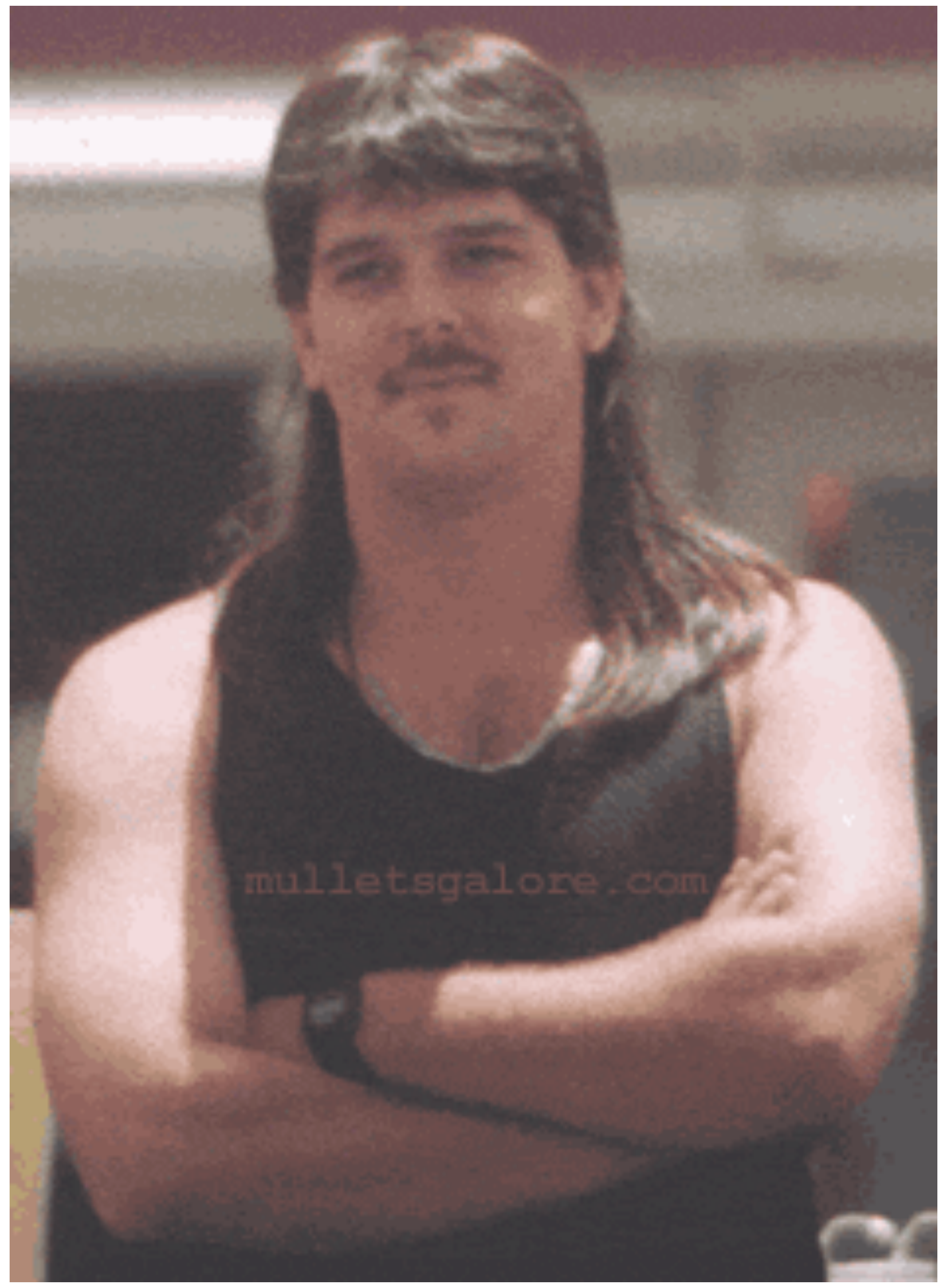
发音不标准。
不是 radiance 而是 ratings 用户对item的评分

Recommender Systems

Samarth Bhargav, Mohammad Aliannejadi, Ilya Markov

University of Amsterdam
Informatics Institute

Introduction



Customer X

Buys Metallica CD

Buys Megadeth CD

Customer Y

Searches ‘Metallica’ ...

