Factorization and Recommendation

Prof. Shou-De Lin

sdlin@csie.ntu.edu.tw

What is the Recommendation System(RS)?

- A recommendation system suggests potentially favored items or contents to users
  - Users may be recommended new items previously unknown to them.
  - The system can find the opposite (e.g.: items the users do not like)

推薦系統就是那個「心魔」

Values of Recommendation Systems

- For service providers
  - Improve trust and customer loyalty
  - Increase sales, click trough rates, etc.
  - Opportunities for promotion
- For customers
  - New interesting items being identified
  - Narrow down the possible choices

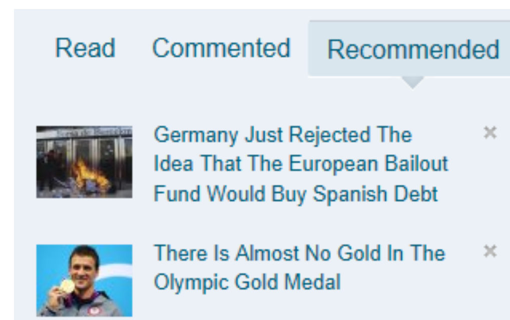
2/3 of the movies watched are recommended

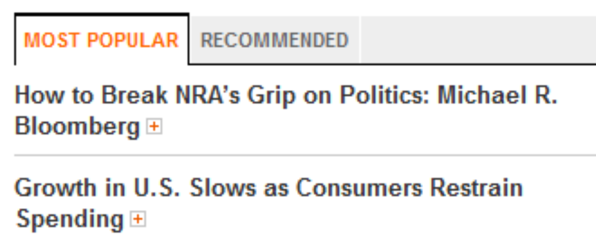
recommendations generate 38% more click-throughs

35% sales from recommendations

28% of the people wou buy more music if they found what they liked

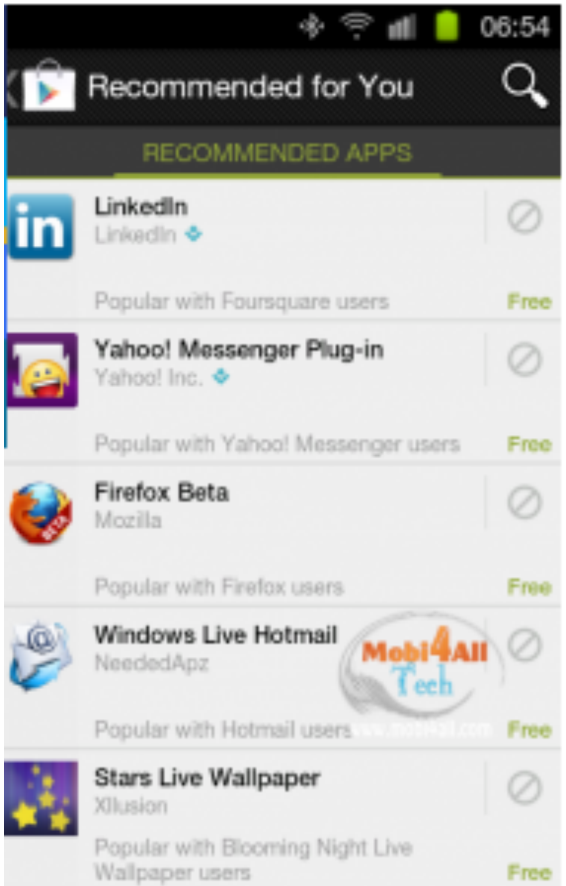
Recommending products, services, and news

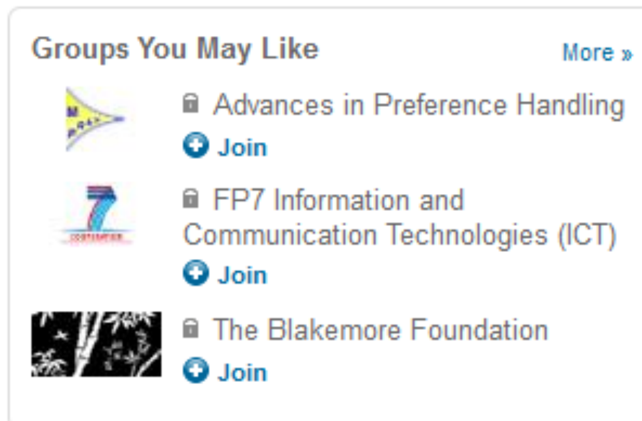


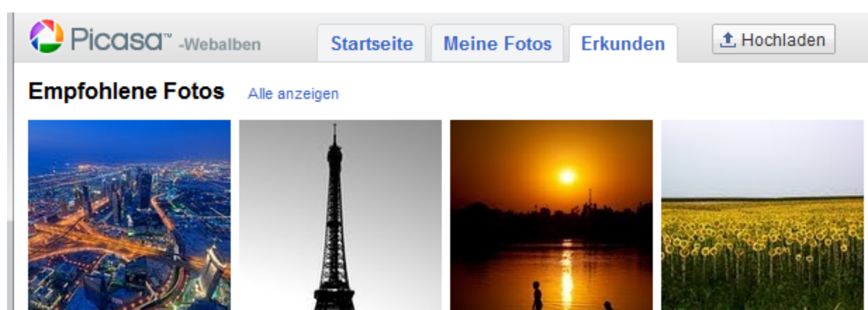


From Dietmar Jannach, Markus Zanker and Gerhard Friedrich

Recommending friends, jobs, and photos







From Dietmar Jannach, Markus Zanker and Gerhard Friedrich

The Netflix Challenge （2006～2009）

- 1M prize to improve the prediction accuracy by 10%

KDD Cup 2011: Yahoo Music Recommendation
KDD Cup 2012: Tencent Advertisement Recommendation

What is a good recommender system?

