From Graph Learning to Open-World Representation Learning

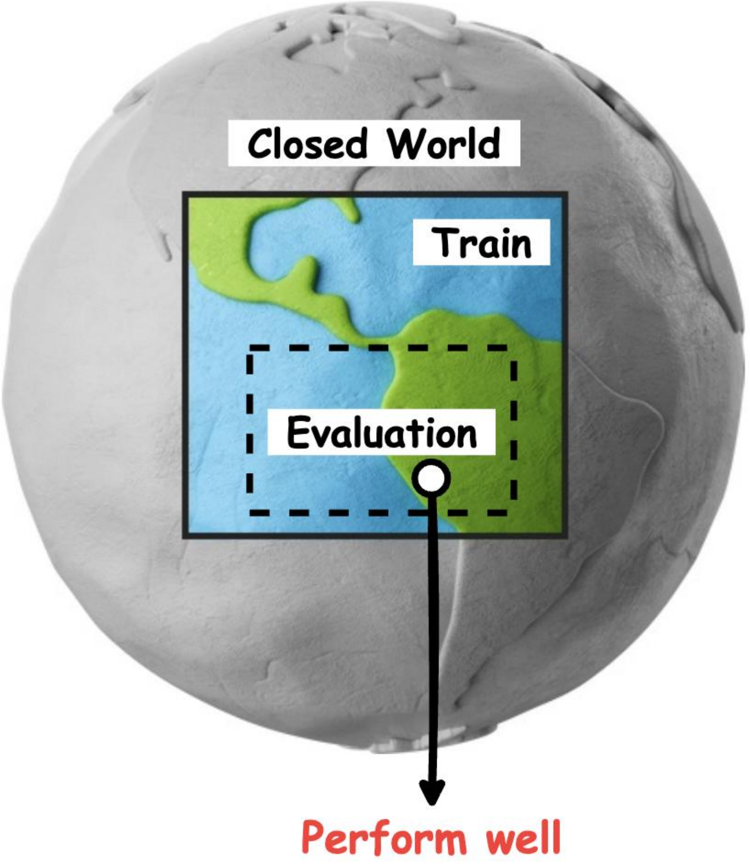
Qitian Wu

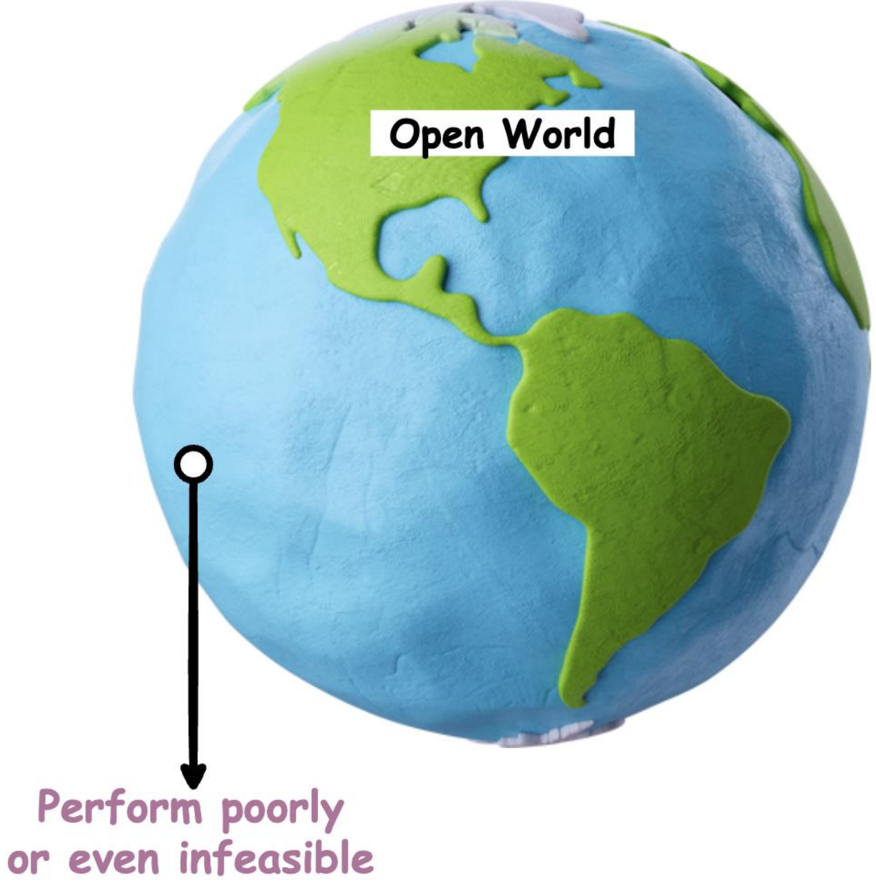
Department of Computer Science and Engineering

