Automated Recommender System (RecSys)

Quanming Yao[1,2], Yong Li[1], Chen Gao[1], Huan Zhao[2], Yongqi Zhang[2]

[1]*Department of Electronic Engineering, Tsinghua University*

[2]*4Paradigm Inc.*

https://quanmingyao.github.io/AutoML.github.io/ijcai21-tutorial.html





Tutorial Outline

1. An introduction to Automated Machine Learning (AutoML)
   - Background on technical tools from machine learning
2. Why AutoML is Needed in RecSys and Recent Advances
   - Exemplar works introducing AutoML into RecSys
3. Automated Graph Representation Learning for RecSys
   - Explore neural architecture search for GNN based RecSys
4. Automated Knowledge Graph (KG) Embedding
   - Explore AutoML for KG Embedding based RecSys

Automated Recommender System (RecSys) Tutorial

Part 2: Why AutoML is Needed in RecSys and Recent Advances

Chen Gao

Department of Electronic Engineering, Tsinghua University

Outline of Part 2

1. What is recommender system

2. Recent advances in recommender system
   a) Neural Networks
   b) Graph Neural Networks
   c) Knowledge Graph Embedding
3. Why AutoML is needed

4. Recent works of automated RecSys

Outline of Part 2

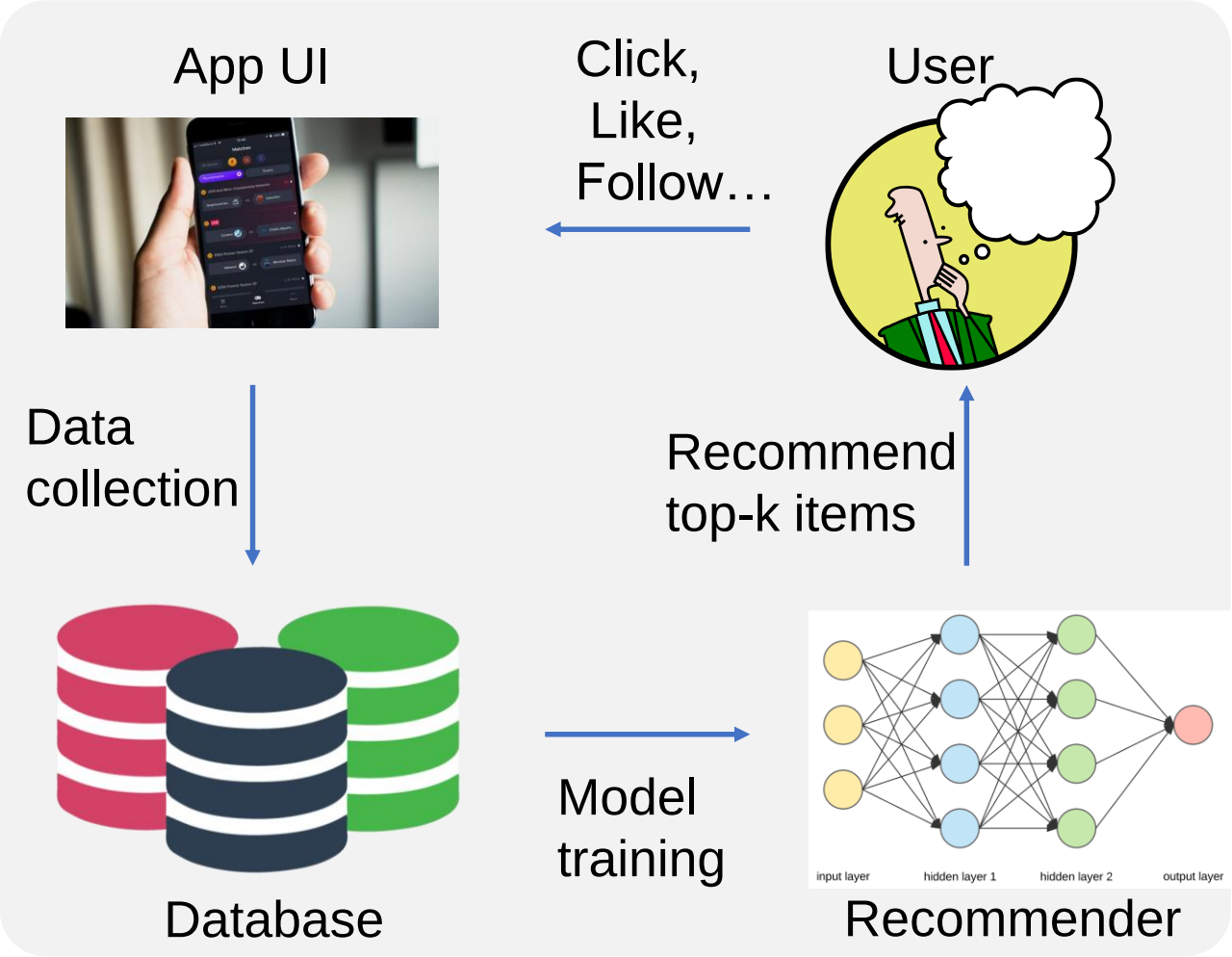
1. **What is recommender system**

2. **Recent advances in recommender system**
   a) **Neural Networks**
   b) **Graph Neural Networks**
   c) **Knowledge Graph Embedding**
3. **Why AutoML is needed**

4. **Recent works of automated RecSys**

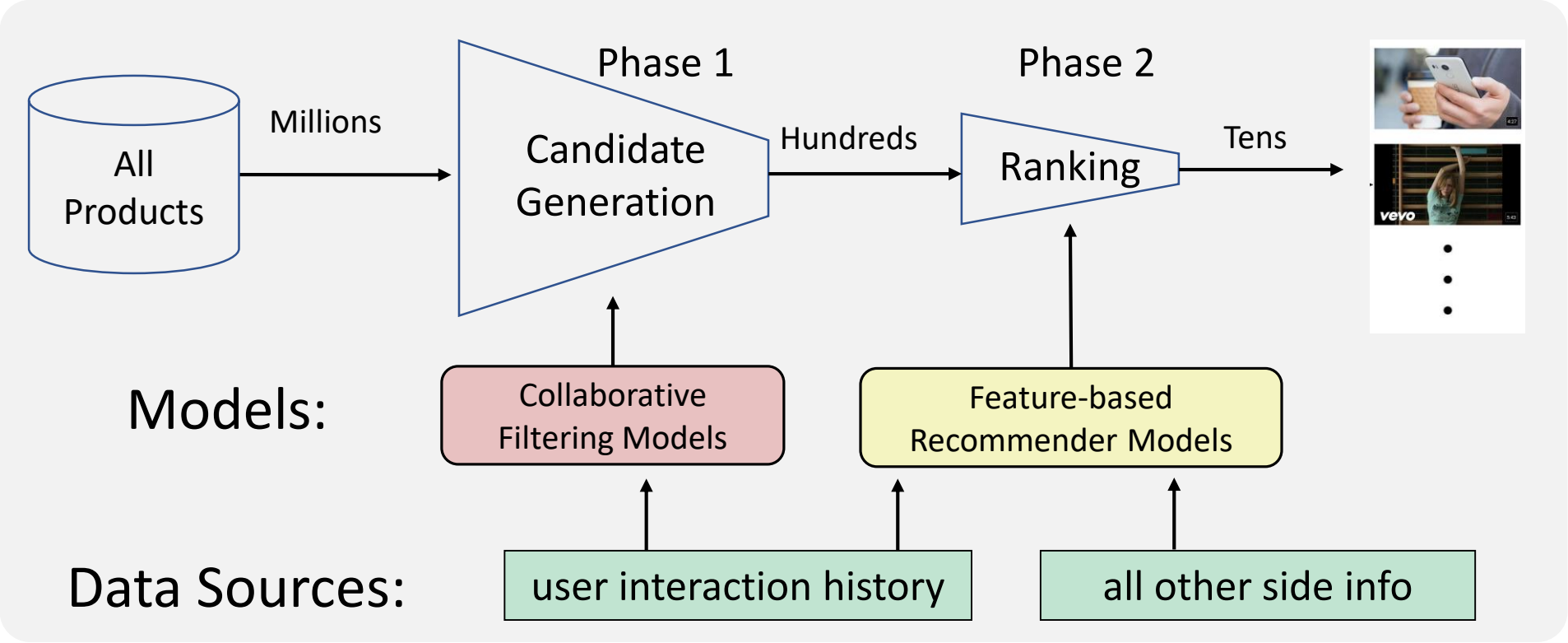
Recommender System

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



Problem Formulation of RecSys

- **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)



Outline of Part 2

1. **What is recommender system**

2. **Recent advances in recommender system**
   a) **Neural Networks**
   b) **Graph Neural Networks**
   c) **Knowledge Graph Embedding**
3. **Why AutoML is needed**

4. **Recent works of automated RecSys**

Deep Neural Networks for Recommendation

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

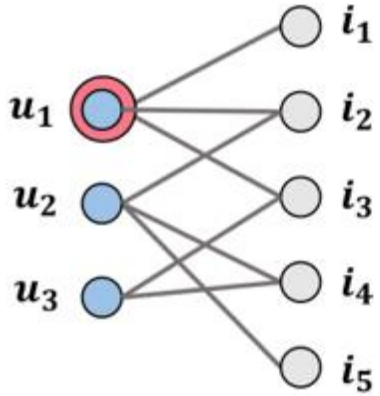
- Deep Neural Networks shows great power in modeling high-order similarity in recommender system, e.g. multi-behavior and social relationship in recommendation…

Collaborative Filtering (CF)

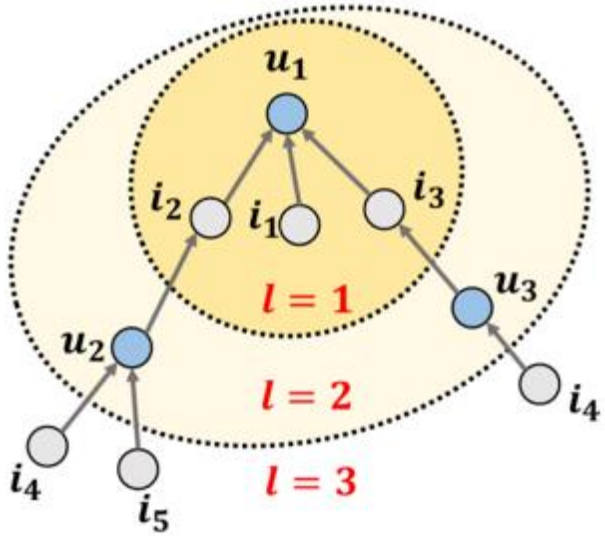
- Revisit CF via **high-order connectivity**
  - The paths that reach u_1 from any node with the path length l larger than 1
  - A natural way to encode collaborative signal is to encode it by the **interaction graph structure**

Why u_1 may like i_4?

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



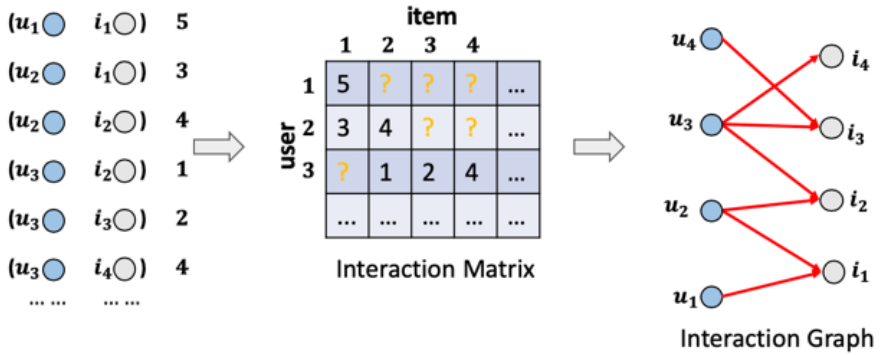
User-Item Interaction Graph

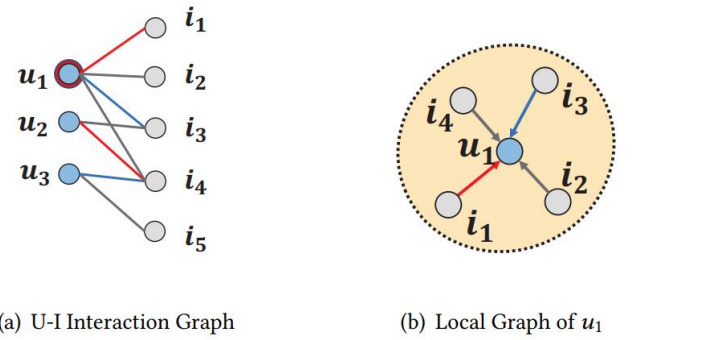


High-order Connectivity for u_1

GNN for CF

- Encode high-order connectivity via GNN → collaborative signals
- Enhance representation learning of users and items → combine user-item bipartite with other features (such as behavioral type, etc.)





