Recommendation System: Basic and Why AutoML is Needed?

Yong Li
(Tsinghua University)

https://sites.google.com/view/kdd20-marketplace-autorecsys/

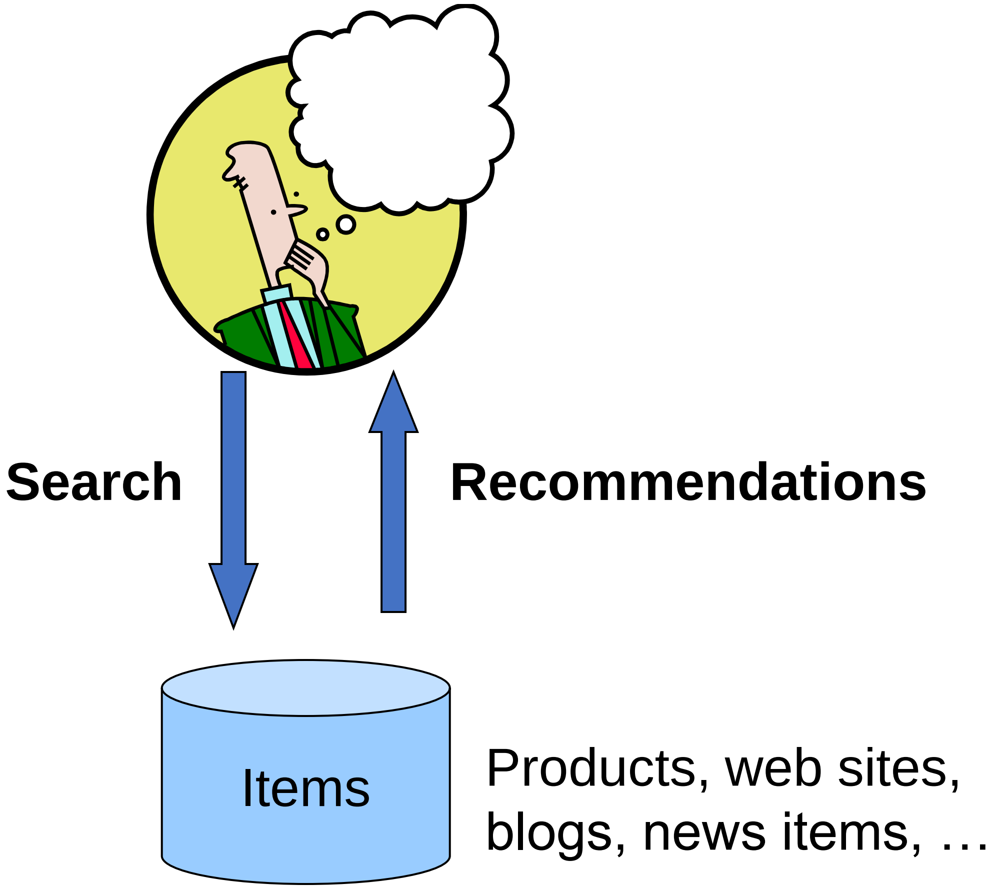
Outline

1. **What is a recommender system**
2. **Recent advances in recommender system**
    a) **Deep Learning**
    b) **Graph Neural Networks**
    c) **Knowledge Graph**
3. **Problem of human-crafted recommender system and why AutoML is needed**

Outline

1. **What is a recommender system**
2. **Recent advances in recommender system**
   a) **Deep Learning**
   b) **Graph Neural Networks**
   c) **Knowledge Graph**
3. **Problem of human-crafted recommender system and why AutoML is needed**

Recommendations

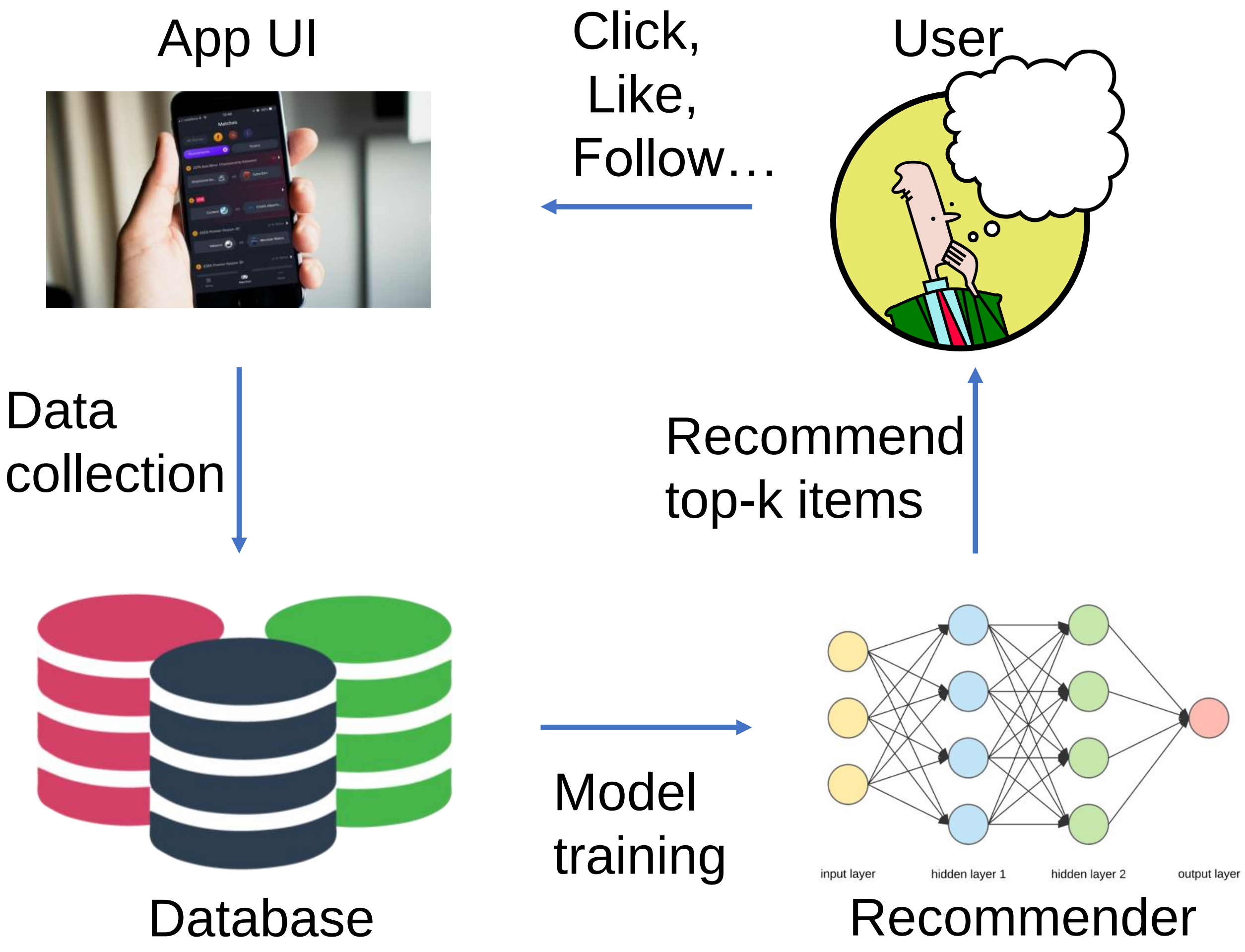


Examples:

amazon.com
PANDORA
StumbleUpon
NETFLIX
Google News
last.fm the social music revolution
XBOX LIVE
YouTube

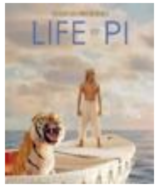
Recommendations

Modelling users' preference towards items based on historical behaviours, such as click, like, follow, etc.



Problem Formulation

- **Input:** historical user-item interactions or additional side information (e.g., user profile, item profile)
- **Output**: given a target Item (e.g., movie, song, product), how likely a user would interact with it (e.g., click, view, or purchase)



User Profile:
- User ID
- Rating history
- Age, Gender
- Clicks
- Income level

……..

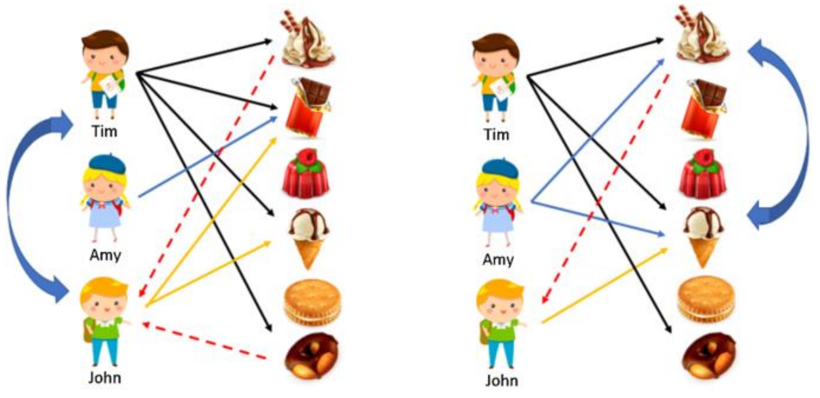
Item Profile:
- Item ID
- Description
- Image
- Category
- Price

……..

Key challenge: user-item semantic gap

- user and item are two **different types of entities.** There may be no overlap between user features and item features.

Collaborative Filtering



(a) User-based filtering (b) Item-based filtering

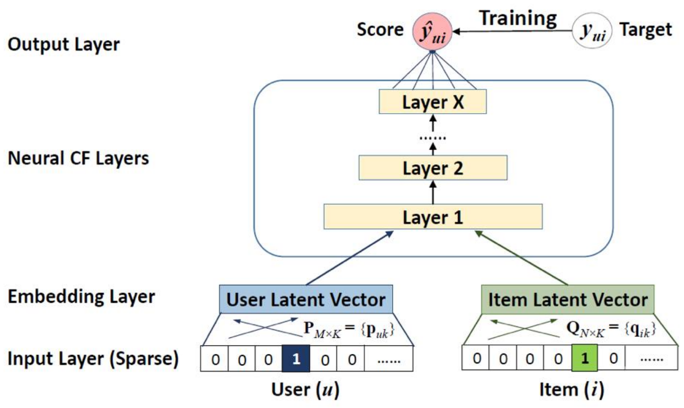
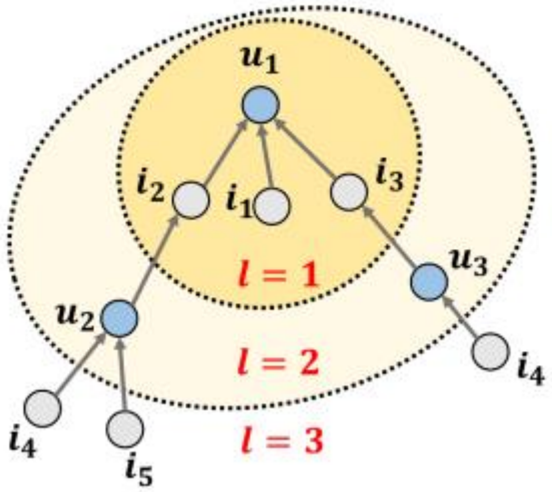


