Recommender Systems

Motivations, Challenges and Seminal Works

UTMIST Academic Talk Series 2018

Soon Chee Loong

2018, November 27

University of Toronto, Professor Scott Sanner

1. Why Learn Recommender Systems ?

2. Recommender Problem

3. Evaluating Recommender Systems

4. Algorithms

5. Further Readings

Why Learn Recommender Systems ?

Recommender Systems as a core CS Curriculum

Recommender Systems as a core CS Curriculum

Programming

- How to code ?

Recommender Systems as a core CS Curriculum

Programming

- How to code ?

Objected-Oriented Programming

- How to write large amount of code (readable, easy to use) ?

Recommender Systems as a core CS Curriculum

Programming

- How to code ?

Objected-Oriented Programming

- How to write large amount of code (readable, easy to use) ?

Algorithms

- How to write efficient code (performance, time, space) ?

Recommender Systems as a core CS Curriculum

Programming

- How to code ?

Objected-Oriented Programming

- How to write large amount of code (readable, easy to use) ?

Algorithms

- How to write efficient code (performance, time, space) ?

Operating Systems

- How to write concurrent code on a single computer (synchronization) ?

Recommender Systems as a core CS Curriculum

Programming

- How to code ?

Objected-Oriented Programming

- How to write large amount of code (readable, easy to use) ?

Algorithms

- How to write efficient code (performance, time, space) ?

Operating Systems

- How to write concurrent code on a single computer (synchronization) ?

Distributed Systems

- How to write parallel code on multiple computers ?

Recommender Systems as a core CS Curriculum

Programming

- How to code ?

Objected-Oriented Programming

- How to write large amount of code (readable, easy to use) ?

Algorithms

- How to write efficient code (performance, time, space) ?

Operating Systems

- How to write concurrent code on a single computer (synchronization) ?

Distributed Systems

- How to write parallel code on multiple computers ?

Databases

- How to store large information ?

Recommender Systems as a core CS Curriculum

Programming

- How to code ?

Objected-Oriented Programming

- How to write large amount of code (readable, easy to use) ?

Algorithms

- How to write efficient code (performance, time, space) ?

Operating Systems

- How to write concurrent code on a single computer (synchronization) ?

Distributed Systems

- How to write parallel code on multiple computers ?

Databases

- How to store large information ?

Information Retrieval

- How to retrieve queried information from databases ?

Recommender Systems as a core CS Curriculum

Programming

- How to code ?

Objected-Oriented Programming

- How to write large amount of code (readable, easy to use) ?

Algorithms

- How to write efficient code (performance, time, space) ?

Operating Systems

- How to write concurrent code on a single computer (synchronization) ?

Distributed Systems

- How to write parallel code on multiple computers ?

Databases

- How to store large information ?

Information Retrieval

- How to retrieve queried information from databases ?

Recommender Systems

- How to recommend things that user may like ?

Recommender Systems as a core CS Curriculum

Programming

- How to code ?

Objected-Oriented Programming

- How to write large amount of code (readable, easy to use) ?

Algorithms

- How to write efficient code (performance, time, space) ?

Operating Systems

- How to write concurrent code on a single computer (synchronization) ?

Distributed Systems

- How to write parallel code on multiple computers ?

Databases

- How to store large information ?

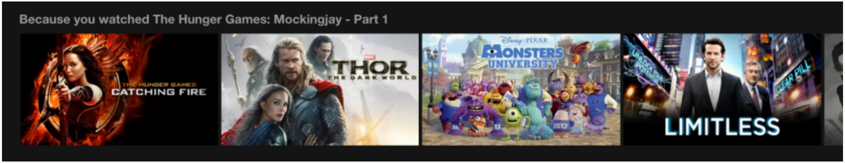
Information Retrieval

- How to retrieve queried information from databases ?
- **The user searches for something they are looking for**

Recommender Systems

- How to recommend things that user may like ?
- **The user discovers something they like but didn't know exist**

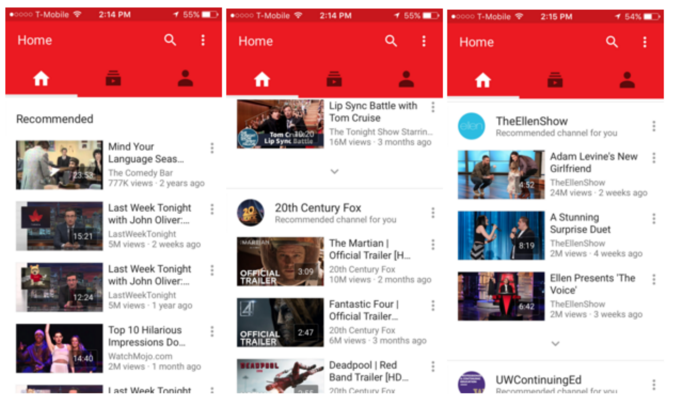
Real world business applications



Netflix Movies Recommendations

Netflix Movie Recommender worth about $1 billion.
Netflix $1 million challenge in 2009 popularize the recommendation problem.

Real world business applications



Youtube Videos Recommendations

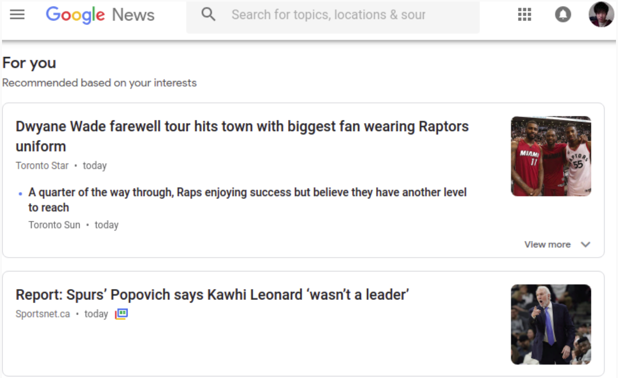
Youtube Video Recommender accounts for [60% of video clicks](#).