- **Requirement 1: finding items that interests the specific user ➔ personalization**
- Requirement 2: recommending diverse items that satisfy all the possible needs of a specific user ➔ diversity
- Requirement 3: making non-trivial recommendation (Harry Potter 1 ,2 ,3 ,4 ➔ 5) ➔ novelty

Inputs/outputs of a Recommendation System

- Given (1 or 2&3):
  1. Ratings of users to items: rating can be either explicit or implicit, for example
     1. Explicit ratings: the level of likeliness of users to items
     2. Implicit ratings: the browsing information of users to items
  2. User features (e.g. preferences, demographics)
  3. Item features (e.g. item category, keywords)
  4. User/user relationships （optional)
- Predict:
  - Ratings of any user to any item, or
  - Ranking of items to each user
- What to recommend?
  - Highly rated or ranked items given each user

Types of Recommender Systems

- **Collaborative Filtering**
- Content-based Recommendation (later on)
- Other solutions

Collaborative Filtering (CF)

- CF is the most successful and common approach to generate recommendations
  - used in Amazon, Netflix, and most of the e-commerce sites
  - many algorithms exist
  - General and can be applied to many domains (book, movies, DVDs, ..)
- Key Idea
  - Recommended the favored items of people who are ‘similar’ to you
  - Need to collect the taste (implicit or explicit) of other people ➔ that’s why we call it ‘collaborative’

Mathematic Form of CF

- Given: some ratings from users to items
- Predict: unknown ratings of users to items

Explicit ratings	Item1	Item2	Item3	Item4
User1	?	3	?	?
User2	1	?	?	5
User3	?	4	?	2
User4	3	?	3	?

Implicit ratings	Item1	Item2	Item3	Item4
User1	?	1	?	?
User2	1	?	?	1
User3	?	1	?	1
User4	1	?	1	?

CF models

- Memory-based CF
  - User-based CF
  - Item-based CF
- Model-based CF

User-Based Collaborative Filtering

- Finding users $N(u)$ most similar to user u (neighborhood of u)
  - Assign $N(u)$'s rating as u's rating
- Prediction
  - $r_{u,i} = \overline{r}_u + \frac{\sum_{v \in N(u)} sim(u,v)(r_{v,i} - \overline{r}_v)}{\sum_{v \in N(u)} sim(u,v)}$
  - $\overline{r}_u$: Average of ratings of user u
- Problem: Usually users do not have many ratings; therefore, the similarities between users may be unreliable

Example of User-Based CF

$$\bar{r}_{John} = \frac{3+0+3+3}{4} = 2.25$$

$$\bar{\boldsymbol{r}}_{\boldsymbol{Joe}} = \frac{5+4+0+2}{4} = \mathbf{2.75}$$

$$\bar{r}_{Jill} = \frac{1+2+4+2}{4} = 2.25$$

$$\bar{r}_{Jane} = \frac{3+1+0}{3} = 1.33$$

$$\bar{\boldsymbol{r}}_{\boldsymbol{Jorge}} = \frac{2+2+0+1}{4} = \mathbf{1.25}$$

$$sim(Jane, John) = \frac{3\times3 + 1\times3 + 0\times3}{\sqrt{3^2+1^2+0^2}\sqrt{3^2+3^2+3^2}} = 0.73$$

$$sim(Jane, \boldsymbol{Joe}) = \frac{3\times5 + 1\times0 + 0\times2}{\sqrt{3^2+1^2+0^2}\sqrt{5^2+0^2+2^2}} = \mathbf{0.88}$$

$$sim(Jane, Jill) = \frac{3\times1 + 1\times4 + 0\times2}{\sqrt{3^2+1^2+0^2}\sqrt{1^2+4^2+2^2}} = 0.48$$

$$sim(Jane, \boldsymbol{Jorge}) = \frac{3\times2 + 1\times0 + 0\times1}{\sqrt{3^2+1^2+0^2}\sqrt{2^2+0^2+1^2}} = \mathbf{0.84}$$

$$r_{Jane,Aladdin} = 1.33 + \frac{0.88(4-2.75) + 0.84(2-1.25)}{0.88+0.84} = 2.33$$

- To predict $r_{Jane,Aladdin}$ using cosine similarity
  - Neighborhood size is 2

Rating	Lion King	Aladdin	Mulan	Anastasia
John	3	0	3	3
Joe	5	4	0	2
Jill	1	2	4	2
Jane	3	?	1	0
Jorge	2	2	0	1

Item-Based Collaborative Filtering

- Similarity is defined between two items
- Finding items $N(i)$ most similar to item i (neighborhood of i)
  - Assign $N(i)$'s rating as i's rating
- Prediction
  - $r_{u,i} = \overline{r}_i + \frac{\sum_{j \in N(i)} sim(i,j)(r_{u,j} - \overline{r}_j)}{\sum_{j \in N(i)} sim(i,j)}$
  - $\overline{r}_i$: Average of ratings of item i
- Items usually have more ratings from many users and the similarities between items are more stable