Shanghai Jiao Tong University

Background and Motivation

- ❑ Machine learning models perform well in CLOSED-world situations
- ❑ Real-world situations are OPEN, dynamic and also uncertain

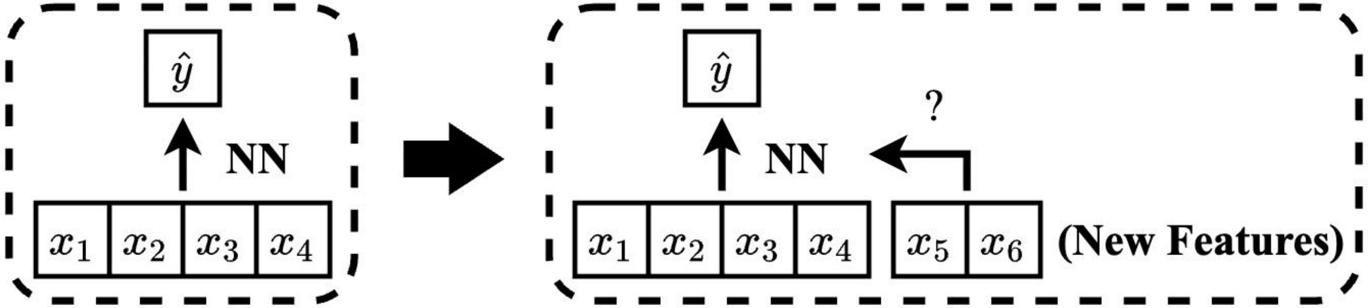




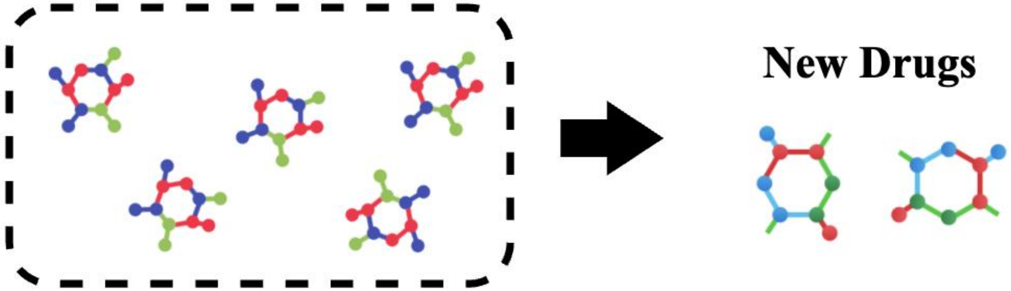