Recommender suggests Megadeth

Widely Used

Formalisation

- Set of users: $\mathscr{U}$
- Set of items: $\mathscr{I}$
- Set of ratings already recorder: $\mathscr{R}$.
  - For user $u \in \mathscr{U}$, $i \in \mathscr{I}$, rating: r_{ui}
  - Subset of users who have rated an item i: $\mathscr{U}_i$ (likewise $\mathscr{I}_u$)
  - $\mathscr{I}_{uv} = \mathscr{I}_u \cap \mathscr{I}_v$
  - $\mathscr{U}_{ij} = \mathscr{U}_i \cap \mathscr{U}_j$
- Set of possible values for a rating: $\mathscr{S}$
- Recommendation problem: Given $\mathscr{R}$,

Formalisation

- Let $f : \mathscr{U} \times \mathscr{I} \rightarrow \mathscr{S}$
- Given these, we can get the best item for a user
  - $i* = \text{argmax}_{j \in \mathscr{I} \setminus \mathscr{I}_u} f(u_a, j)$
- Alternatively
  - The top-N recommendation task: Recommend the best N items: $L(u_a)$

Evaluation

Evaluation

- If ratings are available, an option is to use MAE/RMSE (on a test set)
- More common, however:
  - Treat it as a *ranking* problem
  - Use common ranking metrics like Precision, Recall, NDCG, etc

Recommendation Approaches

Challenges 1

- Cold Start
  - What items to recommend for new users?
  - How to infer preferences for new items?

Challenges 2

- Modelling preferences — Dynamic vs Static Preferences / Short or Long term Preferences
  - Users do not retain the same tastes
  - 'I'm in the mood for something *new*'

Challenges 3

- Exploration vs Exploitation
  - Recommend based on view/rating history? Recommend to understand preferences?
  - Related: *Serendipity* and *Diversity*
- Other: Scalability, Explainability, Transparency, Trust, ...

Recommendation Approaches

1. Content-based
2. Collaborative filtering
3. Deep recommenders
4. Other approaches

build a user profile and item profile. match these two vectors, by computing similarity

use explicit user interaction , eg ratings, likes, adding to watch list/ baskets, purchases

If two users have rated items similarly,
known: user A has rated an item, then we predict that: user B 's rating will be similar as A