Example of Item-Based CF

$$\bar{\boldsymbol{r}}_{\boldsymbol{Lion\ King}} = \frac{3+5+1+3+2}{5} = \mathbf{2.8}$$

$$\bar{r}_{Aladdin} = \frac{0+4+2+2}{4} = 2$$

$$\bar{r}_{Mulan} = \frac{3+0+4+1+0}{5} = 1.6$$

$$\bar{\boldsymbol{r}}_{\boldsymbol{Anastasia}} = \frac{3+2+2+0+1}{5} = \mathbf{1.6}$$

$$sim(Aladdin, \boldsymbol{Lion\ King}) = \frac{0\times3+4\times5+2\times1+2\times2}{\sqrt{0^2+4^2+2^2+2^2}\sqrt{3^2+5^2+1^2+2^2}} = \mathbf{0.84}$$

$$sim(Aladdin, Mulan) = \frac{0\times3+4\times0+2\times4+2\times0}{\sqrt{0^2+4^2+2^2+2^2}\sqrt{3^2+0^2+4^2+0^2}} = 0.32$$

$$sim(Aladdin, \boldsymbol{Anastasia}) = \frac{0\times3+4\times2+2\times2+2\times1}{\sqrt{0^2+4^2+2^2+2^2}\sqrt{3^2+2^2+2^2+1^2}} = \mathbf{0.67}$$

$$r_{Jane,Aladdin} = 2 + \frac{0.84(3-2.8)+0.67(0-1.6)}{0.84+0.67} = 1.40$$

- To predict $r_{Jane,Aladdin}$ using cosine similarity
  - Neighborhood size is 2

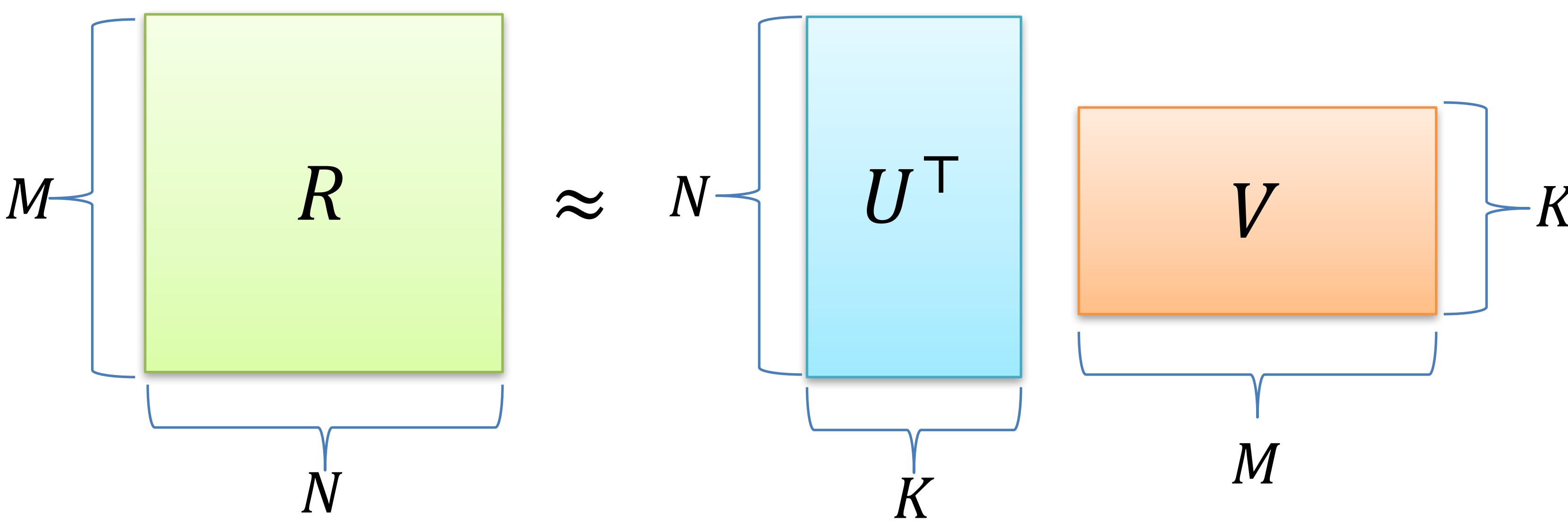
Rating	Lion King	Aladdin	Mulan	Anastasia
John	3	0	3	3
Joe	5	4	0	2
Jill	1	2	4	2
Jane	3	?	1	0
Jorge	2	2	0	1

CF models

- Memory-based CF
  - User-based CF
  - Item-based CF
- Model-based CF
  - Matrix Factorization

Matrix Factorization (MF)

- Given a matrix $R \in \mathbb{R}^{N \times M}$, we would like to find two matrices $U \in \mathbb{R}^{K \times N}, V \in \mathbb{R}^{K \times M}$ such that $U^\top V \approx R$
  - $K \ll \min\{N, M\}$ ➔ we assume R of small rank K
  - A low-rank approximation method
  - Earlier works (before 2007 ~ 2009) call it singular value decomposition (SVD)



Matrix factorization

$$R = U^{\top}V$$

U_k	Dim1	Dim2
Alice	0.4	0.3
Bob	-0.4	0.3
Mary	0.7	-0.6
Sue	0.3	0.9


V_k^T					
Dim1	-0.4	-0.7	0.6	0.4	0.5
Dim2	0.8	-0.6	0.2	0.2	-0.3

Rating (Alice to Harry Potter)=0.4*0.4+0.3*0.2

MF as an Effective CF Realization

- We find two matrices U, V to approximate R
  - Missing entry R_{ij} can be predicted by $U_i^\top V_j$
- Entries in column U_i (or V_j)represent the latent factor (i.e. rating patterns learned from data) of user i (or item j)
- If two users have similar latent factors, then they will give similar ratings to all items
- If two items have similar latent factor, then the corresponding rating for all users are similar

