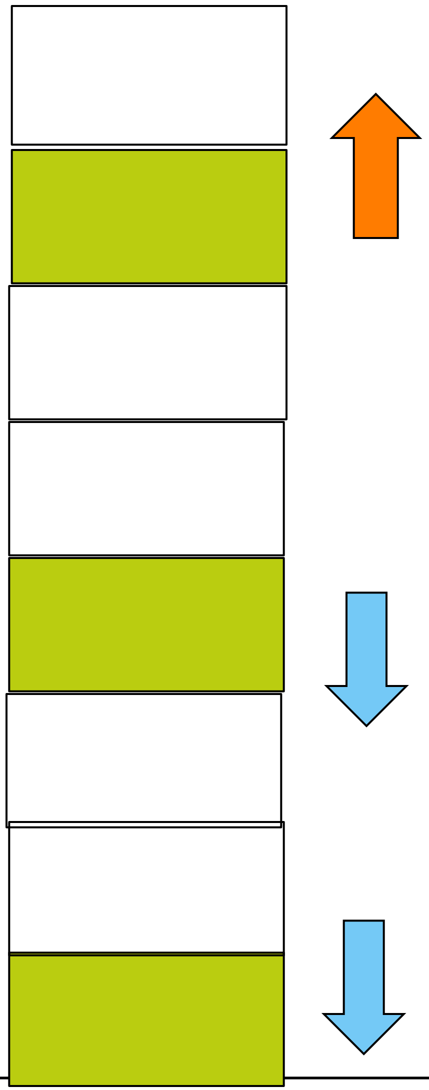
What's the Key?

- CLiMF reciprocal rank loss essentially pushes relevant items apart
- In the process at least one items ends up high-up in the list

Conventional loss

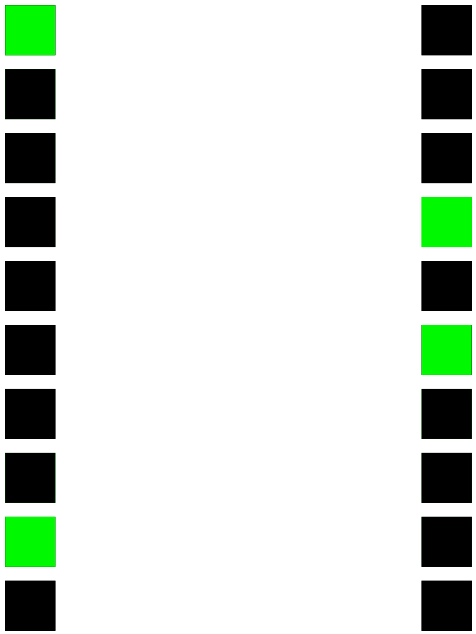


CLiMF MRR-loss



Optimizing RR vs. AUC

Opt. RR　　　　Opt. AUC



Listwise

xCLiMF [Shi 2013b] – Generalization to Ratings

- Expected reciprocal rank (ERR) [Chapelle 2009]
  - A generalization of RR to graded relevance data
  - Cascade nature compared to NDCG