Figure 2: Neural collaborative filtering framework

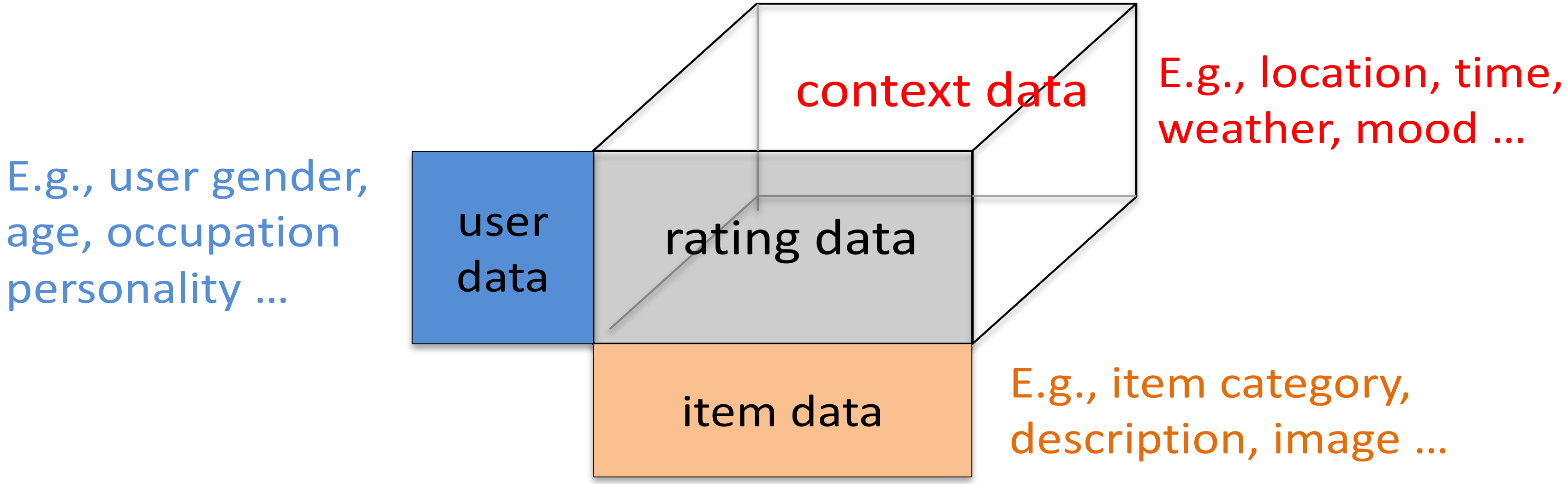


High-order Connectivity for u_1

- Predict users' preference from similar users' records
- Factorize historical behaviours into representations of users and items

Feature-based Recommendation

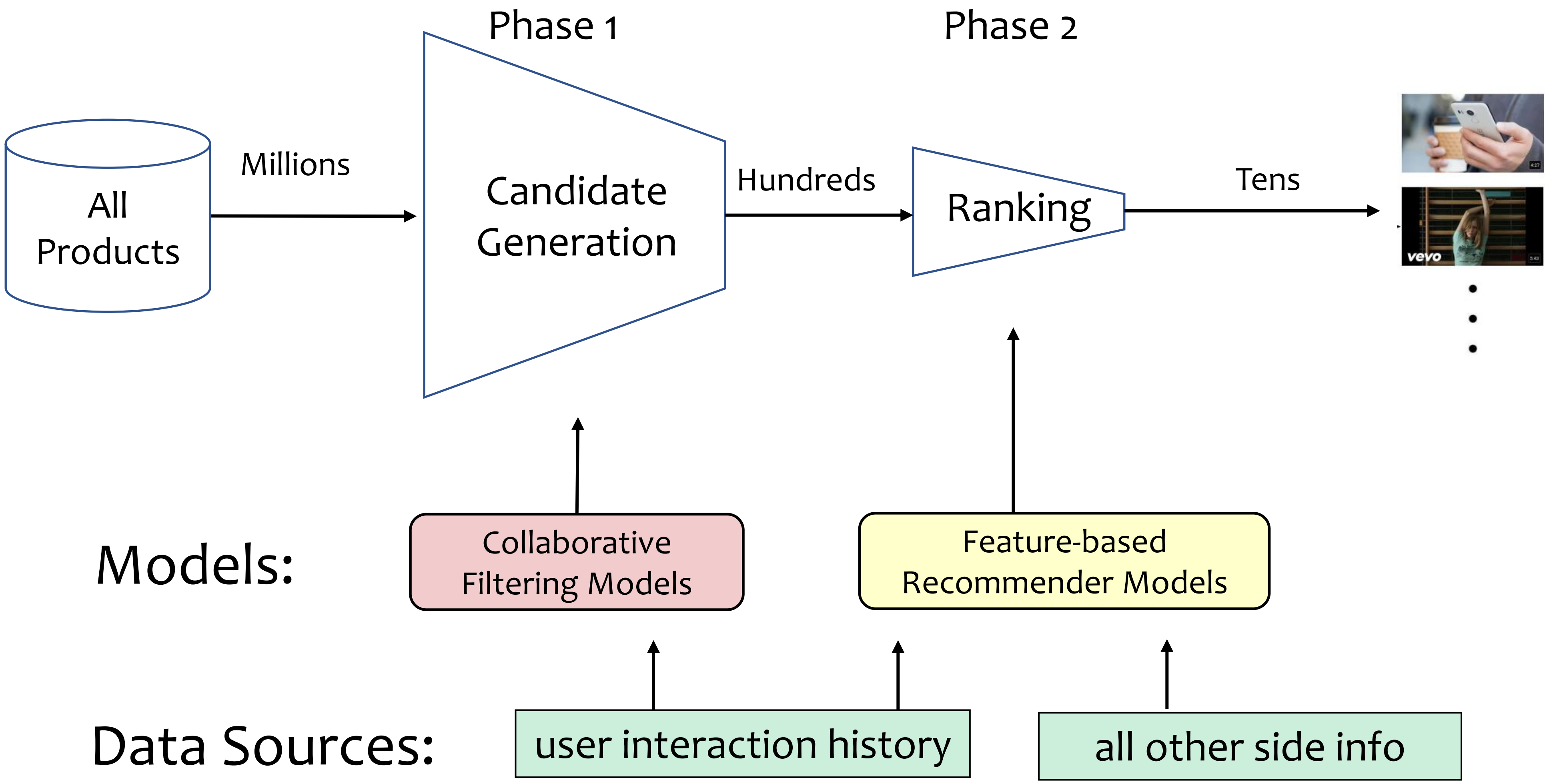
- CF utilizes only the interaction matrix only to build the predictive model.
- How about other information like user/item attributes and contexts?
- Example data used for building a RecSys:



Outline

1. What is a recommender system
2. **Recent advances in recommender system**
   a) **Deep Learning**
   b) Graph Neural Networks
   c) Knowledge Graph
3. Problem of human-crafted recommender system
4. How AutoML comes to help

Modern RecSys Architecture (Covington et al, Recsys’16)



Deep Learning for Recommendation

1. **CF models:**

   Only ID or interaction history is used as input.
   - **DeepMF: Deep Matrix Factorization (Xue et al, IJCAI'17)**
   - **NeuMF: Neural Matrix Factorization (He et al, WWW'17)**
   - **ConvNCF: Outer Product-based NCF (He et al, IJCAI'18)**
   - **AutoRec: Autoencoders Meeting CF (Sedhain et al, WWW'15)**
   - **CDAE: Collaborative Denoising Autoencoder (Wu et al, WSDM'16)**

2. **Feature-based recommendation:**

   Any available data can be used as input.
   - **DCF: Deep Collaborative Filtering via Marginalized DAE (Li et al, CIKM'15)**
   - **Wide&Deep (Cheng et al, DLRS'16)**
   - **DUIF: Deep User-Image Feature (Geng et al, ICCV'15)**
   - **ACF: Attentive Collaborative Filtering (Chen et al, SIGIR'17)**
   - **CKB: Collaborative Knowledge Base Embeddings (Zhang et al, KDD'16)**

Deep Matrix Factorization (Xue et al, IJCAI'17)

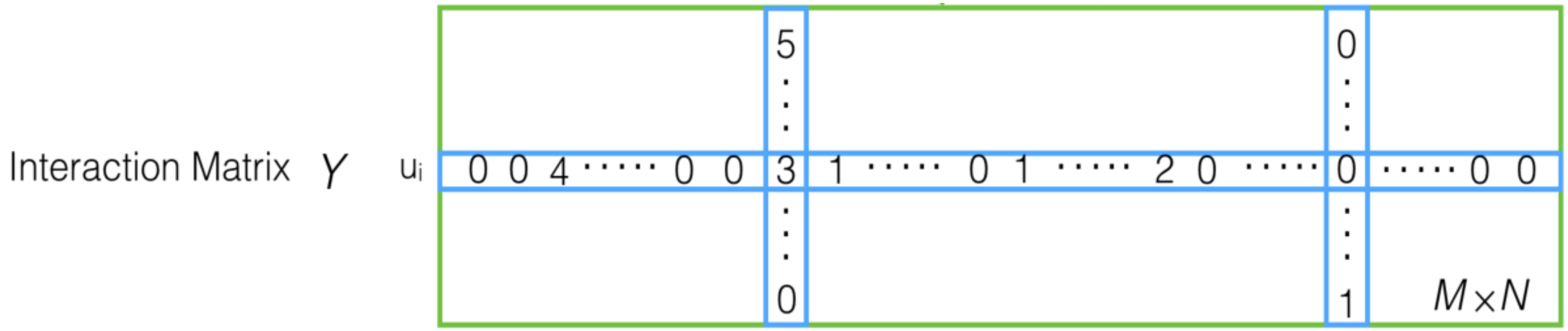
- **Input**:

  user -> historically rated items (multi-hot), i.e., row vector of Y

  indicates the user's global preference

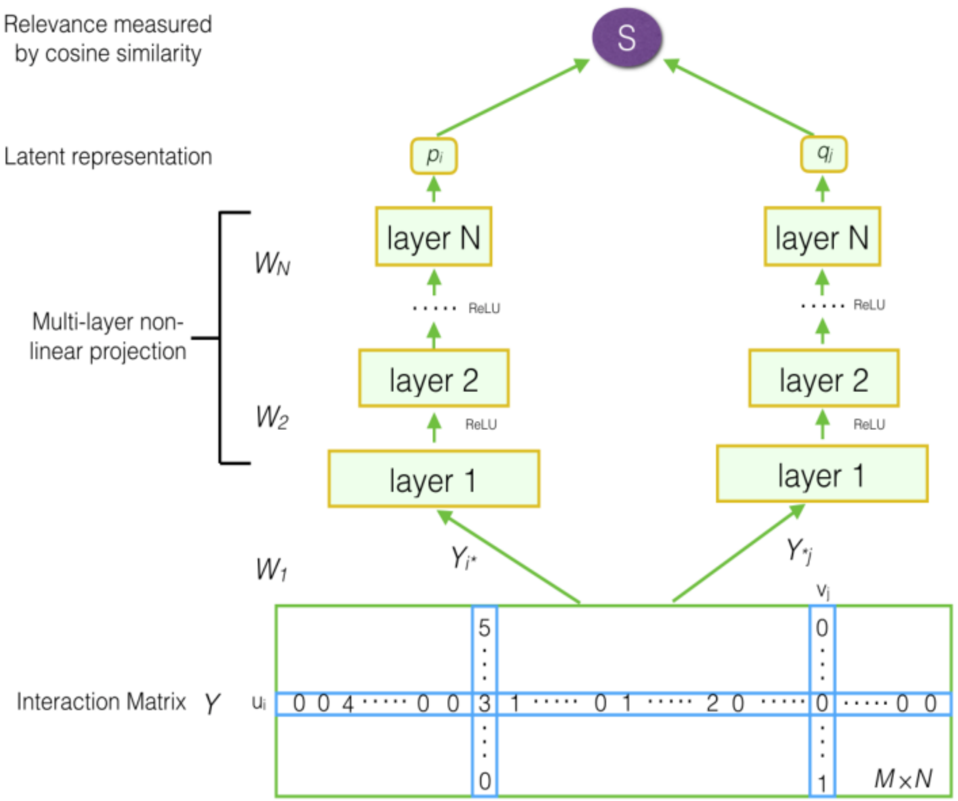
  item -> users who have rated it (multi-hot), i.e., column vector of Y

  indicates the item's rating profile.