use implicit user interaction , eg ratings

Recommendation Approaches

1. Content-based
2. Collaborative filtering
3. Deep recommenders
4. Other approaches

Content-based Recommendations

doc representaiton and matching

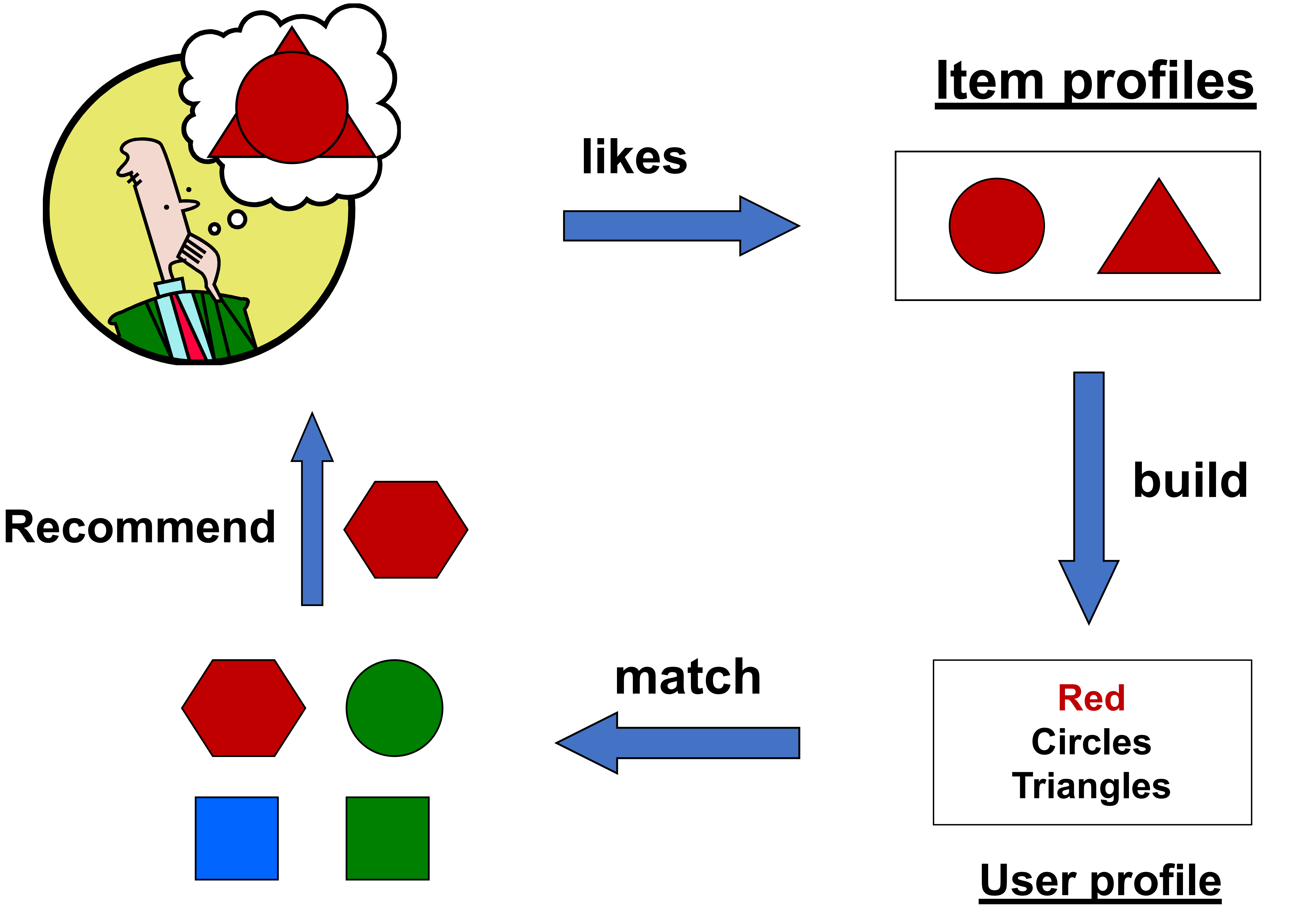
Main idea: Recommend items to customer u similar to previous items $\mathscr{I}_u$ rated highly by u

根据你曾经看过的且喜欢的东西，向你推荐新的东西
we can use similarity func

Movie recommendation: Recommend movies with same actor(s), director, genre, ...

Websites, blogs, news: Recommend other sites with “similar” content

Plan of Action



Item Profiles

For each item, create an *item profile* x_i

Profile is a set (vector) of features

meta data as vector

- Movies: author, title, actor, director,...
- Text: set of "important" words in document

How to pick important features?

- Usual heuristic from text mining is TF-IDF weight

User Profiles

- Simple: (weighted) average of (positively) rated items profiles

user profile

$$x_u = \sum_{i \in \mathcal{I}_u} r_{ui} x_i$$

rating that user gives to item xi
* item as a vector xi

- Variant: normalize weights using average rating of user

- More sophisticated aggregations possible

- Can also build classifiers/regressors to predict if a user likes an item

Recommendations

match user vector (query) with item
vector (doc)

- Suggest items whose feature vector x_i is most similar to profile vector x_u
- Cosine Similarity/Minimum Description Length

Advantages

- User independence — does not need information from other users (Collaborative Filtering requires this)
- Can handle unique tastes of users
- Unpopular items are also recommended
- Transparency — Explanations are straightforward
- Cold start (for items) is a non-issue

Drawbacks

- Feature Engineering / Domain knowledge is often needed
  - Often, content is not the only factor for users interacting with items
- Overspecialisation
  - No serendipitous recommendations (unexpected items)
- Cold start — new users

Recommendation Approaches

1. Content-based
2. Collaborative filtering
3. Deep recommenders
4. Other approaches

Collaborative Filtering

豆瓣电影，如果两个人u and v 对于一些电影打出了相似的评分，则对于一部新的电影，user v所给的评分被预测为 与user u 相近

- Collaborative (Social) filtering
  - Leverage ratings of a user u + ***other users in the system***
  - *Key Idea: If two users u and v have rated items similarly, and user v has rated an item, user u's rating will be similar*
- Content of items no longer needed!
  - Content may be a bad indicator depending on the domain/circumstances