Real world business applications

Amazon Shopping Recommendations

Amazon Shopping Recommender accounts for 35% of sales.

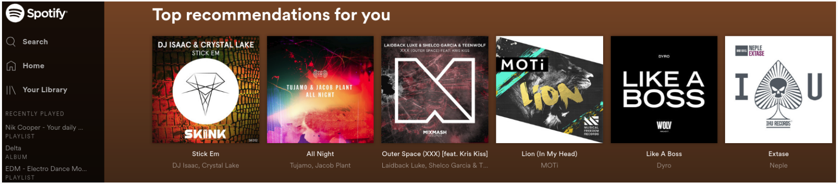
Real world business applications



Google News Recommendations

Google News Recommender generates [38% more clicks](). Popular news (items) changes daily.

Real world business applications



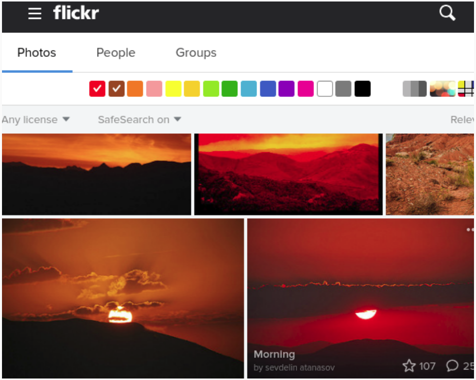
Spotify Music Recommendations

Spotify Music Recommender streams 5 billion tracks in a year.

Spotify RecSys Challenge 2018

User's music taste changes daily.

Real world business applications



Flickr Image Recommendations

Flickr Image Recommender recommends red and brown mountains.

Recommender Systems as Applied Machine Learning

Different types of data and companies.

- E-commerce: Amazon, Alibaba
- Text: Yelp, CiteULike, Kobo
- Image: Instagram, Pinterest, Flickr, 500px
- Video: Netflix, Hulu, Youtube
- Geo-location: Yelp
- Music: Spotify, Pandora, LastFM
- News: Google News, Yahoo! News
- Social: Facebook, Tinder, LinkedIn

Recommender Systems as Applied Machine Learning

Different domain-specific characteristics

- Movie Recsys: Text Reviews, ratings
- News Recsys: Dynamic, hot news changes daily, text content
- Book Recsys: Text, Sequential (Harry Potter 1 till 7)
- Social Recsys: Social Graph
- Travel Recsys: Must satisfy constraints (same location, commute time)
- Image Recsys: Images, Fashion, requires subjective evaluation
- Tag Recsys: Image to text (Photo Tagging), Text summarization (HashTags)
- Music Recsys: can listen more than once, session-based (depends on user's current mood)

Life Hacks

Life Hacks

Control Distraction

- You only get distracted by videos you like.

Life Hacks

Life Hacks

Control Distraction

- You only get distracted by videos you like.
    - Using Incognito mode causes bad recommendations that would not distract you.

Life Hacks

Life Hacks

Control Distraction

- You only get distracted by videos you like.
  - Using Incognito mode causes bad recommendations that would not distract you.

Understanding Recommendations

- Why did I get recommended dating apps on Facebook?

Life Hacks

Life Hacks

Control Distraction

- You only get distracted by videos you like.
  - Using Incognito mode causes bad recommendations that would not distract you.

Understanding Recommendations

- Why did I get recommended dating apps on Facebook?
  - Because your friends use dating apps.

Life Hacks

Life Hacks

Control Distraction

- You only get distracted by videos you like.
  - Using Incognito mode causes bad recommendations that would not distract you.

Understanding Recommendations

- Why did I get recommended dating apps on Facebook?
  - Because your friends use dating apps.

Influencing Recommendations

- I am only attracted to muscular people.

Life Hacks

Life Hacks

Control Distraction

- You only get distracted by videos you like.
  - Using Incognito mode causes bad recommendations that would not distract you.

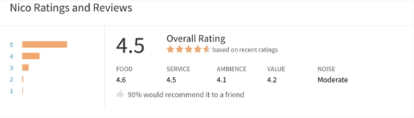
Understanding Recommendations

- Why did I get recommended dating apps on Facebook?
  - Because your friends use dating apps.

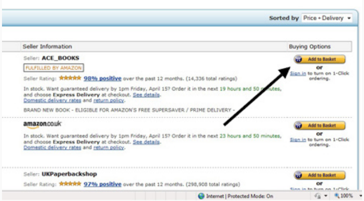
Influencing Recommendations

- I am only attracted to muscular people.
  - Stop swiping right on everyone!
  - Only swipe right to muscular people.

Recommender Problem



Ratings



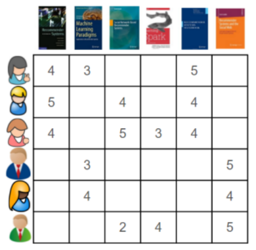
Purchases

What type of data does recommender systems have access to?

- Explicit Data: Ratings, Purchases, Right Swipe, Likes
- Implicit Data: Clicks, views

Recommender Mathematical Formulation

N_u = Number of users
N_i = Number of items
R = Rating Matrix $\in (N_u, N_i)$
R_{train} = Train Rating Matrix
R_{test} = Test Rating Matrix
$R = R_{train} \cup R_{test} = R_{observed} \cup R_{missing}$
$R_{train} \cap R_{test} = \emptyset = R_{observed} \cap R_{missing}$
$\hat{R} = f(R_{train})$ = Predicted Matrix
f = Recommender System

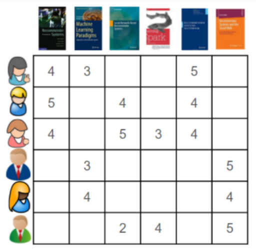


User-Item Rating Matrix.