Deep Matrix Factorization (Xue et al, IJCAI’17)

- **Representation Function**:
  - Multi-Layer Perceptron



Matching Function: cosine similarity

$$cosine(p_i, q_j) = \frac{p_i^T q_j}{\|p_i\| \, \|q_j\|}$$

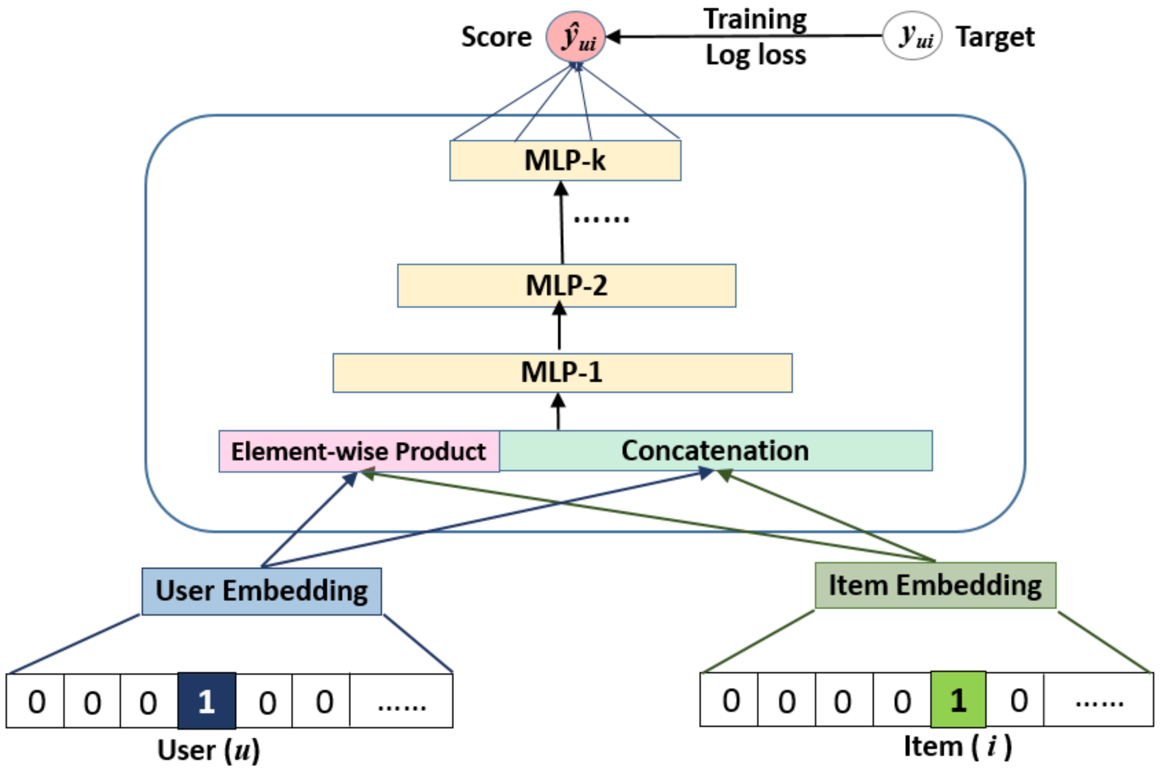
$$l_1 = W_1 x$$
$$l_i = f(W_{i-1} l_{i-1} + b_i), i = 2, ..., N-1$$
$$h = f(W_N l_{N-1} + b_N)$$

$$p_i = f_{\theta_N^U}(...f_{\theta_3^U}(W_{U2} f_{\theta_2^U}(Y_{i*} W_{U1}))...)$$

$$q_j = f_{\theta_N^I}(...f_{\theta_3^I}(W_{V2} f_{\theta_2^I}(Y_{*j}^T W_{V1}))...)$$

NeuMF: Neural Matrix Factorization (He et al, WWW'17)

- NeuMF unifies the strengths of MF and MLP in learning the matching function:
  - MF uses inner product to capture the low-rank relation
  - MLP is more flexible in using DNN to learn the matching function.



Methods of Representation Learning

1. CF models:

   Only ID or interaction history is used as input.

   - DeepMF: Deep Matrix Factorization (Xue et al, IJCAI'17)
   - NeuMF: Neural Matrix Factorization (He et al, WWW'17)
   - ConvNCF: Outer Product-based NCF (He et al, IJCAI'18)
   - AutoRec: Autoencoders Meeting CF (Sedhain et al, WWW'15)
   - CDAE: Collaborative Denoising Autoencoder (Wu et al, WSDM'16)

2. **Feature-based recommendation:**

   Any available data can be used as input.

   - **DCF: Deep Collaborative Filtering via Marginalized DAE (Li et al, CIKM'15)**
   - **Wide&Deep (Cheng et al, DLRS'16)**
   - **DUIF: Deep User-Image Feature (Geng et al, ICCV'15)**
   - **ACF: Attentive Collaborative Filtering (Chen et al, SIGIR'17)**
   - **CKB: Collaborative Knowledge Base Embeddings (Zhang et al, KDD'16)**

Input to Feature-based Models

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	$y^{(7)}$

Raw features:

1. Categorical features
   One-hot encoding on ID features
2. Continuous features
   E.g., time, frequency.
   Need feature normalization

Transformed features:

1. Categorical features
Cross features are important
(e.g., AND (A=true, B=true))

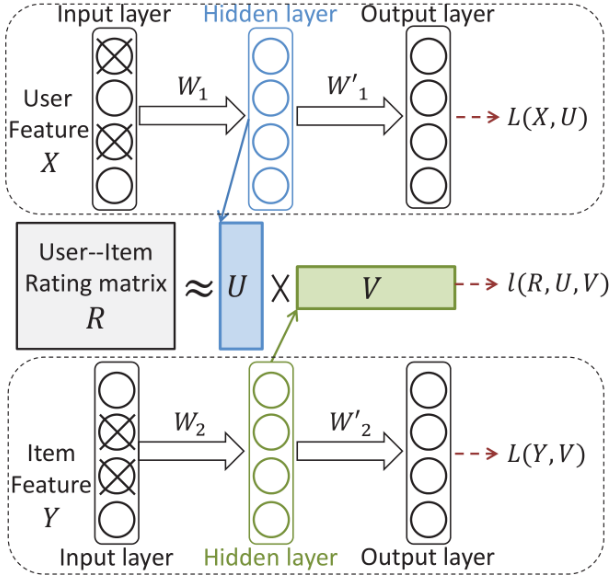
2.Continuous features
E.g., outputs of other models like visual embeddings.

Deep Collaborative Filtering via Marginalized DAE (Li et al, CIKM'15)

- Denoising Auto-Encoder is used to learn features (hidden layers) of user and item from side information.
- The predictive model is MF.

age, gender, city, occupation, locations …

genres, title, texts



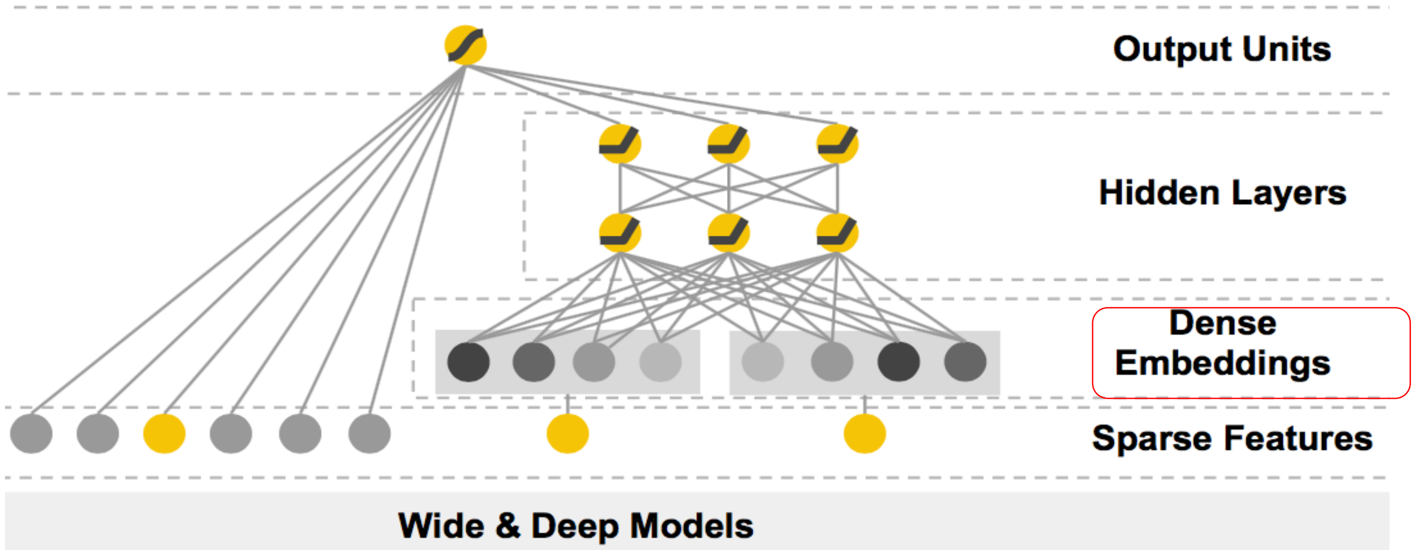
User features reconstruction

Item features reconstruction

$$\arg\min_{\substack{U,V,W_1,\\ W_2,P_1,P_2}} \boxed{\mathcal{L}_U(W_1,P_1,U)} + \boxed{\mathcal{L}_V(W_2,P_2,V)} + \underline{\alpha\|A\odot(R-UV^\top)\|_F^2} + \beta(\|U\|_F^2+\|V\|_F^2)$$

Matrix Factorization Kernel

Wide&Deep (Cheng et al, Recsys'16)



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 for generalizing to unseen feature interactions.**

Short Summary

- Deep Learning is utilized to substitute nearly all components in recommender system.
  - Feature extraction
  - Representation learning
  - Matching function learning

- Deep Learning shows great power in modeling high-order similarity in recommender system, e.g. feature interaction in Wide&Deep, matching function in NeuMF…

Outline

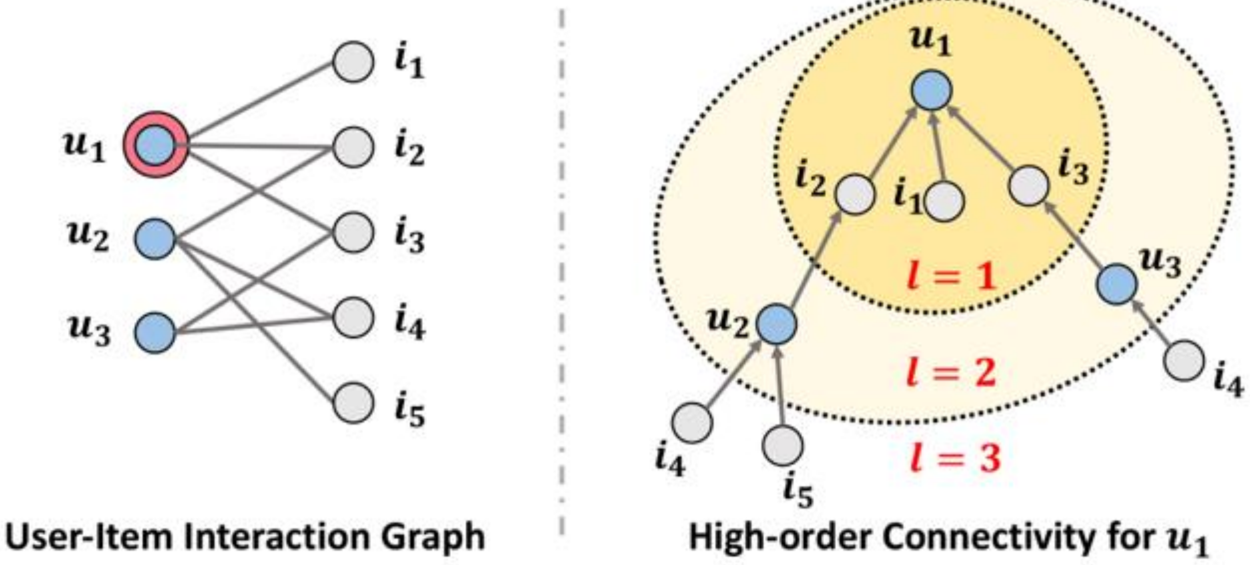
1. What is a recommender system
2. **Recent advances in recommender system**
   a) Deep Learning
   b) **Graph Neural Networks**
   c) Knowledge Graph
3. Problem of human-crafted recommender system
4. How AutoML comes to help