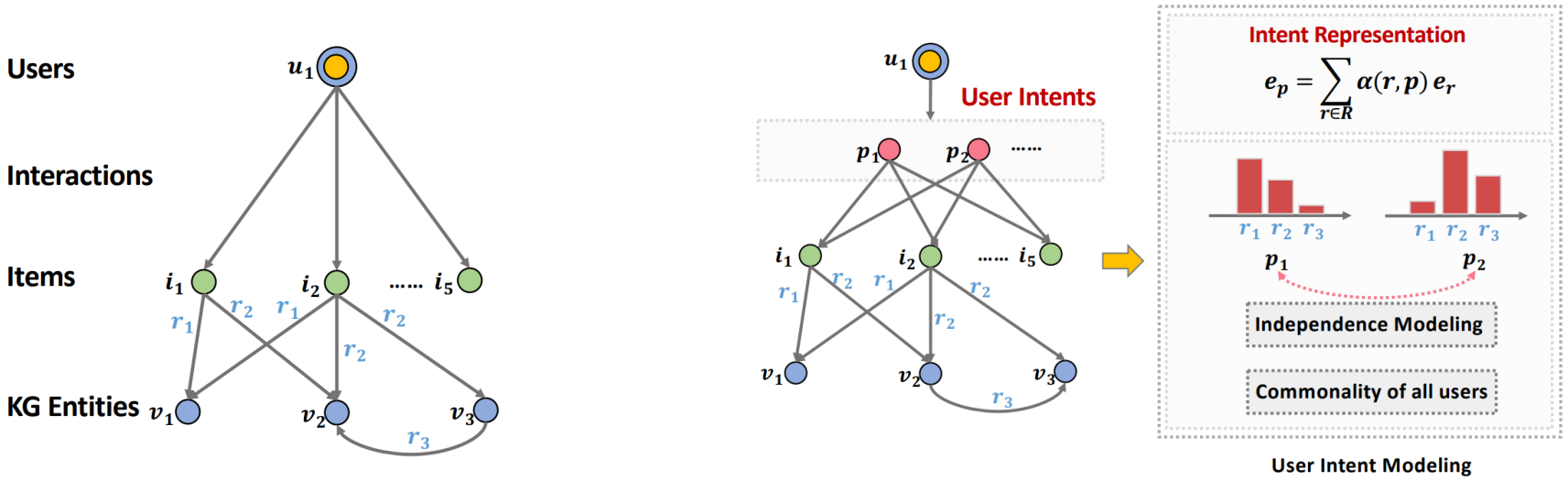
(a) U-I Interaction Graph (b) Local Graph of u_1

Figure 1: An illustration of the user-item multi-behavior graph, where node u_1 is the target user to provide recommendations for.

Jin, B., Gao, C., He, X., Jin, D., & Li, Y. Multi-behavior recommendation with graph convolutional networks. In SIGIR 2020.

KGE for RecSys

- Enrich item by structured data
- Fusing user-item embedding with KG emebdding



Wang, X., Huang, T., Wang, D., Yuan, Y., Liu, Z., He, X., & Chua, T. S. (2021, April). Learning Intents behind Interactions with Knowledge Graph for Recommendation. In *TheWebConf 2021*

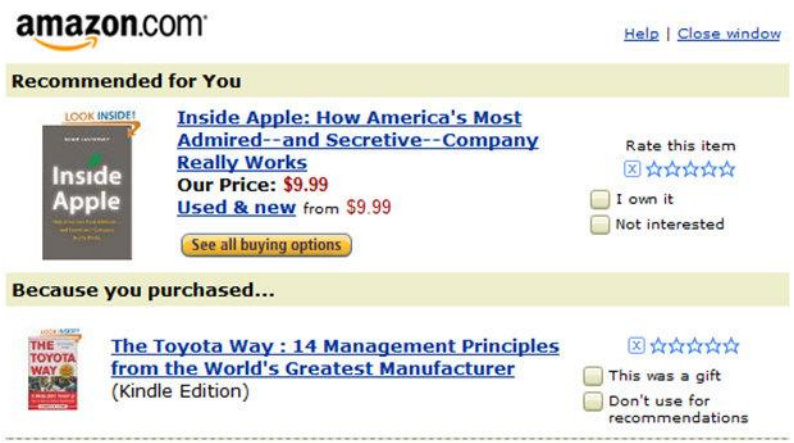
Outline of Part 2

1. **What is recommender system**

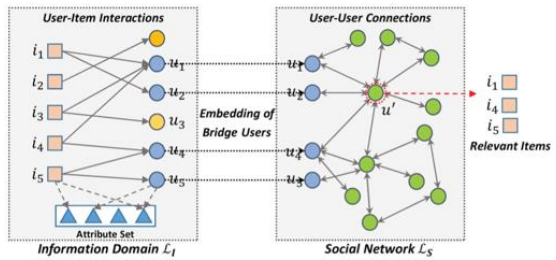
2. **Recent advances in recommender system**
   a) **Neural Networks**
   b) **Graph Neural Networks**
   c) **Knowledge Graph Embedding**
3. **Why AutoML is needed**

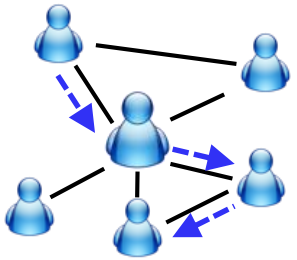
4. **Recent works of automated RecSys**

Evolution of recommendation tasks

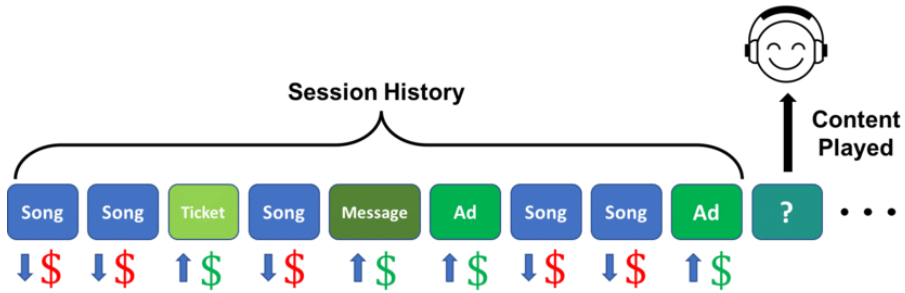


Product rating prediction





Social recommendation

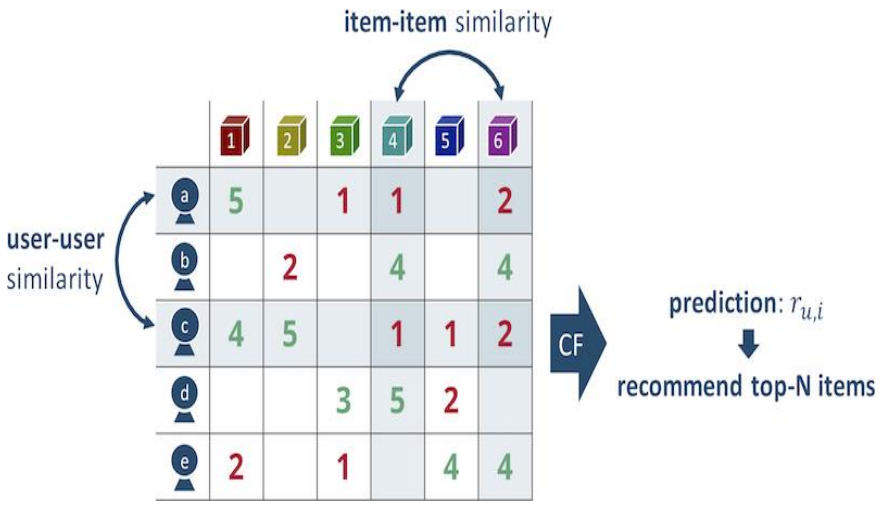


Sequential recommendation

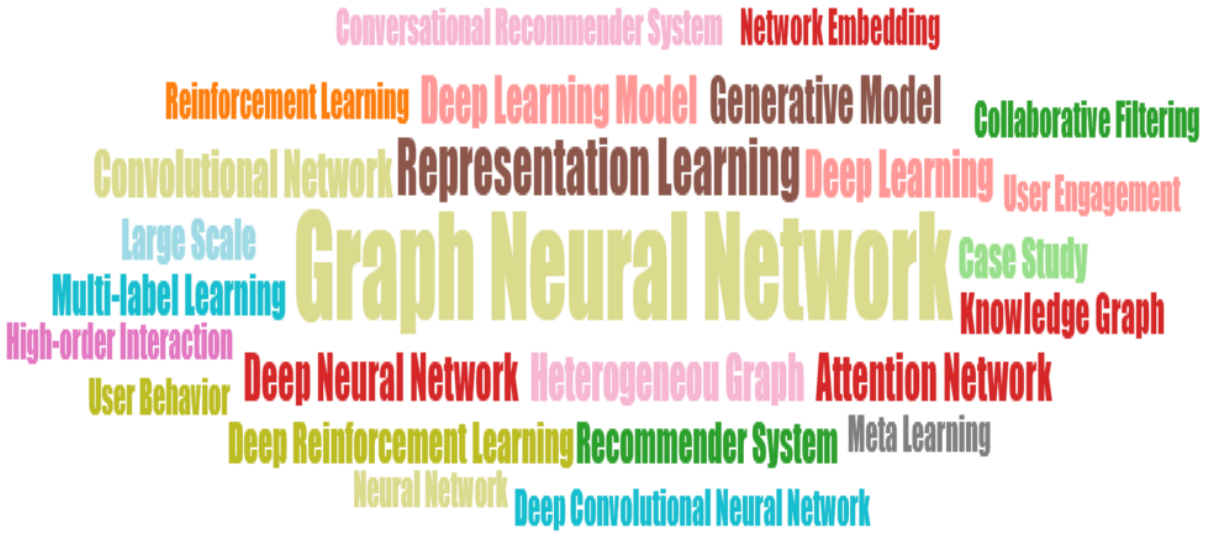
Bundle recommendation

Recommendation tasks are getting more diverse!

Evolution of recommendation models



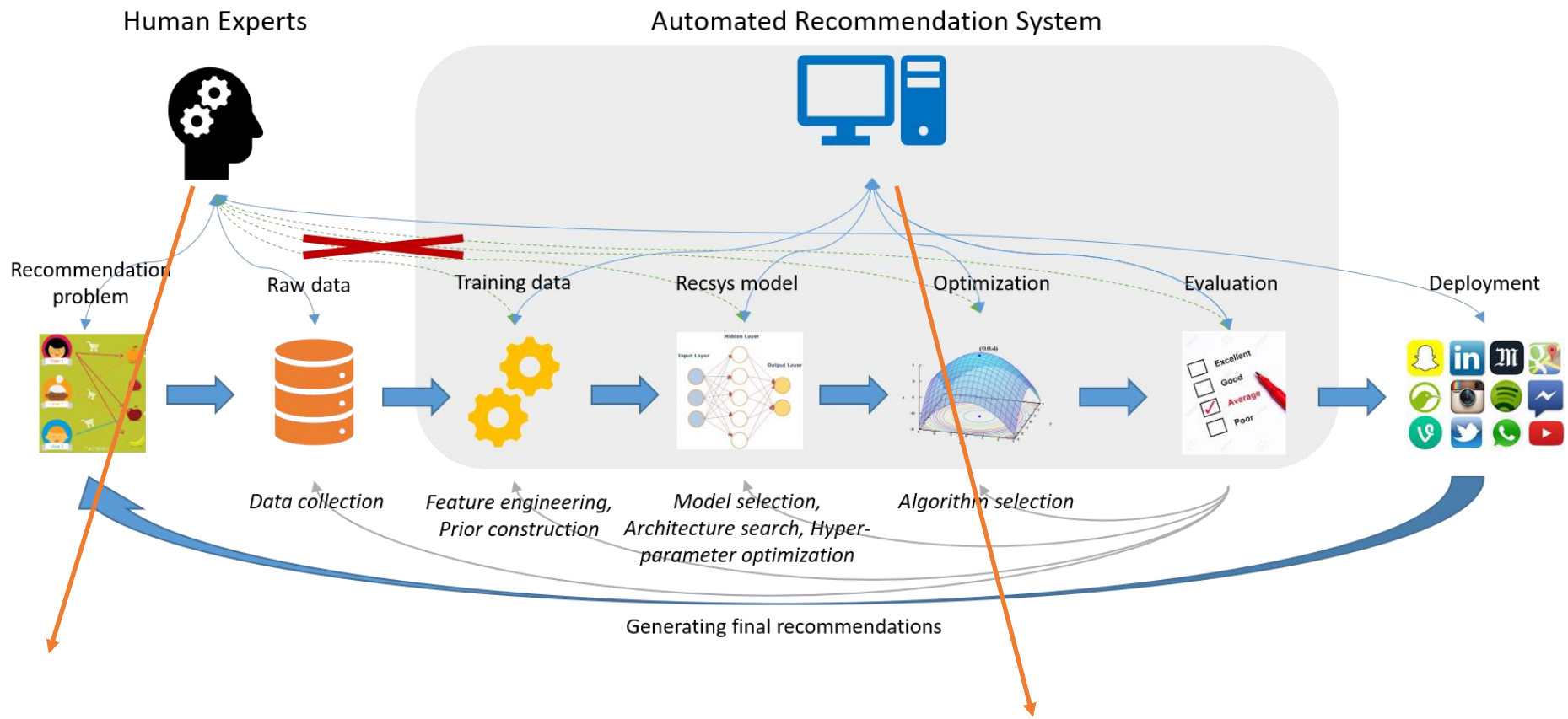
Simple KNN



DNN, GNN, KG, Attention,…

Recommendation models are getting more complicated!