$U^\top$

1	0	2
1	0	1

V

6				
3				
-1				

$\approx$

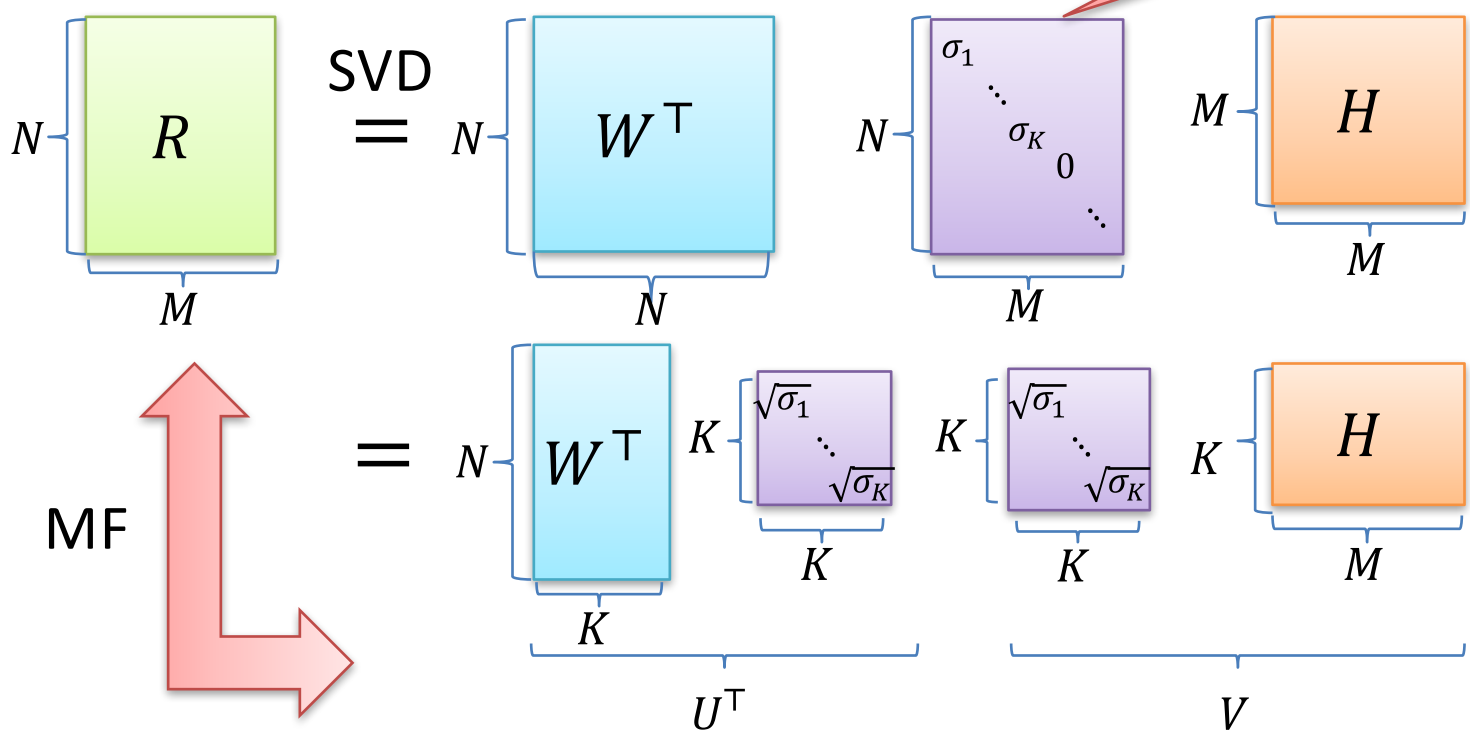
R

1	2		5	
4		2		5
2			4	3
5		1		4
	3	3		

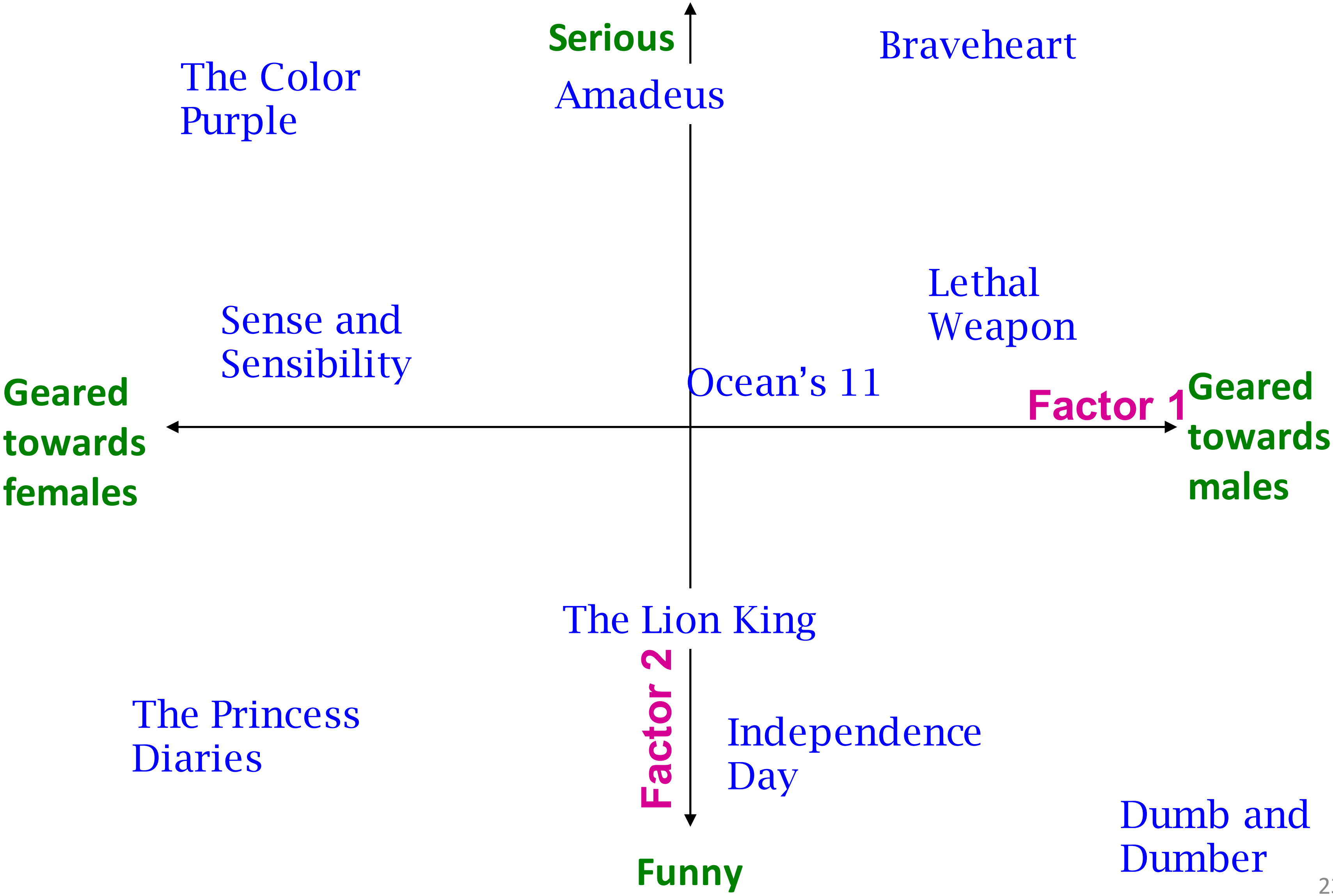
Relationship between MF and SVD

- Singular value decomposition (SVD)
  - Matrix R of rank K should be non-missing
- Matrix factorization (MF)
  - Missing entries can be omitted in learning
  - It is more scalable for large datasets

Diagonal matrix of K positive singular values $\sigma_1 \geq \cdots \geq \sigma_K > 0$ for a rank-K matrix



Latent Factor Examples (Leskovec et al.)