Recap Collaborative Filtering (CF)

- Revisit CF via **high-order connectivity**
  - The paths that reach u_1 from any node with the path length l larger than 1
  - A natural way to encode collaborative signal in the interaction graph structure

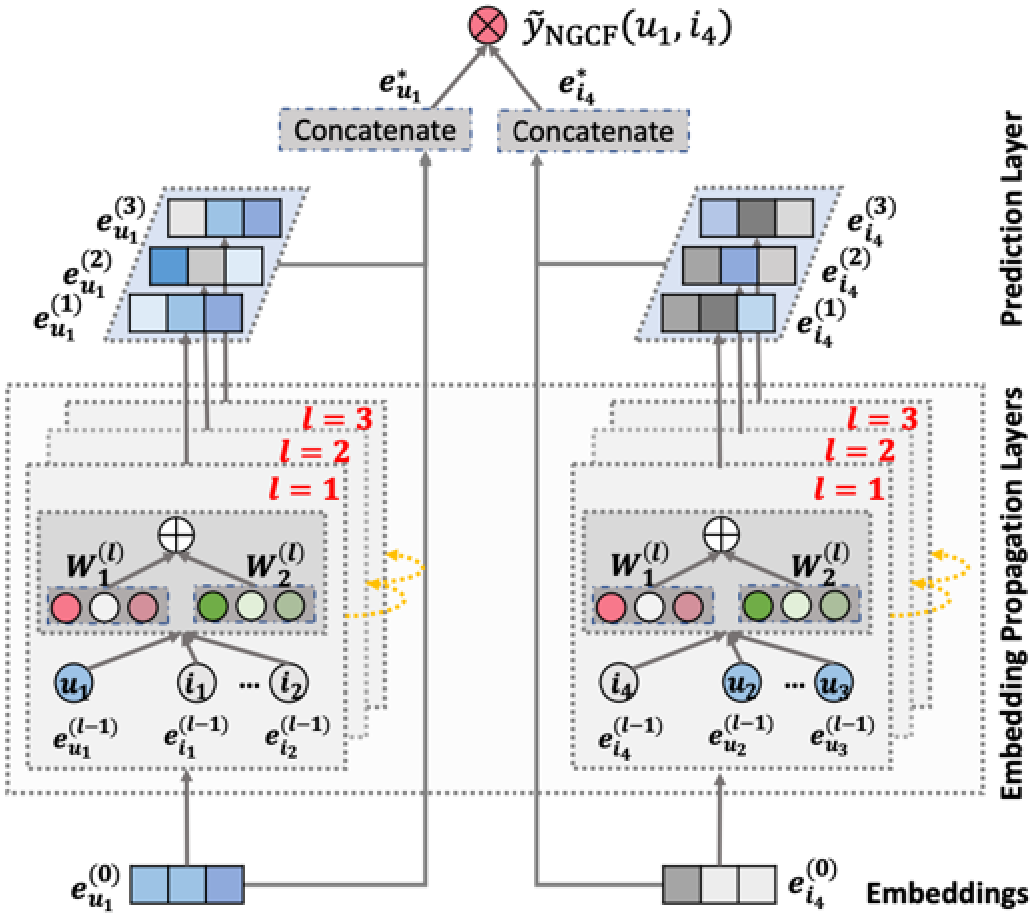
Why u_1 may like i_4?

- $u_1 \leftarrow i_2 \leftarrow u_2 \leftarrow i_4$
- $u_1 \leftarrow i_3 \leftarrow u_3 \leftarrow i_4$



Wang et al. Neural Graph Collaborative Filtering. SIGIR 2019

Neural Graph Collaborative Filtering (Wang et al, SIGIR2020)



Wang et al. Neural Graph Collaborative Filtering. SIGIR 2019

$$\mathbf{e}^*_u = \mathbf{e}^{(0)}_u \| \cdots \| \mathbf{e}^{(L)}_u$$

$$\mathbf{e}^*_i = \mathbf{e}^{(0)}_i \| \cdots \| \mathbf{e}^{(L)}_i$$

$$\hat{y}_{NGCF}(u, i) = {\mathbf{e}^*_u}^\top \mathbf{e}^*_i$$

The representations at different layers

- emphasize the messages passed over different connections
- have different contributions in reflecting user preference

First-order Connectivity Modeling

Inspired by GNNs

1. Propagate embeddings recursively on the user-item graph
2. Construct information flows in the embedding space

- Comp.1: Information Construction:

message passed from i to u

$$\mathbf{m}_{u\leftarrow i} = \frac{1}{\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}}\left(\mathbf{W}_1\mathbf{e}_i + \mathbf{W}_2(\mathbf{e}_i \odot \mathbf{e}_u)\right)$$

discount factor

- message dependent on the affinity, distinct from GCN, GraphSage, etc.
- Pass more information to similar nodes

- Comp.2 & 3: Neighbor Aggregation & Representation Update:

$$\mathbf{e}_u^{(1)} = \mathrm{LeakyReLU}\left(\mathbf{m}_{u\leftarrow u} + \sum_{i\in\mathcal{N}_u}\mathbf{m}_{u\leftarrow i}\right)$$

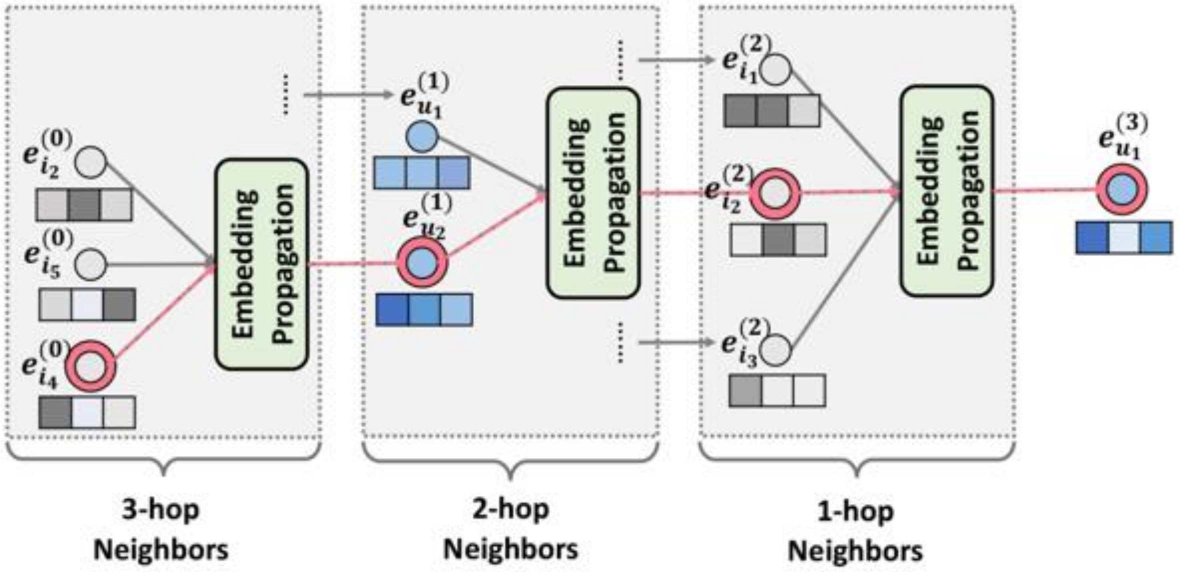
self-connections

all neighbors of u

Wang et al. Neural Graph Collaborative Filtering. SIGIR 2019

High-order Connectivity Modeling

- Stack more embedding propagation layers to explore the high-order connectivity



- The collaborative signal like u1 ← i2 ← u2 ← i4 can be captured in the embedding propagation process.

- **Collaborative signal can be injected into the representation learning process.**

Wang et al. Neural Graph Collaborative Filtering. SIGIR 2019

LightGCN (He et al, SIGIR2020)

- NGCF matrix form

$$\mathbf{E}^{(l)} = \cancel{\mathrm{LeakyReLU}}\Big((\mathcal{L} \cancel{+ \mathbf{I}})\mathbf{E}^{(l-1)}\cancel{\mathbf{W}_1^{(l)}} + \mathcal{L}\mathbf{E}^{\cancel{(l-1)}} \odot \mathbf{E}^{(l-1)}\cancel{\mathbf{W}_2^{(l)}}\Big)$$

- LightGCN matrix form

$$\mathbf{E}^{(k+1)} = \boxed{(\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}})\mathbf{E}^{(k)}}$$

Only simple weighted sum aggregator is remained

- No feature transformation
- No nonlinear activation
- No self connection

$$\begin{aligned}\mathbf{E} &= \alpha_0\mathbf{E}^{(0)} + \alpha_1\mathbf{E}^{(1)} + \alpha_2\mathbf{E}^{(2)} + \ldots + \alpha_K\mathbf{E}^{(K)} \\ &= \boxed{\alpha_0}\mathbf{E}^{(0)} + \boxed{\alpha_1}\tilde{\mathbf{A}}\mathbf{E}^{(0)} + \boxed{\alpha_2}\tilde{\mathbf{A}}^2\mathbf{E}^{(0)} + \ldots + \boxed{\alpha_K}\tilde{\mathbf{A}}^K\mathbf{E}^{(0)}\end{aligned}$$

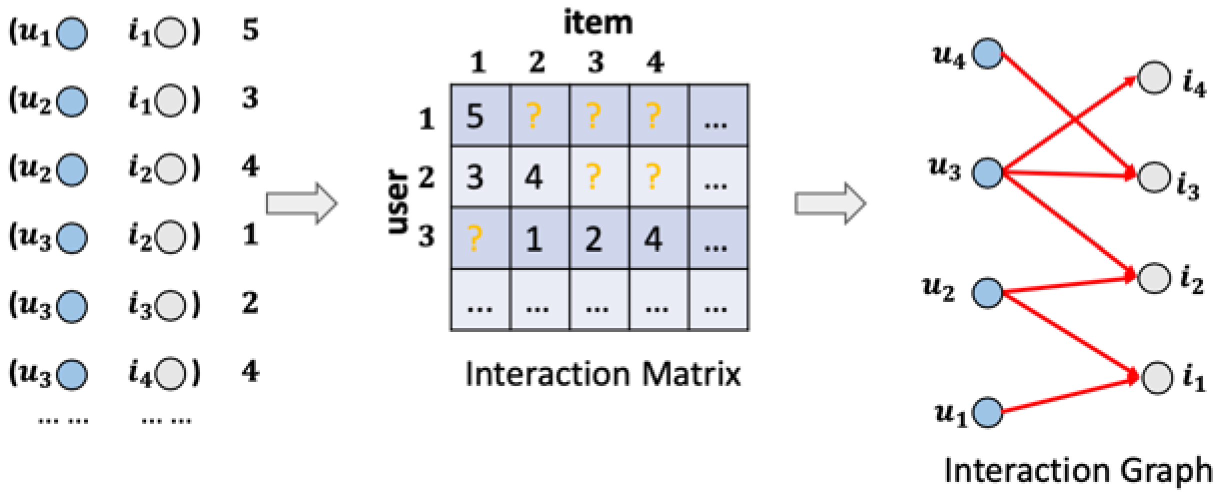
importance of the k-th layer embedding in constituting the final embedding

He et al. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation.