CF Dimensions

- Methodology: Neighbourhood / Model-based
- What is being recommended: User-based / item-based
- Type of ratings: Implicit / Explicit

clicks. users do this without deep thinking

user give 7/10 as a score to a movie.
users put "like" to a movie

User/Item

- User-based

user based approach
豆瓣电影，如果两个人u and v 对于一些电影打出了相似的评分，则对于一部新的电影，user v所给的评分被预测为 与user u 相近

  - Use other users to infer ratings of an item for a user
  - 'Neighbours' — users who have similar ratings
- Item-based

某用户给甜的水果（香蕉，苹果）都打了高分，给苦的水果打了低分。现在有一个新的水果：梨子，求预测该用户打分大约多少。答：应与香蕉苹果相似的分数

  - Use ratings of similar items (that user has rated) to predict the rating for a given item

Implicit/Explicit

- Explicit Ratings
  - Like/Dislike; ImDB ratings
  - Might not be available
- Implicit

suffers from ambiguity. has bias

  - Time spent on web-page; Clicks
  - More available, but intention is unclear

Neighbourhood/Model

- Neighbourhood-based
  - Use stored ratings directly
  - Nearest neighbours
- Model-based
  - Learn a predictive model, model user-item interactions (latent factors)
  - Predict new/incomplete ratings using trained model

Neighbourhood-based Recommenders

- Explicit ratings, not sparse
  - Eric and Lucy have similar tastes
  - Eric and Diane have different tastes
- User representation — Rating vector (dimension = item)
- Consider k-nearest neighbours of user u who *rated item i*: $\mathcal{N}_i(u)$

	The Matrix	Titanic	Die Hard	Forrest Gump	Wall-E
John	5	1		2	2
Lucy	1	5	2	5	5
Eric	2	?	3	5	4
Diane	4	3	5	3	

if lucky, eric and diane
are in the same cluster.
then :

?
= neighbours'value/total
num of neighbours
=(5+3)/2 = 4

$$r_{ui} = \frac{1}{|\mathcal{N}_i(u)|} \sum_{v \in \mathcal{N}_i(u)} r_{vi}$$

Neighbourhood-based Recommenders

- But Lucy is more similar to Eric than Diane

	The Matrix	Titanic	Die Hard	Forrest Gump	Wall-E
John	5	1		2	2
Lucy	1	5	2	5	5
Eric	2	?	3	5	4
Diane	4	3	5	3	

- Consider the similarity:

$$r_{ui} = \frac{\sum_{v \in \mathcal{N}_i(u)} w_{uv} r_{vi}}{\sum_{v \in \mathcal{N}_i(u)} |w_{uv}|}$$

Given similarity of <Eric, Lucy> = 0.75 and <Eric, Diane>=0.15

$$r = \frac{0.75 \times 5 + 0.15 \times 3}{0.75 + 0.15} \approx 4.67$$

eric is more similar to lucy,
less similar to diane. so we
set diff weights: 0.75, 0.55

Neighbourhood-based Recommenders

	The Matrix	Titanic	Die Hard	Forrest Gump	Wall-E
John	5	1		2	2
Lucy	1	5	2	5	5
Eric	2	?	3	5	4
Diane	4	3	5	3	

- Similarity measures
  - Cosine Similarity
  - Pearson Correlation
  - Mean Squared Difference
  - Spearman Rank Correlation
  - ...

Neighbourhood-based Recommenders

	The Matrix	Titanic	Die Hard	Forrest Gump	Wall-E
John	5	1		2	2
Lucy	1	5	2	5	5
Eric	2	?	3	5	4
Diane	4	3	5	3	

- Users may use different rating values — '5' for John (who always rates 1/2/5) might not be equal to '5' from Diane (who rates 3/4/5)
  - Solution: Normalise the ratings per user
  - eg. Mean entering or Z-score normalization
- What if these are sparse?
  - Very likely!
  - Discussed later

5 for john and 5 for diane may be diff 5

NB Recommenders Further Reading

- Regression problem
  - Alternative: Treat it as a classification problem (Section 4.2.2, [1])
- Regression vs Classification (Section 4.2.3)
- User-based
  - Alternative: Item-based (Section 4.2.4)
- User vs Item-based (Section 4.2.5)

NB Recommenders Drawbacks

- Limited Coverage — users can be neighbours only if they rated common items
- Sparsity makes it worse as number of items increase!