Predict missing entries of the Rating Matrix, $R_{missing}$.
Then, recommend the Top-K missing entries for each user.

Recommender Mathematical Formulation

N_u = Number of users
N_i = Number of items
R = Rating Matrix $\in (N_u, N_i)$
R_{train} = Train Rating Matrix
R_{test} = Test Rating Matrix
$R = R_{train} \cup R_{test} = R_{observed} \cup R_{missing}$
$R_{train} \cap R_{test} = \emptyset = R_{observed} \cap R_{missing}$
$\hat{R} = f(R_{train})$ = Predicted Matrix
f = Recommender System



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

User-Item Rating Matrix.

Predict missing entries of the Rating Matrix, $R_{missing}$.
Then, recommend the Top-K missing entries for each user.
Or, recommend Top-K missing entries for each item. (Ads Serving)

Evaluating Recommender Systems

Rating Prediction Error Evaluation

Minimize difference between R_{test} and $\hat{R}_{test}$.

$$minimize \quad ||R_{test} - \hat{R}_{test}||_p$$

Root Mean Square Error (RMSE) , $p = 2$
Mean Absolute Error (MAE) , $p = 1$

Rating Prediction Error Evaluation

Minimize difference between R_{test} and $\hat{R}_{test}$.

$$minimize \quad ||R_{test} - \hat{R}_{test}||_p$$

Root Mean Square Error (RMSE) , $p = 2$
Mean Absolute Error (MAE) , $p = 1$

Problem with Rating Prediction Error Measures

Minimizing RMSE does not necessarily improve Top-K recommendation performance [4].

Set Evaluation

Precision@K

Out of the top-K you recommended (top K items predicted as positive), how precise were you ?

$$Precision@K_u = \frac{|\hat{R}_{test_{(u,topK)}} \cap R_{test_{(u,topK)}}|}{|\hat{R}_{test_{(u,topK)}}|} = \frac{|\hat{R}_{test_{(u,topK)}} \cap R_{test_{(u,topK)}}|}{K}$$

$$Precision@K = \frac{1}{N_u} \sum_{u \in U} Precision@K_u$$

Set Evaluation

Precision@K

Out of the top-K you recommended (top K items predicted as positive), how precise were you ?

$$Precision@K_u = \frac{|\hat{R}_{test_{(u,topK)}} \cap R_{test_{(u,topK)}}|}{|\hat{R}_{test_{(u,topK)}}|} = \frac{|\hat{R}_{test_{(u,topK)}} \cap R_{test_{(u,topK)}}|}{K}$$

$$Precision@K = \frac{1}{N_u} \sum_{u \in U} Precision@K_u$$

Problem with Precision@K

Difficult to select appropriate K.
Every user has varying number of test items.

Set Evaluation

R-Precision

Out of the K possible recalled observed items for current user u, how precise were you ?

$$RPrecision_u = \frac{|\hat{R}_{test_{(u,topK_u)}} \cap R_{test_{(u,topK_u)}}|}{|\hat{R}_{test_{(u,topK_u)}}|} = \frac{|\hat{R}_{test_{(u,topK_u)}} \cap R_{test_{(u,topK_u)}}|}{K_u}$$

$$RPrecision = \frac{1}{N_u} \sum_{u \in U} RPrecision_u \quad , K_u = |\hat{R}_{test_u}|$$

R-Precision

Out of the K possible recalled observed items for current user u, how precise were you ?

$$RPrecision_u = \frac{|\hat{R}_{test_{(u,topK_u)}} \cap R_{test_{(u,topK_u)}}|}{|\hat{R}_{test_{(u,topK_u)}}|} = \frac{|\hat{R}_{test_{(u,topK_u)}} \cap R_{test_{(u,topK_u)}}|}{K_u}$$

$$RPrecision = \frac{1}{N_u} \sum_{u \in U} RPrecision_u \quad , K_u = |\hat{R}_{test_u}|$$

Problem with Set Evaluation

Doesn't account for order within top K recommendations.

Set Evaluation

Problem with Set Evaluation

Doesn’t account for order within top K.

Ranking Binarized Evaluation

AveragePrecision@K

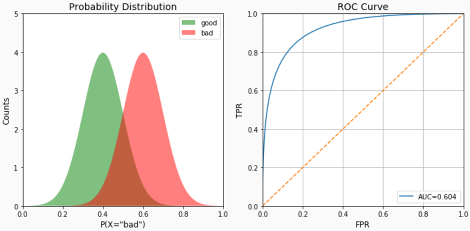
Precision Averaged over K monotonically increasing sets.

$$AveragePrecision@K_u = \frac{1}{K}\sum_{i=1}^{K} Precision@i_u$$

$$AveragePrecision@K = \frac{1}{N_u}\sum_{u \in U} AveragePrecision@K_u$$

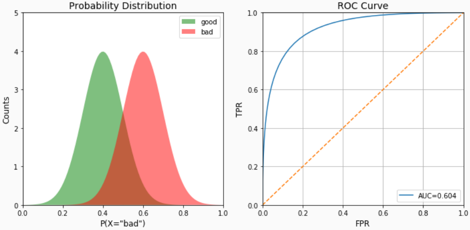
Ranking Binarized Evaluation

Area Under Receiver Operating Characteristic Curve



Visualize performance using ROC Curve

Area Under Receiver Operating Characteristic Curve



Visualize performance using ROC Curve

Can approximate as: [12]

$$AUC_u = \sum_{i=1}^{N_i} \sum_{j=i}^{N_i} \delta(R_{(u,i)} \geq R_{(u,j)})$$

$$i \in \{1st, 2nd, ...\}$$

Ranking Binarized Evaluation

Reciprocal Rank

Minimize number of left swipes before the next right swipe.
Minimize number of songs skipped.