Summary: GNN for CF

Reorganizing the user-item interaction data into a bipartite graph bridges the interaction instances

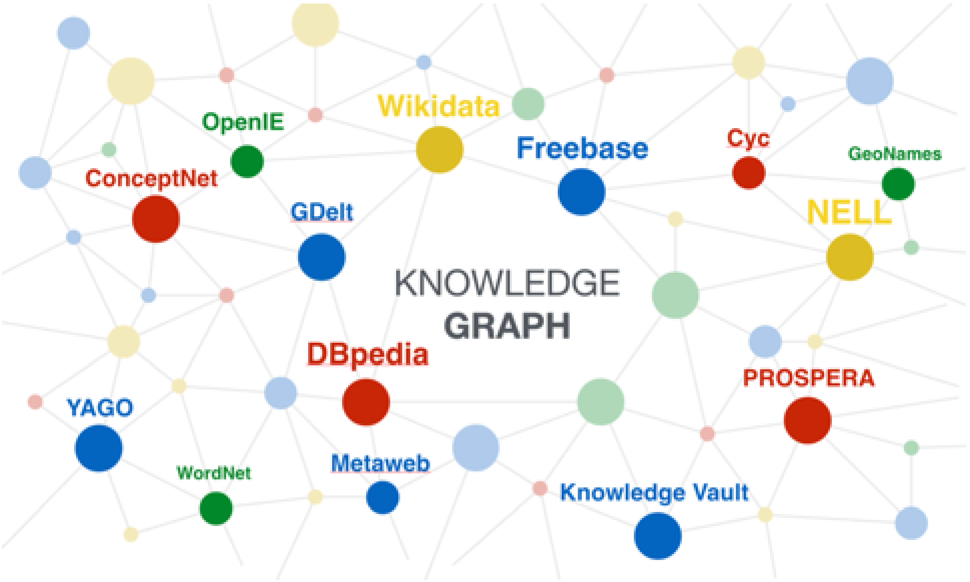
- Exhibit the relationships among users and items → high-order connectivity
- Encode high-order connectivity via GNN → collaborative signals
- It is of great need to reduce unnecessary complexity of GNN.

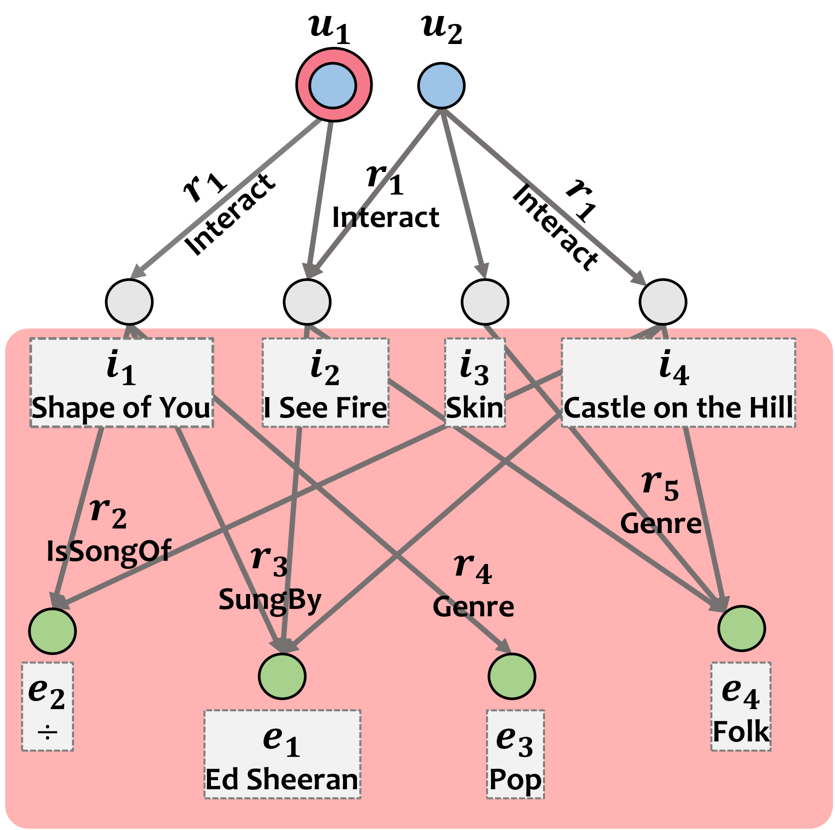


Outline

1. **What is a recommender system**
2. **Recent advances in recommender system**
   a) **Deep Learning**
   b) **Graph Neural Networks**
   c) **Knowledge Graph**
3. **Problem of human-crafted recommender system and why AutoML is needed**

Knowledge Graph-based Recommendation





Knowledge Graph (KG):

- Background knowledge on items
- Rich semantics & Relations
- Structural information

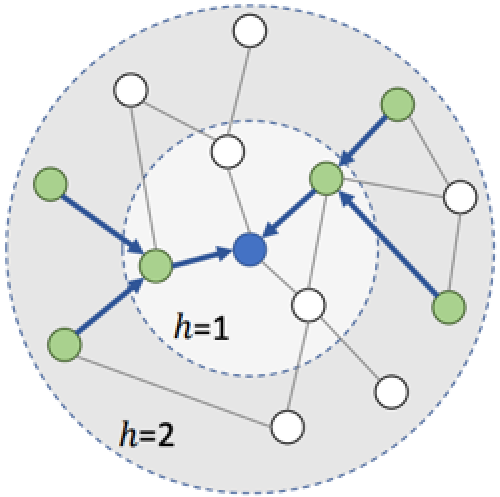
Benefit for Recommendation

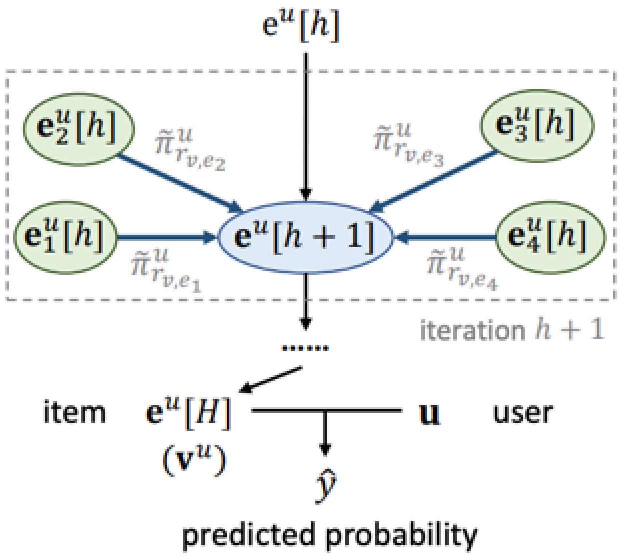
- Narrow down search space
- Explore user interests reasonably
- Offer explanations

Knowledge Graph Convolution Network (KGCN)

KGCN from [Wang et al, WWW'2019]

- Item graph → KG entities are used to enrich item representation





Comp.1 & 2

$$\mathbf{v}^u_{\mathcal{N}(v)} = \sum_{e \in \mathcal{N}(v)} \tilde{\pi}^u_{r_{v,e}} \mathbf{e}$$

Attention score of user-relation

KG entities connected with the target item

Comp.3

$$agg_{sum} = \sigma\left(\mathbf{W} \cdot (\mathbf{v} + \mathbf{v}^u_{\mathcal{S}(v)}) + \mathbf{b}\right)$$

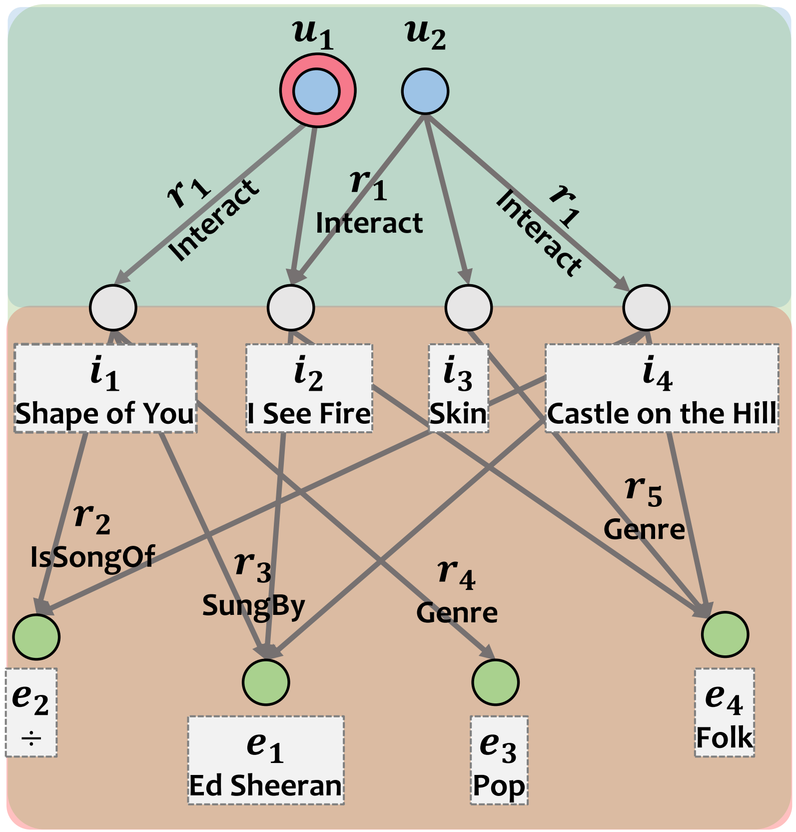
Prediction

$$\hat{y}_{uv} = f(\mathbf{u}, \mathbf{v}^u)$$

User ID embeddings → users are excluded from the propagation.