Equal contribution independent of the items ranked above

NDCG

0		0
5		1
0		0
3	=	3
0		0

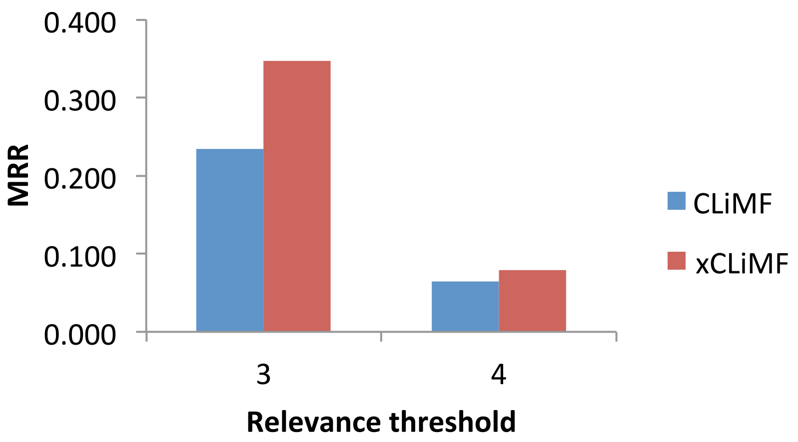
ERR

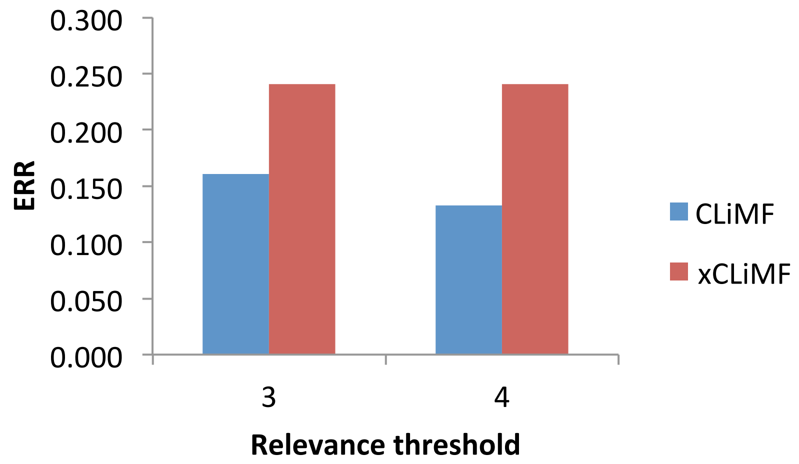
0		0
5		1
0		0
3	<	3
0		0

Different contributions relative to the items ranked above

xCLiMF vs. CLiMF

Results from Tuenti video watching dataset





xCLiMF shows its advantage over CLiMF when applied to graded relevance data

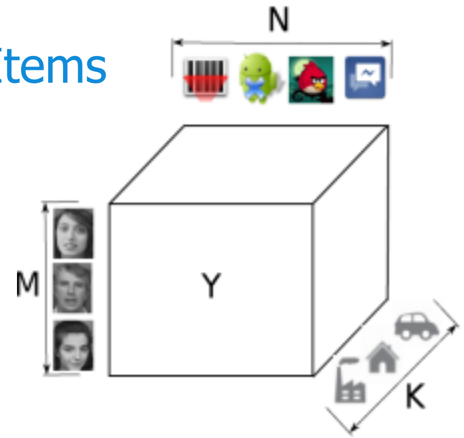
Listwise

TFMAP [Shi 2012a] – Generalization to Context

Context-aware Recommendation

Problem Setting

- Given: Users' implicit feedback on items under different contexts
- Target: To recommend a list of items to each user under any given context, as accurate as possible



Top-N recommendation

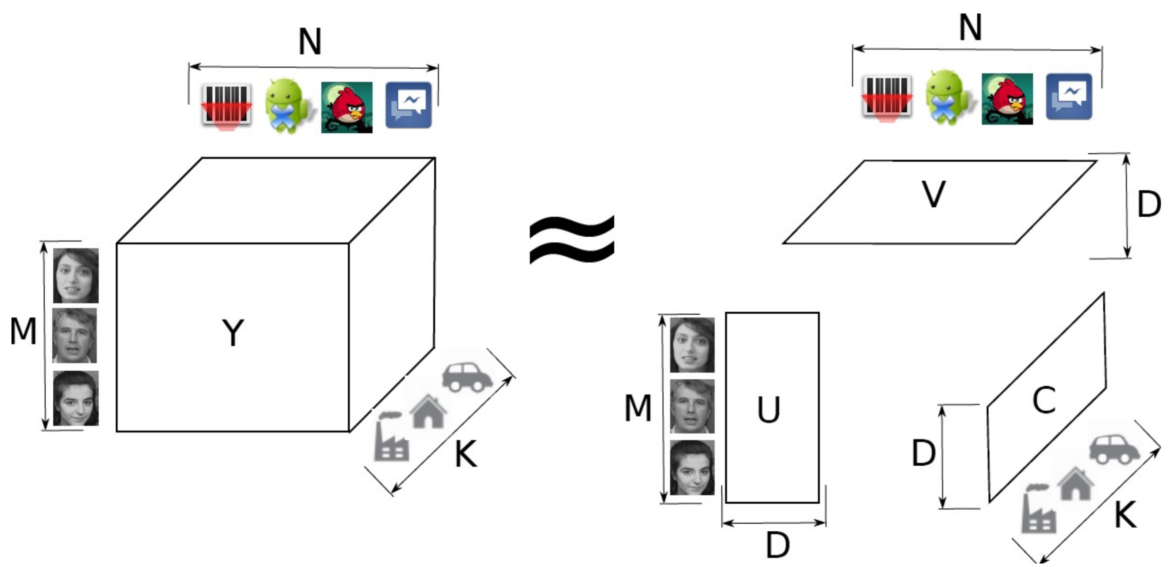
Context-aware

Optimal in terms of MAP

Main Concept of TFMAP

- CP tensor factorization

$$f_{mik} = \langle U_m, V_i, C_k \rangle = \sum_{d=1}^{D} U_{md} V_{id} C_{kd}$$



U,V,C are latent factors (parameters to be learned)

