Introduction to Recommender Systems

Dr. Xiangnan He

School of Data Science

School of Information Science and Technology

Slides are available: http://staff.ustc.edu.cn/~hexn/intro-recsys-Aug2021.pdf

Outline of Tutorial

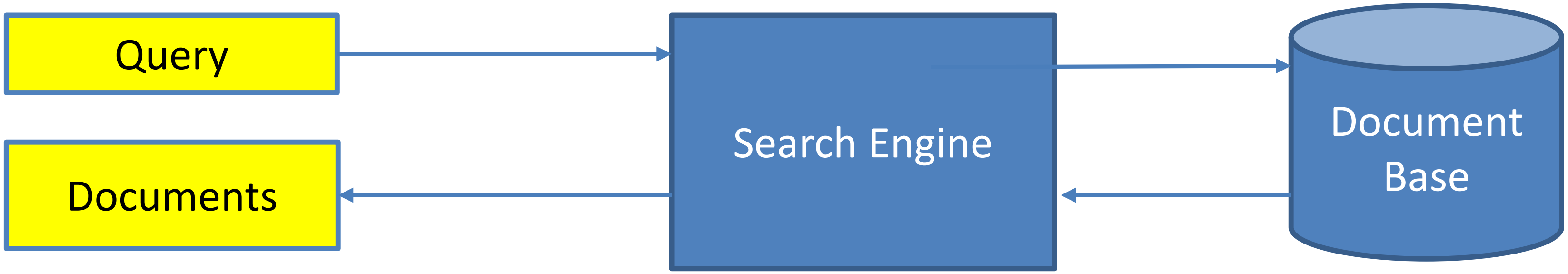
- Background & Basics (15 mins)
- Classical Two-stage Solutions for Recsys (60 mins)
- Rethinking Recsys Ecosystem (15 mins)

Slides are available: http://staff.ustc.edu.cn/~hexn/intro-recsys-Aug2021.pdf

Two Information Seeking Paradigms

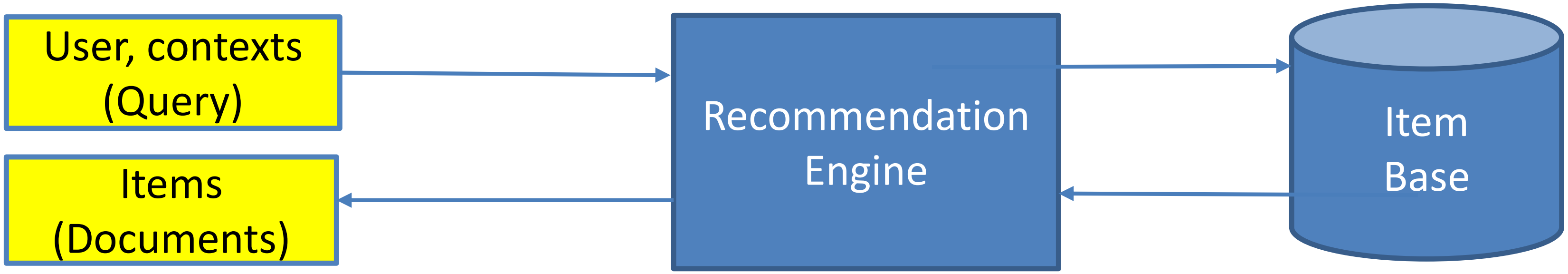
- Retrieval is information pull:
  - User pulls desired information by making a specific request

User intent is **explicitly** reflected in query



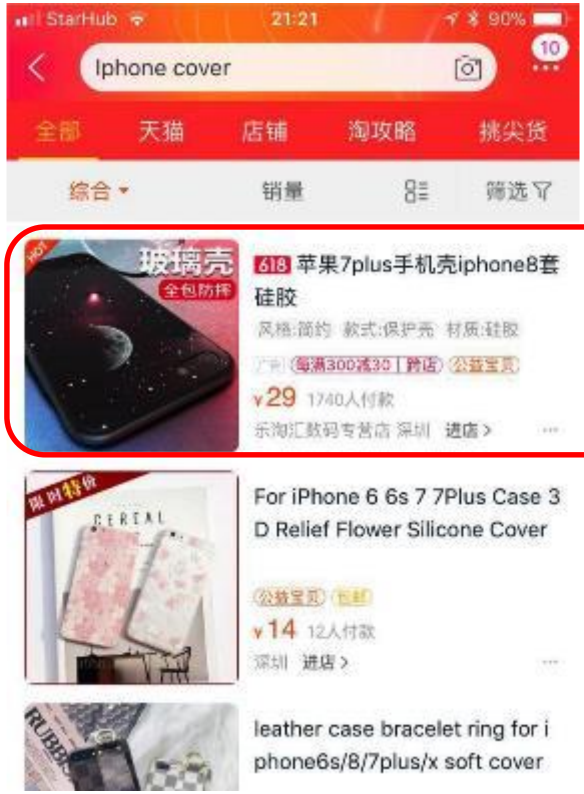
- Recommendation is information push:
  - System pushes desired information to a user by guessing her interest

User intent is **implicitly** reflected in interaction history, profile, contexts etc.



Importance of Recommendation

- Retrieval mostly exists in search engines
- But, recommendation exists everywhere…
  - When you search for a product <= Ad recommendation



618 苹果7plus手机壳iphone8套硅胶

风格:简约 款式:保护壳 材质:硅胶

广告 每满300减30 | 跨店 公益宝贝

¥29 1740人付款

乐淘汇数码专营店 深圳 进店 >

For iPhone 6 6s 7 7Plus Case 3D Relief Flower Silicone Cover

公益宝贝 包邮

¥14 12人付款

深圳 进店 >

leather case bracelet ring for iphone6s/8/7plus/x soft cover

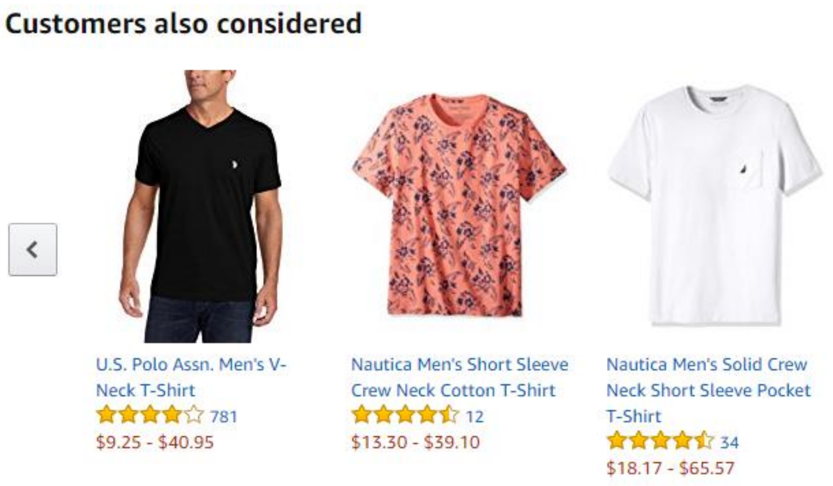
Top result is usually an ad

Search results of Taobao

Importance of Recommendation

- Retrieval mostly exists in search engines
- But, recommendation exists everywhere…
  - When you search for a product => Ad recommendation
  - When you open a product page => Product recommendation

Besides it:



Customers also considered

U.S. Polo Assn. Men's V-Neck T-Shirt
★★★★☆ 781
$9.25 - $40.95

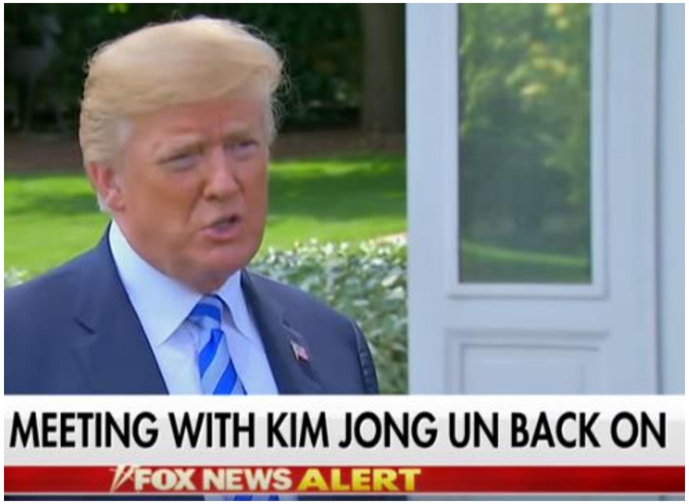
Nautica Men's Short Sleeve Crew Neck Cotton T-Shirt
★★★★½ 12
$13.30 - $39.10

Nautica Men's Solid Crew Neck Short Sleeve Pocket T-Shirt
★★★★½ 34
$18.17 - $65.57

Screenshot of Amazon

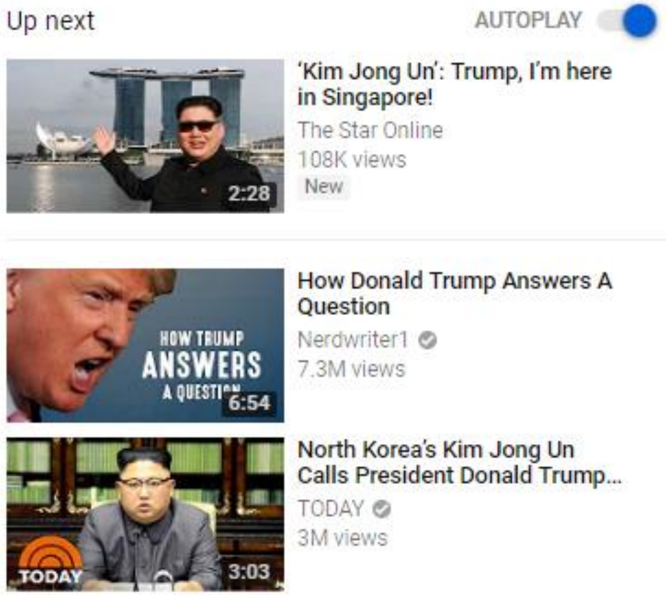
Importance of Recommendation

- Retrieval mostly exists in search engines
- But, recommendation exists everywhere…
  - When you search for a product => Ad recommendation
  - When you open a product page => Product recommendation
  - When you watch a video => Video Recommendation



President Trump reads letter from Kim Jong Un (YouTube)

Besides it:


Up next
AUTOPLAY

'Kim Jong Un': Trump, I'm here in Singapore!
The Star Online
108K views
New

How Donald Trump Answers A Question
Nerdwriter1
7.3M views

North Korea's Kim Jong Un Calls President Donald Trump...
TODAY
3M views

Importance of Recommendation

- Retrieval mostly exists in search engines
- But, recommendation exists everywhere…
  - When you search for a product => Ad recommendation
  - When you open a product page => Product recommendation
  - When you watch a video => Video Recommendation
  - When you read a news => News recommendation
  - When you book a flight => Hotel Recommendation
  - When you use social network => Friend Recommendation
  - When you are hungry => Restaurant Recommendation
  - When you open any webpage/app, there maybe a recommendation list.

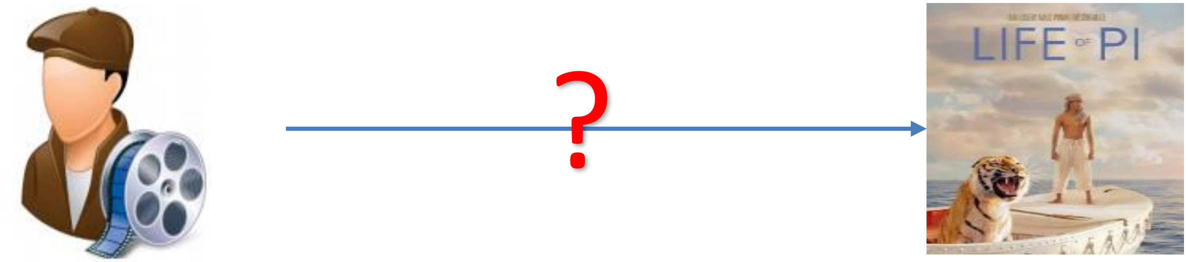
… …

Value of Recommender System (RecSys)

- RecSys has become a major monetization tool for customer-oriented online services
  - E.g., E-commerce, News Portal, Social Networks, etc.
- Ad systems are technically supported by recommendation solutions.
  - The key is Click-Through Rate (CTR) prediction
- Some statistics:
  - YouTube homepage: 60%+ clicks [Davidson et al. 2010]
  - Netflix: 80%+ movie watches, 1billion+ value/year [Gomze-Uribe et al 2016]
  - Amazon: 30%+ page views [Smith and Linden, 2017]

Problem Formulation

- Recommendation generally solves a matching problem.



User Profile (query):
- User ID
- Rating history
- Age, Gender
- Income level
- Time of the day

…….

Item Profile (document):
- Item ID
- Description
- Category
- Price
- Image

…….

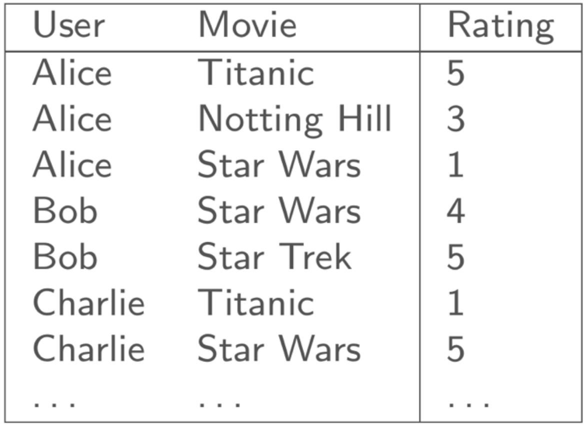
Challenge: no overlap between user features and item features
Matching can’t be done on the superficial feature level!

Collaborative Filtering

- Collaborative Filtering (CF): the most famous behavior-driven technique for recommendation.

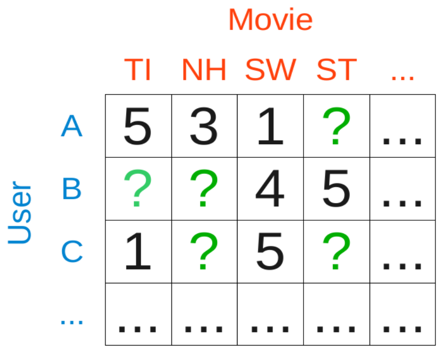
  *"CF makes predictions (**filtering**) about a user's interest by collecting preferences information from many users (**collaborating**)" ---Wikipedia*

- Math formulation: matrix completion problem



User	Movie	Rating
Alice	Titanic	5
Alice	Notting Hill	3
Alice	Star Wars	1
Bob	Star Wars	4
Bob	Star Trek	5
Charlie	Titanic	1
Charlie	Star Wars	5
. . .	. . .	. . .

Input Tabular data



Movie

User	TI	NH	SW	ST	...
A	5	3	1	?	...
B	?	?	4	5	...
C	1	?	5	?	...
...	...	...	...	...	...

Rating Matrix
(Interaction Matrix)

Solving Matrix Completion

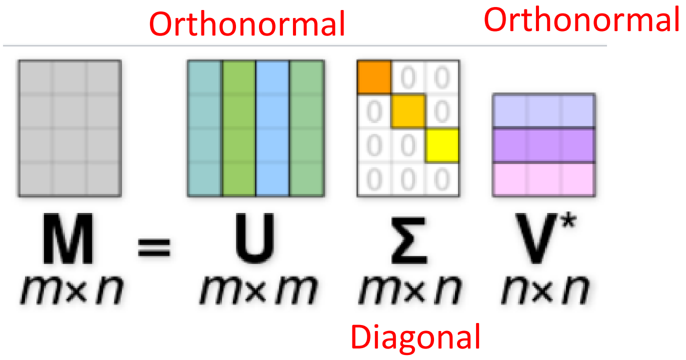
- Singular Value Decomposition (SVD) is the most well-known technique for matrix completion

Movie

User

	TI	NH	SW	ST	...
A	5	3	1	?	...
B	?	?	4	5	...
C	1	?	5	?	...
...	...	...	...	...	...

Rating Matrix

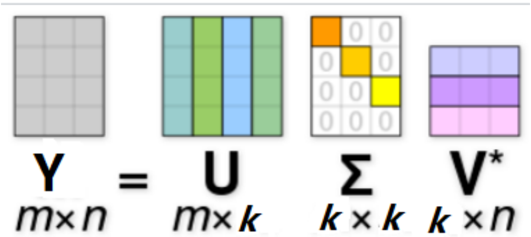


Steps to use SVD for CF:

1. Impute missing data to 0 in **Y**
2. Solving the SVD problem
3. Using only *K* dimensions in **U** and **V** to obtain a low rank model to estimate **Y**

Figure adopted from: https://en.wikipedia.org/wiki/Singular-value_decomposition

SVD is Suboptimal for CF



- In essence, SVD is solving the problem:

$$\arg\min_{\mathbf{U},\Sigma,\mathbf{V}} (\mathbf{Y} - \mathbf{U}\Sigma\mathbf{V}^T)^2$$

$$= \arg\min_{\mathbf{U},\Sigma,\mathbf{V}} \boxed{\sum_{i=1}^{m}\sum_{j=1}^{n}} \left(\overset{\text{Label}}{\boxed{y_{ij}}} - \underset{\text{Model Prediction}}{\boxed{(\mathbf{U}\Sigma\mathbf{V}^T)_{ij}}}\right)^2$$

Training instance

- Several Implications (weaknesses):
  1. Missing data has the same weight as observed data (>99% sparsity)
  2. No regularization is enforced – easy to overfit

Adjust SVD for CF

- The “SVD” model in the context of recommendation:

$$\hat{y}_{ui} = \underbrace{\mathbf{v}_u^T}_{\text{User latent vector}} \underbrace{\mathbf{v}_i}_{\text{Item latent vector}}$$

- Regularized Loss function:

$$L = \underbrace{\sum_u \sum_i w_{ui}(y_{ui} - \hat{y}_{ui})^2}_{\text{Prediction error}} + \lambda(\underbrace{\sum_u ||\mathbf{v}_u||^2 + \sum_i ||\mathbf{v}_i||^2)}_{\text{L2 regularizer}}$$

- This method is also called *Matrix Factorization* (MF) in RecSys:
  - It represents a user and an item as a latent vector (ID embedding).
  - The interaction between user and item is modelled using inner product

  (measure how much user latent “preferences” match with item “properties”

  - Besides L2 regularized loss, other loss can also be used, e.g., cross-entropy, margin-based pairwise loss, etc.