Wang et al. Knowledge Graph Convolutional Networks for Recommender Systems. WWW 2019

Knowledge Graph Attention Network (Wang et al, KDD’2019)



User-Item Bipartite Graph

- User-Item Direct Interactions

$$u_1 \xrightarrow{r_1} i_1$$

+

Knowledge Graph

- Item-Item External Connections

$$i_1 \xrightarrow{r_2} e_1$$

=

Collaborative Knowledge Graph

- High-order connectivity between users and items

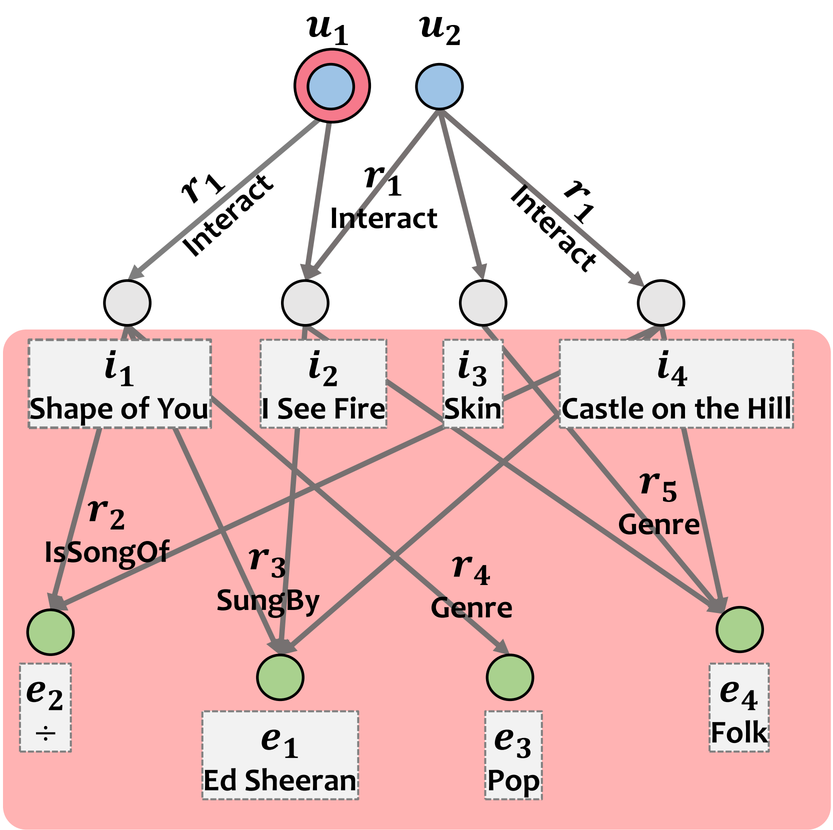
$$u_1 \xrightarrow{r_1} i_1 \xrightarrow{r_2} e_1 \xrightarrow{-r_2} i_2 \Rightarrow u_1 \xrightarrow{r_1} i_2$$

- Reasoning ability & Explainability

Wang et al. KGAT: Knowledge Attention Network for Recommendation. KDD 2019

Summary: KG for CF

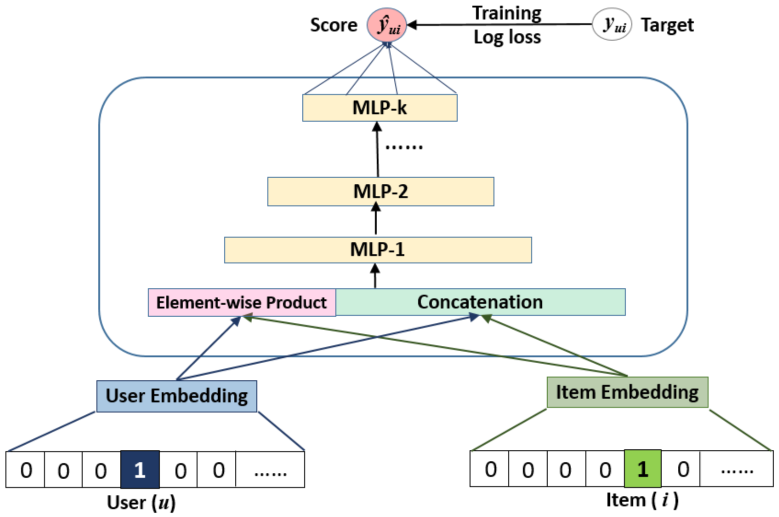
- Integrate user-item interaction, item knowledge and user knowledge into one graph -> combine user-item bipartite with knowledge graph
- Enhance representation learning of users and items with knowledge graph embeddings
- Better explainability of recommender system -> reasoning on knowledge graph



Outline

1. **What is a recommender system**
2. **Recent advances in recommender system**
   a) **Deep Learning**
   b) **Graph Neural Networks**
   c) **Knowledge Graph**
3. **Problem of human-crafted recommender system and why AutoML is needed**

Human-crafted recommender system



Too many decisions to be made!

1. **Input Features**

   feature selection, feature crossing, …

2. **Model Architecture**

   #layers, #blocks, activation functions,…

3. **Interaction Function**

   inner product, minus/plus, min/max, MLP, …

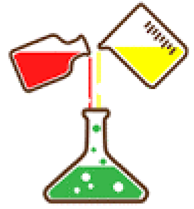
4. **Hyper-parameters**

   embedding_size, regularization, learning rate,…

Human-crafted recommender system

How to make decisions ?

1. **Designed manually by experts**



2. **Enumerated by experiments**

Both are suboptimal and inefficient! Why?

Human-crafted recommender system

1. **Designed manually by experts**
    1. Introduce noise?

       E.g. unnecessary features
    2. Miss counter-intuitive design?

       E.g. novel model architecture
    3. High labor cost

2. **Enumerated by experiments**
    1. Large memory and computation cost

Human-crafted recommender system

- **Most importantly, there is no golden rule that is universally optimal.**

- **Performance of different choices on feature/model architecture/interaction function/hyper-parameter depends on datasets and tasks.**

How to always make good decisions on different datasets and tasks?

AutoML!

AutoML for recommender system

- **Automated interaction function search** (Yao et al, WWW2020)
  - Design interaction function automatically
  - Cover both existing and new interaction functions
  - One-shot search, update interaction function and embedding jointly
  - Better performance than experts with slightly higher computation cost

- Automated feature interaction search (Liu et al, KDD2020)
  - Automatically select important low and high order feature interactions
  - Two stages to search feature interaction and re-train the model
  - Better performance on both online and offline evaluations without much computation cost

Collaborative Filtering – More Example IFCs

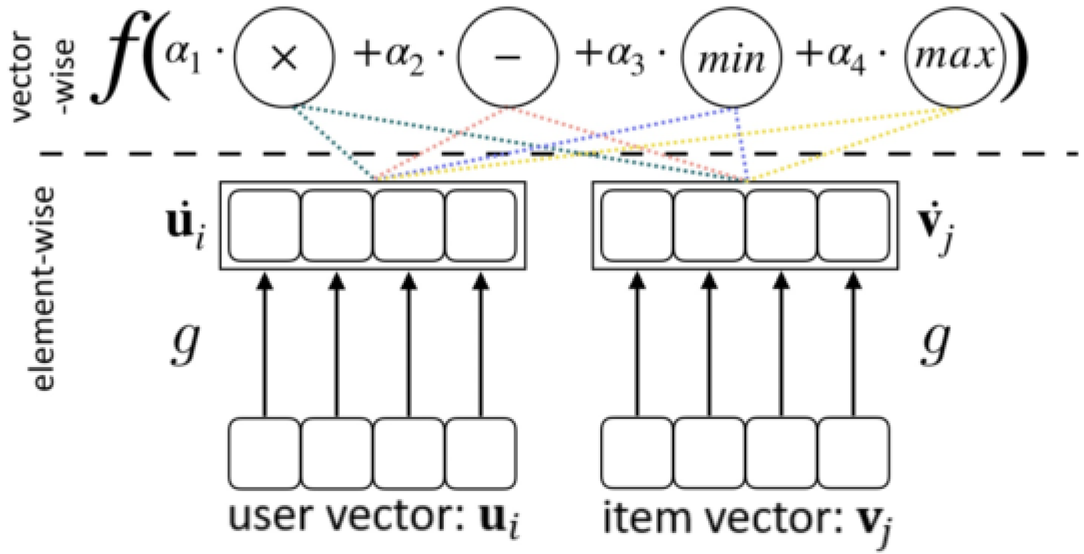
	IFC	operation	space	predict time	recent examples
human-designed	$\langle \boldsymbol{u}_i, \boldsymbol{v}_j \rangle$	inner product	$O((m+n)k)$	$O(k)$	MF [28], FM [37]
	$\boldsymbol{u}_i - \boldsymbol{v}_j$	plus (minus)	$O((m+n)k)$	$O(k)$	CML [19]
	$\max(\boldsymbol{u}_i, \boldsymbol{v}_j)$	max, min	$O((m+n)k)$	$O(k)$	ConvMF [25]
	$\sigma([\boldsymbol{u}_i; \boldsymbol{v}_j])$	concat	$O((m+n)k)$	$O(k)$	Deep&Wide [9]
	$\sigma(\boldsymbol{u}_i \odot \boldsymbol{v}_j + \boldsymbol{H}[\boldsymbol{u}_i; \boldsymbol{v}_j])$	multi, concat	$O((m+n)k)$	$O(k^2)$	NCF [17]
	$\boldsymbol{u}_i * \boldsymbol{v}_j$	conv	$O((m+n)k)$	$O(k \log(k))$	ConvMF [25]
	$\boldsymbol{u}_i \otimes \boldsymbol{v}_j$	outer product	$O((m+n)k)$	$O(k^2)$	ConvNCF [16]

Is there an absolute best IFC? : NO, depends on tasks and datasets [1]

SIF (Yao et al, WWW2020)

IFC	operation
$\langle \boldsymbol{u}_i, \boldsymbol{v}_j \rangle$	inner product
$\boldsymbol{u}_i - \boldsymbol{v}_j$	plus (minus)
$\max(\boldsymbol{u}_i, \boldsymbol{v}_j)$	max, min
$\sigma([\boldsymbol{u}_i; \boldsymbol{v}_j])$	concat
$\sigma(\boldsymbol{u}_i \odot \boldsymbol{v}_j + \boldsymbol{H}[\boldsymbol{u}_i; \boldsymbol{v}_j])$	multi, concat
$\boldsymbol{u}_i * \boldsymbol{v}_j$	conv
$\boldsymbol{u}_i \otimes \boldsymbol{v}_j$	outer product

Cut the search space into two blocks

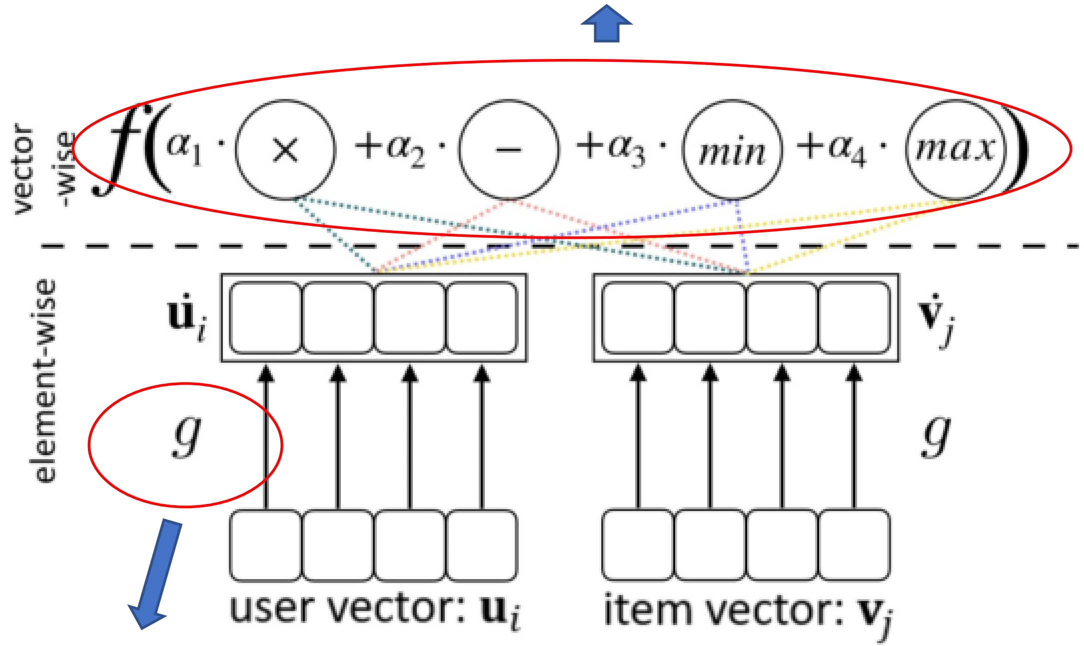


- Vector-level: simple linear algebra operations
- Elementwise: shared nonlinear transformation

Learning from Existing IFCs!