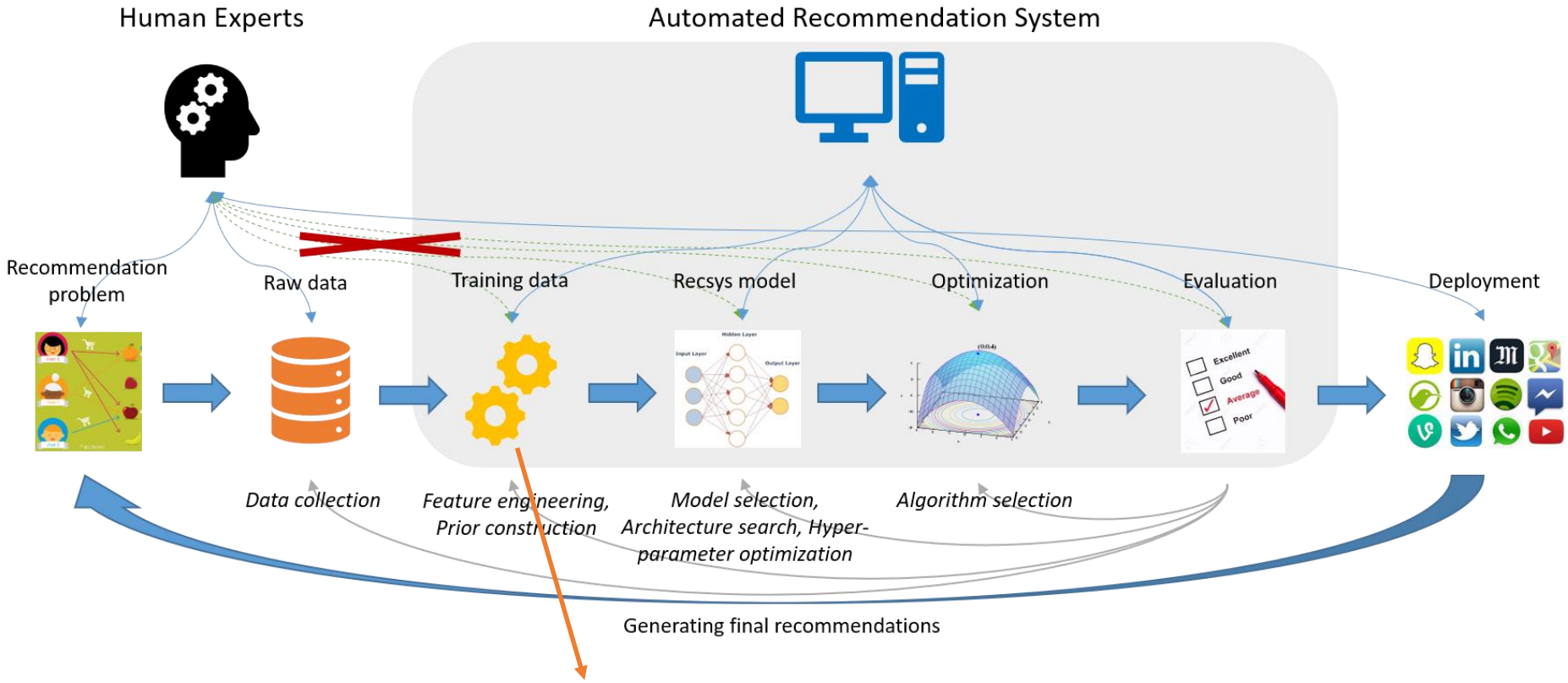
AutoML for Recommendation



Massive human efforts

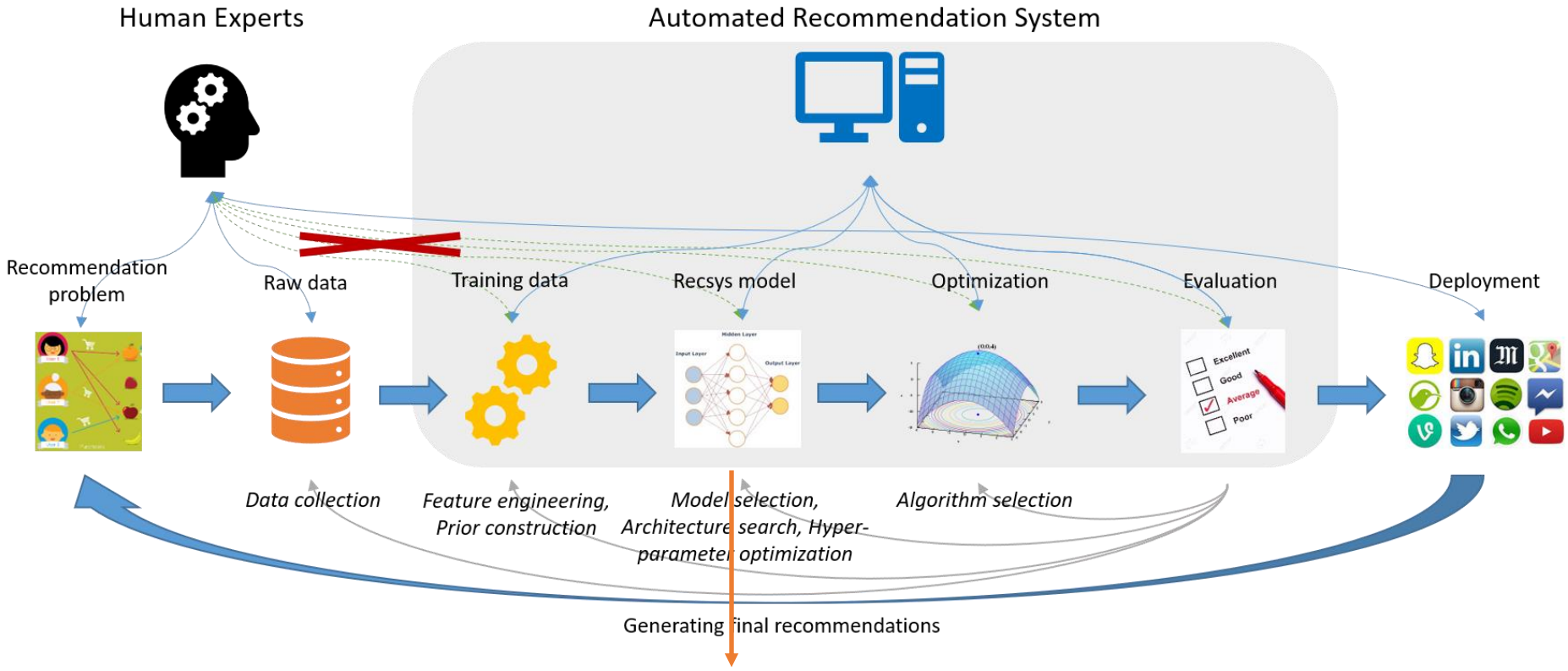
Automatically build recommendation models

What to be automated ?



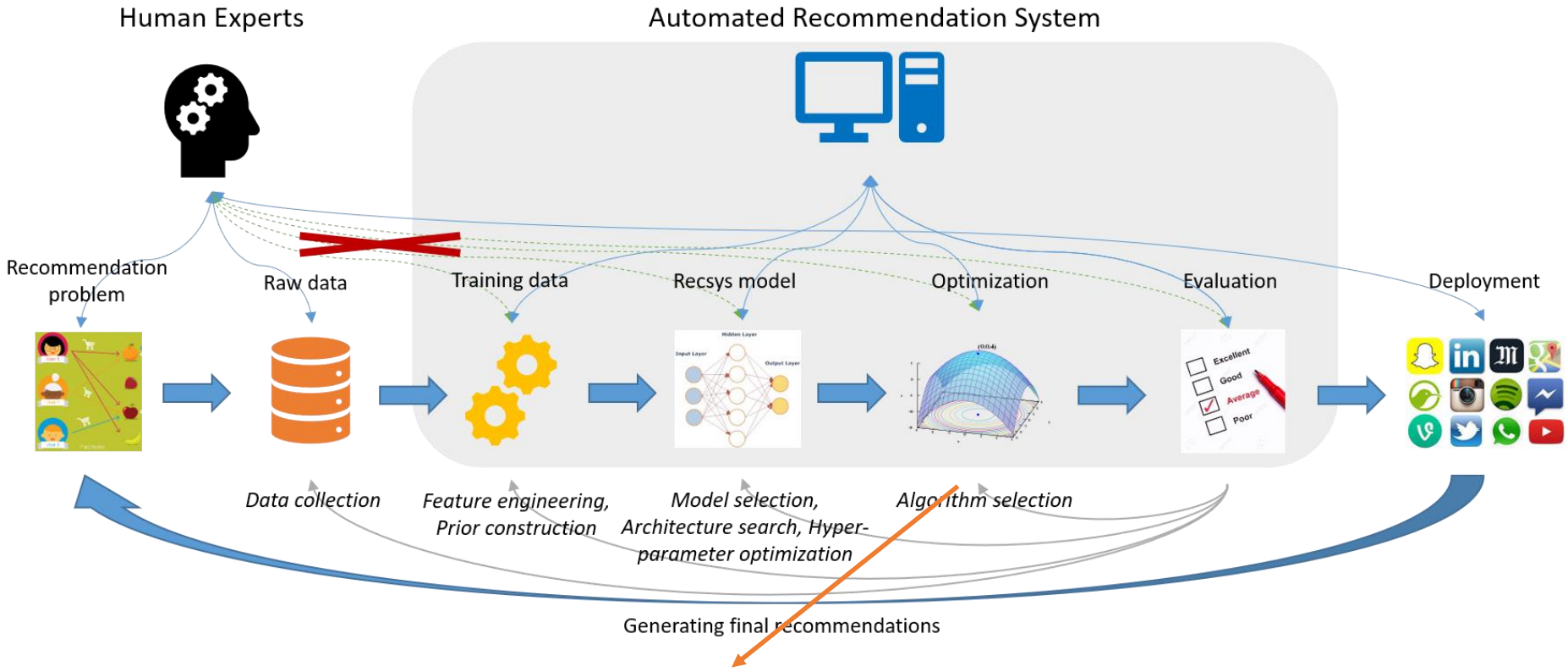
Feature Engineering is a tedious and task-specific work. AutoML helps to generate informative and discriminative features

What to be automated ?



Model Design/Hyperparameter Tuning requires heavy human and computation cost. AutoML helps to identify better architectures than handcrafted ones.

What to be automated ?



Algorithm Selection. Optimization algorithms greatly influence model performance. AutoML helps to set the right configurations for the optimization algorithm.

Human-crafted recommender system

- Most importantly, there is no silver bullet 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!