Why MF Can Capture CF

Item latent vector:

v_i^I

User latent vector:

v_u^U

Movie

User	TI	NH	SW	ST	...
A	5	3	1	?	...
B	?	?	4	5	...
C	1	?	5	?	...
...	...	...	...	...	...

Prediction score:

$$\hat{y}_{ui} = \mathbf{v}_u^T \mathbf{v}_i$$

Train the model on all observed interactions by sharing user embedding and item embedding

Latent Embedding space:

Similar users should have similar embeddings

Similar items should have similar embeddings

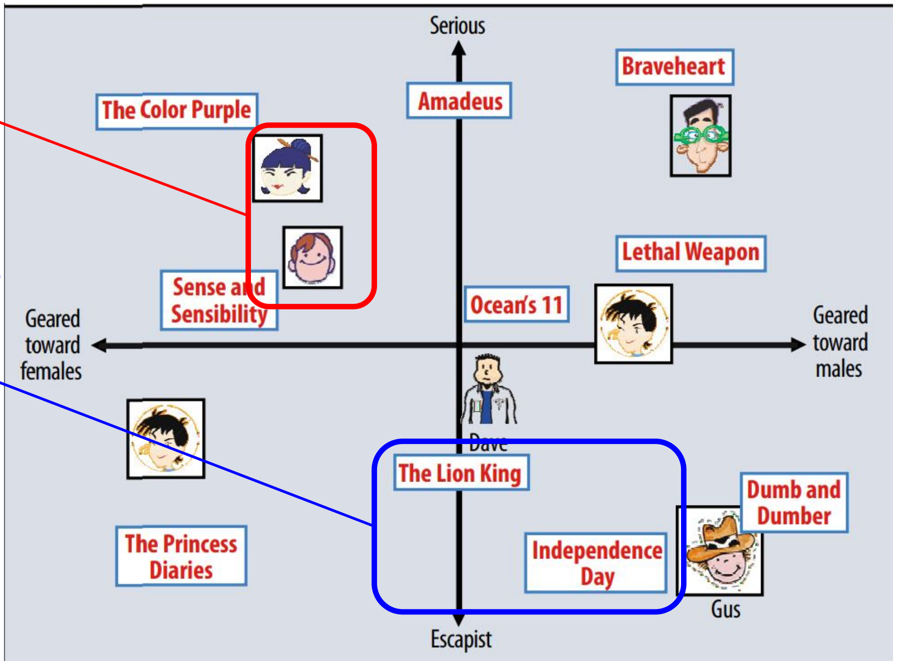


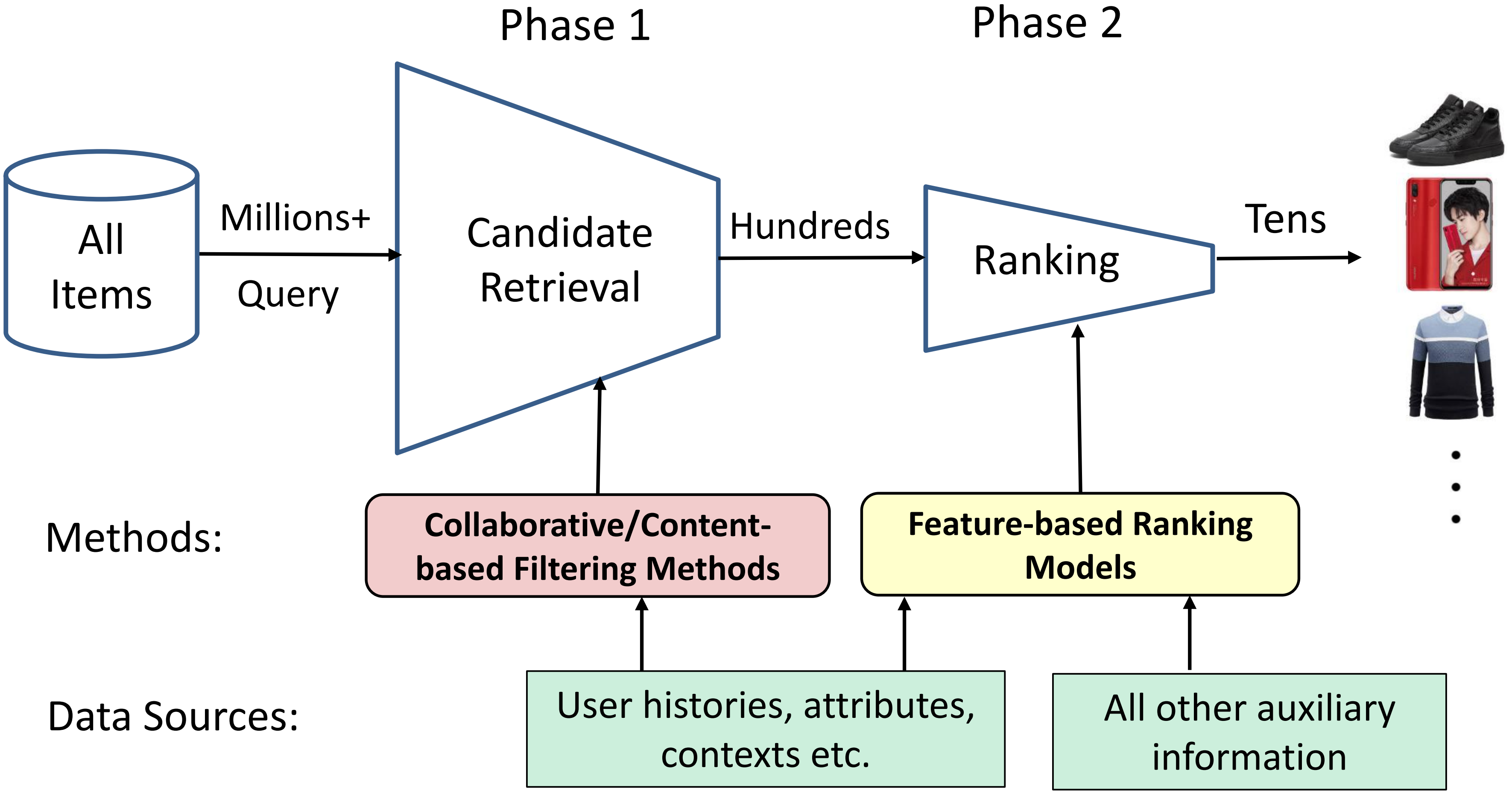
Figure adopted from: https://datajobs.com/data-science-repo/Recommender-Systems-%5BNetflix%5D.pdf

Outline of Tutorial

- Background & Basics (15 mins)
- Classical Two-stage Solutions for Recsys (60 mins)
- Rethinking Recsys Ecosystem (15 mins)

Slides are available: http://staff.ustc.edu.cn/~hexn/intro-recsys-Aug2021.pdf

Recommender System Overview



Phase 1

Phase 2

All Items

Millions+

Query

Candidate Retrieval

Hundreds

Ranking

Tens

Methods:

Collaborative/Content-based Filtering Methods

Feature-based Ranking Models

Data Sources:

User histories, attributes, contexts etc.

All other auxiliary information

(Covington et al, Recsys'16)

Need for Candidate Selection?

- Ranking is an expensive operation
  - ✓ Candidate selection is a first-pass filter to reduce input space to final ranking function

- Recommendations are highly personalized
  - ✓ Users expect a fast, updated, and contextually aware system
  - ✓ Online learning on selected candidate only

- Recall is more important than precision!
  - ✓ Generate a diverse set of items relevant to the *query* (e.g., user profile, target item, contexts)

(Arya et al, SIGIR'17)

Candidate Generation Overview

- Efficiency is the key challenge:
  - ➢ Need to select candidates in milliseconds
  - ➢ Even scanning all items is infeasible -- O(N) cost
  - ➢ The complexity needs to be sublinear in # of items.

- Two types of methods:
  1. Heuristic-based methods

     Define heuristics, e.g., co-occurrence, random walk etc.
  2. Embedding-based methods

     Learn embedding for user and item, and perform kNN search in the embedding space.

Heuristic-based Methods

- Heuristic-based methods are usually simple and easy to implement
  - No objective function is optimized

- Based on item properties, e.g.:
  - Hot sale items
  - Promotion items
- Based on item relations, e.g.:
  - Similar items (metadata, co-purchase, visual, etc)
  - Random walk on item graph
  - Complementary and substitutable items

Find Similar Items (item co-occurrence)

- Item co-occurrence statistics can be evaluated at different levels:
  - A user’s whole history (long-term)
  - A session (short-term)
- Relatedness score between two items:

Co-occurrence count

$$r(v_i, v_j) = \frac{\boxed{c_{ij}}}{\boxed{f(v_i, v_j)}}$$

Normalization function, e.g.,

$$f(v_i, v_j) = c_i \cdot c_j$$

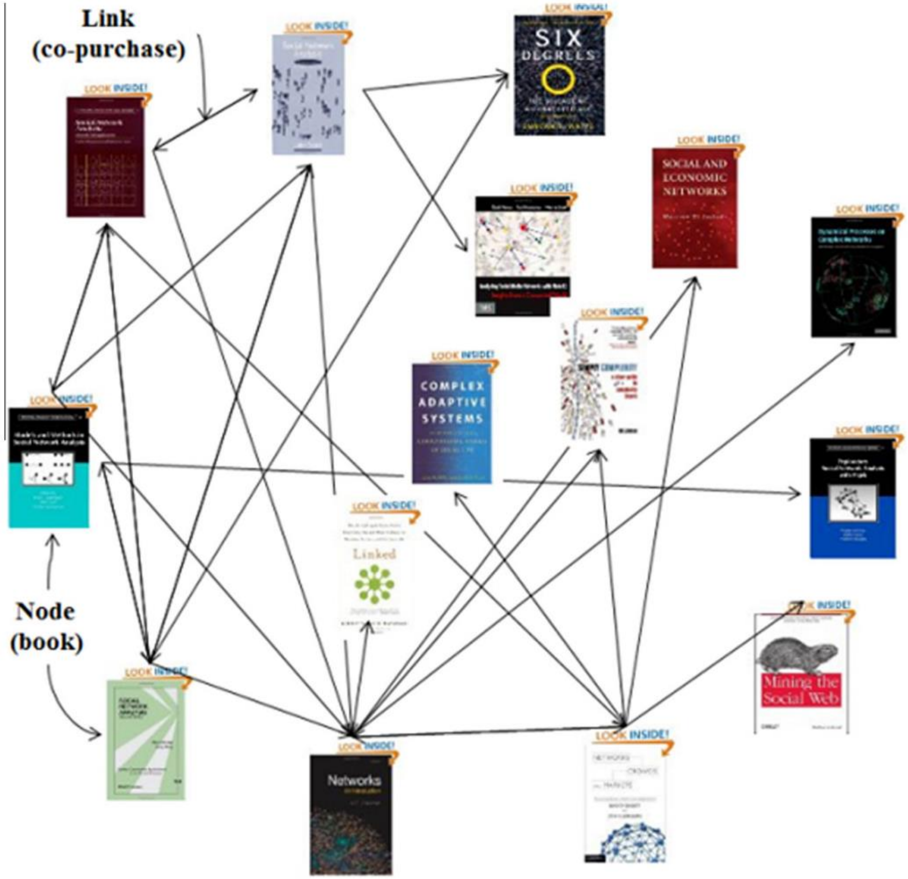
Product of items’ global popularity

- Other normalization functions can be used, e.g., to define *transition probability*.

(Davidson et al, Recsys’10)

Find Similar Items (item graph)

- Build an item graph based on co-occurrence:
  - ➢ Select a threshold to control graph density.



A typical workflow:

Step 1: Build item graph

Step 2: Define the target user's activity (e.g., purchased/clicked products) as seeds

Step 3: Propagating from the seeds on the graph to find more relevant items.

(Leem et al, ESA'14)

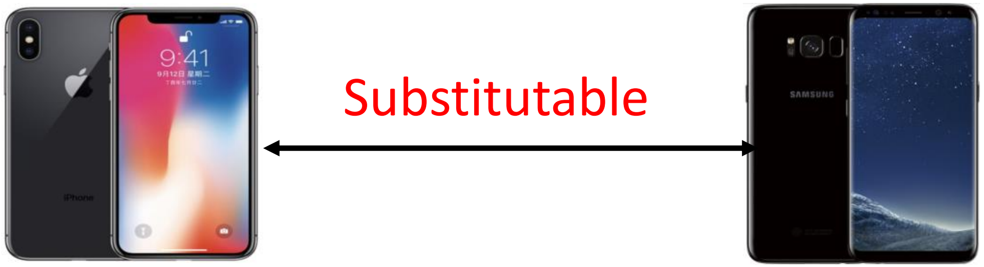
Find Similar Items (graph random walk)

- Global propagation on graph is too costly:
  - ➢ Need to consider the whole graph structure and wait for convergence (consider PageRank)
- More efficient way is to do local random walk!
- Pinterest’s Pixie system:
  1. Starting from each seed
  2. Performing many random walks (parallelized)
  3. Aggregate visit counts of covered items
- Need many tricks to ensure walking depth and personalization quality.
  - ➢ E.g., graph pruning and biased sampling

(Jure et al, WWW’18)

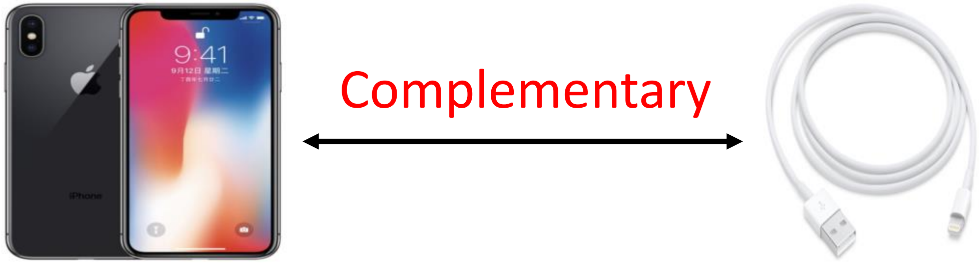
Beyond Similar Items – Compl. & Subst.

- Complementary and substitutable items:
  - ➢ Substitutes: items that are interchangeable (Co-view)



Substitutable

  - ➢ Complements: items that might be purchased together (co-purchase)



Complementary

(Wang et al, WSDM’18)

Benefits of Subst. and Compl.

- Substitutes and Complements are unique source to support specific scenarios:
  - Substitutes: “also viewed”, “buy after viewing”
  - Complements: “also bought”, “frequently buy together”

- Using candidates of substitutes and complements is beneficial to
  - Higher click-through rate
  - Higher conversation rate
  - Increase user stickiness

(Wang et al, WSDM’18)

Learning Substitutes and Complements

- Co-view and Co-purchase statistics provide weak labels for substitutes and complements.
  - Large statistics are more trustable

- How to learn from statistics and generalize to more products?
  - I.e., finding more item pairs having a relation.

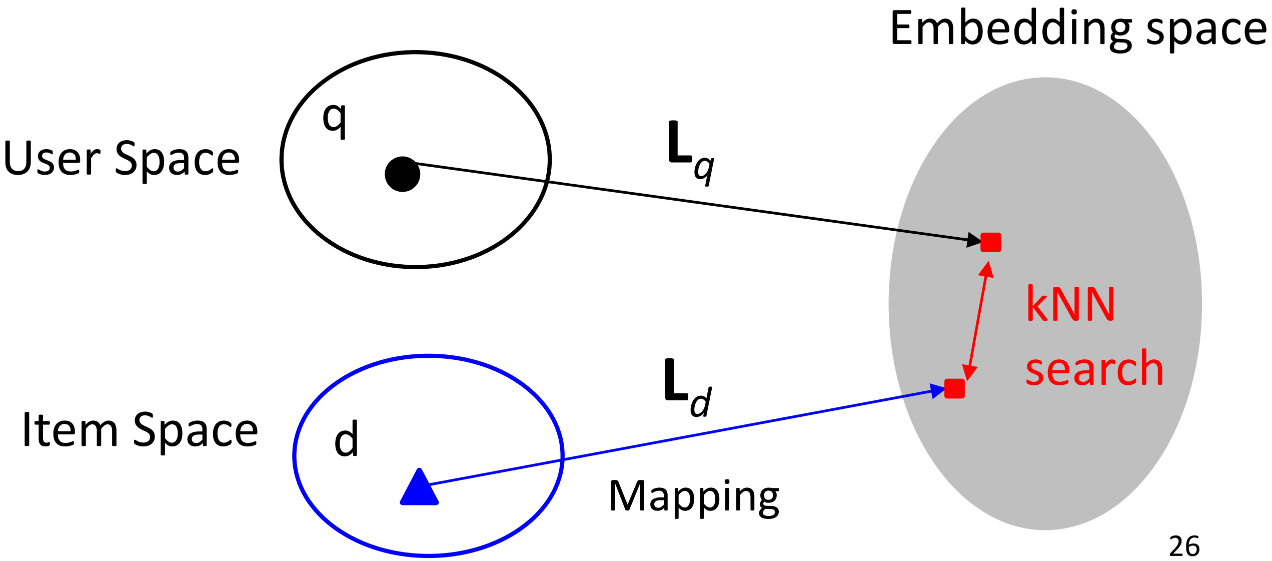
- Can be formulated as a Link Prediction task with two types of relations.
  - Next: Wang et al. WSDM 2018. *A Path-constrained Framework for Discriminating Substitutable and Complementary Products in E-commerce*

Short Summary on Candidate Generation

- ✓ Heuristic methods:
  - ✓ Find similar items
  - ✓ Find complementary and substitutable items

In the next…

- Embedding-based methods
  - ➢ The key is to learn user embedding and item embedding.