Matrix Factorisation

- Attempt to solve sparsity/coverage problems by projecting user/item vectors to a dense latent space
- Two ways:
  - Decompose rating matrix
  - Decompose similarity matrix (not discussed here, see [1], Section 4.4.1.2)

Decomposing Rating Matrix

users

items

- Given R, a $|\mathcal{U}| \times |\mathcal{I}|$ matrix of rank n
  - Approximate it by $\hat{R} = PQ^T$ of rank $k < n$
  - P is a $|\mathcal{U}| \times k$ matrix of users
  - Q is a $|\mathcal{I}| \times k$ matrix of items

P: latent representation of users

similar as LSI: a matrix to represent docs, a matrix to represent query

here, user = query. item =doc

$$err(P, Q) = ||R - PQ^T||_F^2 = \sum_{u,i} (r_{ui} - p_u q_i^T)^2$$

error between true user-ite matrix and approximated matrix

Singular Value Decomposition

- SVD of $R = U\Sigma V^T$
  - U is a $|\mathscr{U}| \times n$ matrix of left singular vectors
  - V is a $|\mathscr{I}| \times n$ matrix of right singular vectors
  - Σ is a $n \times n$ diagonal matrix of singular values

Singular Value Decomposition

goal: infer missing gradients

- Select the subset of the k highest singular values/vectors Σ_k, U_k, V_k
  - $P = U_k \Sigma_k^{1/2}$ and $Q = V_k \Sigma_k^{1/2}$
  - $r_{ui} = p_u q_i^T$

advantage of collaborative filtering. advantage of all approaches: model based, neighbourhood based, matrix factorization, ect

Advantages

- No feature engineering needed!
- Does not rely on content (which may be inaccurate or unavailable)
- Can handle the *overspecialisation/diversity* problem (to an extent)
- Items recommended may not have similar content

Drawbacks

- Neighbourhood based: limited coverage/sparsity (already discussed)

热门电影总是会被推荐不管该用户是否喜欢。因为热门电影总是得到非常多的click。如果用户喜欢冷门电影，很可能此系统无法推荐冷门电影。

- Cold start still an issue!
- Popularity bias: Popular items are often recommended
- Cannot recommend to users with unique tastes (Content-based methods can!)

Recommendation Approaches

1. Content-based

2. Collaborative filtering

3. Deep recommenders

4. Other approaches

Deep Recommenders

- Scalable and can learn complex interactions
- Swap some design choices (number of neighbours, representations, etc) with others (type of NN, loss functions, etc)
- Reproducibility crisis: Only 1 (MultVAE) of 11 deep models work as claimed [2]
- Typical to use Implicit Feedback (lots of data to learn from!)

MultVAE

- Variational autoencoders for collaborative filtering [3]
- Input: Users are represented using a binary vector
- Dimension = Item is 1 if an interaction exists / 0 otherwise