How to use AutoML

Expert in RecSys

1. Define an AutoML problem

- Derive a search space from insights in specific domains
- Search objective is usually validation performance
- Search constraint is usually resource budgets
- Training objective usually comes from classical learning models

Search Space → $\min_{\lambda \in \mathcal{S}} M(F(w^*; \lambda), D_{\text{val}})$ ← Search Objective

s. t. $\min_{w} L(F(w; \lambda), D_{\text{tra}})$ ← Training Objective

$G(\lambda) \leq C$ ← Search Constraints

2. Design or select proper search algorithm

- Reduce model training cost (time to get w^*)

Outline of Part 2

1. **What is recommender system**

2. **Recent advances in recommender system**
   a) **Neural Networks**
   b) **Graph Neural Networks**
   c) **Knowledge Graph Embedding**
3. **Why AutoML is needed**

4. **Recent works of automated RecSys**

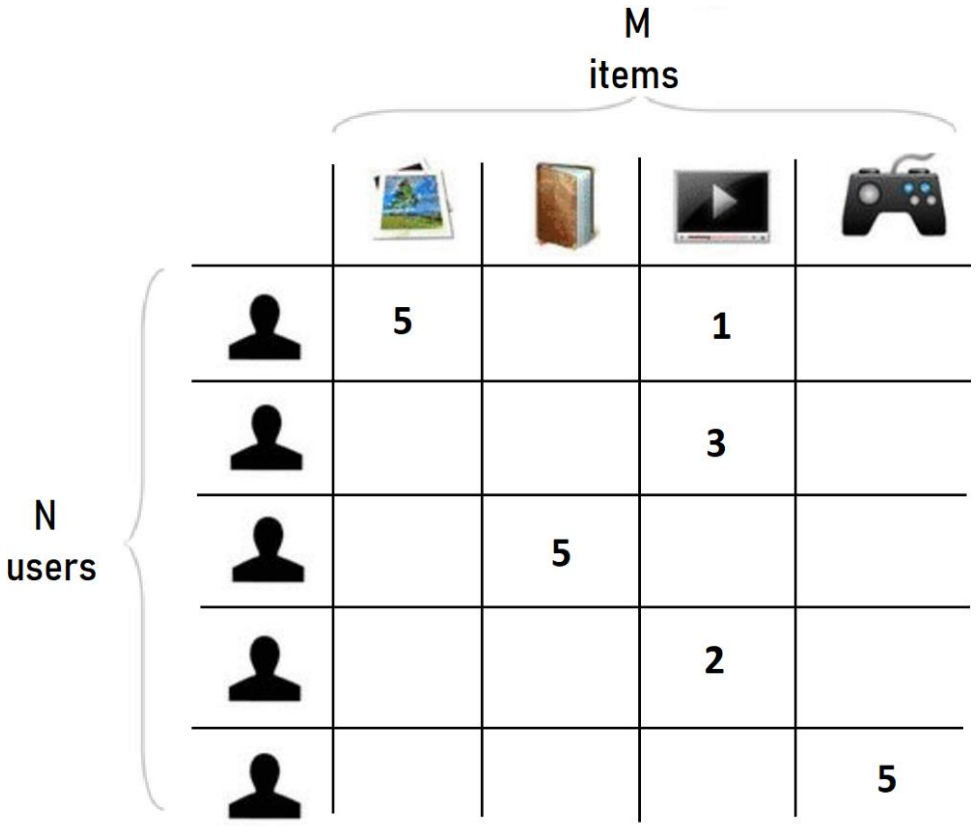
Recent Advances in Automated RecSys

- AutoML for Collaborative Filtering Task
- AutoML for Click-through Rate Prediction Task
- AutoML for Tuning Hyper-parameters in RecSys

Outline

- AutoML for Collaborative Filtering Task
- AutoML for Click-through Rate Prediction Task
- AutoML for Tuning Hyper-parameters in RecSys

Collaborative Filtering – Problem Setup



Study the fundamental CF problem [1]

- Data: a rating matrix with many unknown positions
- Task: estimate ratings on unknown positions
- Measurement: RMSE on estimated ratings.

[1]. Exact Matrix Completion via Convex Optimization. Foundations of Computational Mathematics. 2008

Image is from: https://mc.ai/deep-learning-for-collaborative-filtering-using-fastai/

Interaction Function in CF

	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]

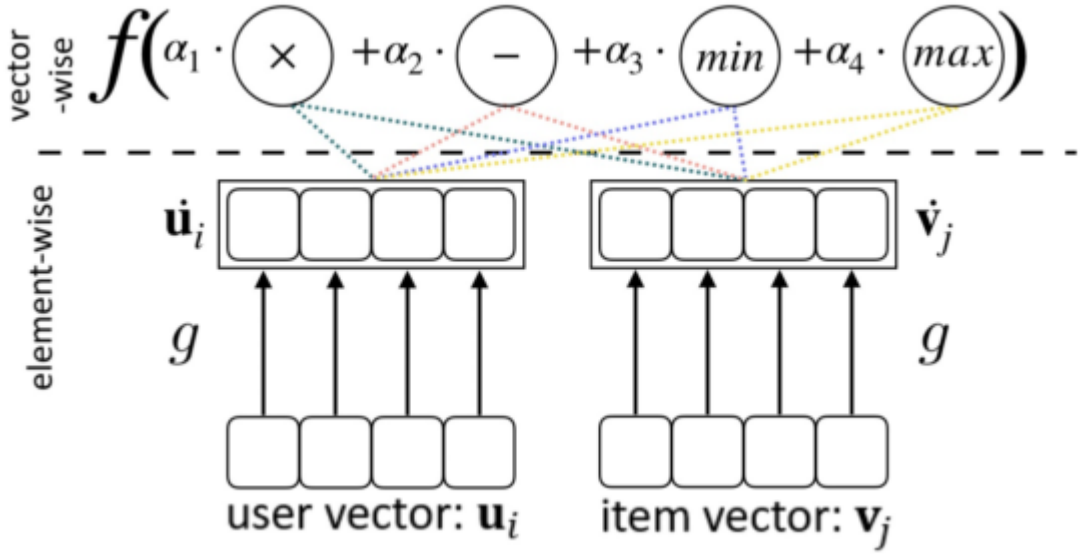
Is there an absolute best IFC? : NO, depends on tasks and datasets [1]

[1]. Are we really making much progress? A worrying analysis of recent neural recommendation approaches. Recsys 2019

Interaction Function Search Space

IFC	operation
$\langle u_i, v_j \rangle$	inner product
$u_i - v_j$	plus (minus)
$\max(u_i, v_j)$	max, min
$\sigma([u_i; v_j])$	concat
$\sigma(u_i \odot v_j + H[u_i; v_j])$	multi, concat
$u_i * v_j$	conv
$u_i \otimes v_j$	outer product

Cut the search space into two blocks



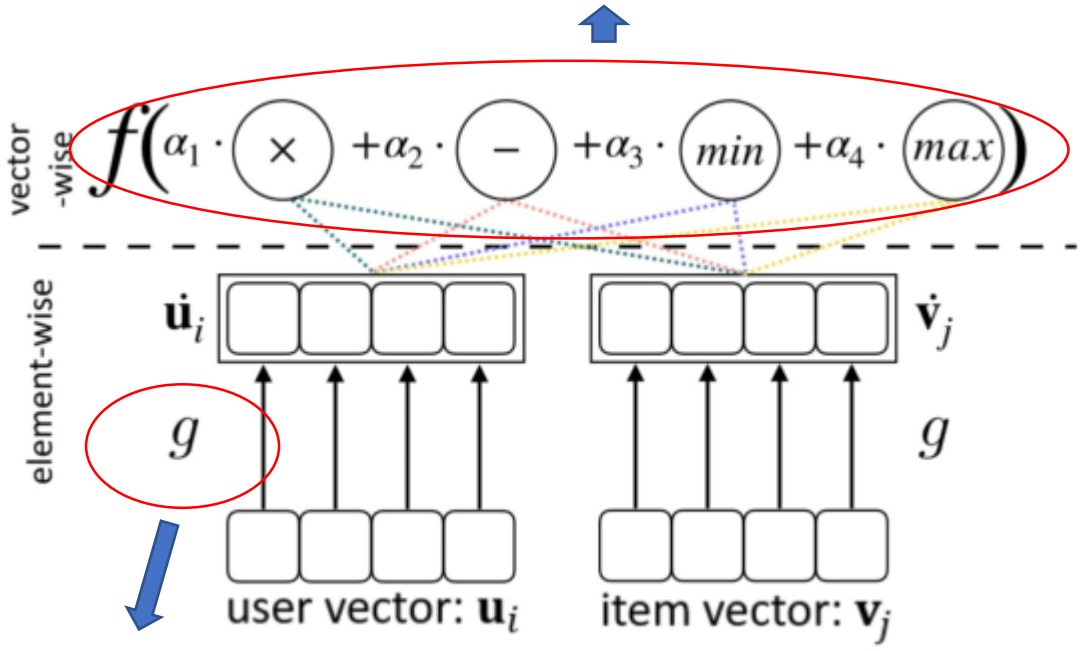
- Vector-level: simple linear algebra operations
- Elementwise: shared nonlinear transformation

Learning from Existing IFCs!

Q. Yao, X. Chen, J. Kwok, Y. Li, C.-J. Hsieh. Efficient Neural Interaction Functions Search for Collaborative Filtering, WWW 2020.

Search Problem

Can be seen as choices on operations



Implement using a small MLP

A Supernet Representation

S: architecture hyper-parameters

T: parameters

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

High level

where F_α is the training objective:

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

Low level

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

Q. Yao, X. Chen, J. Kwok, Y. Li, C.-J. Hsieh. Efficient Neural Interaction Functions Search for Collaborative Filtering, WWW 2020.

General AutoCF Problem Setup

users \ items				
	1	0	0	1
	0	1	0	0
	1	1	0	0
	1	0	0	1

0/1 Interaction matrix

OR

users \ items				
	5	-	-	3
	-	2	-	-
	4	1	-	-
	3	-	-	3

Rating matrix

CF is the most fundamental task in recommender system

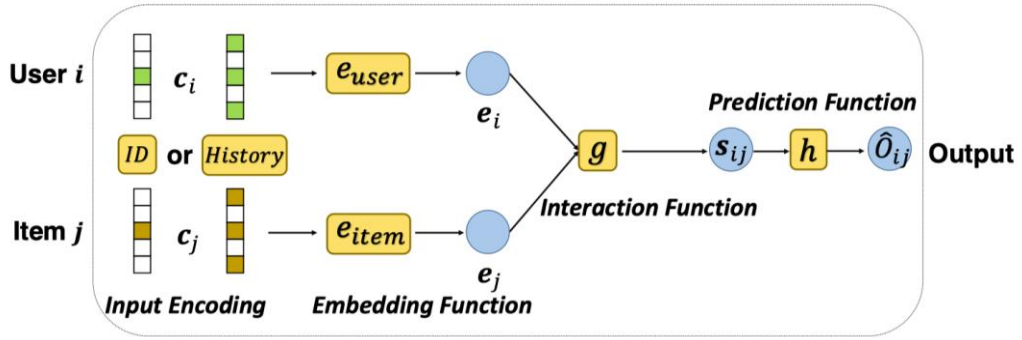
- Implicit CF
- Scenario: e-commerce, ads, etc.
- Data: an interaction matrix (1: observed interaction)
- Task: estimate positive value of unobserved ones
- Measurement: Ranking metrics

- Explicit CF
- Scenario: movie, POI, etc.
- Data: a rating matrix (e.g. 1-5)
- Task: estimate ratings on unknown positions
- Measurement: Regression metrics

Gao, C., Yao, Q., Jin, D., & Li, Y. Efficient Data-specific Model Search for Collaborative Filtering. In KDD 2021

General AutoCF Problem's Search Space

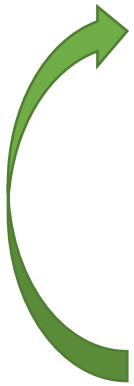
- Representative human-designed CF models under this paradigm.
  - For example, when faced with data sparse issue, history as encoding and less-parameter predictor operation tend to be a better solution