$$ReciprocalRank_u = \frac{1}{rank(firstRelevant)}$$

$$ReciprocalRank = \frac{1}{N_u} \sum_{u \in U} ReciprocalRank_u$$

Ranking Binarized Evaluation

Clicks

For queries, want to minimize number of next page clicks, each page has m results.

$$Clicks_u = \left\lfloor \frac{rank(firstRelevant) - 1}{m} \right\rfloor \quad , rank \in \{1, 2, ...\}$$

$$Clicks = \frac{1}{N_u} \sum_{u \in U} Clicks_u$$

Ranking Binarized Evaluation

Problem with Ranking Binarized Evaluation

Doesn't make use of ordinal values within observed.

Ranking Graded Relevance Evaluation

Normalized Discounted Cumulative Gain

Calculate cumulative gain based on how much is gain. Discount more as you go further down the list.

Ranking Graded Relevance Evaluation

Normalized Discounted Cumulative Gain
Calculate cumulative gain based on how much is gain. Discount more as you go further down the list.

Cumulative Regret
Accumulate regret as decisions are made.

Diversity Evaluation

Maximal Marginal Relevance
NCall@K
Weighted Precision Recall

Click Through Rate

Click Through Rate
Number of clicks on recommendation

Problem with Online Evaluation

Requires real customers.
Expensive (production level code, and reporting)
May lose user's confidence in recommender system.

Algorithms

Collaborative Filtering

Collaborative Filtering [13]

K-Nearest Neighbour on recommender systems.

Collaborative: Predict a user's opinion based on other user's opinions.

Filtering: Filter based on highest similarity score.

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

Rating Matrix

User Similarity, S_u

Cosine Similarity is robust towards vector length.

$$S_{u_{cosine}} = \frac{R_{train} \cdot R_{train}^T}{|R_{train}||R_{train}^T|}$$

$$\hat{R}_{u_{test}} = \frac{S_{u_{cosine}} \cdot R_{test}}{|S_{u_{cosine}}|}$$

1.00	0.75	0.63	0.22	0.30	0.00
0.75	1.00	0.91	0.00	0.00	0.16
0.63	0.91	1.00	0.00	0.00	0.40
0.22	0.00	0.00	1.00	0.97	0.64
0.30	0.00	0.00	0.97	1.00	0.53
0.00	0.16	0.40	0.64	0.53	1.00

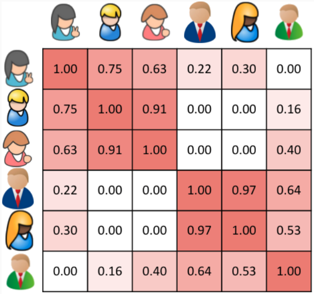
User Similarity Matrix

User Similarity, S_u

Pearson Similarity is robust towards user's feedback biases.

$$S_{u_{pearson}} = \frac{(R_{train} - \bar{u}) \cdot (R_{train}^T - \bar{u})}{|R_{train} - \bar{u}||R_{train}^T - \bar{u}|}$$

$$\hat{R}_{u_{test}} = \frac{S_{u_{pearson}} \cdot (R_{test} - \bar{u})}{|S_{u_{pearson}}|} + \bar{u}$$



1.00	0.75	0.63	0.22	0.30	0.00
0.75	1.00	0.91	0.00	0.00	0.16
0.63	0.91	1.00	0.00	0.00	0.40
0.22	0.00	0.00	1.00	0.97	0.64
0.30	0.00	0.00	0.97	1.00	0.53
0.00	0.16	0.40	0.64	0.53	1.00

User Similarity Matrix

Item Similarity, S_u

$$S_{i_{cosine}} = \frac{R_{train}^T \cdot R_{train}}{|R_{train}^T||R_{train}|}$$

$$\hat{R}_{i_{test}} = \frac{R_{test} \cdot S_{i_{cosine}}}{|S_{i_{cosine}}|}$$

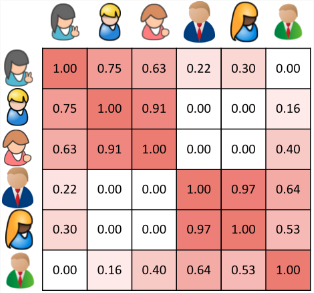
1.00	0.27	0.79	0.32	0.98	0.00
0.27	1.00	0.00	0.00	0.34	0.65
0.79	0.00	1.00	0.69	0.71	0.18
0.32	0.00	0.69	1.00	0.32	0.49
0.98	0.34	0.71	0.32	1.00	0.00
0.00	0.65	0.18	0.49	0.00	1.00

Item Similarity Matrix

Collaborative Filtering

User Similarity, S_u

These are what people similar to you like.



1.00	0.75	0.63	0.22	0.30	0.00
0.75	1.00	0.91	0.00	0.00	0.16
0.63	0.91	1.00	0.00	0.00	0.40
0.22	0.00	0.00	1.00	0.97	0.64
0.30	0.00	0.00	0.97	1.00	0.53
0.00	0.16	0.40	0.64	0.53	1.00

User Similarity Matrix

Item Similarity, S_i

These are items similar to what you have liked before

1.00	0.27	0.79	0.32	0.98	0.00
0.27	1.00	0.00	0.00	0.34	0.65
0.79	0.00	1.00	0.69	0.71	0.18
0.32	0.00	0.69	1.00	0.32	0.49
0.98	0.34	0.71	0.32	1.00	0.00
0.00	0.65	0.18	0.49	0.00	1.00

Item Similarity Matrix

Collaborative Filtering

Problem with Recommender Datasets

Scalability

Lots of users. Lots of items.