Recap: Matrix Factorization

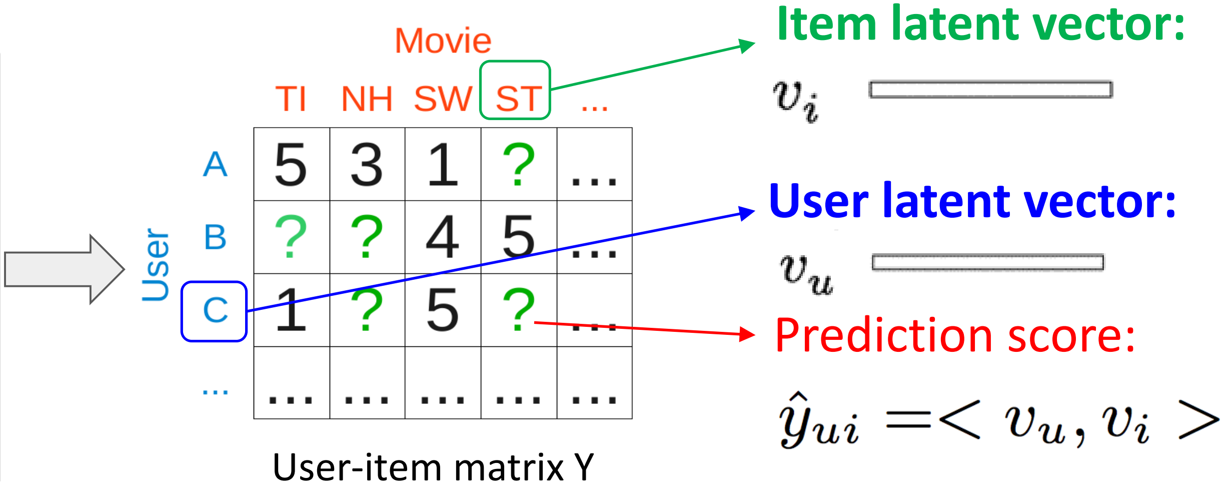
- Matrix Factorization (MF) assumes the user-item interaction matrix has a low-rank structure:

$$\mathbf{Y} \approx \boxed{\mathbf{U}}\boxed{\mathbf{V}}^{T}$$

Item latent matrix

User latent matrix

User	Movie	Rating
Alice	Titanic	5
Alice	Notting Hill	3
Alice	Star Wars	1
Bob	Star Wars	4
Bob	Star Trek	5
Charlie	Titanic	1
Charlie	Star Wars	5
...	...	...



$$\begin{array}{c|ccccc} & \text{TI} & \text{NH} & \text{SW} & \boxed{\text{ST}} & \ldots \\ \hline \text{A} & 5 & 3 & 1 & ? & \ldots \\ \text{B} & ? & ? & 4 & 5 & \ldots \\ \boxed{\text{C}} & 1 & ? & 5 & ? & \ldots \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \end{array}$$

User-item matrix Y

Item latent vector:

v_i

User latent vector:

v_u

Prediction score:

$$\hat{y}_{ui} =< v_u, v_i >$$

- Each user and item is described as an embedding vector
- The score is estimated as the inner product of user embedding and item embedding

(He et al, SIGIR'16)

User Embedding + Rating History

- MF profiles a user with an ID, directly projecting ID to embedding space.
  - ➢ Taking out the *u*-th row in user embedding matrix **U**
- Another more information-rich way for user profile is to use the rating history:

$$\hat{y}_{ui} = (\overbrace{\frac{1}{|\mathcal{R}_u|^{\alpha}}\sum_{\underset{\text{Items rated by } u}{\boxed{j\in\mathcal{R}_u}}}}^{\Leftrightarrow \text{ user embedding}} \boxed{\mathbf{q}_j)^T\mathbf{v}_i}$$

Can be interpreted as the **similarity** between item *i* and *j*

- ➢ Known as *factored item similarity model* (FISM, Kabbur et al, KDD'14), since it factorizes item similarity matrix into two low-rank matrices.

User Embedding + Rating History

- MF profiles a user with her ID
  - ➢ ID embedding encodes user general interest
- FISM profiles a user with her interacted items
  - ➢ Recommend items that are similar to historical items.
- We can profile a user with both her ID and rating history:

$$\hat{y}_{ui} = \overbrace{(\underbrace{\mathbf{v}_u}_{\text{User ID Embedding}} + \underbrace{\frac{1}{|\mathcal{R}_u|^{\alpha}} \sum_{j \in \mathcal{R}_u} \mathbf{q}_j}_{\text{Rating History Embedding}})}^{\text{User Embedding}}{}^{T} \mathbf{v}_i$$

  - ➢ Known as SVD++ model (Koren, KDD'08), the best single model for rating prediction in Netflix challenge (3 years, 1 million price)

User Embedding + Contexts

- User decisions are context-aware:
  - ➢ Contexts: time, location, query, weather etc.

- Context-aware MF model:

$$\hat{y}_{ui} = (\overbrace{\mathbf{v}_u + \sum_{\underbrace{\boxed{c \in C(u)}}_{\text{Current contexts}}} \underbrace{\mathbf{v}_{u,c}}_{\text{Context-aware User Embedding}}}^{\text{User Embedding}})^T \mathbf{v}_i$$

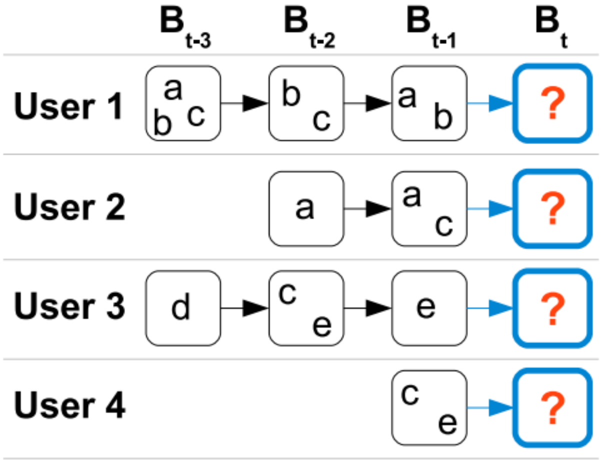
  - ➢ When a context is dense, we can directly learn **$\mathbf{v}_{u,c}$** from data, e.g., timeSVD model (Koren et al, KDD'09)
  - ➢ When the context is sparse, we can decompose **$\mathbf{v}_{u,c}$** to reduce model parameters to avoid overfitting, e.g.:

$$\mathbf{v}_{u,c} = \mathbf{v}_c$$

$$\mathbf{v}_{u,c} = \mathbf{v}_u \odot \mathbf{v}_c$$

User Embedding + Recent Purchases

- Next-basket recommendation in E-commerce:



Two properties:

1. User purchases a basket of products at a time
2. User behaviors are sequential: which products will be purchased in next basket?

➢ Factorizing Personalized Markov Chain (FPMC, Rendle et al, WWW’10):

$$\hat{y}_{uit} = \overbrace{(\mathbf{v}_u + \sum_{\boxed{l\in B_{t-1}}}}^{\text{User Embedding}} \boxed{\mathbf{v}_l)^T \mathbf{v}_i}$$

Estimate transition probability from last item *l* to next item *j*

Items purchased in last basket

Item Embedding + Attributes

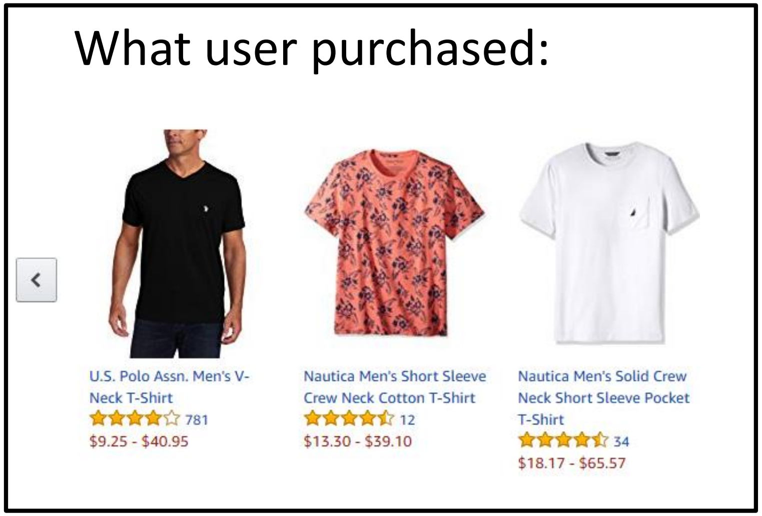
- Attributes of item side can be integrated by using similar way of embedding addition:
  - E.g., product tags, category, price

$$\hat{y}_{ui} = \mathrm{v}_u^T \left(\overbrace{\mathbf{v}_i + \sum_{a \in A(i)} \underbrace{\mathbf{v}_a}_{\text{Attribute Embedding}}}^{\text{Item Embedding}} \right)$$

- In the next, we consider two special "attributes" of E-commerce products:
  - Taxonomy (i.e., category tree)
  - Product Image

Item Embedding + Image

Product images are particularly useful for some categories, such as fashion products.



What user purchased:

U.S. Polo Assn. Men's V-Neck T-Shirt
★★★★☆ 781
$9.25 - $40.95

Nautica Men's Short Sleeve Crew Neck Cotton T-Shirt
★★★★☆ 12
$13.30 - $39.10

Nautica Men's Solid Crew Neck Short Sleeve Pocket T-Shirt
★★★★☆ 34
$18.17 - $65.57

Will user like this one?

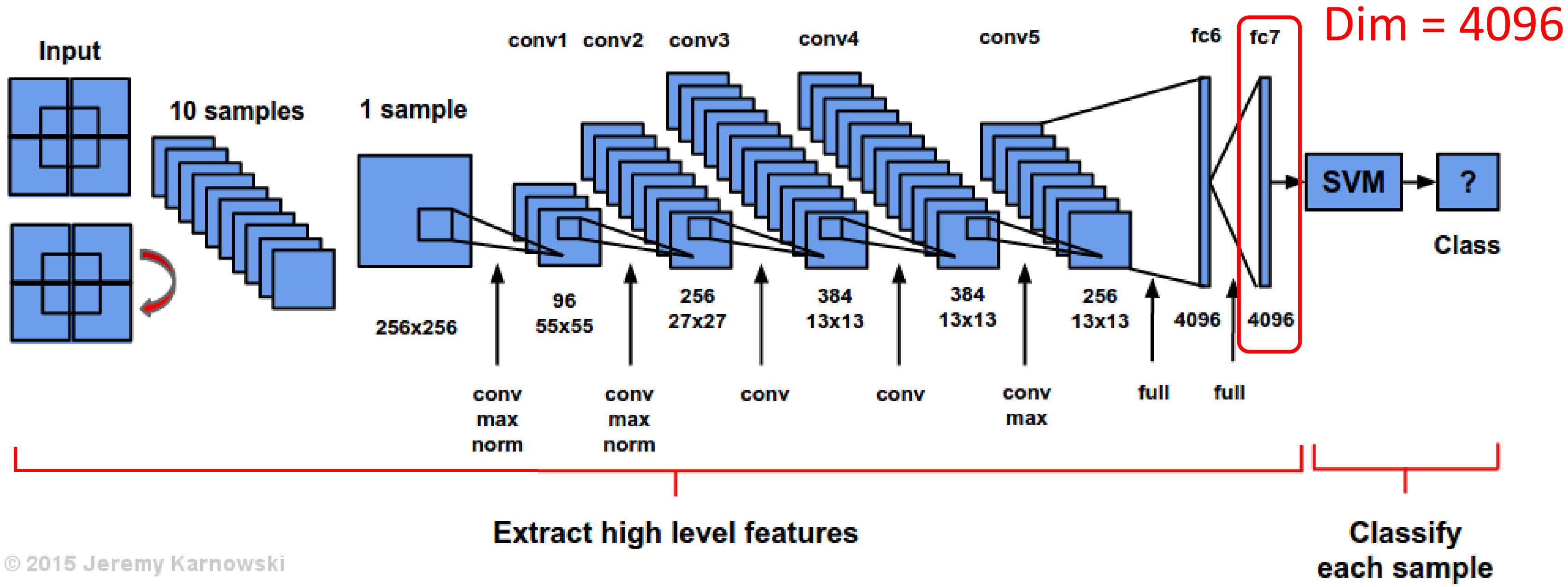
Each product = ID + image

Two key questions:

1. How to understand image?
2. How to integrate image feature into CF model?

Image Understanding

- Traditional (low-level) image features:
  - Pixels, Color histograms
  - SIFT descriptors
- Gap between low-level features and real semantics.
- Recent work uses deep CNN as feature extractor.



Item Embedding + Image

- Let $\mathbf{f}_i$ be CNN features for image *i*:
  - Usually of thousands dimension. E.g., AlexNet: 4096, ResNet: 2048
- MF predicts user rating on image *i*:

$$\hat{y}_{ui} =< \boxed{\mathbf{p}_u}, \mathbf{f}_i >= \mathbf{p}_u^T \mathbf{f}_i$$

User preference on image CNN features

- Problem:
  - $\mathbf{p}_u$ has to be of the same dimension as $\mathbf{f}_i$
  - Too big latent space: too many parameters => overfitting

    E.g., 100 million users * 4096 * 8 B = 3.28 TB
  - Typically, the dimension of CF latent space is hundreds (128, 256) at most.

Item Embedding + Image

- An intuitive solution is to do dimension reduction on CNN features, e.g., PCA
  - However, it will lose signal in CNN features.
  - The objective of dimension reduction is not recommendation.

- Solution: learning a transformation matrix to do the projection based on user-item interactions:

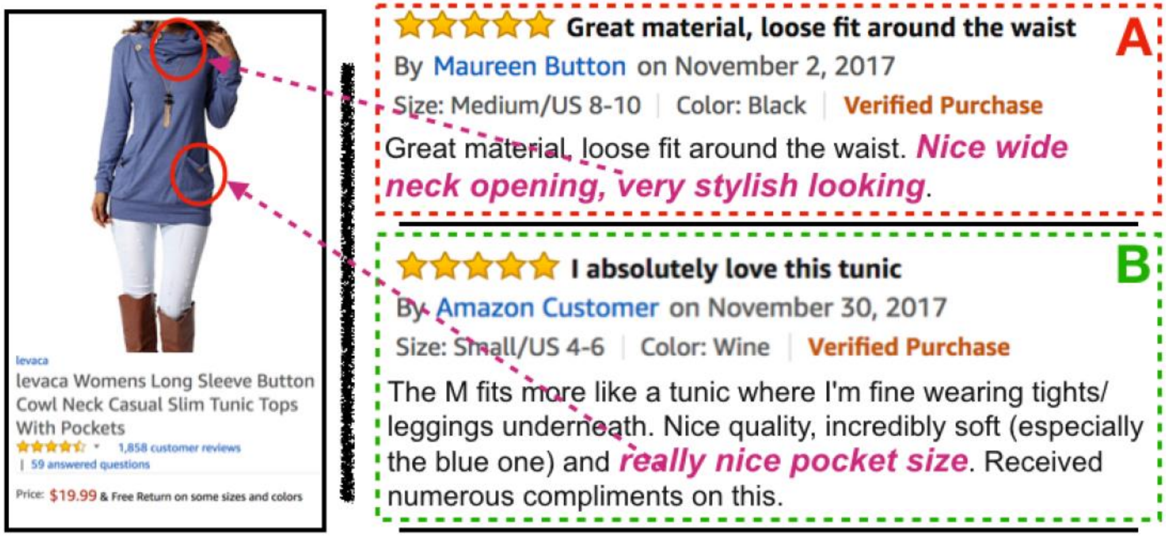
$$\hat{y}_{ui} = \mathbf{p}_u^T(\boxed{\mathbf{E}}\mathbf{f}_i)$$

Transformation matrix that projects CNN features to latent space

  - **E** is optimized for recommendation task.
  - This is the model widely used deep feature-based recsys (Geng et al, ICCV’15, He et al, AAAI’16).