SIF (Yao et al, WWW2020)

Can be seen as choices on operations



Implement using a small MLP

A Supernet Representation

S: architecture hyper-parameters

T: parameters

$$
\begin{aligned}
\min_S \quad & H(S,T) \equiv \sum\nolimits_{(i,j)\in\bar{\Omega}} \mathcal{M}\left(h_\alpha(\boldsymbol{u}_i^*,\boldsymbol{v}_j^*)^\top \boldsymbol{w}_\alpha^*, O_{ij}\right) \qquad (9)\\
\text{s.t.} \quad & \boldsymbol{\alpha} \in \mathcal{C} \text{ and } \boldsymbol{T}^* \equiv \{\boldsymbol{U}^*, \boldsymbol{V}^*, \{\boldsymbol{w}_m^*\}\} = \arg\min_{\boldsymbol{T}} F_\alpha(\boldsymbol{T}; \boldsymbol{S}),
\end{aligned}
$$

High level

where F_α is the training objective:

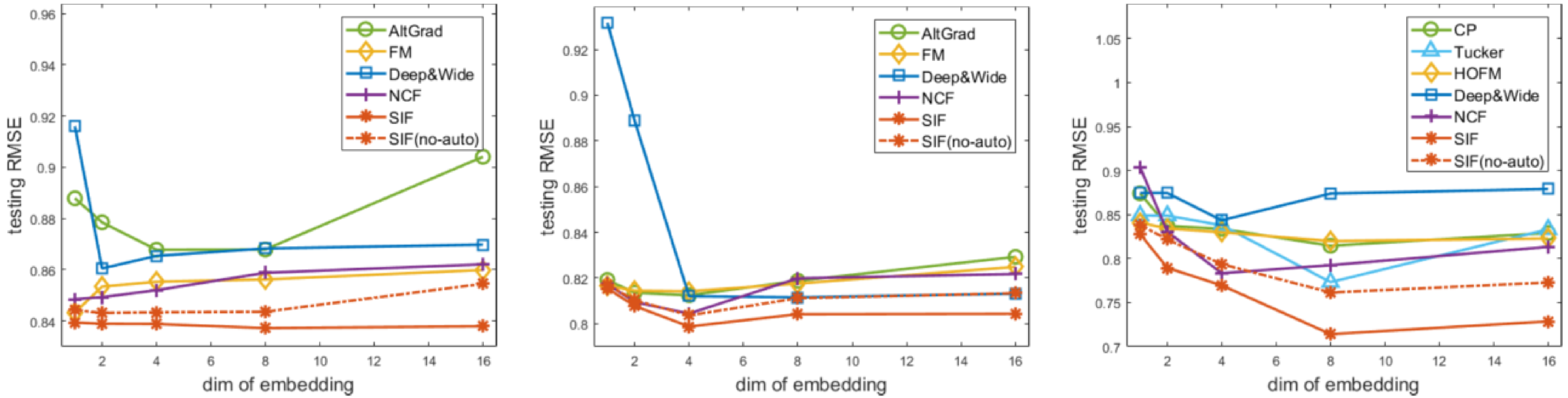
$$
\begin{aligned}
F_\alpha(\boldsymbol{T};\boldsymbol{S}) \equiv & \sum\nolimits_{(i,j)\in\Omega} \ell\left(h_\alpha(\boldsymbol{u}_i,\boldsymbol{v}_j), O_{ij}\right) + \frac{\lambda}{2}\|\boldsymbol{U}\|_F^2 + \frac{\lambda}{2}\|\boldsymbol{V}\|_F^2,\\
& \text{s.t. } \|\boldsymbol{w}_m\|_2 \le 1 \text{ for } m = 1,\dots,|\mathcal{O}|.
\end{aligned}
$$

Low level

- High level: optimize S
- Lew level: optimize T
- Bilevel programming is expensive to solve - T^* needs to be obtained from model training

Comparison with CF Approaches

(i) Alternating gradient descent (“***AltGrad***”); (ii) Factorization machine (“***FM***”); (iii) ***Deep&Wide***; (iv) Neural collaborative filtering (“***NCF***”); (v) ***SIF***; and (iv) ***SIF(no-auto)***, architecture is optimized with training data

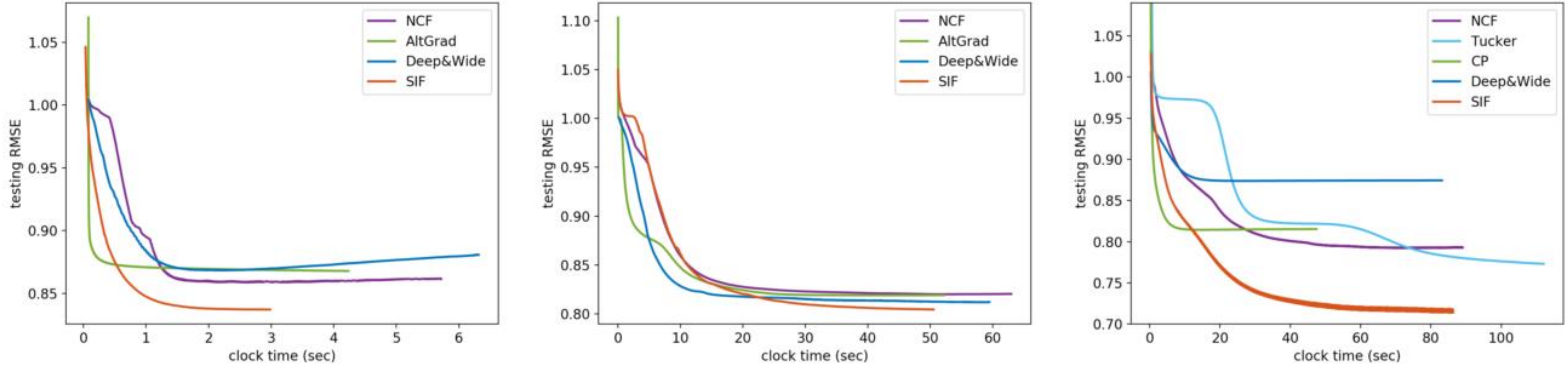


(a) MovieLens-100K. (b) MovieLens-1M. (c) Youtube.

Figure 2: Comparison of testing RMSEs between *SIF* and other CF approaches with different embedding dimension.

SIF is the best, and validation set helps architecture search

Comparison with CF Approaches



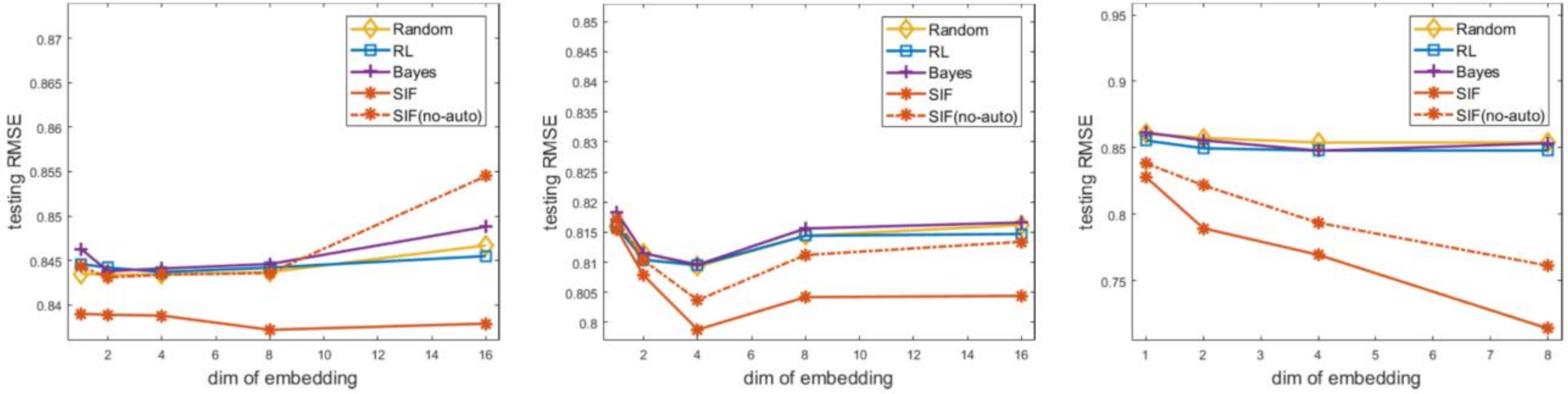
(a) MovieLens-100K. (b) MovieLens-1M. (c) Youtube.

Figure 3: Comparison of the convergence between *SIF* (with searched IFC) and other CF methods when embedded dimension is 8. *FM* and *HOFM* are not shown as their code donot support a callback to record testing performance.

Interaction function obtained from SIF can be trained as fast as state-of-the-art

Comparison with AutoML Approaches

(i) “Random”; (ii) “RL”: reinforcement learning; (iii) “Bayes”: HyperOpt



(a) MovieLens-100K. (b) MovieLens-1M. (c) Youtube.

Figure 4: Comparison of testing RMSEs between *SIF* and other AutoML approaches with different embedding dimensions. *Gen-approx* is slow with bad performance, thus is not run on Youtube.

SIF can find better architecture than other AutoML search algorithms

Comparison with AutoML Approaches

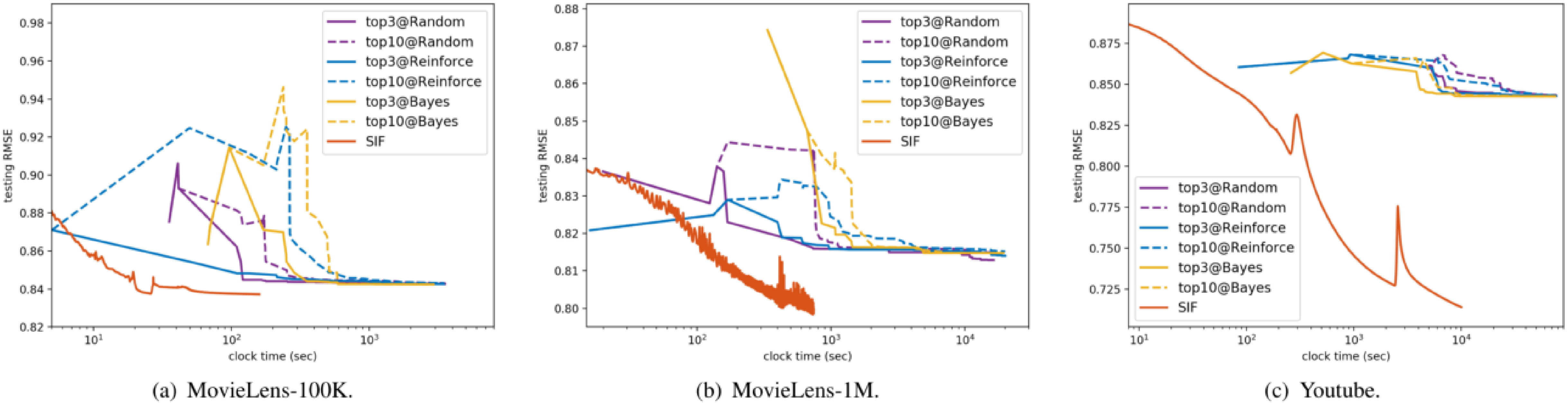


Figure 5: Comparison of search efficiency among *SIF* and other AutoML approaches when embedded dimension is 8.