Category	Name	Model Formulation	User Enc. c_i	Emb. e_{user}	Item Enc. c_j	Emb. e_{item}	Interacton g	Prediction h
Single	MF	$\mathbf{u}_i \cdot \mathbf{v}_j$.	ID	MAT	ID	MAT	multiply	SUM
	FISM [14]	$\text{MAT}(\mathbf{r}_i) \cdot \mathbf{v}_j$	History	ID	MAT	MAT	multiply	SUM
	GMF [10]	$\text{VEC}(\mathbf{u}_i \odot \mathbf{v}_j)$	ID	MAT	ID	MAT	multiply	VEC
	MLP [10]	$\text{MLP}(\mathbf{u}_i; \mathbf{v}_j)$	ID	MAT	ID	MAT	concat	MLP
	CMF [11]	$\|\|\mathbf{u}_i - \mathbf{v}_j\|\|$	ID	MAT	ID	MAT	minus	Norm
	DMF [39]	$\text{MLP}(\mathbf{r}_i) \cdot \text{MLP}(\mathbf{r}_j)$	History	MLP	History	MLP	multiply	SUM
	JNCF-Cat [5]	$\text{MLP}(\text{MLP}(\mathbf{r}_i); \text{MLP}(\mathbf{r}_j))$	History	MLP	History	MLP	concat	MLP
	JNCF-Dot [5]	$\text{MLP}(\text{MLP}(\mathbf{r}_i) \odot \text{MLP}(\mathbf{r}_j))$	History	MLP	History	MLP	multiply	MLP
Fused	NeuMF [10]	$\text{VEC}(\mathbf{u}_i \odot \mathbf{v}_j) + \text{MLP}(\mathbf{u}_i; \mathbf{v}_j)$	ID	MAT	ID	MAT	multiply	VEC
			ID	MAT	ID	MAT	concat	MLP
	SVD++ [16]	$\mathbf{u}_i \cdot \mathbf{v}_j + \text{MAT}(\mathbf{r}_i) \cdot \mathbf{v}_j$	ID	MAT	ID	MAT	multiply	SUM
			History	MAT	ID	MAT	multiply	SUM
	DELF [6]	$\text{MLP}(\mathbf{u}_i; \mathbf{v}_j)$ $+\text{MLP}(\mathbf{u}_i; \text{MLP}(\mathbf{r}_j))$ $+\text{MLP}(\text{MLP}(\mathbf{r}_i); \mathbf{v}_j)$ $+\text{MLP}(\text{MLP}(\mathbf{r}_i); \text{MLP}(\mathbf{r}_j))$	ID	MAT	ID	MAT	concat	MLP
			ID	MAT	History	MAT	concat	MLP
			History	MAT	ID	MAT	concat	MLP
			History	MAT	History	MAT	concat	MLP

Gao, C., Yao, Q., Jin, D., & Li, Y. Efficient Data-specific Model Search for Collaborative Filtering. In KDD 2021

General AutoCF Problem and ARSP Solution



- Initialize predictor
- Sample from space
- Fast estimation with predictor
- Filtering candidates
- Evaluation with full training
- Loop until meeting criteria
- Obtain designed CF models

Algorithm 1: *OurARSP*

input : Search space $\mathcal{F}$, a learnable predictor $\mathcal{P}$, performance measurement $\mathcal{M}$, a empty set $\mathcal{H}$.

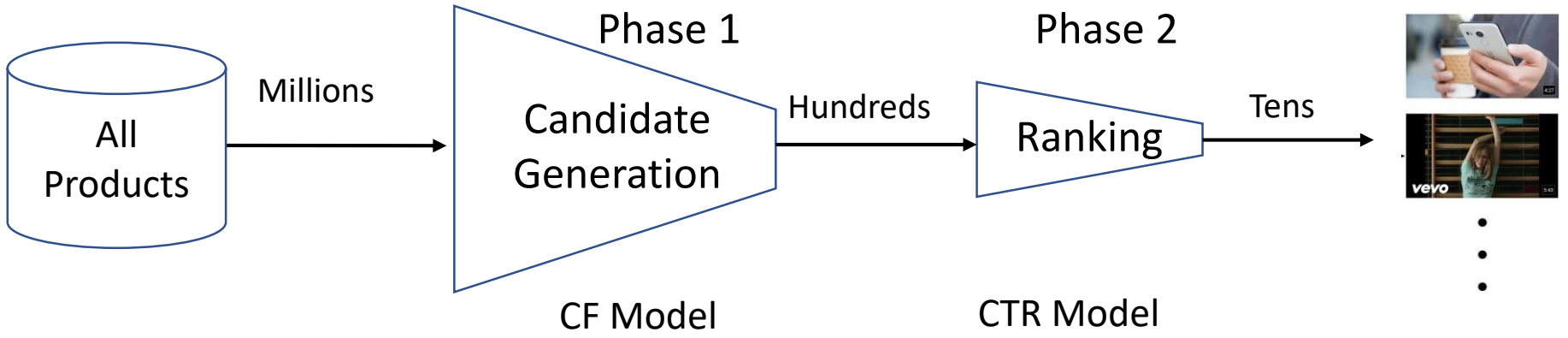
Initialize the predictor $\mathcal{P}$ with random parameters;
do
  Randomly select a $(K_1 + K_2)$-size model set $\mathcal{F}^b$ from $\mathcal{F}$;
  Generate one-hot encodings $\mathbf{x}_o$ to represent models in $\mathcal{F}^b$;
  Estimate the performance of models in $\mathcal{F}^b$ with $\mathcal{P}$;
  Choose top-K_1 model sets $\mathcal{F}^t$ to train with $\mathcal{S}$;
  Evaluate the trained models in $\mathcal{F}^t$ with $\mathcal{M}$;
  Set of evaluated-model $\mathcal{H} \leftarrow \mathcal{H} \cup \{(f, \mathcal{M}(f)) | f \in \mathcal{F}^t\}$;
  Update $\mathcal{P}$ with records in $\mathcal{H}$ via loss function in (8).
while *not meet stop criteria*;
return desired CF models in $\mathcal{H}$.

Gao, C., Yao, Q., Jin, D., & Li, Y. Efficient Data-specific Model Search for Collaborative Filtering. In KDD 2021

Outline

- AutoML for Collaborative Filtering Task
- AutoML for Click-through Rate Prediction Task
- AutoML for Tuning Hyper-parameters in RecSys

A pipeline of modern recommendation engine

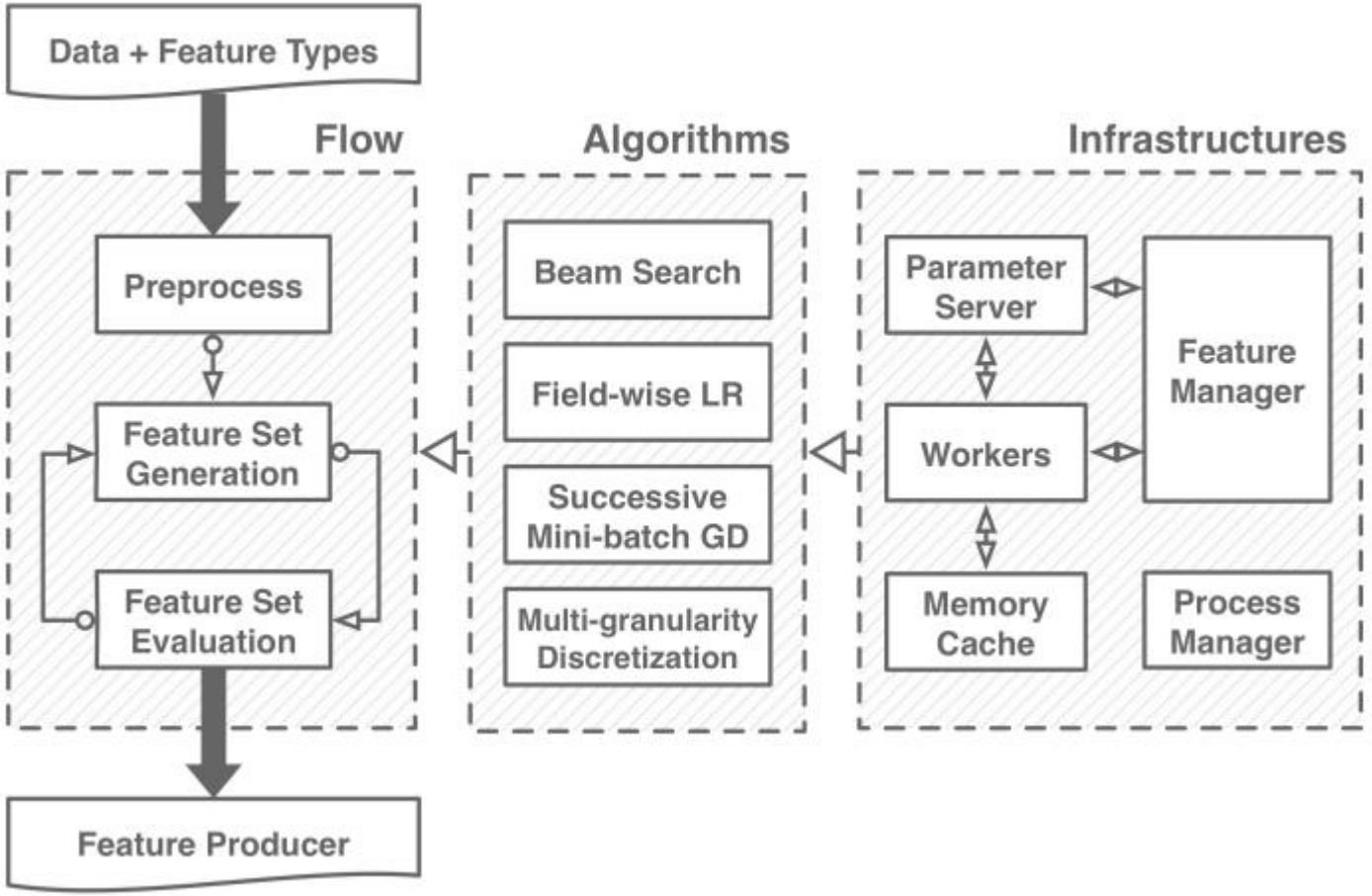


Data Input : User-item interaction/rating data

Data Input : Rich attributes and context

System Framework of AutoCross

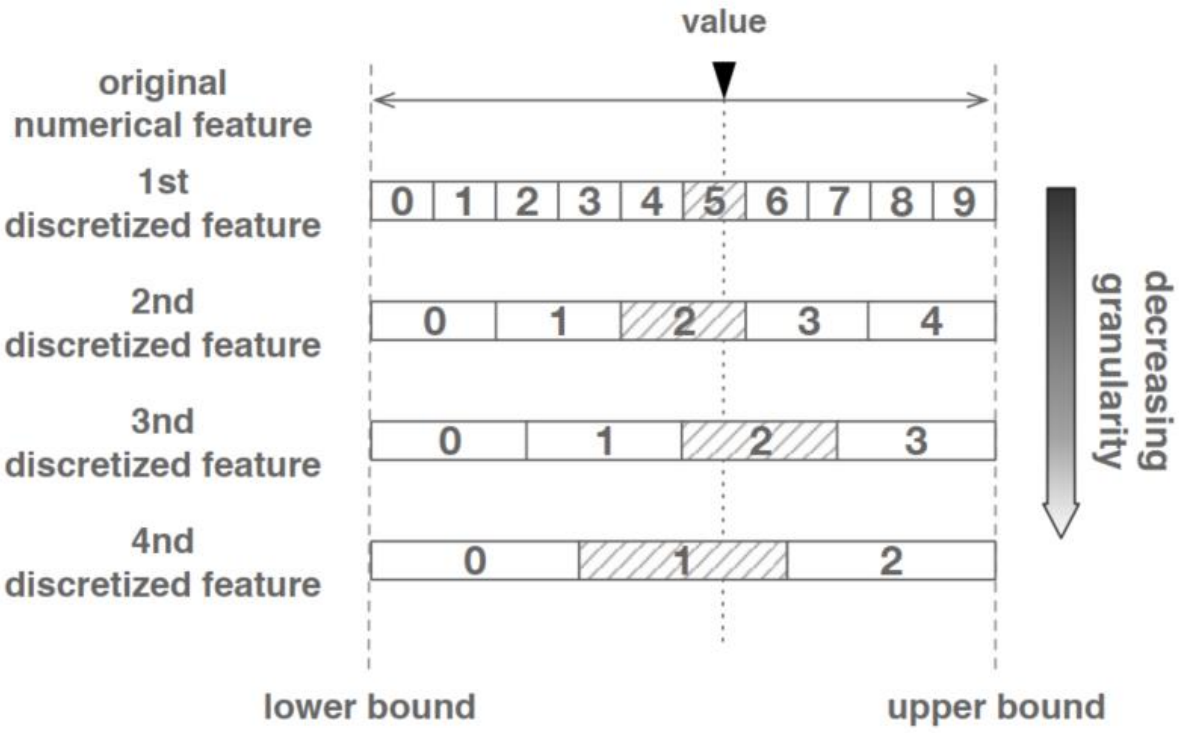
- Input
  - Training data
  - Feature type
- Output
  - Feature producer