More Specific Examples

- ❑ New users/items in recommender systems

- ❑ New features collected by new released platforms for decisions

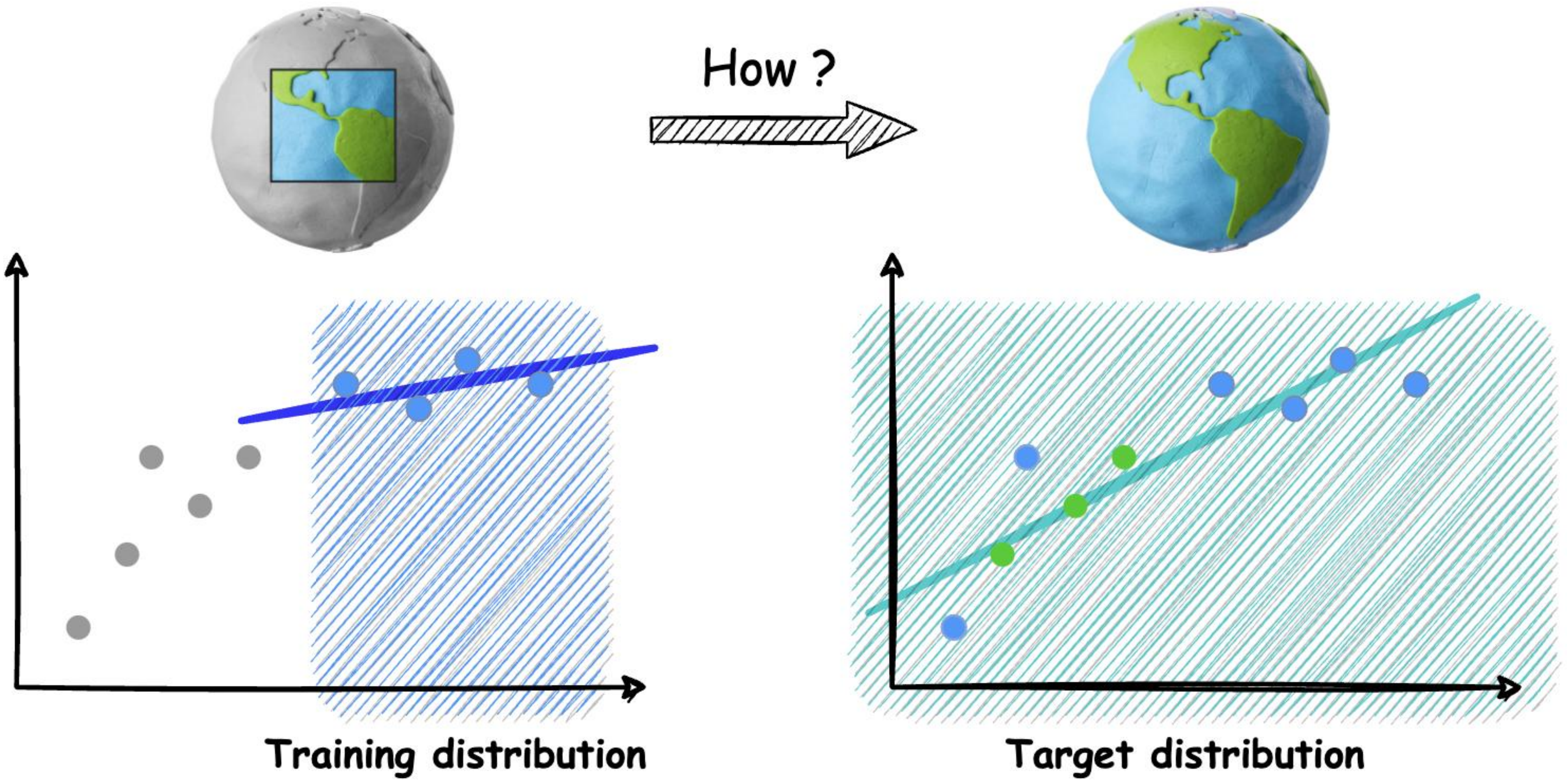


- ❑ New developed drugs or combinations for treatment