Sparse Matrix

Mostly unobserved

Collaborative Filtering

Problem with Recommender Datasets

Scalability

Lots of users. Lots of items.

Sparse Matrix

Mostly unobserved

Problem with Collaborative Filtering

Similarity Matrix is Dense (can't scale to real-world dataset).

Sparse Matrix results in 0 similarity between most pairs.

Collaborative Filtering

Problem with Recommender Datasets

Scalability

Lots of users. Lots of items.

Sparse Matrix

Mostly unobserved

Problem with Collaborative Filtering

Similarity Matrix is Dense (can't scale to real-world dataset).

Sparse Matrix results in 0 similarity between most pairs.

Approaches to Dense Similarity Matrix

Use Sparse Matrix Computation (scipy.sparse) to solve memory issue.

Run in parallel on many machines (Apache Spark) to solve runtime issue.

Cluster user and items to minimize computation on only K Nearest Neighbours.

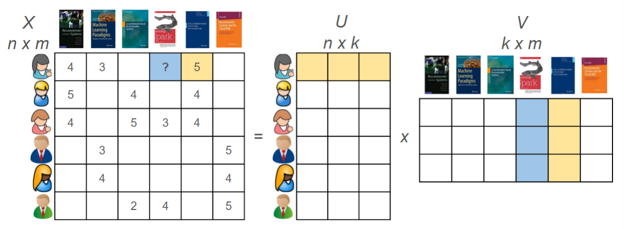
Matrix Factorization

Matrix Factorization solves Sparse Matrix

Reduce dimension first, then do dot product.

$$U \in (n_u, \mathbf{d}) \quad , I \in (n_i, \mathbf{d})$$

$$\hat{R} = U \cdot I^T$$



Matrix Factorization

Matrix Factorization

Instead of projecting into User Space or Item Space, just project into the same shared latent space.

Connections to Collaborative Filtering

$$U = R \quad , I = R^T$$

$$d_u = n_i \quad , d_i = n_u, \mathbf{d} << min(n_i, nu)$$

Requires two-step process

$$S_u \approx R \cdot R^T = U \cdot U^T \quad , \hat{R} \approx S_u \cdot R$$

$$S_i \approx R^T \cdot R = I \cdot I^T \quad , \hat{R} \approx R^T \cdot S_i$$

Matrix Factorization

Pure Singular Value Decomposition (PureSVD) [4]

Literally do SVD.

Eckart Young Theorem [5]

SVD is the optimal low rank approximation to minimize Frobenius Norm.

$$Loss = ||R_{train} - \hat{R}_{train}||_F^2$$

$$\hat{R}_{train} = U\Sigma V^T = SVD(R_{train})$$

Pure Singular Value Decomposition (PureSVD) [4]

Literally do SVD.

Eckart Young Theorem [5]

SVD is the optimal low rank approximation to minimize Frobenius Norm.

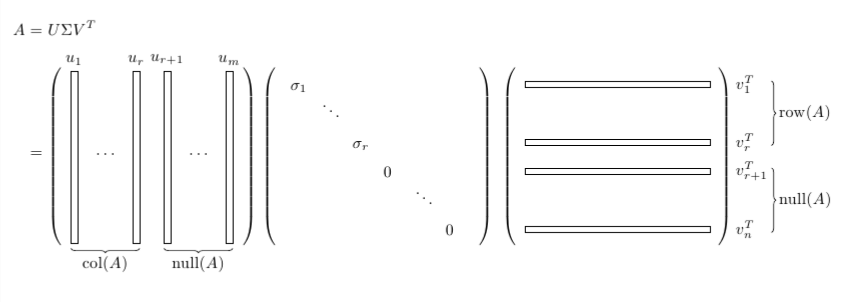
$$Loss = ||R_{train} - \hat{R}_{train}||_F^2$$

$$\hat{R}_{train} = U\Sigma V^T = SVD(R_{train})$$

Problem with PureSVD

Overfits to zeroes in the sparse matrix.

Matrix Factorization



Singular Value Decomposition

For prediction, index into the trained user and item features.

$$\hat{R}_{test_{(u,i)}} = U[u] \cdot \Sigma \cdot V[i]^T$$

Matrix Factorization

Probabilistic Matrix Factorization (PMF) [9]

Train only on observed.

Optimize loss function directly using Stochastic Gradient Descent.

Derive loss function from log of Multivariate Gaussian.

$$Loss = \sum_{u \in R} \sum_{i \in R} I_{observed} * (R_{train_{(u,i)}} - (U_u \cdot I_i^T))^2 + regularizationPriors$$

Matrix Factorization

Probabilistic Matrix Factorization (PMF) [9]

Train only on observed.

Optimize loss function directly using Stochastic Gradient Descent.

Derive loss function from log of Multivariate Gaussian.

$$Loss = \sum_{u \in R} \sum_{i \in R} I_{observed} * (R_{train_{(u,i)}} - (U_u \cdot I_i^T))^2 + regularizationPriors$$

Problem with Probabilistic Matrix Factorization

Only trains on the observed, doesn't train on unobserved.

Matrix Factorization

Other notable Matrix Factorization Models:
MMMF [16], NNMF [6], fLDA [1], LLORMA [8]

Problem with Matrix Factorization

Simple model: Only linear dot product.
Optimizes for rating in it's objective NOT ranking.
Doesn't make use of cosine similarity due to non-convexity of optimization. Hence, will overshoot.