U, V, C are learned from optimizing the smoothed MAP

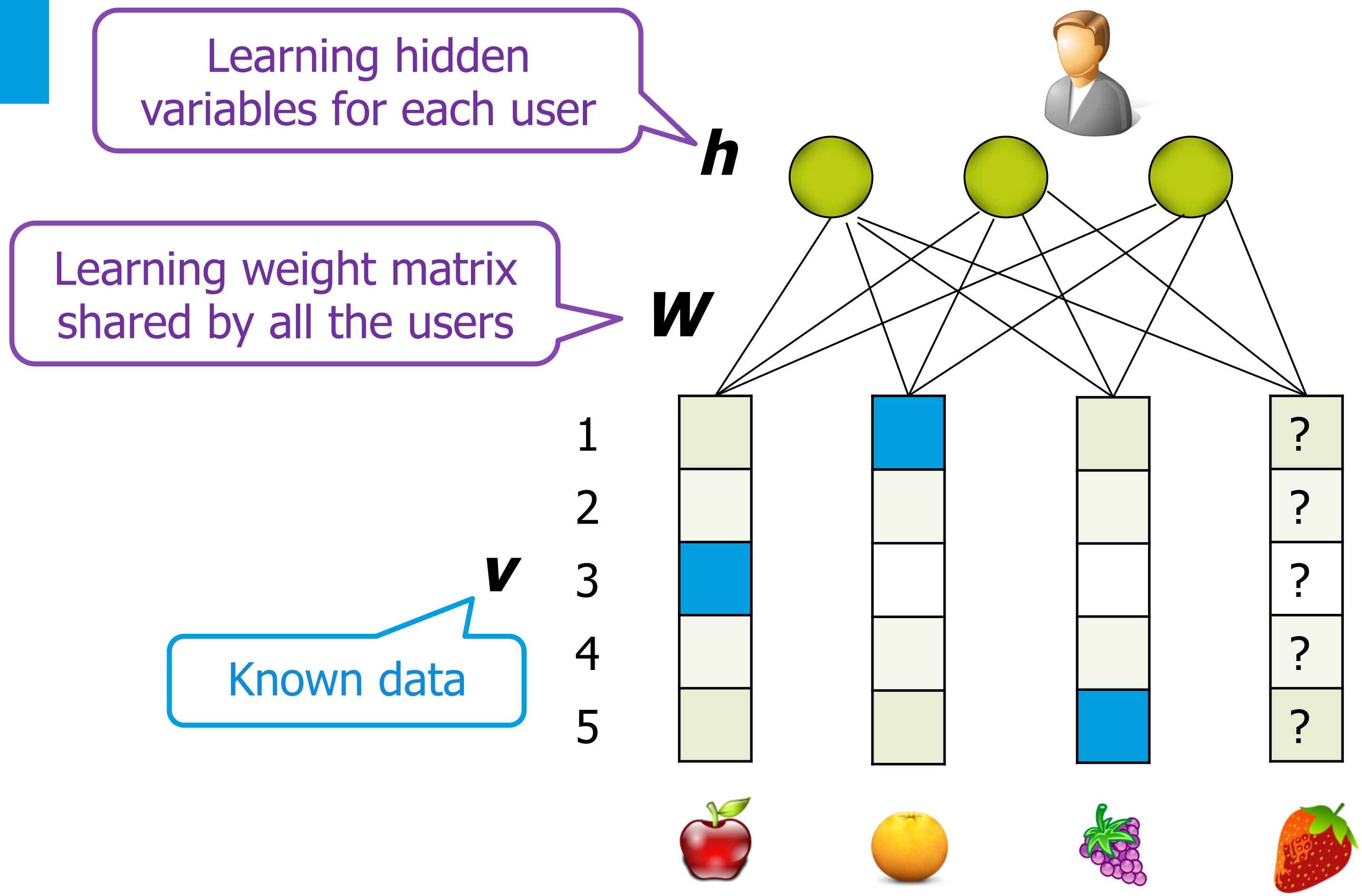
- Intro to in Ranking
  - Ranking measures
- Learning to Rank for Recommender Systems
  - Classification of approaches
- **Trends and Challenges**

TRENDS AND CHALLENGES

Deep Learning
Restricted Boltzmann Machines (RBMs)

- A (generative stochastic) Neural Network
- Learns a probability distribution over its inputs
- Used in dimensionality reduction, CF, topic modeling, feature learning
- Essential components of Deep Learning methods (DBN's, DBM's)

RBMs for CF
[Salakhutdinov 2007]



Bandits

- In many domains items are constantly new (e.g. news recommendation, computational advertisement)
- Contextual Bandits are on-line learning algorithms that work by "exploring" the user preference space and "exploiting" the resulting models in serving recommendations
- There has been little work in finding optimal strategies in terms of the ranking of items