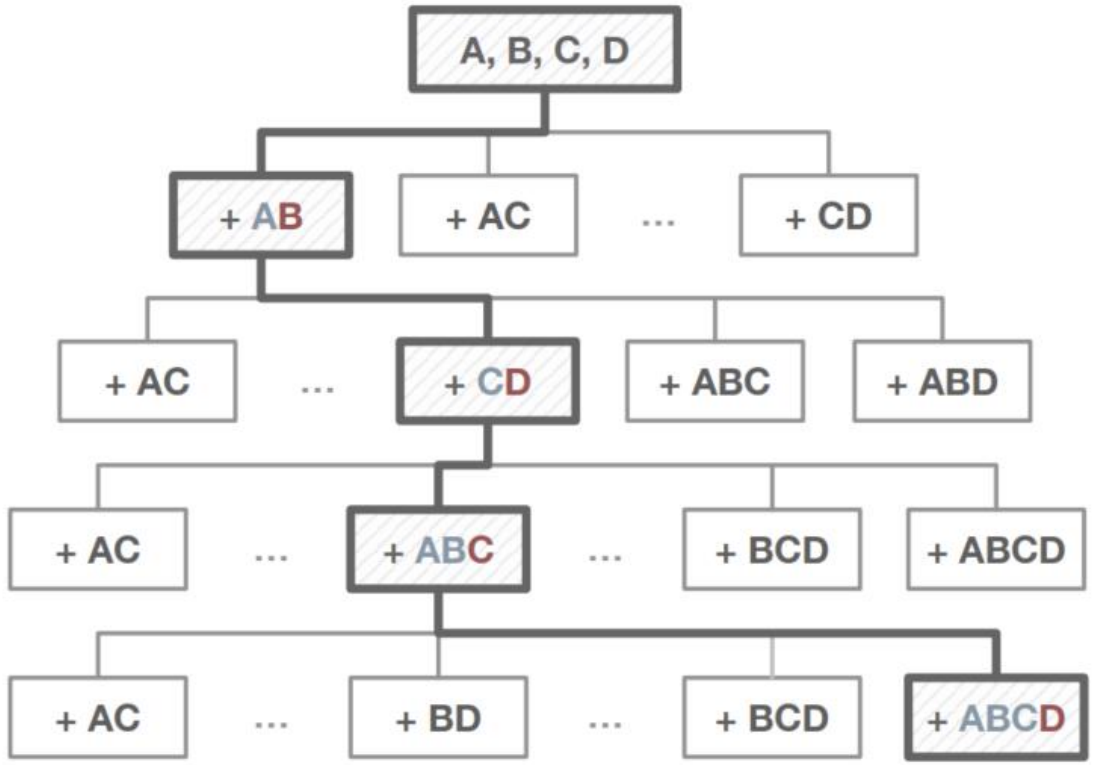
Luo, Y et al., Autocross: Automatic feature crossing for tabular data in real-world applications. KDD 2019.

Method (feature search)

- multi-granularity discretization

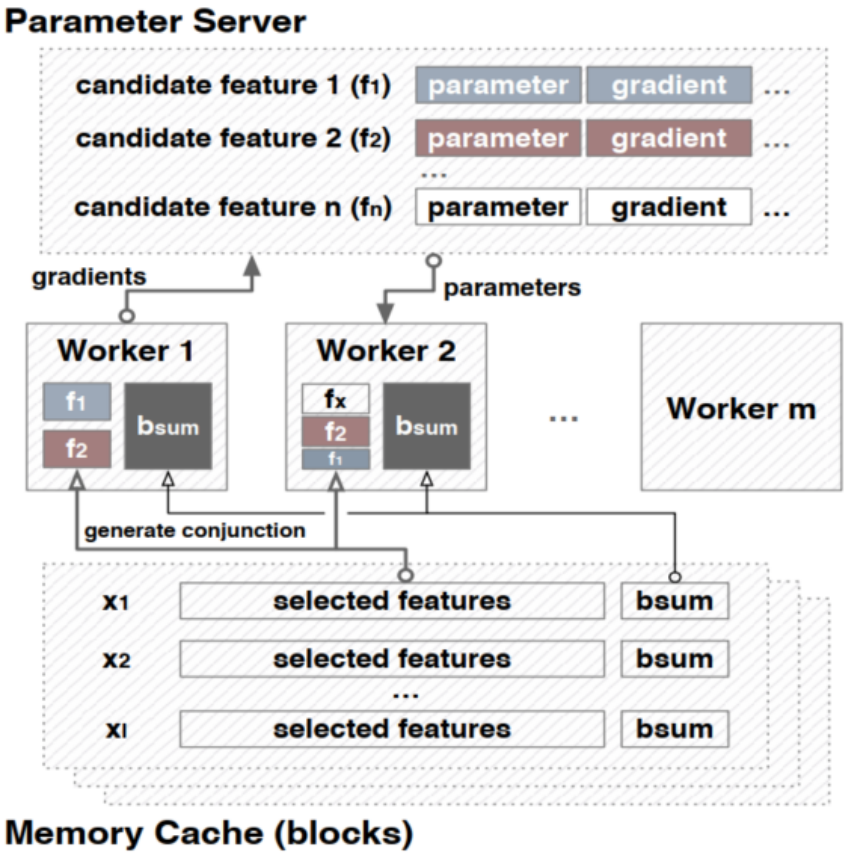


- beam search



Method (feature evaluation)

- field-wise logistic regression



- successive mini-batch gradient descent

Algorithm 2 Successive Mini-batch Gradient Descent (SMBGD).

Require: set of candidate feature sets $\mathbb{S} = \{\mathcal{S}_i\}_{i=1}^{n}$, training data equally divided into $N \geq \sum_{k=0}^{\lceil \log_2 n \rceil - 1} 2^k$ data blocks.

Ensure: best candidate $\mathcal{S}'$.

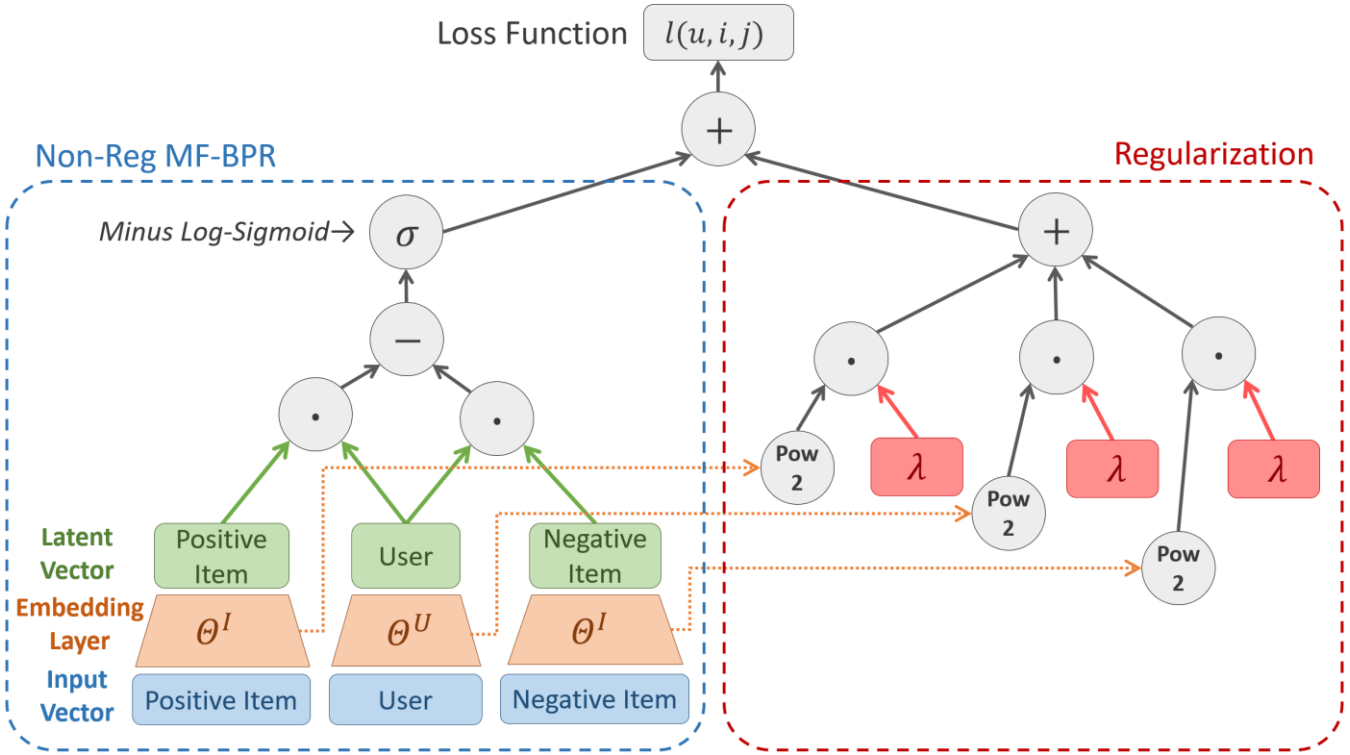
1: **for** $k = 0, 1, \cdots, \lceil \log_2 n \rceil - 1$ **do**
2: use additional 2^k data blocks to update the field-wise LR models of all $\mathcal{S} \in \mathbb{S}$, with warm-starting;
3: evaluate the models of all $\mathcal{S}$'s with validation AUC;
4: keep the top half of candidates in $\mathbb{S}$: $\mathbb{S} \leftarrow$ `top_half`($\mathbb{S}$) (rounding down);
5: break if $\mathbb{S}$ contains only one element;
6: **end for**
7: **return** $\mathcal{S}'$ (the singleton element of $\mathbb{S}$).

Outline

- AutoML for Collaborative Filtering Task
- AutoML for Click-through Rate Prediction Task
- AutoML for Tuning Hyper-parameters in RecSys

Regularization term in RecSys

(take MF-BPR as an example)

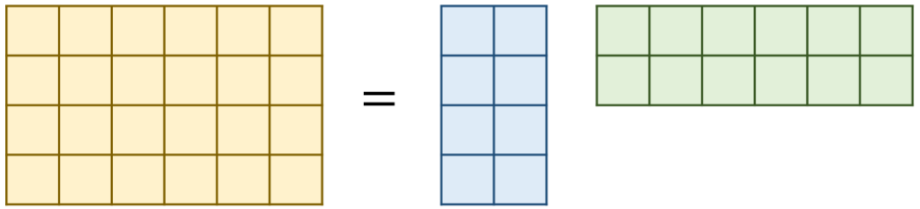


$$l_{S_T}(\Theta|\lambda) = \tilde{l}_{S_T}(\Theta) + \Omega(\Theta|\lambda)$$
$$= - \sum_{(u,i,j)\in S_T} \ln(\sigma(\hat{y}_{ui}(\Theta) - \hat{y}_{uj}(\Theta))) + \Omega(\Theta|\lambda)$$

Chen, Y., Chen, B., He, X., Gao, C., Li, Y., Lou, J. G., & Wang, Y.. λopt: Learn to regularize recommender models in finer levels. In KDD 2019

A common concern of RecSys models: Regularization Tuning Headache

Bob

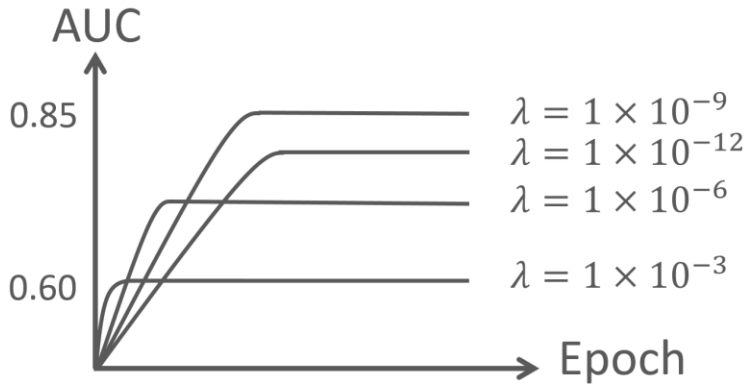


Model

$$l = \tilde{l}(\theta) + \lambda\|\theta\|^2$$

Penalty

Regularized Loss



What if we can do the regularization automatically?