Non-Linearity

Replace prediction with nonlinear model, $f_{nonlinear}$.

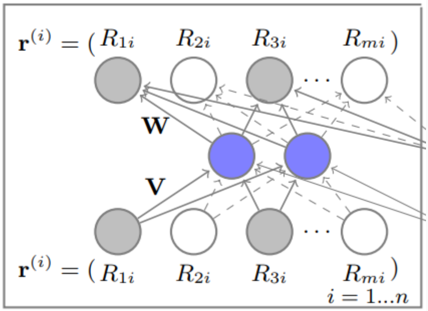
$$Loss = ||R_{train} - \hat{R}_{train}||$$

$$\hat{R}_{train} = f_{nonlinear}(R_{train})$$

Non-Linearity

AutoEncoder Recommender (AutoRec) [15]

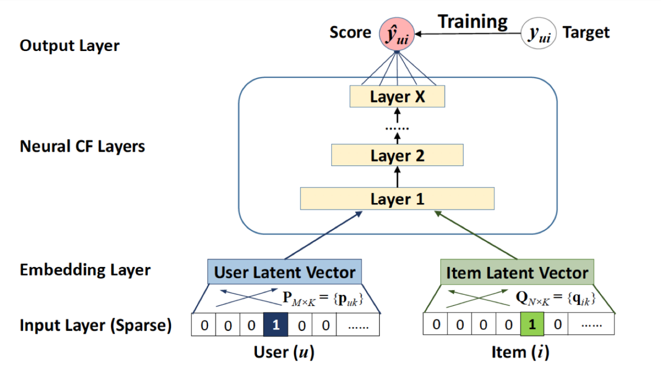
Neural Networks with Encoder Decoder models



AutoRec

Non-Linearity

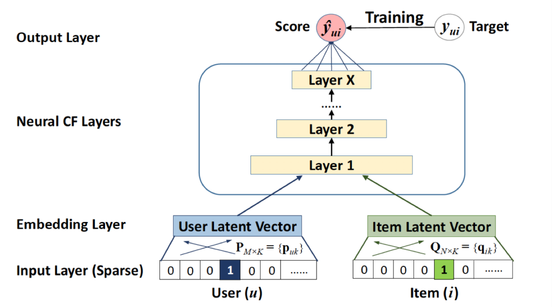
Neural Collaborative Filtering (NCF) [7]



NCF

Non-Linearity

Neural Collaborative Filtering (NCF) [7]



NCF

Neural Networks Problem

Many local optima.
Annoying to train.
Can't interpret
(no similarity matrix).

One-Class Implicit Feedback

Problem with Recommender Datasets

One-Class Feedback

Only have data for positive (purchases) items

A form of **Semi-supervised Learning**.

Unobserved are either positive or negative.

One-Class Implicit Feedback

Problem with Recommender Datasets

One-Class Feedback

Only have data for **positive** (purchases) items

A form of **Semi-supervised Learning**.

Unobserved are either positive or negative.

Approaches to One-Class Implicit Feedback Dataset

Cost Sensitive Learning

- Unobserved popular more likely to be negative.
- Unobserved unpopular more likely to be unknown.

One-Class Implicit Feedback

Weighted Regularized Matrix Factorization (WRMF) [11]

Uses weighted hyperparameter as a form of Cost Sensitive Learning.

$$weight_{(u,i)} = cost_{(u,i)} = c_{(u,i)} = \beta + \alpha * R_{train_{(u,i)}}$$

Locally Weighted Regression [3] objective

$$Loss = \sum_{u \in R} \sum_{i \in R} c_{(u,i)} * (R_{train_{(u,i)}} - (U_u \cdot I_i^T))^2 + regularization$$

note: Unlike PMF, it accounts for unobserved in it's objective. Hence, it looks dense!

One-Class Implicit Feedback

Weighted Regularized Matrix Factorization (WRMF) [11]

Closed-Form Solution using Alternating Least Square

$$U_u = (I^T \cdot C_u \cdot I)^{-1} \cdot I^T \cdot C_u \cdot R_{train_u}$$

$$I_i = (U^T \cdot C_i \cdot U)^{-1} \cdot U^T \cdot C_i \cdot R_{train_i}$$

Major contribution of WRMF

$$I^T \cdot C_u \cdot I = I^T \cdot diag(\beta) \cdot I + I^T \cdot (C_u - diag(\beta)) \cdot I$$

Hence, optimization is independent of unobserved although it accounts for unobserved in its objective.

One-Class Implicit Feedback

Weighted Regularized Matrix Factorization (WRMF) [11]

Closed-Form Solution using Alternating Least Square

$$U_u = (I^T \cdot C_u \cdot I)^{-1} \cdot I^T \cdot C_u \cdot R_{train_u}$$

$$I_i = (U^T \cdot C_i \cdot U)^{-1} \cdot U^T \cdot C_i \cdot R_{train_i}$$

Major contribution of WRMF

$$I^T \cdot C_u \cdot I = I^T \cdot diag(\beta) \cdot I + I^T \cdot (C_u - diag(\beta)) \cdot I$$

Hence, optimization is independent of unobserved although it accounts for unobserved in its objective.

Problem with WRMF

Can only give a fixed constant cost to unobserved.
ALS is slow (alternate, not parallelizable).

One-Class Implicit Feedback

Bayesian Personalized Ranking (BPR) [12]

Non-Uniform Sampling as a form of Cost Sensitive Learning.

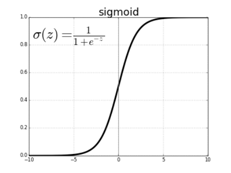
Ranking Objective Optimize AUC directly

Train on pairs of ratings instead of single ratings.

$$Loss = -\sum_{u \in U} \sum_{i \in R_u} \sum_{j \notin R_u} P(R_{train_{(u,i)}} \geq R_{train_{(u,j)}})$$

Uses sigmoid function to be differentiable.