Item Embedding + Image

- User may care about different parts on a product:



levaca

levaca Womens Long Sleeve Button Cowl Neck Casual Slim Tunic Tops With Pockets

1,858 customer reviews

| 59 answered questions

Price: $19.99 & Free Return on some sizes and colors

Great material, loose fit around the waist

By Maureen Button on November 2, 2017

Size: Medium/US 8-10 | Color: Black | Verified Purchase

Great material, loose fit around the waist. ***Nice wide neck opening, very stylish looking***.

I absolutely love this tunic

By Amazon Customer on November 30, 2017

Size: Small/US 4-6 | Color: Wine | Verified Purchase

The M fits more like a tunic where I'm fine wearing tights/ leggings underneath. Nice quality, incredibly soft (especially the blue one) and ***really nice pocket size***. Received numerous compliments on this.

(Chen et al, 2018)

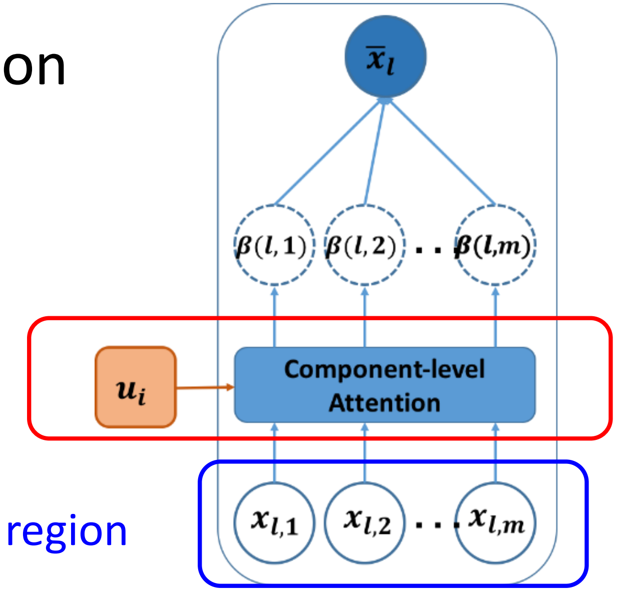
- User-sensitive Image Representation

(Chen et al, SIGIR’17)

Attention Net determines region’s weight:

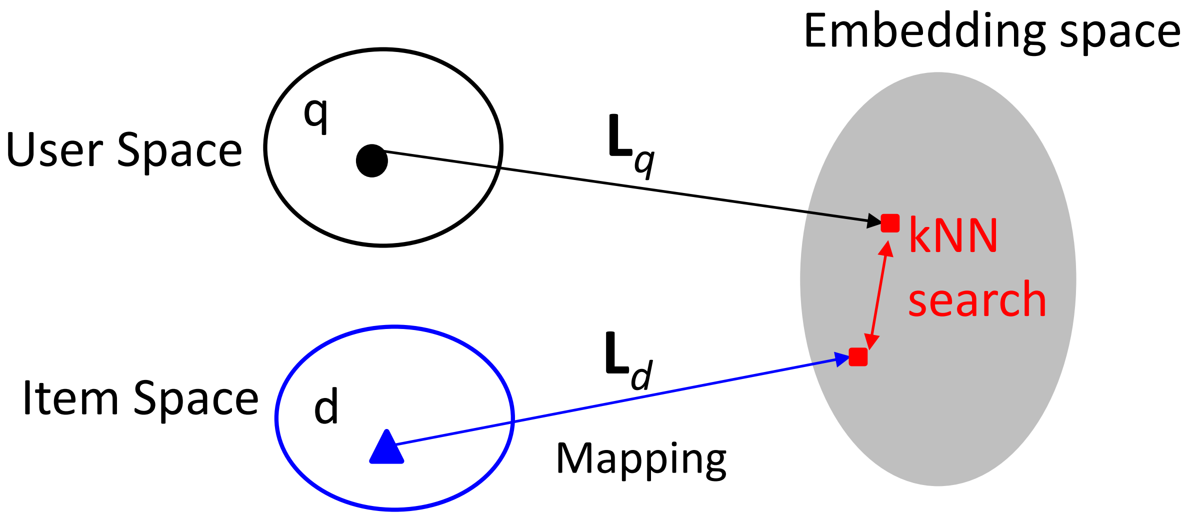
- Input: user embedding and region feature

$$b(i,l,m) = \mathbf{w}_2^T \phi(\mathbf{W}_{2u}\mathbf{u}_i + \mathbf{W}_{2x}\mathbf{x}_{lm} + \mathbf{b}_2) + \mathbf{c}_2$$



Feature of each region

Short Summary



- We have covered the model design for user embedding and item embedding.
  - ➢ User Embedding + Rating History / Contexts / Recent Purchases
  - ➢ Item Embedding + Attributes / Taxonomy / Image
- It is natural to combine all above info to build a unified embedding-based predictive model.
- Next: how to learn embeddings?
  - ➢ I.e., the optimization process

User Feedback Data

Explicit Feedback conveys user preference explicitly

- E.g., user ratings
- Usually real-values
- Higher score => positive signal
- Lower scores => negative signal

Movie

User

	TI	NH	SW	ST	...
A	5	3	1	?	...
B	?	?	4	5	...
C	1	?	5	?	...
...	...	...	...	...	...

Implicit Feedback conveys user preference implicitly:

- E.g., clicks, purchases
- Usually binary 0/1
- Observed data => positive signal
- Unobserved => negative signal

Movie

User

	TI	NH	SW	ST	...
A	1	1	1	?	...
B	?	?	1	1	...
C	1	?	1	?	...
...	...	...	...	...	...

In E-Commerce, most feedback data are implicit feedback, much more than explicit ratings!

- Cheaper and easier to collect (e.g., server logs)

Rating Prediction is Suboptimal

- Old-style work on recommendation optimize L2 loss on observed user-item interactions:

$$L = \sum_{\boxed{(u,i)\in\mathcal{R}}} w_{ui}(\hat{y}_{ui} - \hat{y}_{ui})^2 + \lambda\Big(\sum_{u} ||\mathbf{v}_u||^2 + \sum_{i} ||\mathbf{v}_i||^2\Big)$$

Observed interactions

- ➢ But many empirical evidence show that:

A lower error rate does not lead to a good ranking performance…

- ➢ Possible Reasons:

1) Discrepancy between error measure (e.g., RMSE) and ranking measure.
2) Survival bias – users tend to consume items they like and ignore items they dislike

=> important to account for **missing data**!

Towards Top-K Recommendation

- Recommendation is a personalized ranking task by nature, rather than rating prediction (regression).
  - ➢ Evaluated by Precision/Reall/AUC etc, rather than RMSE!
- Optimizing the relative ranking of a user on two items are more advantageous:
  - ➢ Higher rating > Lower rating (explicit feedback)
  - ➢ Observed interaction > Unobserved interaction (implicit feedback)

sigmoid Positive prediction Negative prediction

$$L_{BPR} = \arg\max_{\Theta} \sum_{\boxed{(u,i,j)\in\mathcal{R}_B}} \ln\sigma\left(\boxed{\hat{y}_{ui}} - \boxed{\hat{y}_{uj}}\right) - \lambda||\Theta||^2$$

Pairwise training examples: *u* prefers *i* over *j*

- ➢ Known as the Bayesian Personalized Ranking objective (BPR, Rendle et al, UAI’09)

Training Procedure

$$L_{BPR} = \arg\max_{\Theta} \sum_{\boxed{(u,i,j)\in\mathcal{R}_B}} \ln \sigma(\hat{y}_{ui} - \hat{y}_{uj}) + \lambda||\Theta||^2$$

Pairwise examples: *u* prefers *i* over *j*

For a positive instance (*u, i*), e.g., a purchase, all non-purchased items of *u* can be used as negative instances.

$$\mathcal{D} := \{(u,i,j) | \underset{\text{Items purchased by } u}{\underline{i \in \mathcal{Y}_u^+}} \wedge \underset{\text{Items not purchased by } u}{\underline{j \notin \mathcal{Y}_u^+}}\}$$

Using SGD (stochastic gradient descent) for optimization:

- Step 1: Sample a positive instance (*u, i*)
- Step 2: Sample a negative instance (*u, j*) to pair with (*u, i*)
- Step 3: Update parameters w.r.t. this stochastic instance

Which negative items to sample?

Static Negative Sampling

- Sampling from a static distribution:
  - Uniform distribution: all items are equally likely to be disliked by user (vanilla BPR sampler)
  - Popularity-aware distribution: popular items are more likely to be disliked by user (He et al, SIGIR'16)

- Note that the objective of BPR learning is to increase the margin $\hat{y}_{ui} - \hat{y}_{uj}$ as much as possible.
  - Problem: If the current model already scores (*u, i*) much higher than (*u, j*), sampling (*u, j*) as negative has fewer gain to model update.
  - In other words, the gradients of BPR objective w.r.t. (*u, i, j*) are close to 0.

Dynamic Negative Sampling

- Basic idea: sampling hard negatives leads to more gain to the current model.
  - The negative sampling distribution dynamically changes with model updates.
- Difficulty: evaluating model prediction on all items is too time-consuming!
- An approximate algorithm to DNS:
  - Step 1: Randomly sample *K* negatives (e.g., *K=20*)
  - Step 2: Use current model to score the *K* negatives
  - Step 3: Pick the negative with the highest score as the “true negative” for model update
- DNS always leads to faster convergence and better performance.

(Zhang et al, SIGIR’13)

Other Optimization Alternatives

- Besides optimizing BPR objective, there are other optimization choices, e.g.:
  - ➢ Margin-based pairwise loss:

$$Loss = \sum_{(u,i,j)\in\mathcal{D}} \max(0, \boxed{\Delta} + \hat{y}_{uj} - \hat{y}_{ui})$$

Expected minimum margin between positive prediction and negative prediction

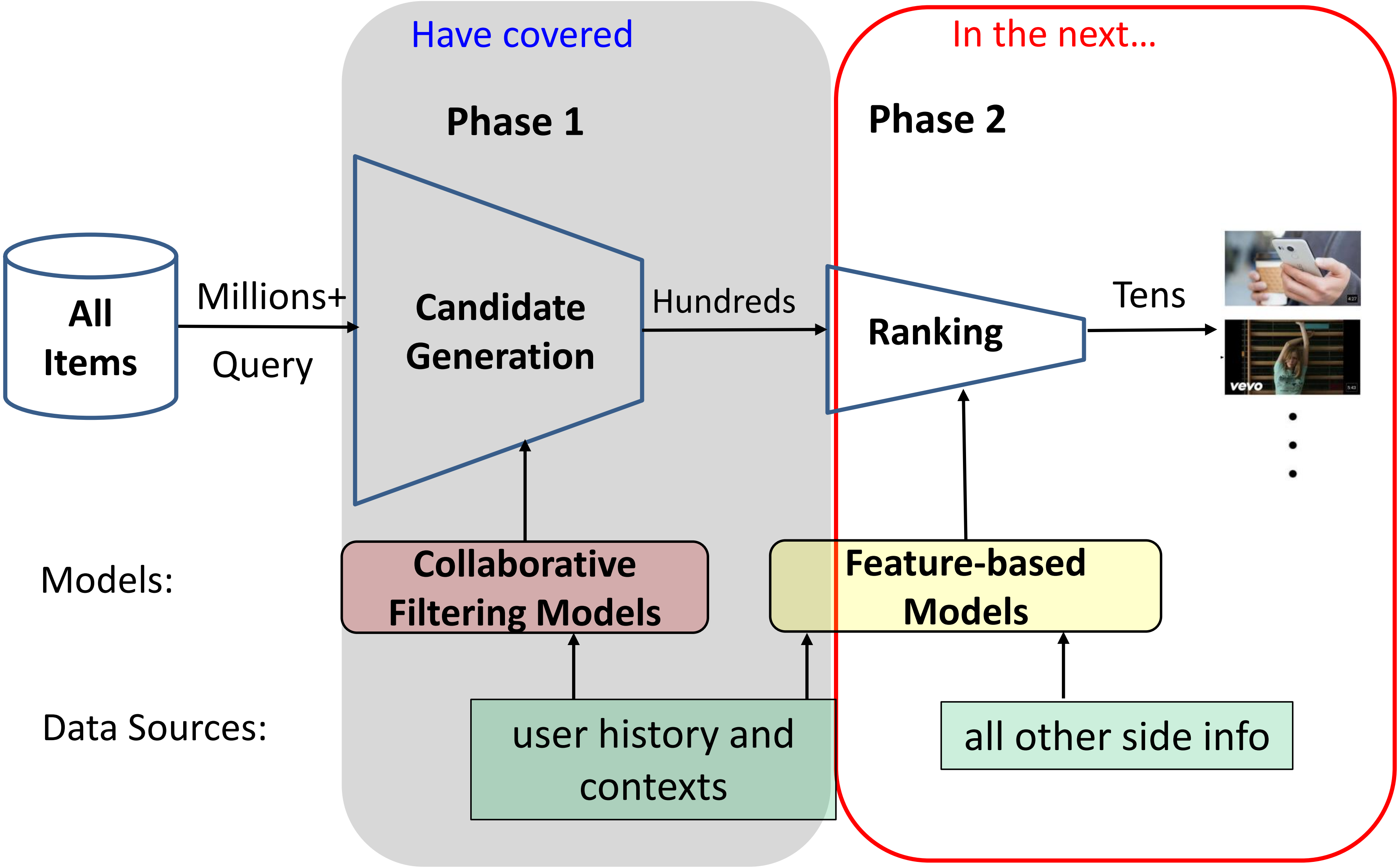
- More commonly used in Knowledge Graph Completion
- Recently used in Recsys (Ying et al, KDD'18)

  - ➢ Point-wise classification loss:

$$L = \sum_{u} [\underbrace{\sum_{i\in\mathcal{R}_u} \log\sigma(\hat{y}_{ui})}_{\text{Observed interactions (positive)}} + w_0 \underbrace{\sum_{j\in\mathcal{R}_u^-} \log(1-\sigma(\hat{y}_{uj}))}_{\text{Unobserved interactions (negative)}}]$$

- Treat recommendation as a classification task.
- Also commonly used in Recsys (He et al, WWW'17)

Recommender System Overview



Have covered

In the next…

Phase 1

Phase 2

All Items

Millions+

Query

Candidate Generation

Hundreds

Ranking

Tens

Models:

Collaborative Filtering Models

Feature-based Models

Data Sources:

user history and contexts

all other side info

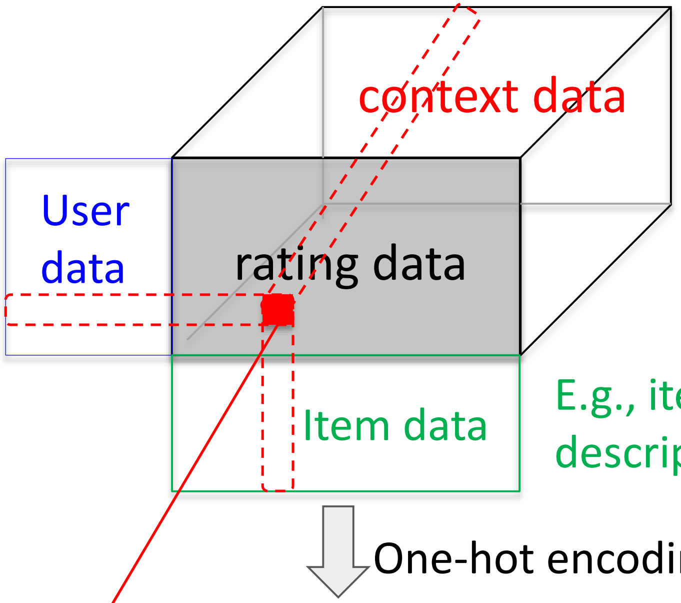
(Covington et al, Recsys’16)

Requirements for Ranking

- Fine-tuning the results to present to end user
  - ➢ Ensemble different candidate sources
- Only a few hundred items are being scored
  - ➢ Low requirement on model efficiency
  - ➢ More complicated models can be used
  - ➢ More features can be used, e.g., user profiles, contexts, combinatorial features …
- Powered by supervised learning methods. Optimization objective can be tailored for different scenarios, such as optimizing for:
  - ➢ Click-through rate (CTR)
  - ➢ Purchase
  - ➢ Impression

  ……

Input to Feature-based Models



One-hot encoding

E.g., recent history, location, time, weather, mood …

E.g., user gender, age, occupation personality …

E.g., item category, description, image

Input Features:

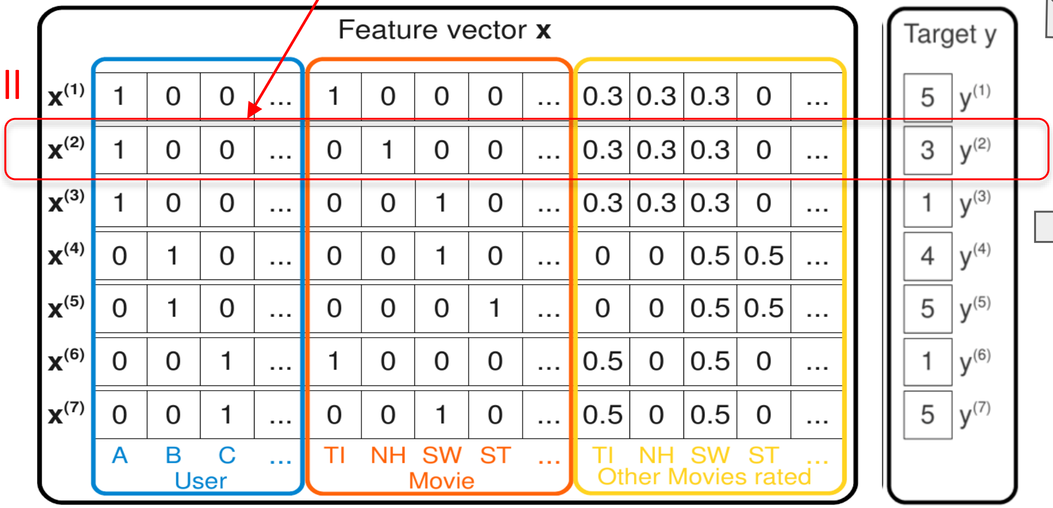
1. Categorical features:

   user/item ID, bag-of-words, cross features…

2. Numerical features:

   textual/visual embeddings, converted features (e.g. TFIDF, GBDT)…

Each row encodes all info for a rating



Feature vector $\mathbf{x}$

	User				Movie					Other Movies rated					Target y	
	A	B	C	...	TI	NH	SW	ST	...	TI	NH	SW	ST	...		
$\mathbf{x}^{(1)}$	1	0	0	...	1	0	0	0	...	0.3	0.3	0.3	0	...	5	$y^{(1)}$
$\mathbf{x}^{(2)}$	1	0	0	...	0	1	0	0	...	0.3	0.3	0.3	0	...	3	$y^{(2)}$
$\mathbf{x}^{(3)}$	1	0	0	...	0	0	1	0	...	0.3	0.3	0.3	0	...	1	$y^{(3)}$
$\mathbf{x}^{(4)}$	0	1	0	...	0	0	1	0	...	0	0	0.5	0.5	...	4	$y^{(4)}$
$\mathbf{x}^{(5)}$	0	1	0	...	0	0	0	1	...	0	0	0.5	0.5	...	5	$y^{(5)}$
$\mathbf{x}^{(6)}$	0	0	1	...	1	0	0	0	...	0.5	0	0.5	0	...	1	$y^{(6)}$
$\mathbf{x}^{(7)}$	0	0	1	...	0	0	1	0	...	0.5	0	0.5	0	...	5	$y^{(7)}$

Predictive Model

(Rendle, ICDM’10)

Key to Feature-based Models

- Feature vector is high-dimensional but sparse
  - Consider the CF case: feature vector = user ID + item ID
  - Need to discover prediction patterns in nonzero features
- The interactions between features are important

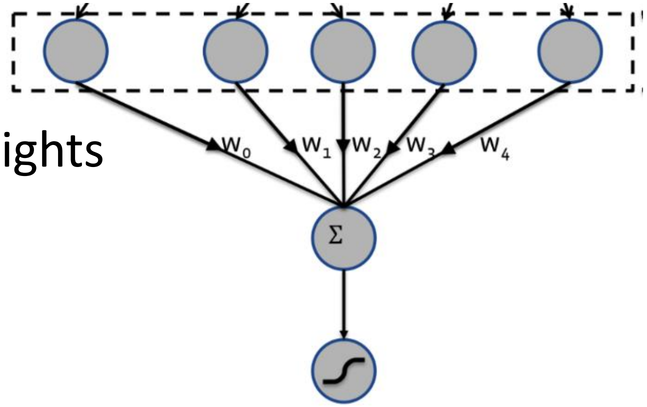
  E.g., users like to use food delivery apps at meal-time

  => Order-2 interactions between app category and time

  E.g., male teenagers like shooting games

  => Order-3 interactions between gender, age, and app category.
- Crucial for feature-based models to capture feature interactions (aka., cross features)

Logistic Regression (LR)



Features

Weights

Model Equation:

$$\hat{y}(\mathbf{x}) = \mathbf{w}^T\mathbf{x} = \sum_{i=1}^{n} w_i x_i$$

- Each feature *i* has a weight w_i

An example of CTR prediction:

Publisher	Advertiser
ESPN	Nike

$$s = w_{\text{ESPN}} + w_{\text{Nike}}$$

Pros:

- Simple & Easy to interpret
- Easy to do online learning

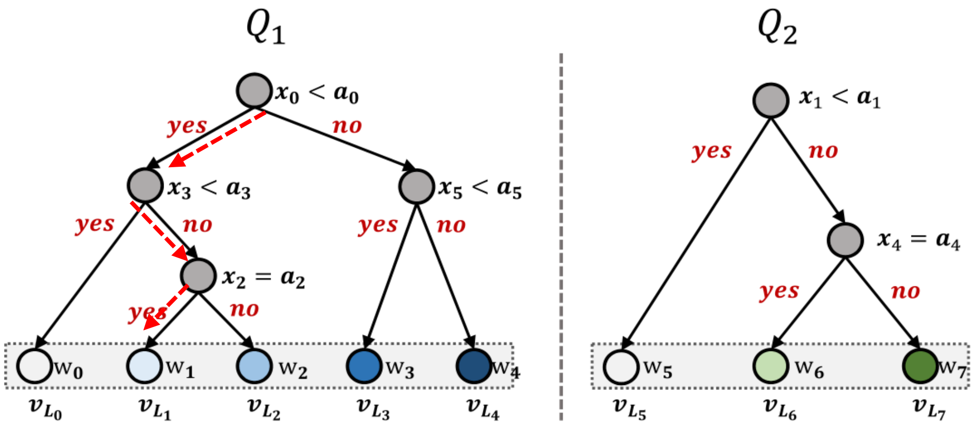
Cons:

- Features are independent
- Need manual feature engineering to design cross features.

Tree-based Models

Decision Tree (DT):

- A node splits a feature into decision edges based on its value.
- A **path** from root to leaf forms a decision rule (i.e., **cross feature**).
- Leaf node stores **prediction value**.



leaf node v_{L_2} represents $[x_0 < a_0]\&[x_3 \geq a_3]\&[x_2 \neq a_2]$

Gradient Boosting Decision Trees (GBDT):

- Build multiple trees
- Combine predictions of multiple trees in an **additive** way

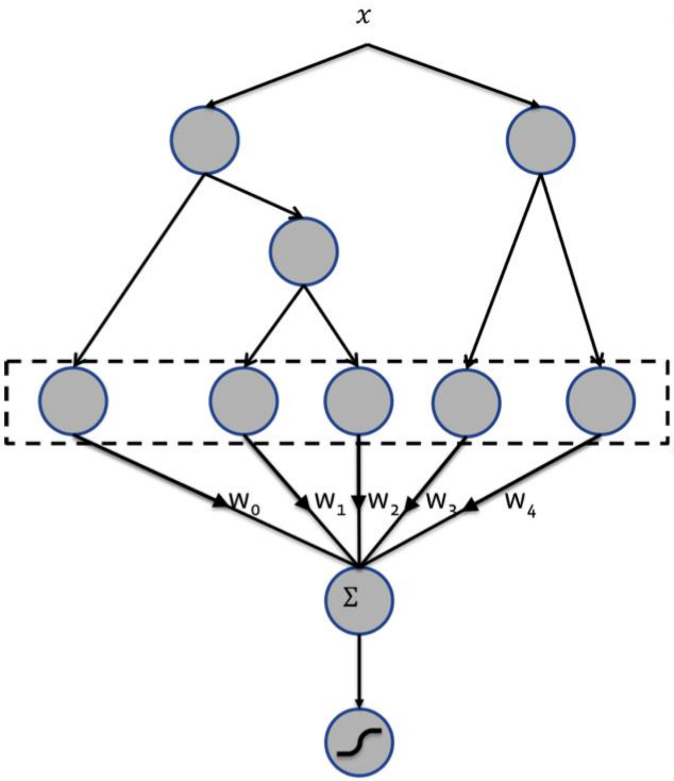
$$\hat{y}_{GBDT}(\mathbf{x}) = \sum_{s=1}^{\overset{\text{\# of trees}}{S}} \underbrace{\hat{y}_{DT_s}(\mathbf{x})}_{\text{Prediction of the } s\text{-th tree}},$$

- Can capture more complex decision patterns than a single tree.

(Wang et al, WWW'18)

GBDT + LR

- Early Facebook CTR prediction solution (He et al, ADKDD'14)



GBDT: converts feature vector to multiple cross features

- Acts as a non-linear feature transformer.

LR: reassigns the weights of cross features for a prediction

Pros:

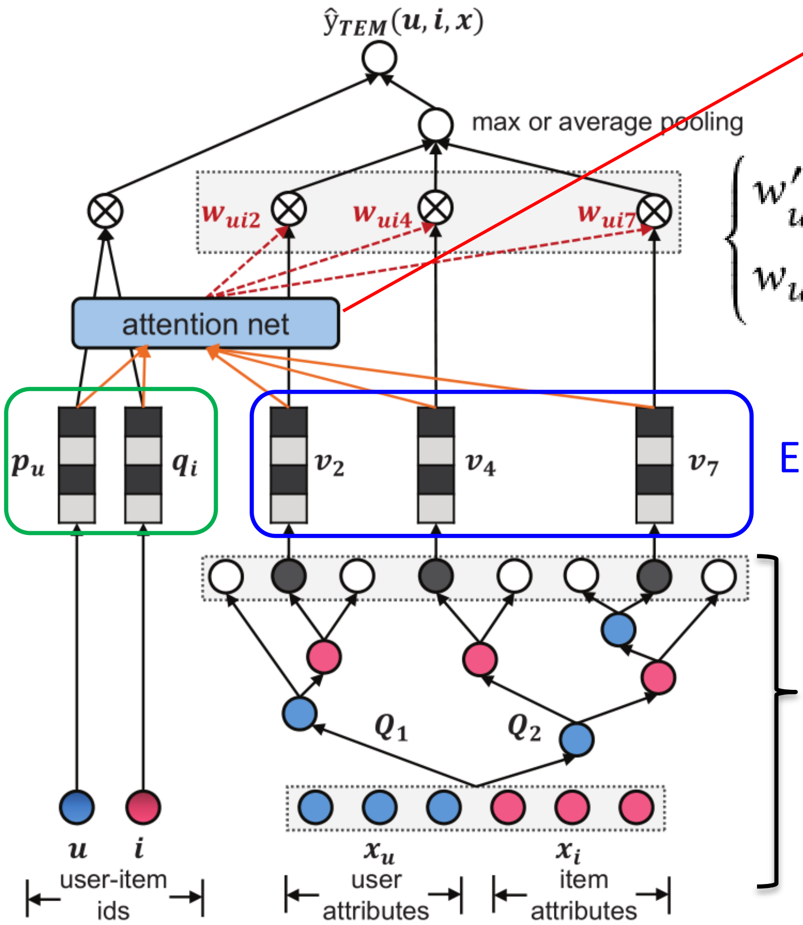
- Alleviate the need for manual feature engineering

Cons:

- Limited expressiveness: the weight of a cross feature is unchanged for all predictions.

GBDT + Attention Net

- Tree-enhanced Embedding Model (TEM, Wang et al, WWW'17) adapts the weights of cross features by using neural attention network.



Attention Net outputs weight of a cross feature for prediction (*u, i*) based on their embeddings:

$$\begin{cases} w'_{uil} = \mathbf{h}^{\top} ReLU(\mathbf{W}([\mathbf{p}_u \odot \mathbf{q}_i, \mathbf{v}_l]) + \mathbf{b}) \\ w_{uil} = \dfrac{exp(w'_{uil})}{\sum_{(u,i,x)\in O} exp(w'_{uil})} \end{cases}$$

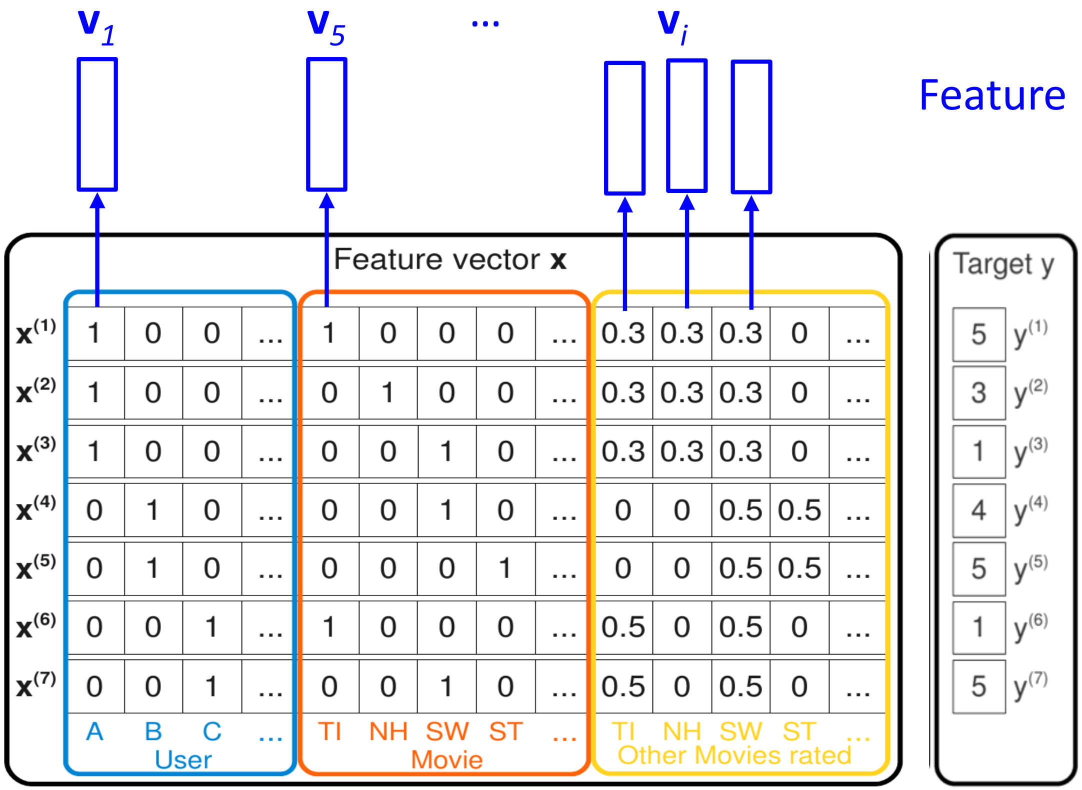
Embeddings of user and item

Embeddings of cross features

GBDT: generating cross features (each tree activates one leaf node)

Embedding-based Models

- Learning the semantics of features in vector space.
  - ➢ Each feature is associated with an embedding vector.
  - ➢ Model prediction is a function of embeddings, rather than raw features.



Feature Embeddings

Feature vector **x**

	A	B	C	...	TI	NH	SW	ST	...	TI	NH	SW	ST	...	Target y	
$\mathbf{x}^{(1)}$	1	0	0	...	1	0	0	0	...	0.3	0.3	0.3	0	...	5	$y^{(1)}$
$\mathbf{x}^{(2)}$	1	0	0	...	0	1	0	0	...	0.3	0.3	0.3	0	...	3	$y^{(2)}$
$\mathbf{x}^{(3)}$	1	0	0	...	0	0	1	0	...	0.3	0.3	0.3	0	...	1	$y^{(3)}$
$\mathbf{x}^{(4)}$	0	1	0	...	0	0	1	0	...	0	0	0.5	0.5	...	4	$y^{(4)}$
$\mathbf{x}^{(5)}$	0	1	0	...	0	0	0	1	...	0	0	0.5	0.5	...	5	$y^{(5)}$
$\mathbf{x}^{(6)}$	0	0	1	...	1	0	0	0	...	0.5	0	0.5	0	...	1	$y^{(6)}$
$\mathbf{x}^{(7)}$	0	0	1	...	0	0	1	0	...	0.5	0	0.5	0	...	5	$y^{(7)}$
	User				Movie					Other Movies rated						

In the next:

- Factorization Machine (Rendle et al, ICDM’10)
- Wide&Deep (Cheng et al, Recsys’16)
- Deep Crossing (Shan et al, KDD’16)
- Neural FM (He et al, SIGIR’17)
- Attentional FM (Xiao et al, IJCAI’17)

Factorization Machine

- Extend LR by modeling pairwise interactions between feature embeddings with inner product:

Only nonzero features are considered

$$\hat{y}(\mathbf{x}) = w_0 + \underbrace{\sum_{i=1}^{p} w_i x_i}_{\text{First-order: Linear Regression}} + \underbrace{\sum_{i=1}^{p} \sum_{j>i}^{p} < \mathbf{v}_i, \mathbf{v}_j > \boxed{x_i x_j}}_{\text{Second-order: pair-wise interactions between features}}$$

  - Note: self-interaction is not included: $\cancel{< \mathbf{v}_i, \mathbf{v}_i >}$.

- Example:

Publisher (P)	Advertiser (A)	Gender (G)
ESPN	Nike	Male

$y = w_{ESPN} + w_{Nike} + w_{Gender} + <\boldsymbol{v}_{ESPN},\boldsymbol{v}_{Nike}> + < \boldsymbol{v}_{ESPN},\boldsymbol{v}_{Male} > + < \boldsymbol{v}_{Nike},\boldsymbol{v}_{Male} >$

Pros:

- Feature interactions are learned automatically.