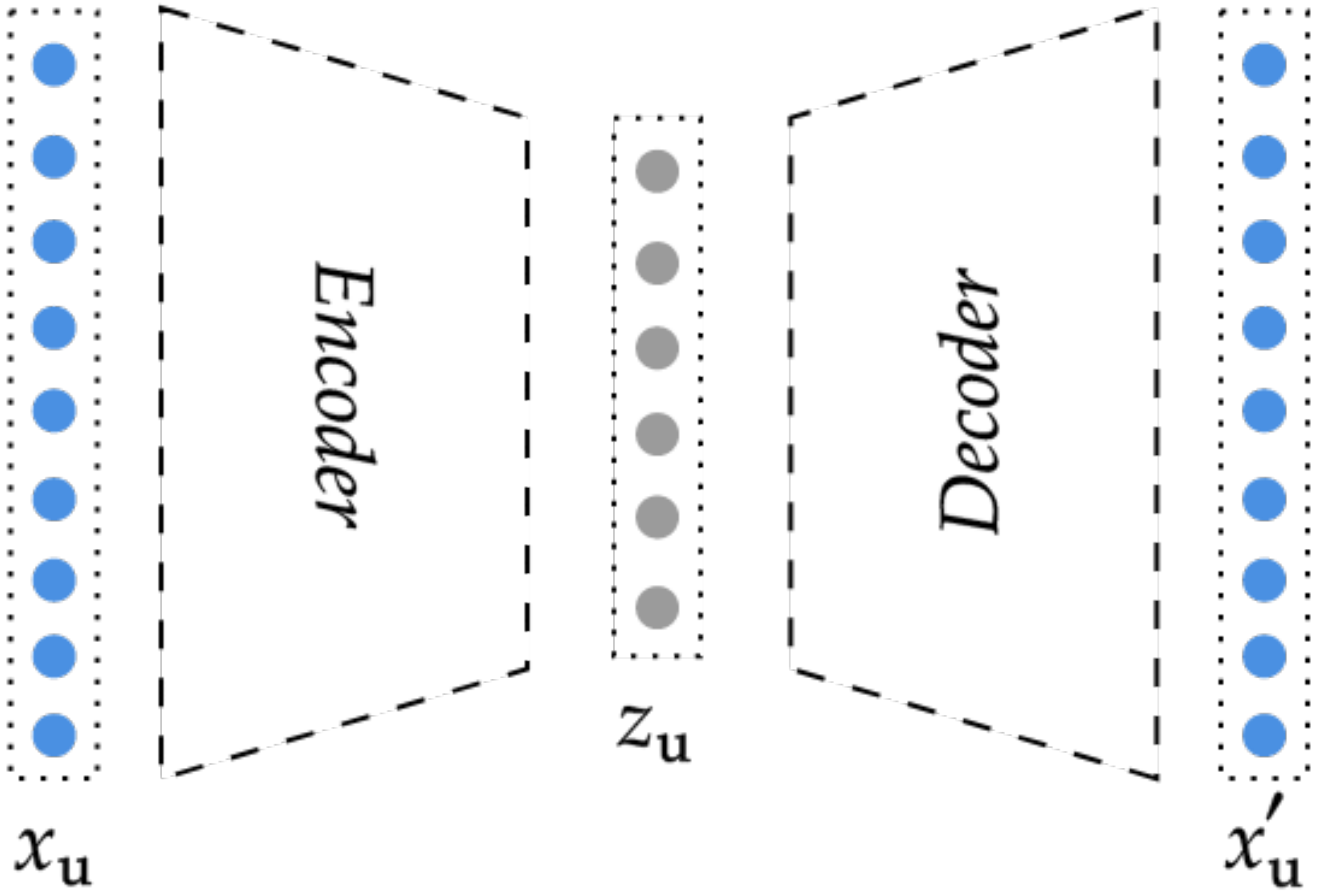
MultVAE

- Assume output distribution is Multinomial

- Regular VAE Loss

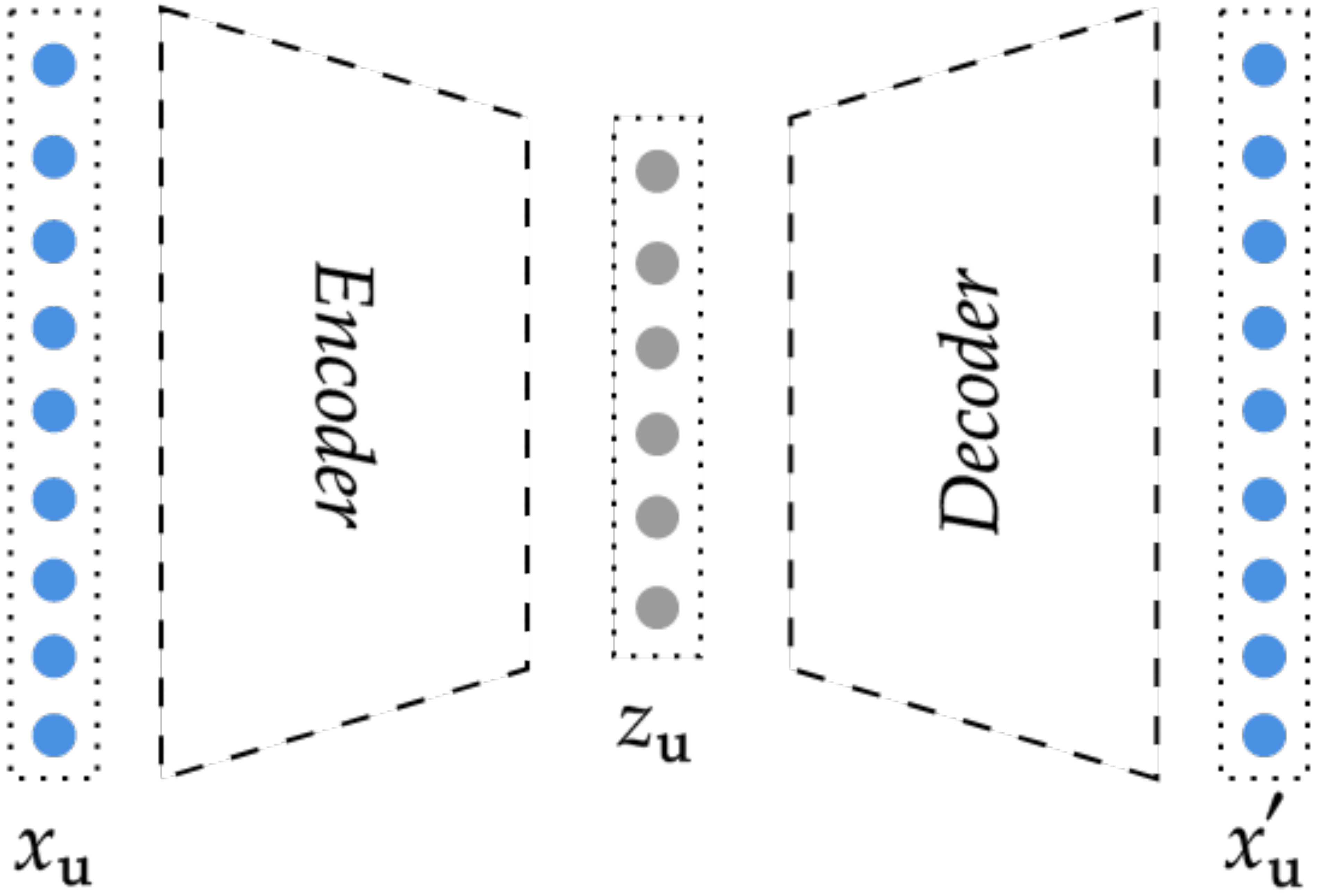
    - Reconstruction — try to reconstruct the input vector

    - Regularisation — KL Divergence between prior (Isotropic Gaussian) and posterior distribution



MultVAE

- Inference
    - Encode user history x_u and feed to encoder — decode the representation z_u
    - (after removing previously interacted items) pick items with the highest score in the output x'_u



MultVAE Experiments

- ML-20M, Netflix, Million Songs Dataset
- Explicit —> Implicit
- If Ratings >= 3.5 set to 1 or 0 otherwise

	ML-20M	**Netflix**	**MSD**
# of users	136,677	463,435	571,355
# of items	20,108	17,769	41,140
# of interactions	10.0M	56.9M	33.6M
% of interactions	0.36%	0.69%	0.14%
# of held-out users	10,000	40,000	50,000

MultVAE Results

- MultDAE — Variant with input dropout + reconstruction (no KL term)
- Baselines
  - Non-neural
    - WMF — Weighted Matrix Factorization [5]
    - SLIM — Sparse linear method [6]