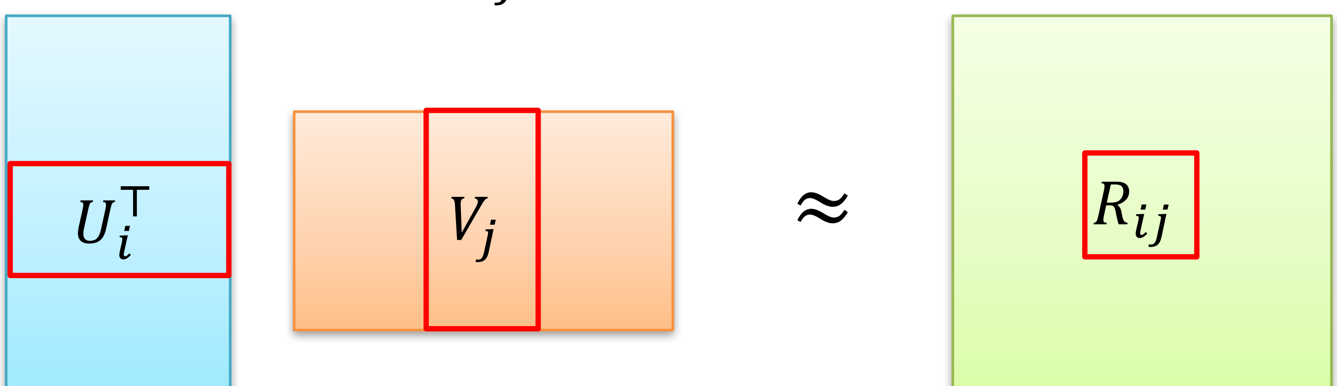


How to Train an MF Model (1/2)

- MF as a Minimization problem

$$\arg\min_{U,V} \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{M} \delta_{ij} \overbrace{\left(U_i^\top V_j - R_{ij}\right)}^{\text{Squared error}}{}^2 + \overbrace{\frac{\lambda_U}{2} \|U\|_F^2 + \frac{\lambda_V}{2} \|V\|_F^2}^{\text{Regularization}}$$

  - $\|U\|_F^2 = \sum_{i=1}^{N} \|U_i\|_2^2 = \sum_{i=1}^{N} \sum_{k=1}^{K} U_{ki}^2$: squared Frobenius norm
  - $\delta_{ij} \in \{0,1\}$: rating R_{ij} is observed in R



How to Train an MF Model (2/2)