Chen, Y., Chen, B., He, X., Gao, C., Li, Y., Lou, J. G., & Wang, Y.. λopt: Learn to regularize recommender models in finer levels. In KDD 2019

Alternating Optimization

$$\min_{\Lambda} \sum_{\{(u',i',j')\in S_V\}} l(u',i',j'|\arg\min_{\Theta} \sum_{\{(u,i,j)\in S_T\}} l(u,i,j|\Theta,\Lambda))$$

At iteration t

Train the wheel 

- Fix Λ, Optimize Θ

  $\rightarrow$ Conventional MF-BPR except λ is fine-grained now

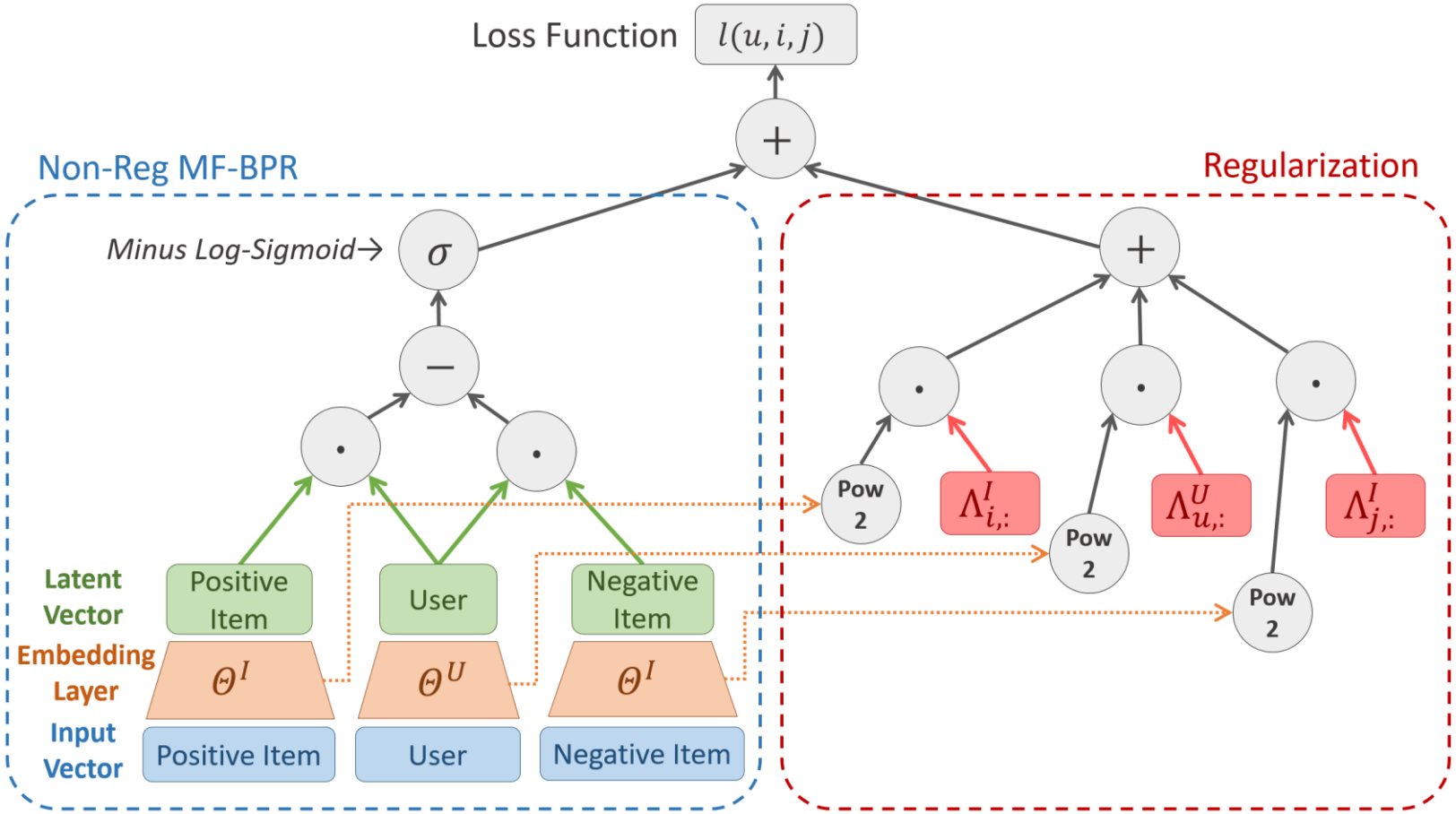
Train the brake 

- Fix Θ, Optimize Λ

  $\rightarrow$ Find Λ which achieve the smallest validation loss

Chen, Y., Chen, B., He, X., Gao, C., Li, Y., Lou, J. G., & Wang, Y.. λopt: Learn to regularize recommender models in finer levels. In KDD 2019

MF-BPR with fine-grained regularization



Chen, Y., Chen, B., He, X., Gao, C., Li, Y., Lou, J. G., & Wang, Y.. λopt: Learn to regularize recommender models in finer levels. In KDD 2019

Fix Θ, Optimize Λ

Taking a greedy perspective, we look for Λ which can minimize the next-step validation loss

- If we keep using current Λ for next step, we would obtain $\overline{\Theta}_{t+1}$
- Given $\overline{\Theta}_{t+1}$, our aim is $\min_{\Lambda} l_{S_V}(\overline{\Theta}_{t+1})$ with the constraint of non-negative Λ

But how to obtain $\overline{\Theta}_{t+1}$ without influencing the normal Θ update?

- Simulate* the MF update!
  - Obtain the gradients by combining the non-regularized part and penalty part

*: Using – over the letters to distinguish the simulated ones with normal ones

$$\overline{\frac{\partial l_{S_T}}{\partial \Theta_t}} = \frac{\overline{\partial \tilde{l}_{S_T}}}{\partial \Theta_t} + \frac{\partial \Omega}{\partial \Theta_t}$$

Λ is the only variable here

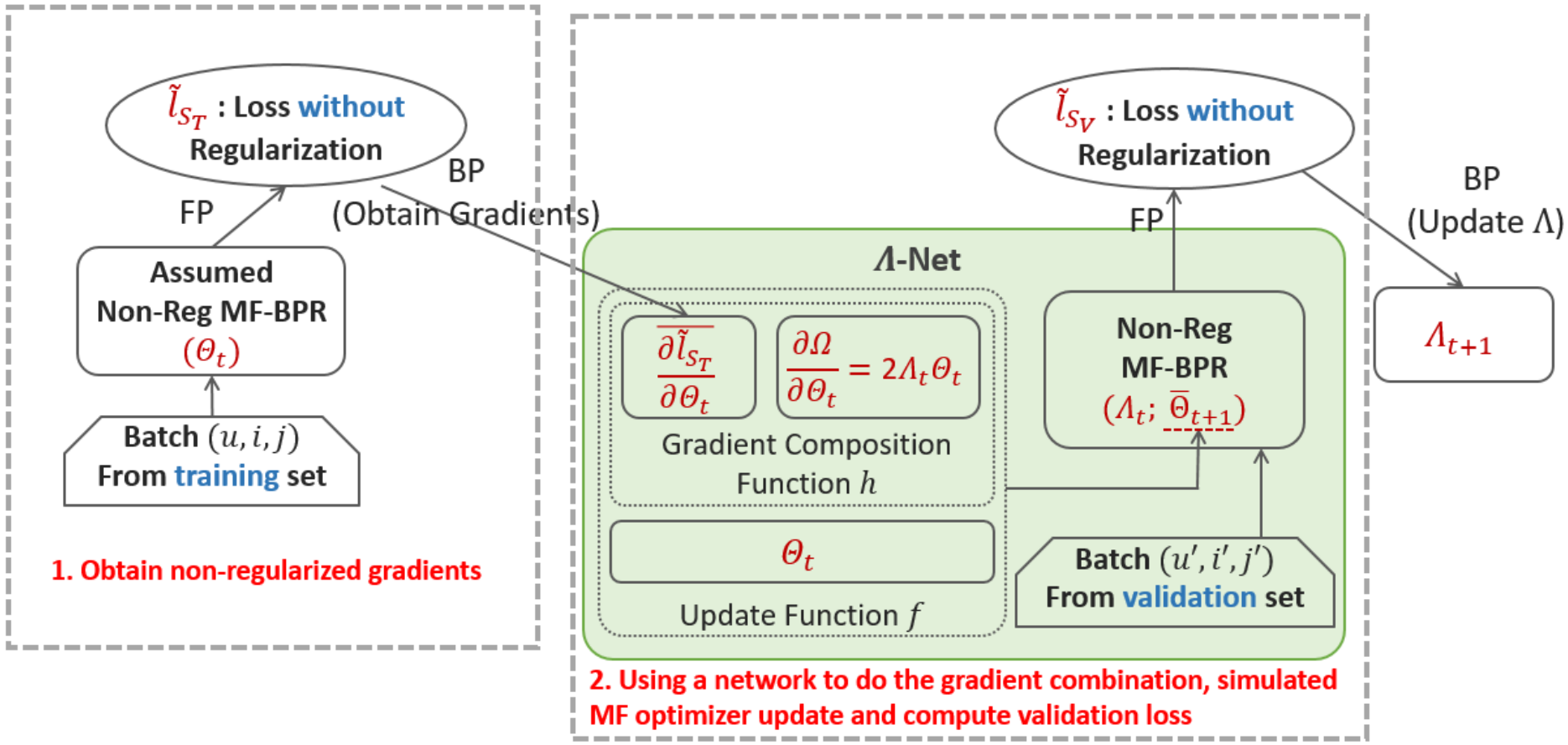
  - Simulate the operations that the MF optimizer would take

$$\overline{\Theta}_{t+1} = f(\Theta_t, \overline{\frac{\partial l_{S_T}}{\partial \Theta_t}})$$

f denotes the MF update function

Chen, Y., Chen, B., He, X., Gao, C., Li, Y., Lou, J. G., & Wang, Y.. λopt: Learn to regularize recommender models in finer levels. In KDD 2019

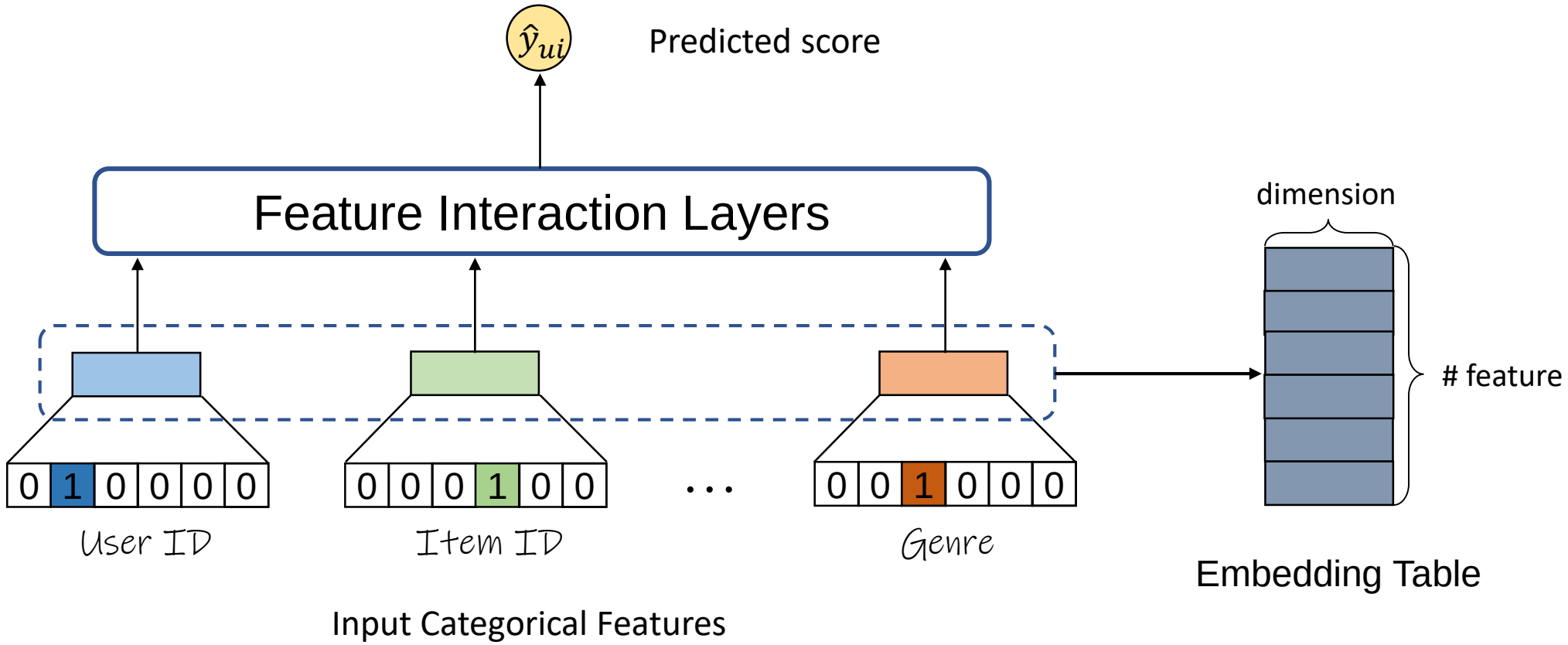
Fix Θ, Optimize Λ in Auto-Differentiation



Chen, Y., Chen, B., He, X., Gao, C., Li, Y., Lou, J. G., & Wang, Y.. λopt: Learn to regularize recommender models in finer levels. In KDD 2019

Automated Embedding Size Learning for RecSys

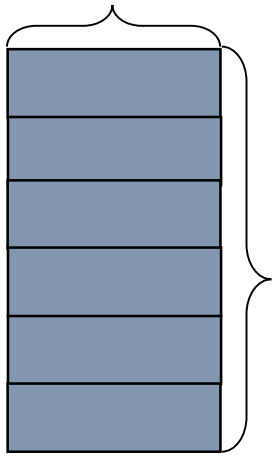
- Embedding in Recommender Systems



Backgrounds

- Problems in Traditional Embedding Method

dimension →



- All features share the same dimension → hard to handle the heterogeneity among different features.
  - Large: lead to overfitting for sparse features.
  - Small: the less representative ability for high-frequent features.

feature →

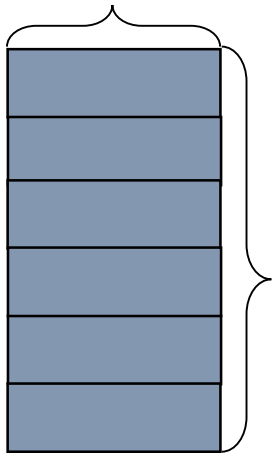
- Usually contains millions even billions of features → need huge memory to storage the embedding table.