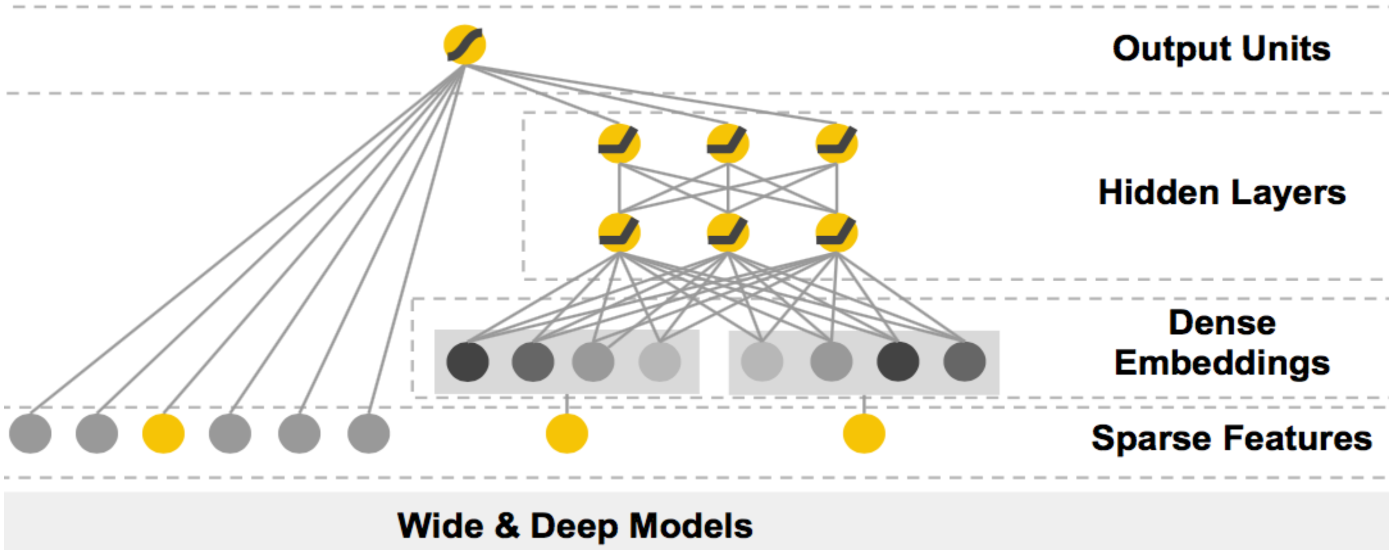
Cons:

- Only 2nd-order feature interactions. (inefficient for higher order interactions)

(Rendle et al, ICDM’10)

Wide&Deep

Google's App Recommender Ranking Solution in 2016



Standard MLP

Embedding Concatenation

- The wide part is linear regression for memorizing seen feature interactions, which requires careful engineering on cross features.

  e.g., *AND(gender=female, language=en)* is 1 iff both single features are 1

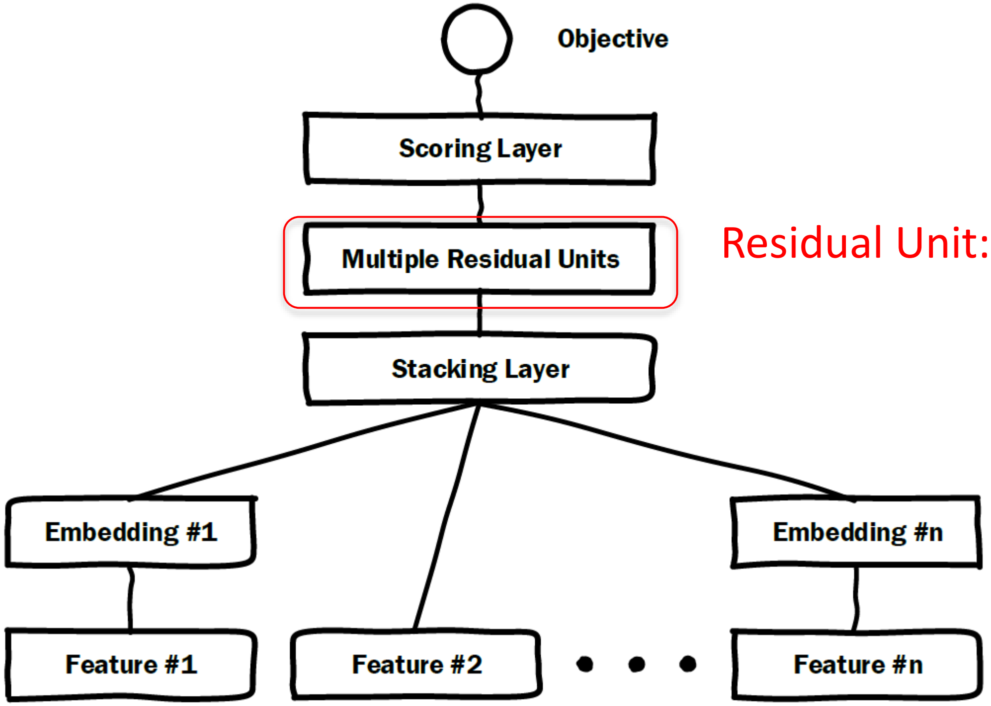
- The deep part is DNN for generalizing to unseen feature interactions.

  Feature interactions are captured in an implicit way.

(Cheng et al, Recsys'16)

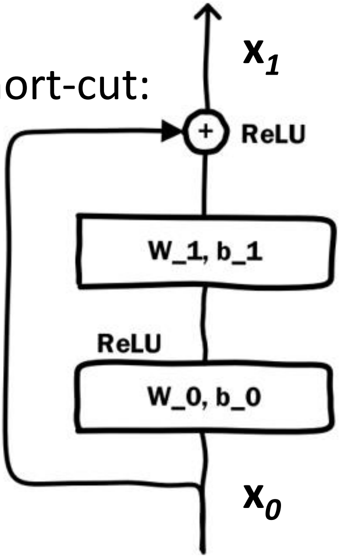
Deep Crossing

Microsoft’s CTR Prediction Solution in 2016:



Residual Unit:

short-cut:



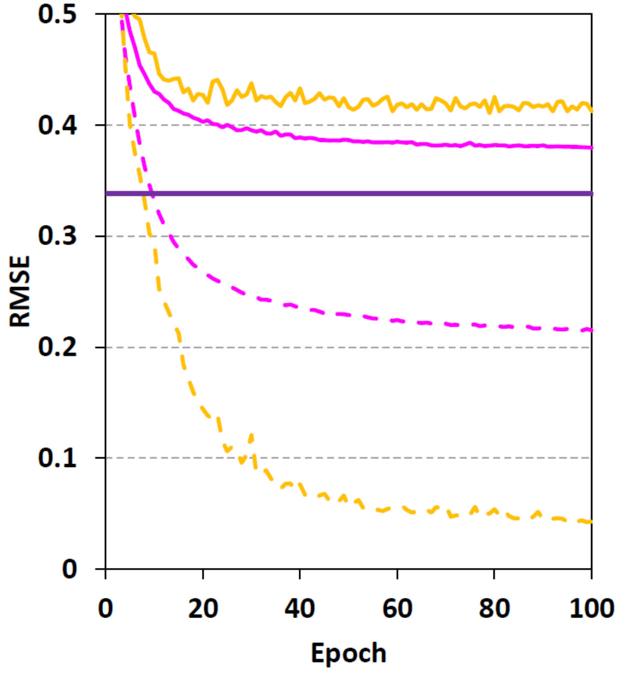
The main difference from Wide&Deep is the use of residual layers, which allow deeper network to be built (~10 layers).

(Shan et al, KDD’16)

Empirical Evidence

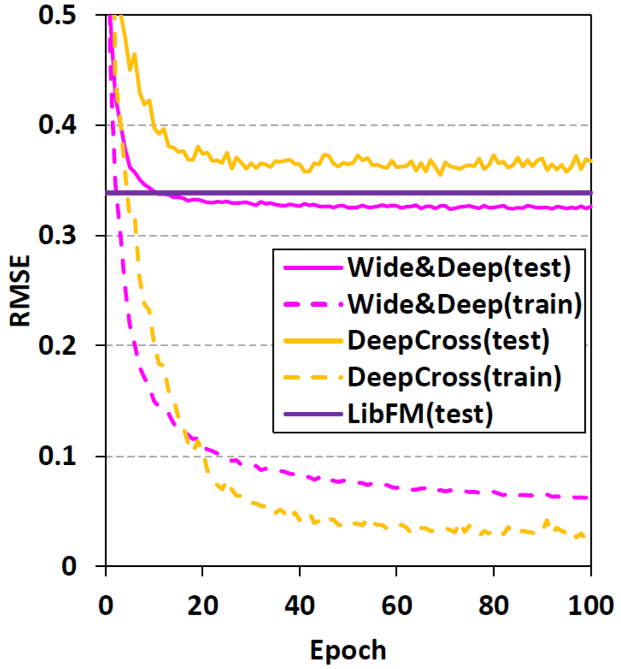
- However, when only raw features are used, Wide&Deep and DeepCross don't perform well in learning feature interactions.

Solid line: testing loss; Dashed line: training loss



(a) Random initialization

With random initialization, two deep methods underperform FM.



(b) FM as pre-training

With FM embeddings as pre-training, Wide&Deep slightly outperforms FM.

Some issues of DL methods:

Easy to overfit

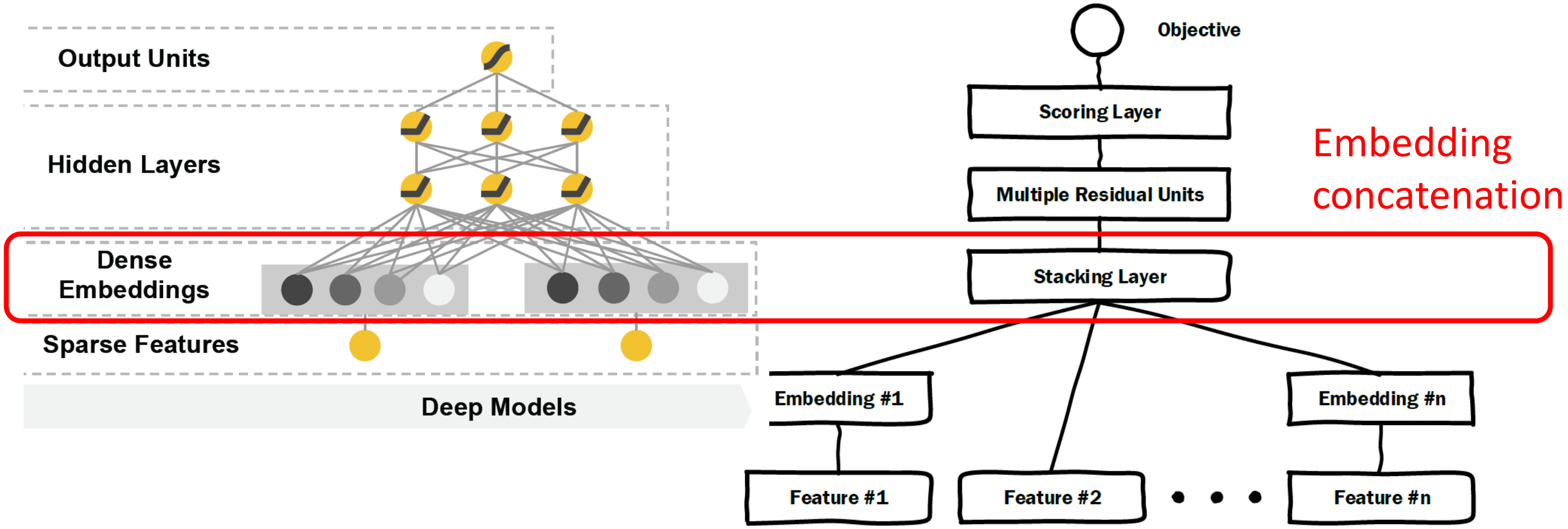
Hard to train well

Need good init.

(He and Chua, SIGIR'17)

Why DNN is Ineffective?

Besides optimization difficulties, one reason is in model design:



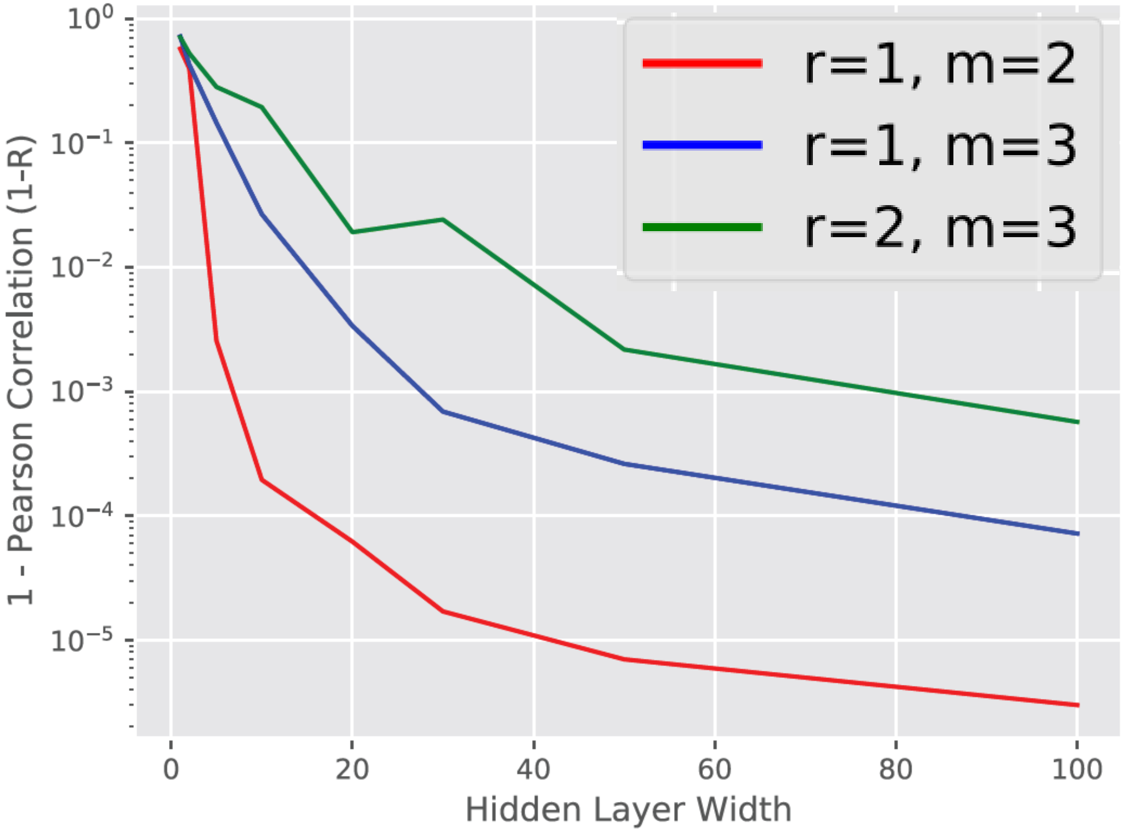
Embedding concatenation

1. Embedding concatenation carries little information about feature interactions in the low level!
2. The structure of Concat+MLP is ineffective in learning the multiplicative relation (Beutel et al, WSDM’18).

DNN is Weak in Capturing Multiplicative Relation

- Evidence from Google researchers (Beutel et al, WSDM’18)
  - ➢ Setting: generate low-rank data, and use one-layer MLP to fit it

r: rank size; m: data dimension (2 -> matrix; 3 -> 3D tensor).



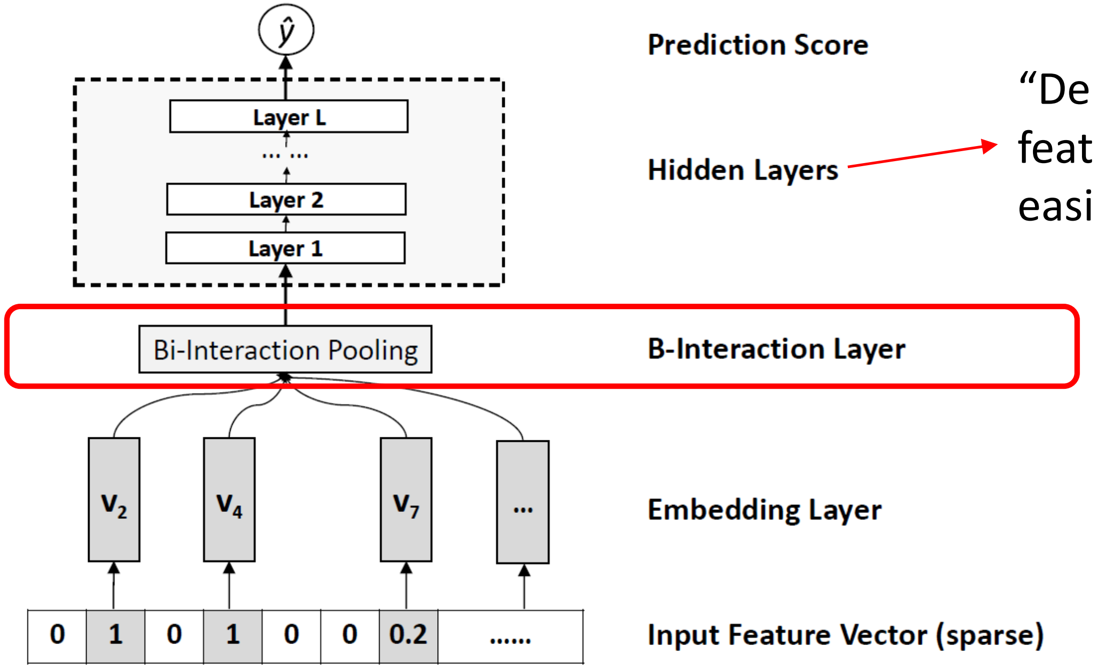
MLP can learn low-rank relation, but is inefficient in doing so!

- Need to use 100 neurons to fit a rank-1 matrix.