- MF with bias terms

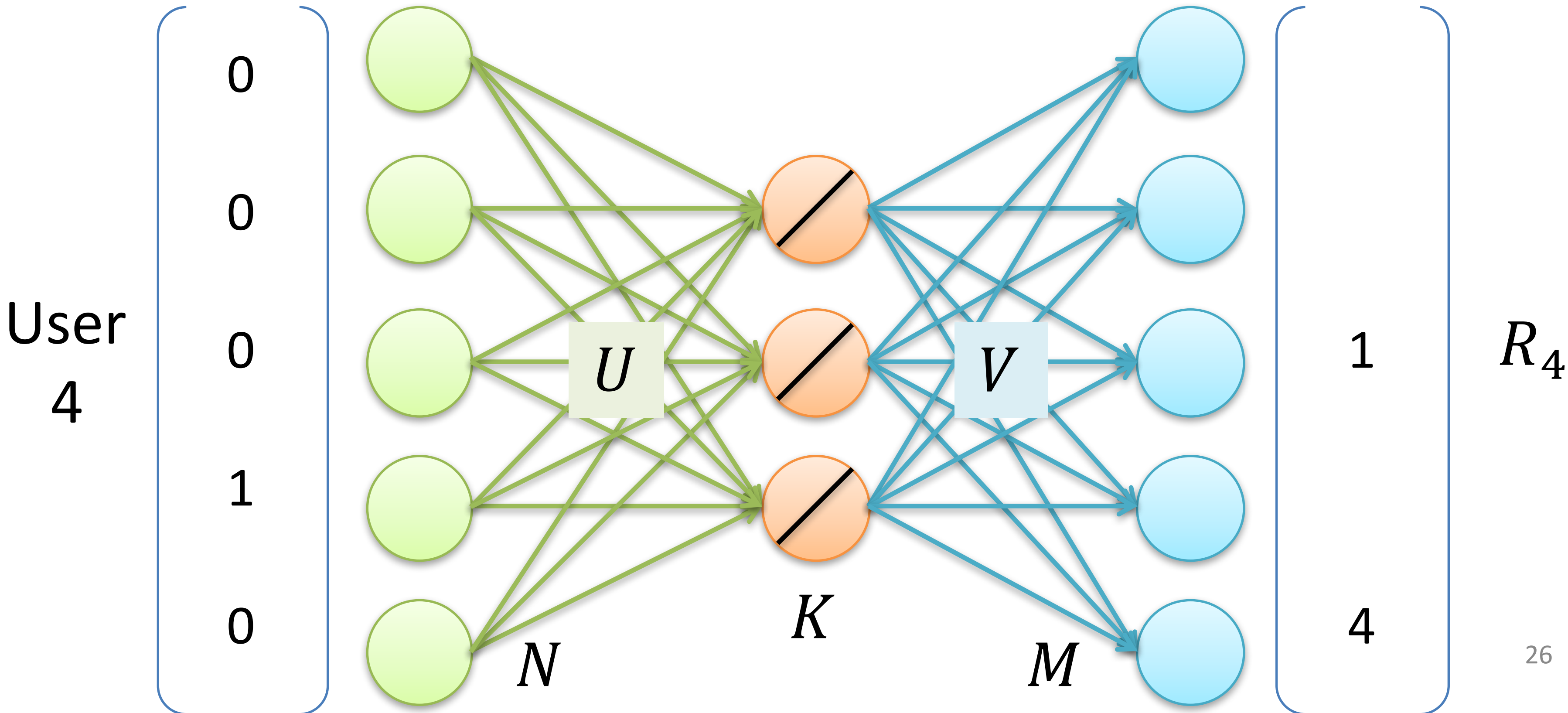
$$\arg\min_{U,V} \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M}\delta_{ij}\left(U_i^\top V_j + b_i + c_j + \mu - R_{ij}\right)^2 + \frac{\lambda_U}{2}\|U\|_F^2 + \frac{\lambda_V}{2}\|V\|_F^2 + \frac{\lambda_b}{2}\|b\|_2^2 + \frac{\lambda_c}{2}\|c\|_2^2 + \frac{\lambda_\mu}{2}\mu^2$$

  - b: rating mean vector for each user
  - c: rating mean vector for each item
  - μ: overall mean of all ratings
- Some MF extension works omit the bias terms

Koren, Yehuda, Robert Bell, and Chris Volinsky.
"Matrix factorization techniques for recommender systems."
Computer 42.8 (2009): 30-37.

MF as Neural Network (NN)

- Shallow NN with identity activation function
  - N input neurons: user i as one-hot encoding
  - M output neurons: row i in rating matrix R



MF as Probabilistic Graphical Model (PGM)

- Bayesian network with normal distributions
- Maximum a posteriori (MAP)

$$\underbrace{\arg\max_{U,V} \prod_{i=1}^{N} \prod_{j=1}^{M} \mathcal{N}\left(R_{ij} \middle| U_i^\top V_j, \sigma_R^2\right)^{\delta_{ij}}}_{\substack{\text{Likelihood} \\ \text{(normal distribution)}}} \underbrace{\prod_{i=1}^{N} \mathcal{N}(U_i | 0, \sigma_U^2 I) \prod_{j=1}^{M} \mathcal{N}(V_j | 0, \sigma_V^2 I)}_{\substack{\text{Zero-mean spherical Gaussian prior} \\ \text{(multivariate normal distribution)}}}$$

Learning in MF

- Stochastic gradient descent (SGD)
- Alternating least squares (ALS)
- Variational expectation maximization (VEM)

Stochastic Gradient Descent (SGD) (1/2)

- Gradient descent: updating variables based on the direction of negative gradients
  - SGD updates variables instance-wise
- Let L be the objective function

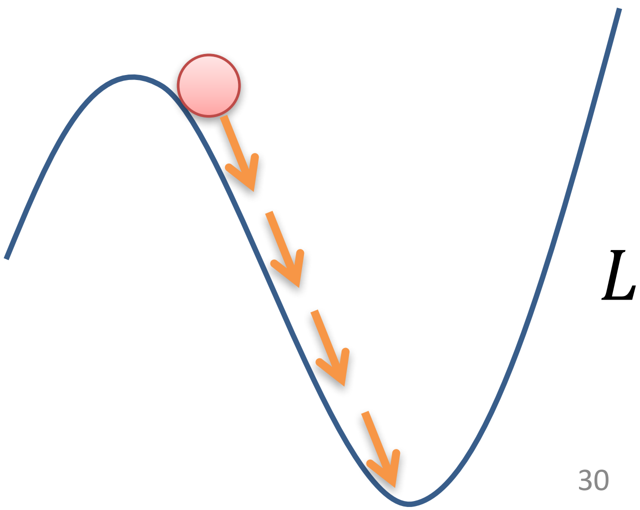
$$L = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M}\delta_{ij}\left(U_i^{\top}V_j - R_{ij}\right)^2 + \frac{\lambda_U}{2}\|U\|_F^2 + \frac{\lambda_V}{2}\|V\|_F^2$$