Liu, S., Gao, C., Chen, Y., Jin, D., & Li, Y. Learnable Embedding sizes for Recommender Systems. In *ICLR 2021.*

Backgrounds

➢ Problems in Traditional Embedding Method

dimension →



➢ All features share the same dimension → hard to handle the heterogeneity among different features.

Large: lead to overfitting for sparse features.

Small: the less representative ability for high-frequent features.

feature →

➢ Usually contains millions even billions of features → need huge memory to storage the embedding table.

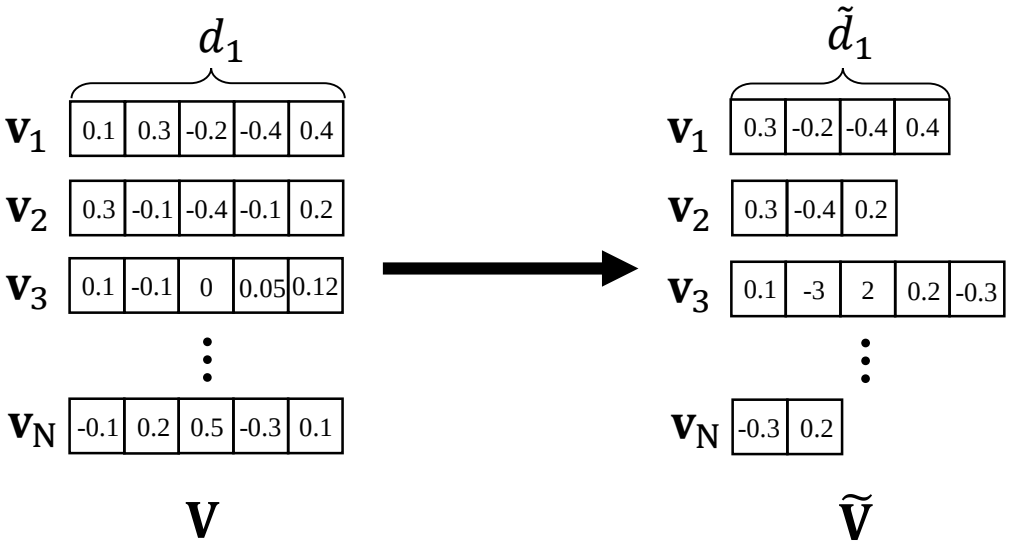
Solution:

- ☐ For dimension, make the embedding size non-uniform.
- ☐ For features, make some features share some embedding parameters.

Liu, S., Gao, C., Chen, Y., Jin, D., & Li, Y. Learnable Embedding sizes for Recommender Systems. In *ICLR 2021.*

Methodology

Automatically assign different embedding sizes $\tilde{d}_i$ for different features embeddings $\mathbf{v}_i$



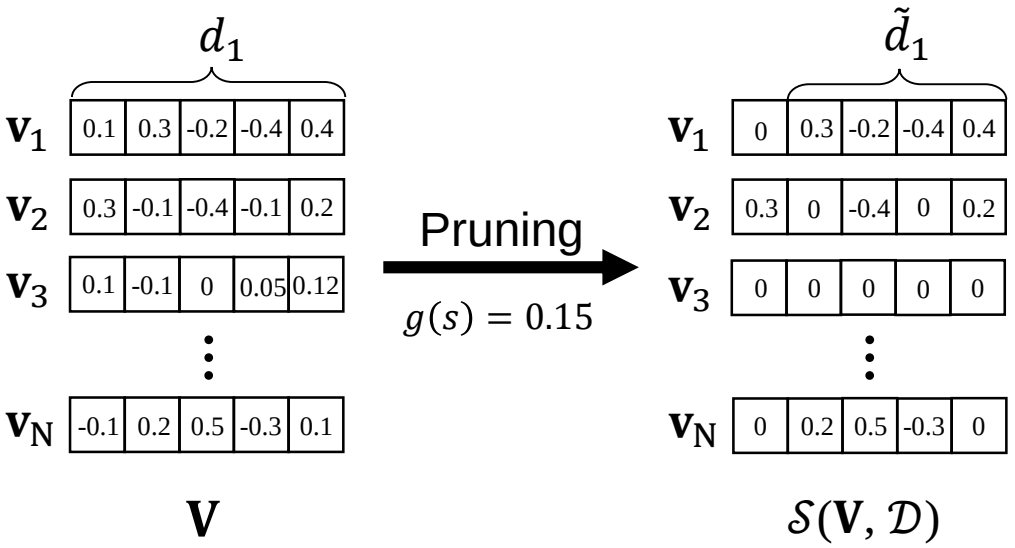
Challenge:

Discreteness and extremely-large optimization space make directly learning impossible.

There are d^N possible solutions, so it is a NP-hard problems.

Methodology

How to learn?
prune the less important values in $\mathbf{V}$ with learnable thresholds.



$$\hat{\mathbf{V}} = \mathcal{S}(\mathbf{V}, s) = sign(\mathbf{V}) ReLU(|\mathbf{V}| - g(s)),$$

$$g(s) = \frac{1}{1 + e^{-s}}$$

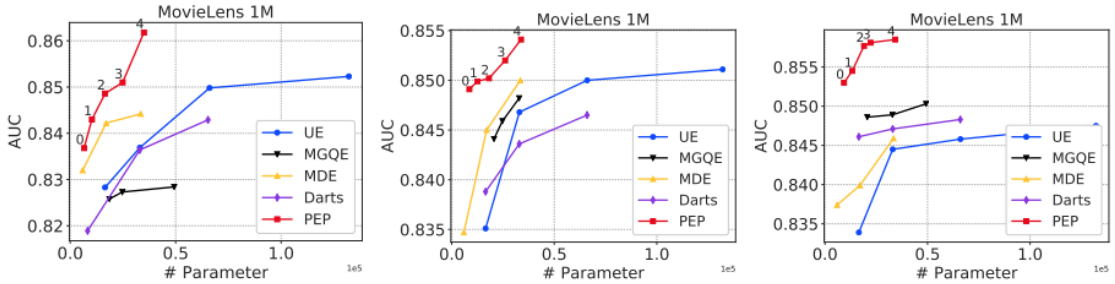
s : control the value of thresholds and update the value with other parameters by gradient descent.

$$\min \mathcal{L}(\mathbf{V}, \Theta, \mathcal{D}), \longrightarrow \min \mathcal{L}(\mathcal{S}(\mathbf{V}, s), \Theta, \mathcal{D}).$$

The learnable threshold prevent the computational bi-level optimization process while make the embedding size learnable !

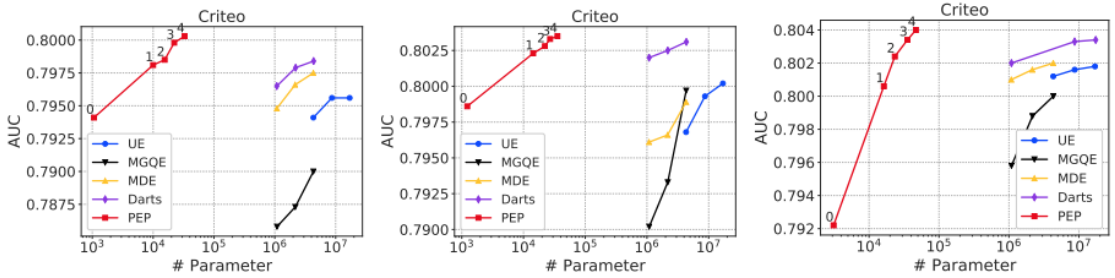
Liu, S., Gao, C., Chen, Y., Jin, D., & Li, Y. Learnable Embedding sizes for Recommender Systems. In *ICLR 2021.*

Empirical Study



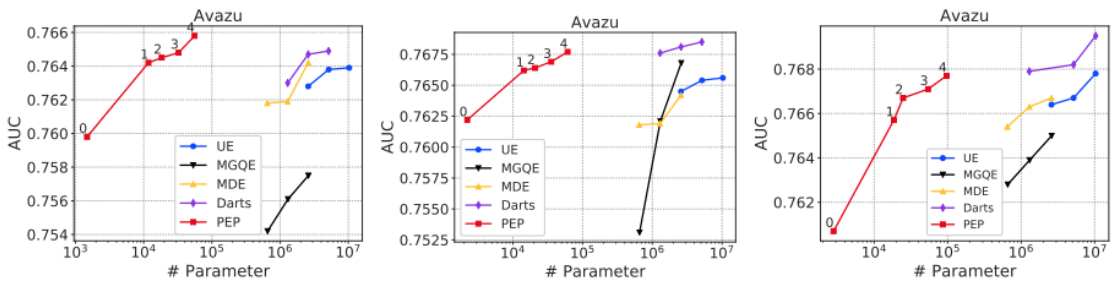
(a) FM (b) DeepFM (c) AutoInt

Figure 2: Parameter-AUC curve on MovieLens-1M with three base models.



(a) FM (b) DeepFM (c) AutoInt

Figure 3: Parameter-AUC curve on Criteo with three base models.



(a) FM (b) DeepFM (c) AutoInt

The PEP method shows State-of-the-art Performance of CTR prediction task

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 engineering, model architecture design and algorithm selection.

- AutoML help automatically make reasonable decisions on different datasets and tasks.

References

- Jin, B., Gao, C., He, X., Jin, D., & Li, Y. Multi-behavior recommendation with graph convolutional networks. In SIGIR 2020.
- Wang, X., Huang, T., Wang, D., Yuan, Y., Liu, Z., He, X., & Chua, T. S. Learning Intents behind Interactions with Knowledge Graph for Recommendation. In TheWebConf 2021.
- Yao, Q., Chen, X., Kwok, J. T., Li, Y., & Hsieh, C. J. Efficient neural interaction function search for collaborative filtering. In TheWebConf 2020.
- Gao, C., Yao, Q., Jin, D., & Li, Y. Efficient Data-specific Model Search for Collaborative Filtering. In KDD 2021.
- Luo, Y., Wang, M., Zhou, H., Yao, Q., Tu, W. W., Chen, Y., ... & Yang, Q. Autocross: Automatic feature crossing for tabular data in real-world applications. In KDD 2019.
- Chen, Y., Chen, B., He, X., Gao, C., Li, Y., Lou, J. G., & Wang, Y. λopt: Learn to regularize recommender models in finer levels. In KDD 2019.
- Liu, S., Gao, C., Chen, Y., Jin, D., & Li, Y. Learnable Embedding sizes for Recommender Systems. In ICLR 2021.

Next…

1. An introduction to Automated Machine Learning (AutoML)
   - Background on technical tools from machine learning
2. Why AutoML is Needed in RecSys and Recent Advances
   - Exemplar works introducing AutoML into RecSys
3. Automated Graph Representation Learning for RecSys
   - Explore neural architecture search for GNN based RecSys
4. Automated Knowledge Graph (KG) Embedding
   - Explore AutoML for KG Embedding based RecSys