Insight: need to augment DNN with multiplicative relation modeling

Neural Factorization Machine

- Neural FM "deepens" FM by placing DNN above second-order interaction modeling.
  - Interaction is modeled with multiplication between embeddings



"Deep layers" learn **higher-order** feature interactions only, being much easier to train.

Bilinear Interaction Pooling:

$$f_{BI}(\mathcal{V}_x) = \sum_{i=1}^{n} \sum_{j=i+1}^{n} x_i \mathbf{v}_i \odot x_j \mathbf{v}_j,$$

Figure 2: Neural Factorization Machines model (the first-order linear regression part is not shown for clarity).

(He and Chua, SIGIR'17)

Empirical Evidence

All methods are fed into raw features without any feature engineering

Task #1: Context-aware App Usage Prediction
- Frappe data: instance #: 288,609, feature #: 5,382

Task #2: Personalized Tag Recom
- MovieLens data: Inst #: 2,006,859, Feat #: 90,445

Table: Parameter # and testing RMSE at embedding size 128

	Frappe		MovieLens	
Method	**Param#**	**RMSE**	**Param#**	**RMSE**
Logistic Regression	5.38K	0.5835	0.09M	0.5991
FM	1.38M	0.3385	23.24M	0.4735
High-order FM	2.76M	0.3331	46.40M	0.4636
Wide&Deep (3 layers)	4.66M	0.3246	24.69M	0.4512
DeepCross (10 layers)	8.93M	0.3548	25.42M	0.5130
Neural FM (1 layer)	**1.45M**	**0.3095**	**23.31M**	**0.4443**

1. Embedding methods learn interactions, better than simple linear models

2. Deep embedding methods:
Wide&Deep = Concat+3 layers
DeepCross = Concat+10 layers

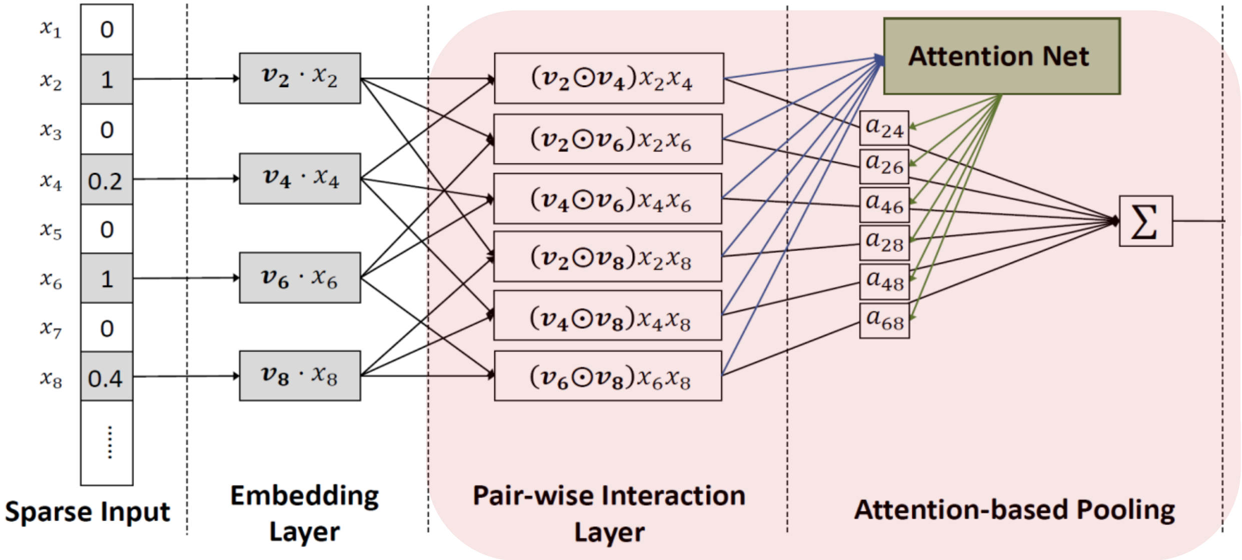
3. Neural FM
= BI pooling + 1 layer
Shallower but outperforming existing deeper methods with less parameters.

Codes: github.com/hexiangnan/neural_factorization_machine

(He and Chua, SIGIR'17)

Attentional FM

- Neural FM treats all second-order feature interactions as contributing equally.
  - ➢ However, some interactions may not be important.
- Attentional FM uses an attention network to learn the weight of a feature interaction.



$$f_{ABI}(\mathcal{V}_x) = \sum_{i=1}^{n}\sum_{j=i+1}^{n}(x_i\mathbf{v}_i \odot x_j\mathbf{v}_j)a_{ij}$$

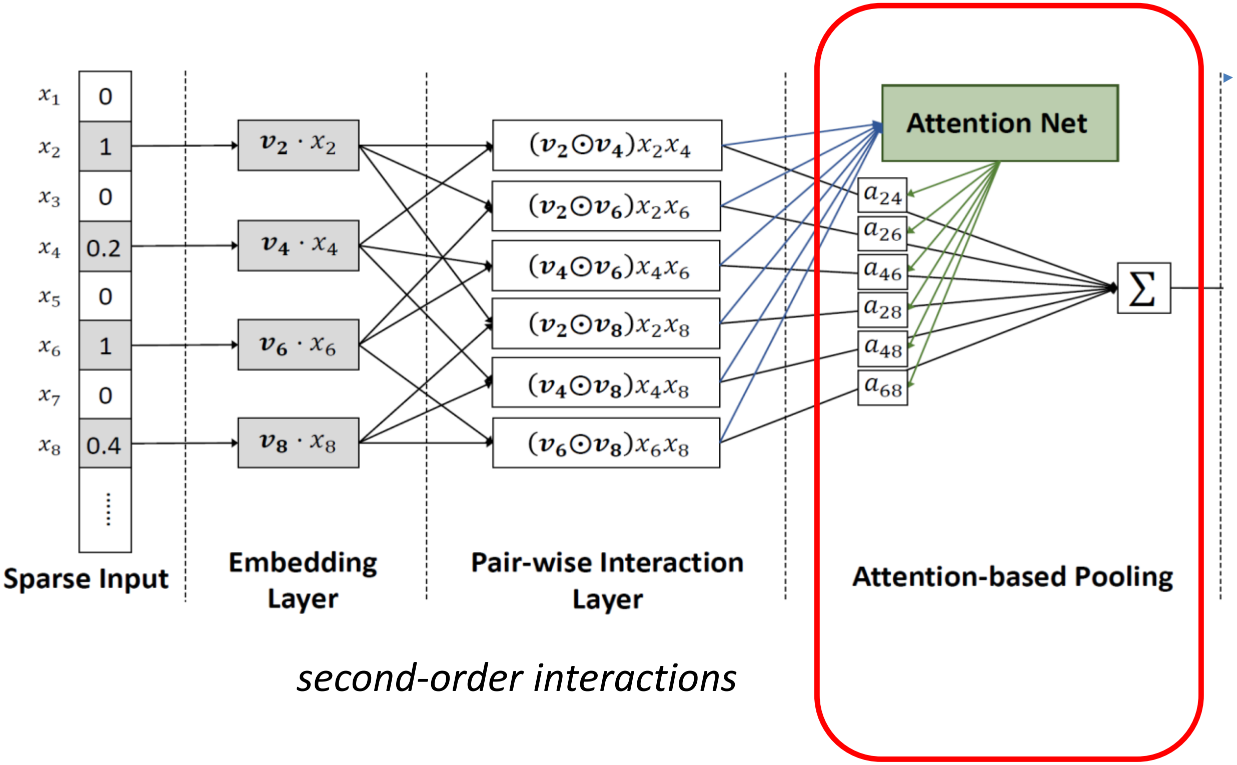
$$a'_{ij} = \mathbf{h}^T ReLU(\mathbf{W}(\mathbf{v}_i \odot \mathbf{v}_j)x_ix_j + \mathbf{b}),$$

$$a_{ij} = \frac{\exp(a'_{ij})}{\sum_{(i,j)\in\mathcal{R}_x}\exp(a'_{ij})},$$

(Xiao et al, IJCAI’17)

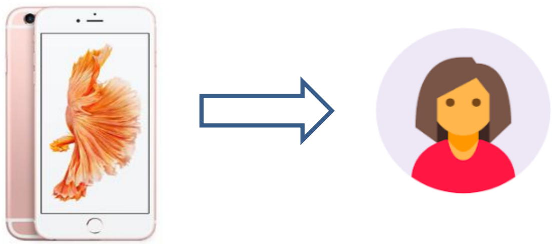
Explaining Recommendation with AFM

The attention scores can be used to select the most predictive second-order feature interactions as explanations.



second-order interactions

Example: explainable recommendation with second-order cross features:



<Female, Age 20>
<Age 20, iPhone>
<Female, Color Pink>
……

Empirical Evidence

All methods are fed into raw features without any feature engineering

Task #1: Context-aware App Usage Prediction
- Frappe data: instance #: 288,609, feature #: 5,382

Task #2: Personalized Tag Recom
- MovieLens data: Inst #: 2,006,859, Feat #: 90,445

Table: Parameter # and testing RMSE at embedding size 128

	Frappe		MovieLens	
Method	**Param#**	**RMSE**	**Param#**	**RMSE**
Logistic Regression	5.38K	0.5835	0.09M	0.5991
FM	1.38M	0.3385	23.24M	0.4735
High-order FM	2.76M	0.3331	46.40M	0.4636
Wide&Deep (3 layers)	4.66M	0.3246	24.69M	0.4512
DeepCross (10 layers)	8.93M	0.3548	25.42M	0.5130
Neural FM (1 layer)	**1.45M**	**0.3095**	**23.31M**	**0.4443**
AFM (0 layer)	**1.45M**	**0.3102**	**23.26M**	**0.4325**

AFM without hidden layers can even be better than NFM with 1 hidden layer.

Adding hidden layers to AFM further improves.

Codes: github.com/hexiangnan/neural_factorization_machine

(Xiao et al, IJCAI'17)

Summary of Recommendation

- Candidate Generation and Ranking are two major components for E-Commerce recommender systems
- For Candidate Generation:
  - Efficiency and recall are the main concern
  - The key to kNN-based solution is how to learn good features for user and item
- For Ranking:
  - Precision is the main concern.
  - The key to feature-based models is how to effectively learn feature interactions.
- Currently, the two steps are separated tuning.
  - A future direction is to jointly optimize both steps
  - E.g., using rewards from ranking stage to improve candidate generation.

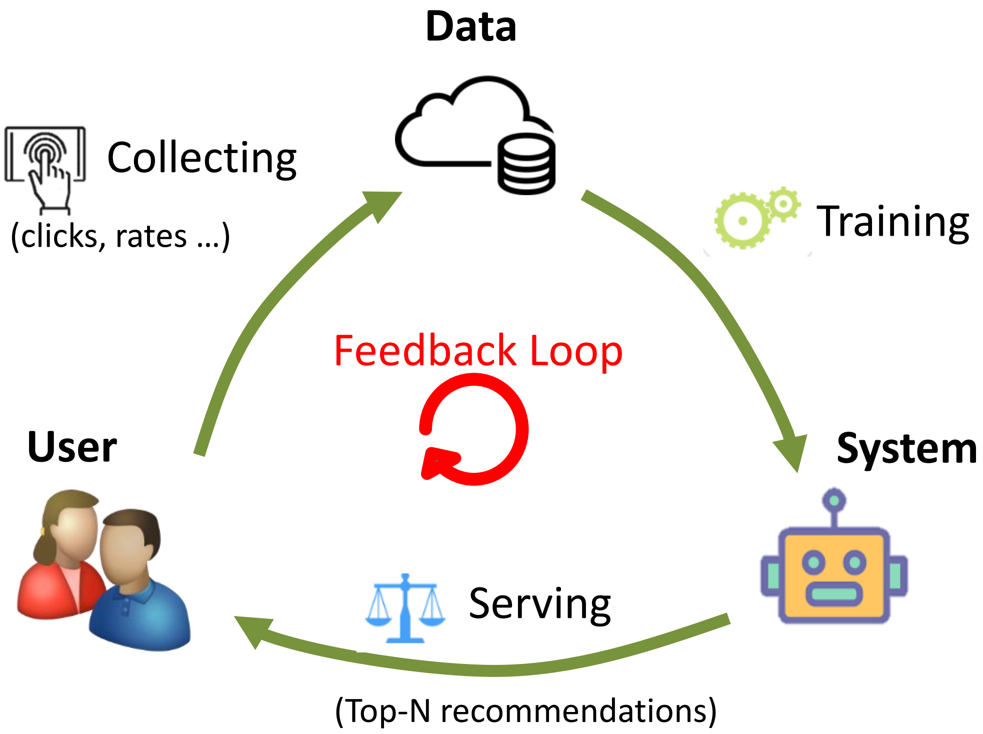
Outline of Tutorial

- Background & Basics (15 mins)
- Classical Two-stage Solutions for Recsys (60 mins)
- Rethinking Recsys Ecosystem (15 mins)

Slides are available: http://staff.ustc.edu.cn/~hexn/intro-recsys-Aug2021.pdf

Ecosystem of RecSys

- Workflow of RS
  - **Training**: RS is trained/updated on observed user-item interaction data.
  - **Serving**: RS infers user preference over items and exposes top-n items.
  - **Collecting**: User actions on exposed items are merged into the training data.
- Forming a Feedback Loop

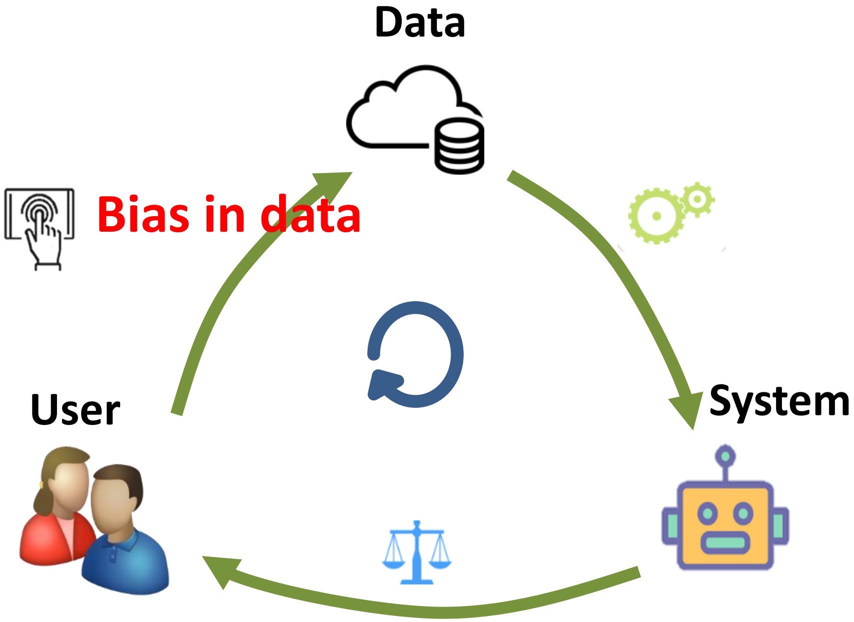


Chen et al. arxiv 2021. Bias and Debias in Recommender System: A Survey and Future Directions

Where Bias Comes?

- Bias in data (Collecting):
  - Data is observational rather than experimental (missing-not-at-random)
  - Affected by many factors:
    - The exposure mechanism
    - Display position
    - Public opinions

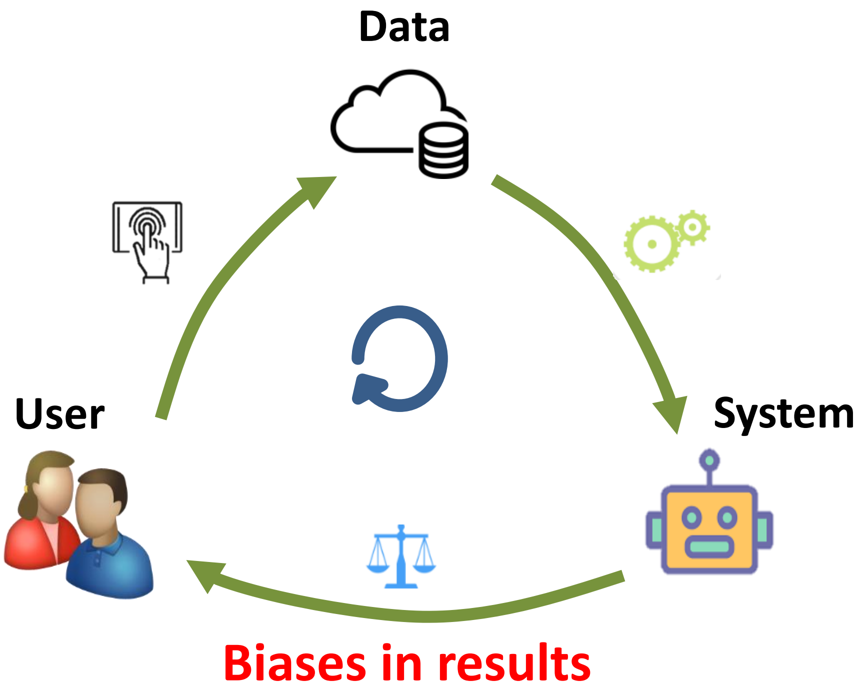
    ……
  - The collected data deviates from user true preference.



Chen et al. arxiv 2021. Bias and Debias in Recommender System: A Survey and Future Directions

Where Bias Comes?