  - Objective function for each training rating

$$L_{ij} = \frac{1}{2}\left(U_i^{\top}V_j - R_{ij}\right)^2 + \frac{\lambda_U}{2}\|U_i\|_2^2 + \frac{\lambda_V}{2}\left\|V_j\right\|_2^2$$

Stochastic Gradient Descent (SGD) (2/2)

- Gradient
  - $-\frac{\partial L_{ij}}{\partial U_i} = \left(U_i^\top V_j - R_{ij}\right)V_j + \lambda_U U_i$
  - $-\frac{\partial L_{ij}}{\partial V_j} = \left(U_i^\top V_j - R_{ij}\right)U_i + \lambda_V V_j$
- Update rule
  - $U_i \leftarrow U_i - \eta \frac{\partial L_{ij}}{\partial U_i}$
  - $V_j \leftarrow V_j - \eta \frac{\partial L_{ij}}{\partial V_j}$
  - η: learning rate or step size



Matrix factorization: Stopping criteria

- When do we stop updating?
  - Improvement drops (e.g. <0)
  - Reached small error
  - Achieved predefined # of iterations
  - No time to train anymore

Alternating Least Squares (ALS)

- Stationary point: zero gradient
  - We can find the closed-form solution of U with V fixed, and vise versa
- Zero gradient
  - $\frac{\partial L}{\partial U_i} = \sum_{j=1}^{M} \delta_{ij} V_j \left(U_i^\top V_j - R_{ij}\right) + \lambda_U U_i = 0$
  - $\frac{\partial L}{\partial V_j} = \sum_{i=1}^{N} \delta_{ij} U_i \left(U_i^\top V_j - R_{ij}\right) + \lambda_V V_j = 0$
- Closed-form solution i.e. update rule
  - $U_i = \left(\sum_{j=1}^{M} \delta_{ij} V_j V_j^\top + \lambda_U I\right)^{-1} \left(\sum_{j=1}^{M} \delta_{ij} R_{ij} V_j\right)$
  - $V_j = \left(\sum_{i=1}^{N} \delta_{ij} U_i U_i^\top + \lambda_V I\right)^{-1} \left(\sum_{i=1}^{N} \delta_{ij} R_{ij} U_i\right)$

SGD vs. ALS

$$\arg\min_{U,V} \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{M}\delta_{ij}\left(U_i^\top V_j - R_{ij}\right)^2 + \frac{\boxed{\lambda_U}}{2}\|U\|_F^2 + \frac{\boxed{\lambda_V}}{2}\|V\|_F^2$$

- SGD
  - It is easier to develop MF extensions since we do not require the closed-form solutions
- ALS
  - We are free from determining the learning rate η
  - It allows parallel computing
- Drawback for both
  - Regularization parameters λ need careful tuning using validation → we have to run MF for multiple times
- Variational-EM (VEM) learns regularization parameters

Extensions of MF

- Matrix R can be factorized into $U^\top U, U^\top VU, UVW, \ldots$
- SVD++
  - MF with implicit interactions among items
- Non-negative MF (NMF)
  - non-negative entries for the factorized matrices
- Tensor factorization (TF), Factorization machines (FM)
  - Additional features involved in learning latent factors
- Bayesian PMF (BPMF)
  - Further modeling distributions of PMF parameters θ
- Poisson factorization (PF)
  - PMF normal likelihood replaced with Poisson likelihood to form a probabilistic nonnegative MF

Applications of MF

- Recommender systems
- Filling missing features
- Clustering
- Link prediction
  - Predict future new edges in a graph
- Community detection
  - Cluster nodes based on edge density in a graph
- Word embedding
  - Word2Vec is actually an MF

Limitations on Collaborative filtering

1. Cannot consider features of items and users
   - Solution: factorization machine
2. Cannot consider cold-start situation
   - Solution: transfer recommendation

Demo: A joke recommendation system using CM models

- http://eigentaste.berkeley.edu/index.html

Evaluating Recommendation Systems

- Accuracy of predictions
  - How close predicted ratings are to the true ratings
- Relevancy of recommendations
  - Whether users find the recommended items relevant to their interests
- Ranking of recommendations
  - Ranking products based on their levels of interestingness to the user

Accuracy of Predictions

- Mean absolute error (MAE)
    - $MAE = \frac{\sum_{ij}|\hat{r}_{ij} - r_{ij}|}{n}$
    - $\hat{r}_{ij}$: Predicted rating of user i and item j
    - r_{ij}: True rating
- Normalized mean absolute error (NMAE)
    - $NMAE = \frac{MAE}{r_{max} - r_{min}}$
- Root mean squared error (RMSE)
    - $RMSE = \sqrt{\frac{1}{n}\sum_{ij}\left(\hat{r}_{ij} - r_{ij}\right)^2}$
    - Error contributes more to the RMSE value

Relevancy of Recommendations

- Precision
  - $P = \frac{N_{rs}}{N_s}$
- Recall
  - $R = \frac{N_{rs}}{N_r}$
- F-measure (F_1 score)
  - $F = \frac{2PR}{P+R}$

		Recommended Items		
		Selected	Not Selected	Total
Relevancy	Relevant	N_{rs}	N_{rn}	N_r
	Irrelevant	N_{is}	N_{in}	N_i
	Total	N_s	N_n	N