Optimize using SGD.



Sigmoid

Problem with BPR

Sampling is slow.

Linear Optimization

Alternating over both user and item features is slow.
Just optimize over item similarity matrix directly.

$$Loss = ||R_{train} - (Rtrain \cdot W)||$$

Sparse Linear Methods (SLIM) [10]

$$Loss = ||R_{train} - (R_{train} \cdot W)||_F^2$$

To prevent the trivial solution of Identity Matrix.

$$diag(W) = 0$$

note: Initialize diagonals to 0 and don't train it instead to indirectly satisfy trivial solution constraint.

Problem with SLIM

Inversion is still cubic (n_i^3) operation.
Similarity matrix is dense.

Projected Linear Recommender (PLRec, Linear Flow) [14]

Reduce dimension first. Then, optimize.

$$U \cdot \Sigma \cdot V^T = SVD(R_{train})$$

$$Loss = ||R_{train} - (R_{train} \cdot V \cdot \sqrt{\Sigma} \cdot W)||_F^2$$

To prevent the trivial solution of Identity Matrix.

$$diag(W) = 0$$

Hence, W is a much smaller item similarity matrix.

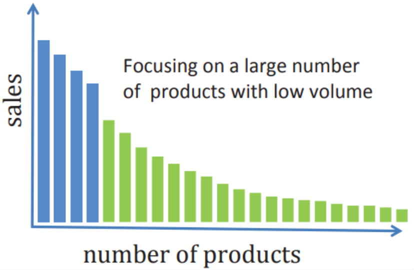
Problem with Recommender Datasets

Long Tail (popularity-bias) [2]

Most data belong to a small subset of items.

Known as **Imbalanced Dataset** in Machine Learning.

Training is biased towards popular items.



Long Tail [19]

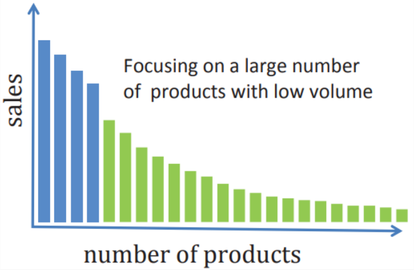
Diversity

However, want recommendation to be diverse as more profit [19].
How to recommend a diverse set of items instead of just popular ones?

Diversity

Diversity

Random exploration will be difficult to improve ranking more than popularity baseline on standard ranking metrics.



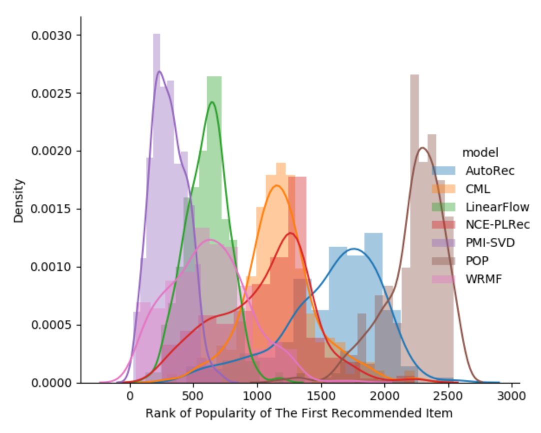
Long Tail

Predicting popular items only has good chance of hit.

Noise Contrastive Estimation PLRec (NCE-PLRec) [18]

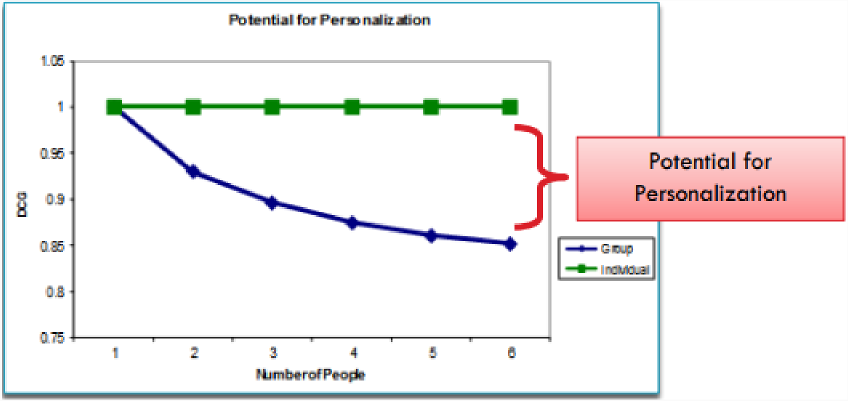
Derivation leads to de-popularized matrix.

Transfer probability mass from unobserved popular to observed.



Diverse Recommendations of NCE-PLRec

Personalization



Potential For Personalization [17]

A single universal ranking is suboptimal.

How to personalized recommendation to users instead of recommend similar items to all users?

How to personalized recommendation to users instead of recommend similar items to all users?

Approaches to Personalized Models

Extra user-specific parameters

- Train different models for each user
- Train different models for each cluster of users.
- Predict user separately based on historical purchases.

How to personalized recommendation to users instead of recommend similar items to all users?

Approaches to Personalized Models

Extra user-specific parameters

- Train different models for each user
- Train different models for each cluster of users.
- Predict user separately based on historical purchases.

Cost-Sensitive Learning based on popularity of observed item

- Popular, less personalized
- Unpopular, more personalized

Cold-Start

Problem with Recommender Datasets

Cold Start

- What item to recommend to a new user?
- Can’t train on user with no items at all.
- Can’t predict on user with no items. Dot product is 0.