  - Neural
    - CDAE — Collaborative denoising auto-encoders [7]

(a) ML-20M

	Recall@20	Recall@50	NDCG@100
Mult-VAE^{PR}	**0.395**	**0.537**	**0.426**
Mult-DAE	0.387	0.524	0.419
WMF	0.360	0.498	0.386
SLIM	0.370	0.495	0.401
CDAE	0.391	0.523	0.418

(b) Netflix

	Recall@20	Recall@50	NDCG@100
Mult-VAE^{PR}	**0.351**	**0.444**	**0.386**
Mult-DAE	0.344	0.438	0.380
WMF	0.316	0.404	0.351
SLIM	0.347	0.428	0.379
CDAE	0.343	0.428	0.376

(c) MSD

	Recall@20	Recall@50	NDCG@100
Mult-VAE^{PR}	**0.266**	**0.364**	**0.316**
Mult-DAE	**0.266**	**0.363**	0.313
WMF	0.211	0.312	0.257
SLIM	—	—	—
CDAE	0.188	0.283	0.237

Advantages

- All the advantages that come with DL — Powerful, Flexible, Scalable, etc
- Cross-pollination — adoption of advances from ML/DL

Drawbacks

- Cold start still a problem
- No consensus if performance gains are significant
  - Reproducibility [2]
  - ‘Deep’ models might not be needed — EASE — Embarrassingly shallow autoencoders [8] outperforms (most) neural baselines
  - Recent models are promising see RecVAE [9] (current SOTA)