Ranking of Recommendations

- Spearman's rank correlation: For n items
  - $\rho = 1 - \frac{6\sum_{i=1}^{n}(x_i - y_i)^2}{n^3 - n}$
  - $1 \leq x_i \leq n$: Predicted rank of item i
  - $1 \leq y_i \leq n$: True rank of item i
- Kendall's tau: For all $\binom{n}{2}$ pairs of items (i, j)
  - $\tau = \frac{c-d}{\binom{n}{2}}$, in range $[-1,1]$
  - There are c concordant pairs
    - $x_i > x_j, y_i > y_j$ or $x_i < x_j, y_i < y_j$
  - There are d discordant pairs
    - $x_i > x_j, y_i < y_j$ or $x_i < x_j, y_i > y_j$

Library for Recommender Systems

LIBMF: A Matrix-factorization Library for Recommender Systems

URL: https://www.csie.ntu.edu.tw/~cjlin/libmf/

Language: C++

Focus: solvers for Matrix-factorization models

R interface: package “recosystem”

https://cran.r-project.org/web/packages/recosystem/index.html

Features:

- solvers for real-valued MF, binary MF, and one-class MF
- parallel computation in a multi-core machine
- less than 20 minutes to converge on a data set of 1.7B ratings
- supporting disk-level training, which largely reduces the memory usage

MyMediaLite - a recommender system algorithm library

URL: http://mymedialite.net/

Language: C#

Focus: rating prediction and item prediction from positive-only feedback

Alogrithms: kNN, BiasedMF, SVD++…

Features:

- Measure: MAE, RMSE, CBD, AUC, prec@N, MAP, and NDCG

LibRec: A Java Library for Recommender Systems

URL: http://www.librec.net/index.html

Language: Java

Focus: algorithms for **rating prediction** and **item ranking**

Algorithms: kNN, PMF, NMF, BiasedMF, SVD++, BPR, LDA…more at: http://www.librec.net/tutorial.html

Features:

- Faster than MyMediaLite

Collection of Datasets: MovieLens 1M, Epinions, Flixster...

- http://www.librec.net/datasets.html

mrec: recommender systems library

URL: http://mendeley.github.io/mrec/

Language: Python

Focus: item similarity and other methods for implicit feedback

Algorithms: item similarity methods, MF, weighted MF for implicit feedback

Features:

- train models and make recommendations in parallel using IPython
- utilities to prepare datasets and compute quality metrics

SUGGEST: Top-N recommendation engine

Python interface: “pysuggest”

https://pypi.python.org/pypi/pysuggest

Focus: collaborative filtering-based top-N recommendation algorithms (user-based and item-based)

Algorithms: user-based or item-based collaborative filtering based on various similarity measures

Features:

low latency: compute top-10 recommendations in less that 5us

Conclusion

- Recommendation is arguably the most successful AI/ML solutions till now.
  - It is not just for customer-product
  - Matching users and users
  - Matching users and services
  - Matching users and locations
  - …
- The basic skills and tools are mature, but the advanced issues are not fully solved.
  - Lack of data for cold start users is still the main challenging task to be solved.

References (1/3)

- SVD++
  - Koren, Yehuda. "Factorization meets the neighborhood: a multifaceted collaborative filtering model." *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2008.
- TF
  - Karatzoglou, Alexandros, et al. "Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering." *Proceedings of the fourth ACM conference on Recommender systems*. ACM, 2010.
- FM
  - Rendle, Steffen, et al. "Fast context-aware recommendations with factorization machines." *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. ACM, 2011.
- BPR-MF
  - Rendle, Steffen, et al. "BPR: Bayesian personalized ranking from implicit feedback." *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*. AUAI Press, 2009.
- BPMF
  - Salakhutdinov, Ruslan, and Andriy Mnih. "Bayesian probabilistic matrix factorization using Markov chain Monte Carlo." *Proceedings of the 25th international conference on Machine learning*. ACM, 2008.
- PF
  - Gopalan, Prem, Jake M. Hofman, and David M. Blei. "Scalable recommendation with poisson factorization." *arXiv preprint arXiv:1311.1704* (2013).

References (2/3)

- Link prediction

  - Menon, Aditya Krishna, and Charles Elkan. "Link prediction via matrix factorization." *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer Berlin Heidelberg, 2011.

- Community detection

  - Yang, Jaewon, and Jure Leskovec. "Overlapping community detection at scale: a nonnegative matrix factorization approach." *Proceedings of the sixth ACM international conference on Web search and data mining*. ACM, 2013.
  - Psorakis, Ioannis, et al. "Overlapping community detection using bayesian non-negative matrix factorization." *Physical Review E* 83.6 (2011): 066114.

- Clustering

  - Ding, Chris HQ, Xiaofeng He, and Horst D. Simon. "On the Equivalence of Nonnegative Matrix Factorization and Spectral Clustering." *SDM*. Vol. 5. 2005.
  - Kuang, Da, Sangwoon Yun, and Haesun Park. "SymNMF: nonnegative low-rank approximation of a similarity matrix for graph clustering." *Journal of Global Optimization* 62.3 (2015): 545-574.

- Word embedding

  - Levy, Omer, and Yoav Goldberg. "Neural word embedding as implicit matrix factorization." *Advances in neural information processing systems*. 2014.
  - Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. "Glove: Global Vectors for Word Representation." *EMNLP*. Vol. 14. 2014.

References (3/3)

- The Matrix Cookbook
  - http://www2.imm.dtu.dk/pubdb/views/edoc_download.php/3274/pdf/imm3274.pdf
  - Collection of matrix facts, including derivatives and expected values