Ranking aggregation

- Ensemble methods and Hybrid methods are commonly used in industry
- Aggregating the resulting rankings provided by different recommendations methods is not trivial
- Currently an open research field

Modelling Dependencies in Rankings

- Most ranking methods are oblivious to the other items in the recommended ranked list.
- For example Diversity is not taken into account in most ranking methods
- Basket recommendation or next item recommendation cannot be handled by most current methods

Meet Okapi

- A new Collaborative Ranking framework that runs on apache Giraph
- Written in Java
  - accessible for a large(r) audience
  - good for unit testing
- Distributed, "large scale ready" computing
- Reproducibility
  - has a distributed evaluation framework
- http://baltrunas.info/research-menu/okapi

Thank you !

Telefonica Research is looking for interns!

Contact: alexk@tid.es or linas@tid.es

References

- Balakrishnan, S., & Chopra, S. (2012, February). Collaborative ranking. In *Proceedings of the fifth ACM international conference on Web search and data mining* (pp. 143-152). ACM.
- Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., & Hullender, G. (2005, August). Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning* (pp. 89-96). ACM.
- Cao, Z., Qin, T., Liu, T. Y., Tsai, M. F., & Li, H. (2007, June). Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning* (pp. 129-136). ACM.
- Chapelle, O., Metlzer, D., Zhang, Y., & Grinspan, P. (2009, November). Expected reciprocal rank for graded relevance. In *Proceedings of the 18th ACM conference on Information and knowledge management* (pp. 621-630). ACM.
- Joachims, T. (2002, July). Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 133-142). ACM.
- Koren, Y. (2008, August). Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 426-434). ACM.
- Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, *42*(8), 30-37.
- Koren, Y., & Sill, J. (2011, October). OrdRec: an ordinal model for predicting personalized item rating distributions. In *Proceedings of the fifth ACM conference on Recommender systems* (pp. 117-124). ACM.
- Liu, N. N., & Yang, Q. (2008, July). Eigenrank: a ranking-oriented approach to collaborative filtering. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 83-90). ACM.

- Liu, N. N., Zhao, M., & Yang, Q. (2009, November). Probabilistic latent preference analysis for collaborative filtering. In *Proceedings of the 18th ACM conference on Information and knowledge management* (pp. 759-766). ACM.
- Quoc, C., & Le, V. (2007). Learning to rank with nonsmooth cost functions. *Proceedings of the Advances in Neural Information Processing Systems*, *19*, 193-200.
- Rendle, S., Freudenthaler, C., Gantner, Z., & Schmidt-Thieme, L. (2009, June). BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence* (pp. 452-461). AUAI Press.
- Rendle, S. (2012). Factorization machines with libFM. *ACM Transactions on Intelligent Systems and Technology (TIST)*, *3*(3), 57.
- Salakhutdinov, R., Mnih, A., & Hinton, G. (2007, June). Restricted Boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning* (pp. 791-798). ACM.
- Shi, Y., Larson, M., & Hanjalic, A. (2010, September). List-wise learning to rank with matrix factorization for collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender systems* (pp. 269-272). ACM.
- Shi, Y., Karatzoglou, A., Baltrunas, L., Larson, M., Hanjalic, A., & Oliver, N. (2012a, August). TFMAP: Optimizing MAP for top-n context-aware recommendation. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval* (pp. 155-164). ACM.
- Shi, Y., Karatzoglou, A., Baltrunas, L., Larson, M., Oliver, N., & Hanjalic, A. (2012b, September). CLiMF: learning to maximize reciprocal rank with collaborative less-is-more filtering. In *Proceedings of the sixth ACM conference on Recommender systems* (pp. 139-146). ACM.
- Taylor, M., Guiver, J., Robertson, S., & Minka, T. (2008, February). Softrank: optimizing non-smooth rank metrics. In *Proceedings of the international conference on Web search and web data mining* (pp. 77-86). ACM.

- Volkovs, M., & Zemel, R. S. (2012). Collaborative ranking with 17 parameters. In *Advances in Neural Information Processing Systems* (pp. 2303-2311).
- Weimer, M., Karatzoglou, A., Le, Q. V., & Smola, A. J. (2007). Cofi rank-maximum margin matrix factorization for collaborative ranking. In *Advances in neural information processing systems* (pp. 1593-1600).
- Xu, J., & Li, H. (2007, July). Adarank: a boosting algorithm for information retrieval. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 391-398). ACM.
- Yue, Y., Finley, T., Radlinski, F., & Joachims, T. (2007, July). A support vector method for optimizing average precision. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 271-278). ACM.