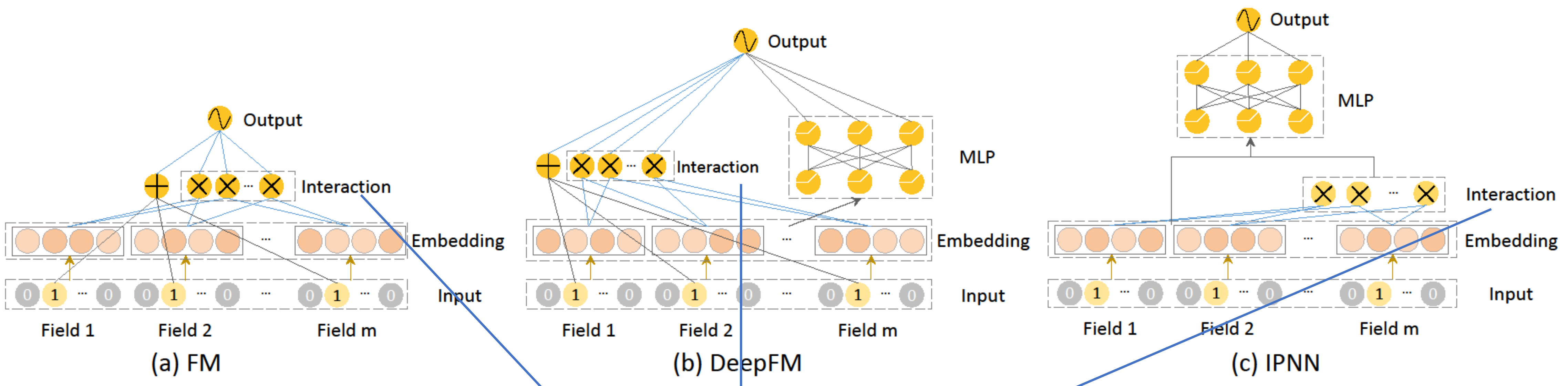
SIF is much faster than other AutoML search algorithms

AutoML for recommender system

- Automated interaction function search (Yao et al, WWW2020)
  - Design interaction function automatically
  - Cover both existing and new interaction functions
  - One-shot search, update interaction function and embedding jointly
  - Better performance than experts with slightly higher computation cost

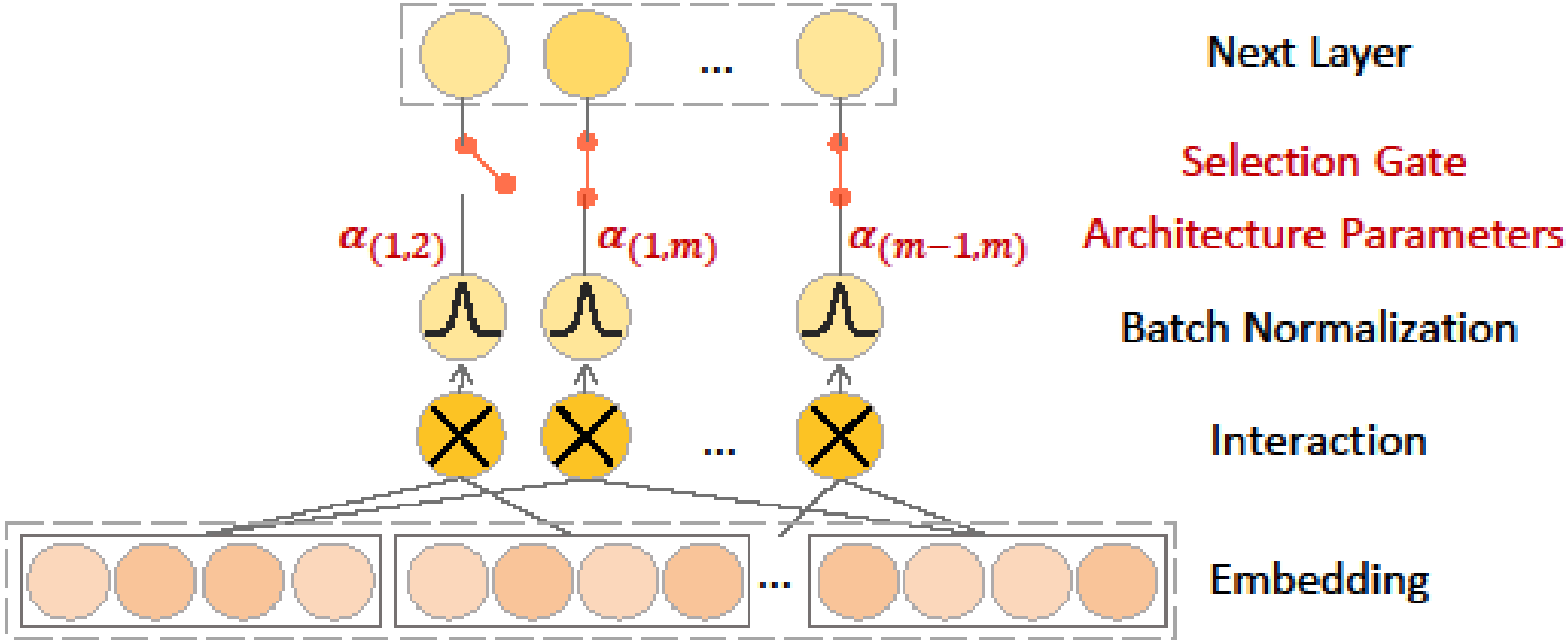
- **Automated feature interaction search (Liu et al, KDD2020)**
  - **Automatically select important low and high order feature interactions**
  - **Two stages to search feature interaction and re-train the model**
  - **Better performance on both online and offline evaluations without much computation cost**

Factorization Models



Feature interactions

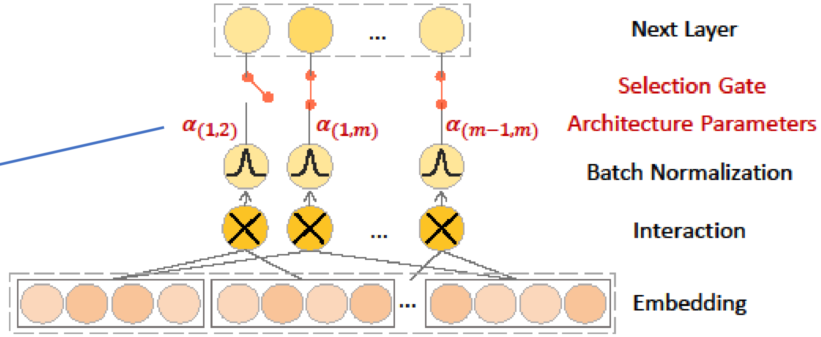
AutoFIS (Liu et al, KDD2020)



Automatically select important feature interactions!

AutoFIS (Liu et al, KDD2020)

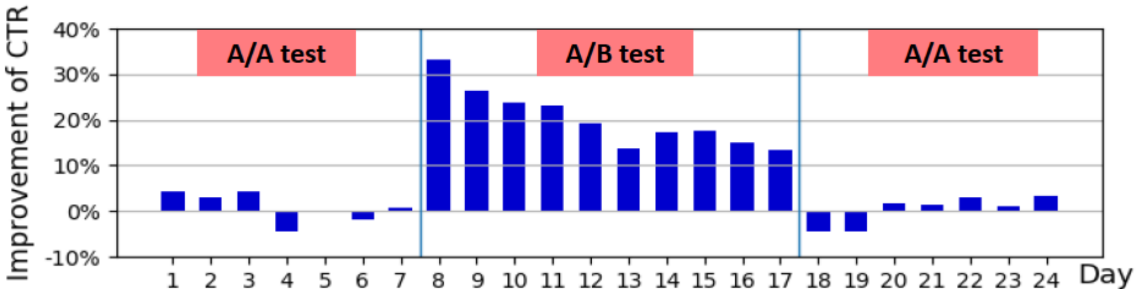
- Stage 1: Search
  - Gate: whether to select a feature interaction
  - Relax discrete choices to continuous α
  - GRDA optimizer to get sparse network

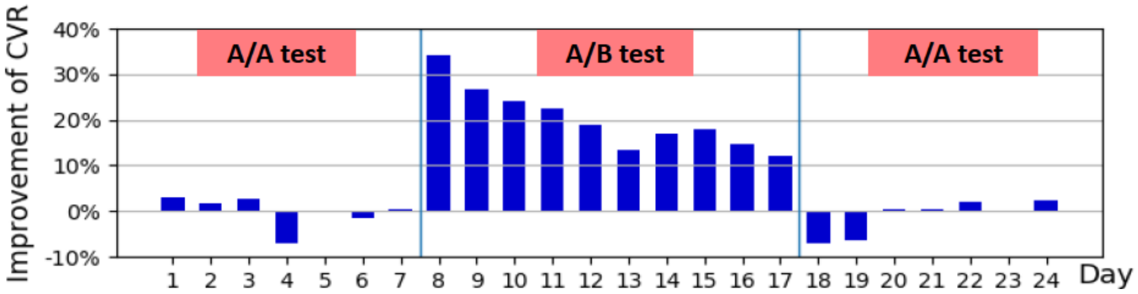


- Stage 2: Re-train
  - Remove unimportant feature interactions
  - Re-train the new model with α as attention units

AutoFIS (Liu et al, KDD2020)

Model	Avazu						Criteo					
	AUC	log loss	top	time (s)	search + re-train cost (min)	Rel. Impr.	AUC	log loss	top	time (s)	search + re-train cost (min)	Rel. Impr.
FM	0.7793	0.3805	100%	0.51	0 + 3	0	0.7909	0.5500	100%	0.74	0 + 11	0
FwFM	0.7822	0.3784	100%	0.52	0 + 4	0.37%	0.7948	0.5475	100%	0.76	0 + 12	0.49%
AFM	0.7806	0.3794	100%	1.92	0 + 14	0.17%	0.7913	0.5517	100%	1.43	0 + 20	0.05%
FFM	0.7831	0.3781	100%	0.24	0 + 6	0.49%	0.7980	0.5438	100%	0.49	0 + 39	0.90%
DeepFM	0.7836	0.3776	100%	0.76	0 + 6	0.55%	0.7991	0.5423	100%	1.17	0 + 16	1.04%
GBDT+LR	0.7721	0.3841	100%	0.45	8 + 3	-0.92%	0.7871	0.5556	100%	0.62	40 + 10	-0.48%
GBDT+FFM	0.7835	0.3777	100%	2.66	6 + 21	0.54%	0.7988	0.5430	100%	1.68	9 + 57	1.00%
AutoFM(2nd)	0.7831*	0.3778*	29%	**0.23**	4 + 2	0.49%	0.7974*	0.5446*	51%	**0.48**	14 + 9	0.82%
AutoDeepFM(2nd)	**0.7852***	**0.3765***	**24%**	0.48	7 + 4	0.76%	**0.8009***	**0.5404***	**28%**	0.69	22 + 11	1.26%
FM(3rd)	0.7843	0.3772	100%	5.70	0 + 21	0.64%	0.7965	0.5457	100%	8.21	0 + 72	0.71%
DeepFM(3rd)	0.7854	0.3765	100%	5.97	0 + 23	0.78%	0.7999	0.5418	100%	13.07	0 + 125	1.14%
AutoFM(3rd)	0.7860*	0.3762*	**25% / 2%**	**0.33**	22 + 5	0.86%	0.7983*	0.5436*	35% / 1%	**0.63**	75 + 15	0.94%
AutoDeepFM(3rd)	**0.7870***	**0.3756***	21% / 10%	0.94	24 + 10	0.99%	**0.8010***	**0.5404***	**13% / 2%**	0.86	128 + 17	1.28%





Better offline and online performance.

Summary

- **Advanced techniques are incorporated into recommender systems, such as deep learning, graph neural networks and knowledge graph. Better performance is achieved.**
- **Human-crafted recommender system requires heavy manual designs or computation cost on multiple components, including feature, interaction function, model architecture and hyper-parameter.**
- **AutoML comes to help to automatically make reasonable decisions on different datasets and tasks.**

Thank You!

liyong07@Tsinghua.edu.cn

https://sites.google.com/view/kdd20-marketplace-autorecsys/