Recommendation Approaches

1. Content-based

2. Collaborative filtering

3. Deep recommenders

4. Other approaches

Sequential Recommendation

- Short-term vs long-term preferences
- *Current* intent
  - e.g. Infer based on items viewed in the current session
- Next-item/next-basket
- See [10] for a (DL-based SeqRec) overview

Exploration vs Exploitation

- Exploitation — recommend only items that are likely to interest a user
  - What about a new user?
- Exploration — recommend other items
  - Learn user's preferences over un-encountered items

Exploration vs Exploitation

- Common approaches:
  - (Contextual, Multi-armed) Bandit [11]
  - e.g ϵ-greedy
- See [11], [12]

Conversational Recommendation

- Coming full circle
  - Tackles cold start, dynamic preferences
- If unsure about attribute/item preference — ask!
- See recent survey [13] or tutorial in SIGIR [14]

Conclusions

Summary

- We discussed
  - Content-based
  - Collaborative filtering
  - Deep recommenders
- Also *very* important (not discussed)
  - The human component
  - Explainability (See [1], Chapter 15), Trust (Chapter 20), Diversity, ...

Connections to Search/Ranking

ranking works on implicit feedback
recommendation works on explicit feedback

- Very similar
- Similar evaluation methodology — NDCG/Recall
- Users are at the centre of both
- Push (Recommendation) vs Pull (Search)

Recommended Reading

- Recommender Systems Handbook [1]
  - Chapters 1, 4, 5, 8, 15
- Nice overview of Deep Recommenders + bonus lesson on reproducibility [2]
- Sequential Recommendation [10]
- Conversational Recommandation [13, 14]

References

[1] Ricci, Francesco, Lior Rokach, and Bracha Shapira. "Introduction to recommender systems handbook." *Recommender systems handbook*. Springer, Boston, MA, 2011. 1-35.

[2] Dacrema, Maurizio Ferrari, Paolo Cremonesi, and Dietmar Jannach. "Are we really making much progress? A worrying analysis of recent neural recommendation approaches." *Proceedings of the 13th ACM Conference on Recommender Systems*. 2019.

[3] Liang, Dawen, et al. "Variational autoencoders for collaborative filtering." *Proceedings of the 2018 world wide web conference*. 2018.

[4] Kingma, Diederik P., and Max Welling. "Auto-encoding variational bayes." *arXiv preprint arXiv:1312.6114* (2013).

[5] Hu, Yifan, Yehuda Koren, and Chris Volinsky. "Collaborative filtering for implicit feedback datasets." *2008 Eighth IEEE International Conference on Data Mining*. Ieee, 2008.

[6] Ning, Xia, and George Karypis. "Slim: Sparse linear methods for top-n recommender systems." *2011 IEEE 11th International Conference on Data Mining*. IEEE, 2011.

[7] Wu, Yao, et al. "Collaborative denoising auto-encoders for top-n recommender systems." *Proceedings of the ninth ACM international conference on web search and data mining*. 2016.

[8] Steck, Harald. "Embarrassingly shallow autoencoders for sparse data." *The World Wide Web Conference*. 2019.

[9] Shenbin, Ilya, et al. "RecVAE: A new variational autoencoder for Top-N recommendations with implicit feedback." *Proceedings of the 13th International Conference on Web Search and Data Mining*. 2020.

[10] Fang, Hui, et al. "Deep Learning for Sequential Recommendation: Algorithms, Influential Factors, and Evaluations." *ACM Transactions on Information Systems (TOIS)* 39.1 (2020): 1-42.

References

[11] Li, Lihong, et al. "A contextual-bandit approach to personalized news article recommendation." *Proceedings of the 19th international conference on World wide web.* 2010.

[12] Afsar, M. Mehdi, Trafford Crump, and Behrouz Far. "Reinforcement learning based recommender systems: A survey." *arXiv preprint arXiv:2101.06286* (2021).

[13] Gao, Chongming, et al. "Advances and Challenges in Conversational Recommender Systems: A Survey." *arXiv preprint arXiv:2101.09459* (2021).

[14] Lei, Wenqiang, et al. "Conversational Recommendation: Formulation, Methods, and Evaluation." Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval. 2020.