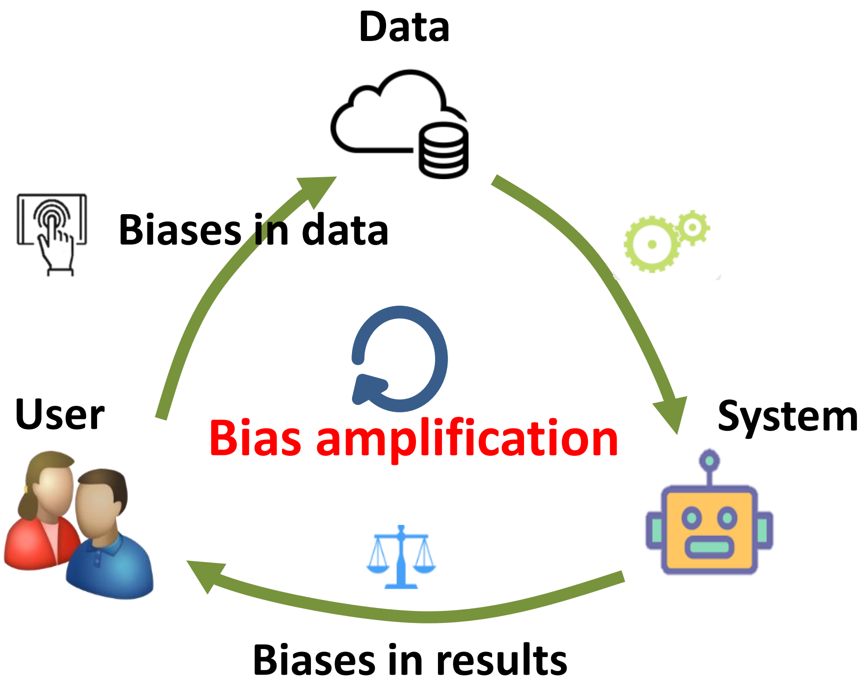
- Bias in results (Serving):
  - Unbalanced training data
  - Recommendations are in favor of some item groups
  - E.g., popularity bias, category-aware unfairness
  - Hurting user experience and satisfaction



Chen et al. arxiv 2021. Bias and Debias in Recommender System: A Survey and Future Directions

Matthew Effect: Bias + Loop

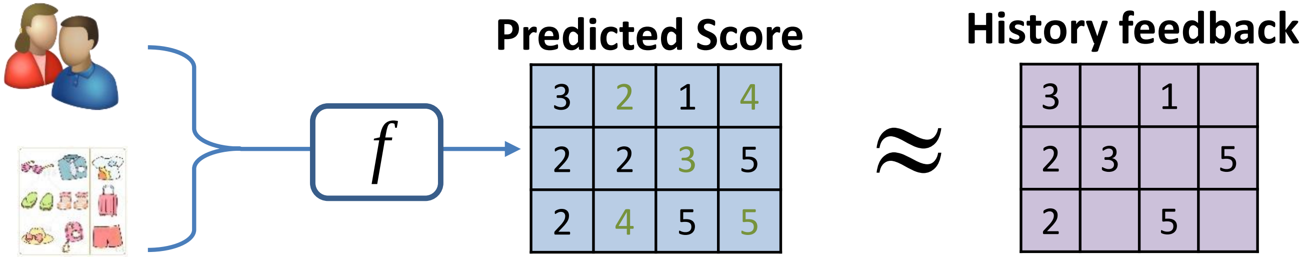
- Biases amplification along the loop:
  - Biases would be circled back into the collected data
  - Resulting in "Matthew effect" issue: the rich gets richer
  - Damaging the ecosystem of RS



Chen et al. arxiv 2021. Bias and Debias in Recommender System: A Survey and Future Directions

Mainstream Models: Fitting Historical Data

- Minimizing the difference between historical feedback and model prediction



Predicted Score

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

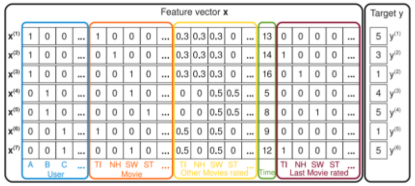
≈

History feedback

3		1	
2	3		5
2		5	

➢ Collaborative filtering

- Matrix factorization & factorization machines



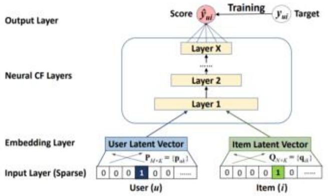
Feature vector **x**

	User				Movie					Other Movies rated					Time	Last Movie rated					Target y	
	A	B	C	...	TI	NH	SW	ST	...	TI	NH	SW	ST	...		TI	NH	SW	ST	...		
$x^{(1)}$	1	0	0	...	1	0	0	0	...	0.3	0.3	0.3	0	...	13	0	0	0	0	...	5	$y^{(1)}$
$x^{(2)}$	1	0	0	...	0	1	0	0	...	0.3	0.3	0.3	0	...	14	1	0	0	0	...	3	$y^{(2)}$
$x^{(3)}$	1	0	0	...	0	0	1	0	...	0.3	0.3	0.3	0	...	16	0	1	0	0	...	1	$y^{(2)}$
$x^{(4)}$	0	1	0	...	0	0	1	0	...	0	0	0.5	0.5	...	5	0	0	0	0	...	4	$y^{(3)}$
$x^{(5)}$	0	1	0	...	0	0	0	1	...	0	0	0.5	0.5	...	8	0	0	1	0	...	5	$y^{(4)}$
$x^{(6)}$	0	0	1	...	1	0	0	0	...	0.5	0	0.5	0	...	9	0	0	0	0	...	1	$y^{(5)}$
$x^{(7)}$	0	0	1	...	0	0	1	0	...	0.5	0	0.5	0	...	12	1	0	0	0	...	5	$y^{(6)}$

Factorization Machines

➢ Deep learning approaches

- Neural factorization machines & graph neural networks

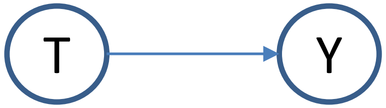


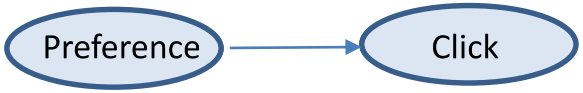
Neural Collaborative Filtering

Shortcomings of Data-Driven Methods

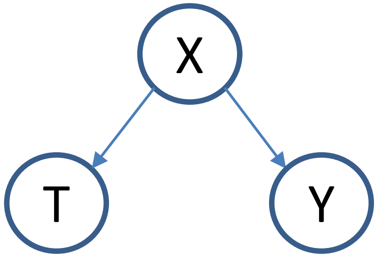
- Data-driven methods suffer from biases & feedback loop as they are only able to capture correlations, rather than causality.

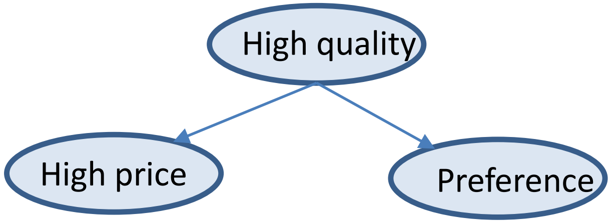
- Three basic types of correlations:
  - **Causation**
    - Stable and explainable



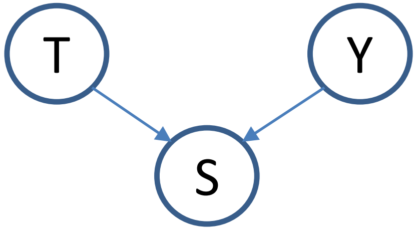


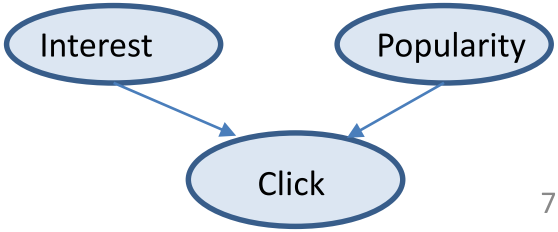
  - **Confounding**
    - Ignoring X
    - Spurious correlation





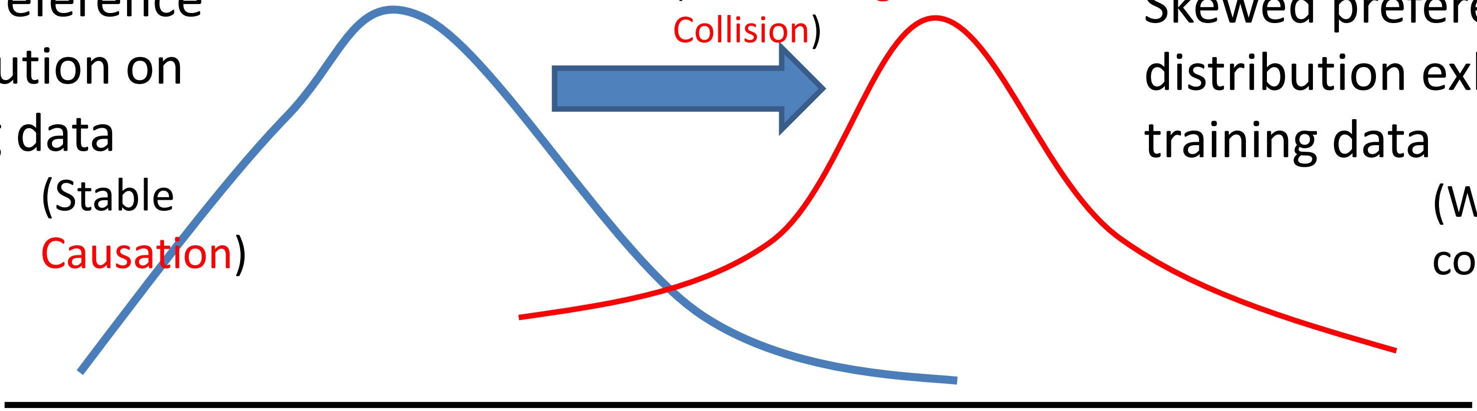
  - **Collision**
    - Condition on S
    - Spurious correlation





Shortcomings of Data-Driven Methods

- Data-driven methods would learn skewed user preference:



Biases

(Confounding, Collision)

True preference distribution on testing data

(Stable Causation)

Skewed preference distribution exhibited on training data

(With spurious correlation)

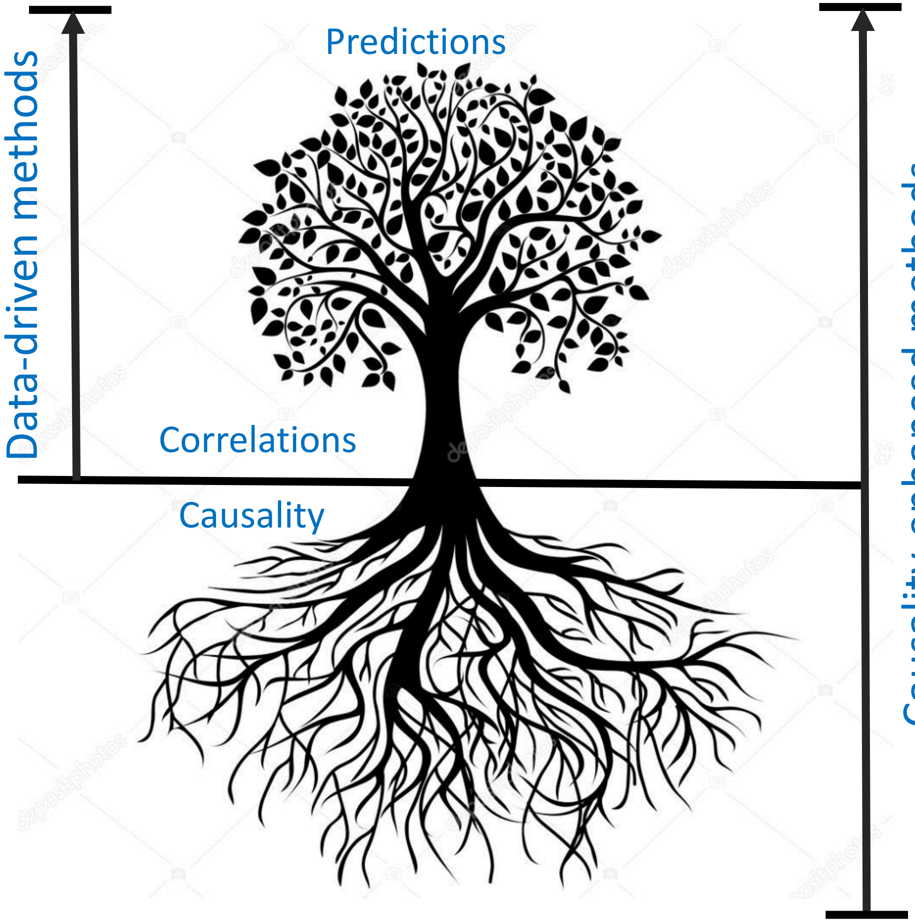
- Data-driven methods may infer spurious correlations, which are deviated from reflecting user true preference and lack interpretation.

Chen et al. arxiv 2021. Bias and Debias in Recommender System: A Survey and Future Directions

Why Causal Inference?

- Aim: Understanding the inherent causal mechanism of user behavior
  - Capturing user true preference

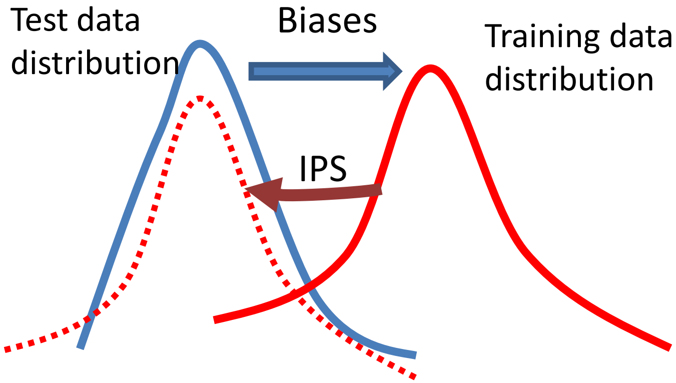
- Making reliable & explainable recommendations
  - Correlation + Causality > Correlation



Existing Causal^{+} Method: IPS

- Basic idea: intervene data distribution by sample reweighting:

$$L_{ips} = \frac{1}{U \cdot I} \sum_{(u,i) \in D_T} \frac{1}{\boxed{ps(u,i)}} \delta(y_{ui}, \hat{y}_{ui})$$



- Properly defining propensity scores can lead to *unbiased estimator* of the ideal:

$$L_{ideal} = \frac{1}{U \cdot I} \sum_{u \in U, i \in I} \delta(y_{ui}, \hat{y}_{ui}) \neq E(L_{naive}) = E\left(\frac{1}{|\{(u,i): O_{ui} = 1\}|} \sum_{u \in U, i \in I} \delta(y_{ui}, \hat{y}_{ui})\right.$$

$$= E(L_{ips}) = \frac{1}{U \cdot I} \sum_{u \in U, i \in I} E_{O_{ui}} \frac{O_{ui}}{ps(u,i)} \delta(u,i) = \frac{1}{U \cdot I} \sum_{u \in U, i \in I} \delta(y_{ui}, \hat{y}_{ui})$$

👎 But, finding good propensity is not easy.

Subjected to high variance.

Schnabel, Tobias, et al. "Recommendations as treatments: Debiasing learning and evaluation." international conference on machine learning. PMLR, 2016.

Existing Causal^{+} Method: Causal Embedding

- Utilizing cause-specific data (e.g., uniform data) to guide model learning.

E.g., Joint training (CausE [1]):

$$\min_{\mathcal{W}_c,\mathcal{W}_t} \overbrace{\frac{1}{|S_c|}\sum_{(i,j)\in S_c} \ell\left(y_{ij},\hat{y}^{c}_{ij}\right)}^{\text{On uniform data}} + \overbrace{\frac{1}{|S_t|}\sum_{(i,j)\in S_t} \ell\left(y_{ij},\hat{y}^{t}_{ij}\right)}^{\text{On biased data}} + \lambda_c R\left(\mathcal{W}_c\right) + \lambda_t R\left(\mathcal{W}_t\right) + \lambda^{CausE}_{tc}\left\|\mathcal{W}_t - \mathcal{W}_c\right\|^2_F,$$

Guiding term

- Other ways: knowledge distillation [2]

But, obtaining uniform data is not easy.

Uniform data is much smaller.

[1] Bonner, Stephen et.al. "Causal embeddings for recommendation.” In RecSys 2018.

[2] Liu, Dugang, et al. "A general knowledge distillation framework for counterfactual recommendation via uniform data.” In SIGIR 2020.

Our Recent Work

- Work#1: Improving IPS via learning propensity scores from uniform data.

  SIGIR 2021. AutoDebias: Learning to Debias for Recommendation.

- Work#2: Improving Causal Embedding via pairwise cause-specific data.

  WWW 2021. Disentangling User Interest and Conformity for Recommendation with Causal Embedding

- Work#3: Eliminating popularity bias via counterfactual inference.

  KDD 2021. Model-Agnostic Counterfactual Reasoning for Eliminating Popularity Bias in Recommender System

- Work#4: Leveraging popularity bias via causal intervention.

  SIGIR 2021 Best Paper Honorable Mention. Causal Intervention for Leveraging Popularity Bias in Recommendation

Thanks!

Slides are available: http://staff.ustc.edu.cn/~hexn/intro-recsys-Aug2021.pdf