Cold-Start

Problem with Recommender Datasets

Cold Start

- What item to recommend to a new user?
- Can't train on user with no items at all.
- Can't predict on user with no items. Dot product is 0.

Approaches to Cold-Start Problem

Content-based Collaborative Filtering

Use extra content information where there are overlaps (not cold)

Cold-Start

Problem with Recommender Datasets

Cold Start

- What item to recommend to a new user?
- Can't train on user with no items at all.
- Can't predict on user with no items. Dot product is 0.

Approaches to Cold-Start Problem

Content-based Collaborative Filtering

Use extra content information where there are overlaps (not cold)

- Form similarity using content, then proceed as usual.

Cold-Start

Problem with Recommender Datasets

Cold Start

- What item to recommend to a new user?
- Can’t train on user with no items at all.
- Can’t predict on user with no items. Dot product is 0.

Approaches to Cold-Start Problem

Content-based Collaborative Filtering

Use extra content information where there are overlaps (not cold)

- Form similarity using content, then proceed as usual.
- Tensor Factorization.

Cold-Start

Problem with Recommender Datasets

Cold Start

- What item to recommend to a new user?
- Can't train on user with no items at all.
- Can't predict on user with no items. Dot product is 0.

Approaches to Cold-Start Problem

Content-based Collaborative Filtering

Use extra content information where there are overlaps (not cold)

- Form similarity using content, then proceed as usual.
- Tensor Factorization.

Bandit Approaches

Change behavior based on feedback to new recommendations.

Further Readings

Major Topics Uncovered

Further Topics

- **Sequential Recommender**: Order in recommendation
- **Dynamic Recommender**: Changing user preference
- **Graph Recommender**: Matrix as Bipartite Graph
- **Ensemble Recommender**: User, Item Features

Research Conferences for Recommender Systems

- RecSys (Think of it as CVPR for Recommender)
- KDD (Think of it as NeurIPS for Applied Machine Learning, practical empirical success)
- WSDM (Think of it as ICML for Web-related research, theoretical)
- WWW (Recommender works in the web)
- SIGIR (Think of it as CVPR for Information Retrieval)
- ICML/NIPS/AAAI (Recommender is a semi-supervised ranking problem)

References i

[1] D. Agarwal and B.-C. Chen.
flda: matrix factorization through latent dirichlet allocation.
In *Proceedings of the third ACM international conference on Web search and data mining*, pages 91–100. ACM, 2010.

[2] C. Anderson.
The long tail.
Wired magazine, 12(10):170–177, 2004.

[3] A. Christopher, M. Andrew, and S. Stefan.
Locally weighted learning.
Artif Intell Rev, 11(1-5):11–73, 1997.

References ii

[4] P. Cremonesi, Y. Koren, and R. Turrin.
Performance of recommender algorithms on top-n recommendation tasks.
In *Proceedings of the fourth ACM conference on Recommender systems*, pages 39–46. ACM, 2010.

[5] C. Eckart and G. Young.
The approximation of one matrix by another of lower rank.
Psychometrika, 1(3):211–218, 1936.

[6] N. Gillis.
Introduction to nonnegative matrix factorization.
arXiv preprint arXiv:1703.00663, 2017.

References iii

[7] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua.
Neural collaborative filtering.
In *Proceedings of the 26th International Conference on World Wide Web*, pages 173–182. International World Wide Web Conferences Steering Committee, 2017.

[8] J. Lee, S. Kim, G. Lebanon, Y. Singer, and S. Bengio.
Llorma: local low-rank matrix approximation.
The Journal of Machine Learning Research, 17(1):442–465, 2016.

[9] A. Mnih and R. R. Salakhutdinov.
Probabilistic matrix factorization.
In *Advances in neural information processing systems*, pages 1257–1264, 2008.

References iv

[10] X. Ning and G. Karypis.
Slim: Sparse linear methods for top-n recommender systems.
In *2011 11th IEEE International Conference on Data Mining*, pages 497–506. IEEE, 2011.

[11] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. Lukose, M. Scholz, and Q. Yang.
One-class collaborative filtering.
In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*, pages 502–511. IEEE, 2008.

[12] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme.
Bpr: Bayesian personalized ranking from implicit feedback.
In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, pages 452–461. AUAI Press, 2009.

References v

[13] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl.
Item-based collaborative filtering recommendation algorithms.
In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001.

[14] S. Sedhain, H. H. Bui, J. Kawale, N. Vlassis, B. Kveton, A. K. Menon, T. Bui, and S. Sanner.
Practical linear models for large-scale one-class collaborative filtering.
In *IJCAI*, pages 3854–3860, 2016.

[15] S. Sedhain, A. K. Menon, S. Sanner, and L. Xie.
Autorec: Autoencoders meet collaborative filtering.
In *Proceedings of the 24th International Conference on World Wide Web*, pages 111–112. ACM, 2015.

[16] N. Srebro, J. Rennie, and T. S. Jaakkola.
Maximum-margin matrix factorization.
In *Advances in neural information processing systems*, pages 1329–1336, 2005.

[17] J. Teevan, S. T. Dumais, and E. Horvitz.
Potential for personalization.
ACM Transactions on Computer-Human Interaction (TOCHI), 17(1):4, 2010.

[18] G. Wu, M. Volkovs, C. L. Soon, S. Sanner, and H. Rai.
Noise contrastive estimation for scalable linear models for one-class collaborative filtering.
arXiv preprint arXiv:1811.00697, 2018.

[19] H. Yin, B. Cui, J. Li, J. Yao, and C. Chen.
Challenging the long tail recommendation.
Proceedings of the VLDB Endowment, 5(9):896–907, 2012.

References vii