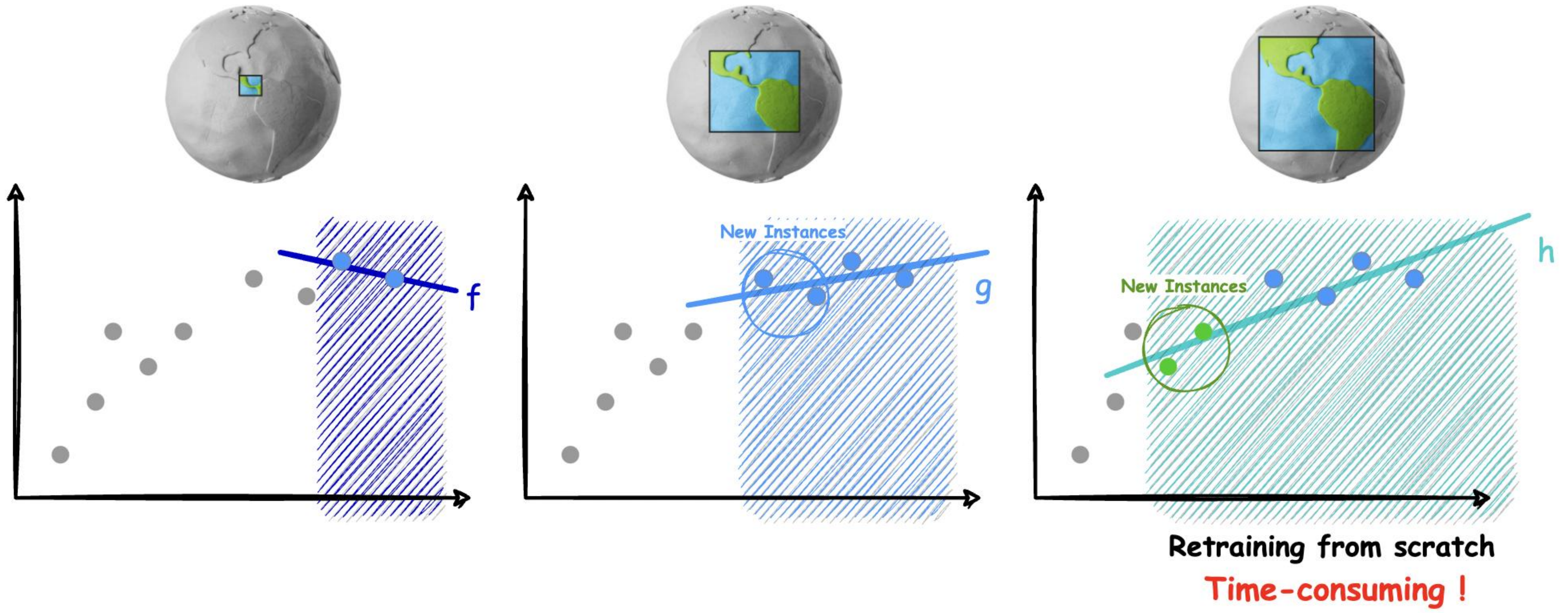
Technical Challenges

- ❑ **Open-world learning requires out-of-distribution generalization**



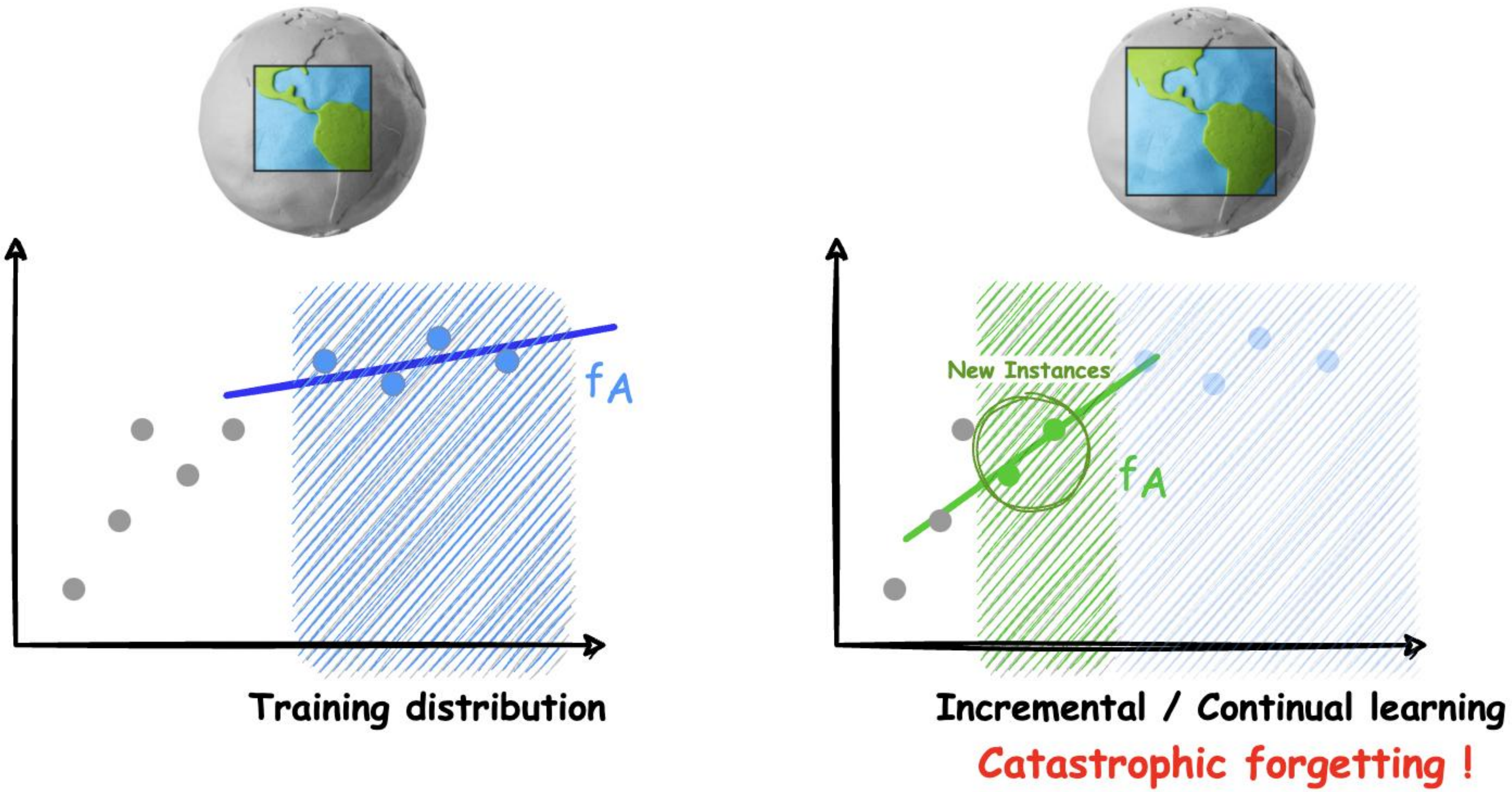
Technical Challenges

- ❑ **Retraining model from scratch for each new data is time-consuming**



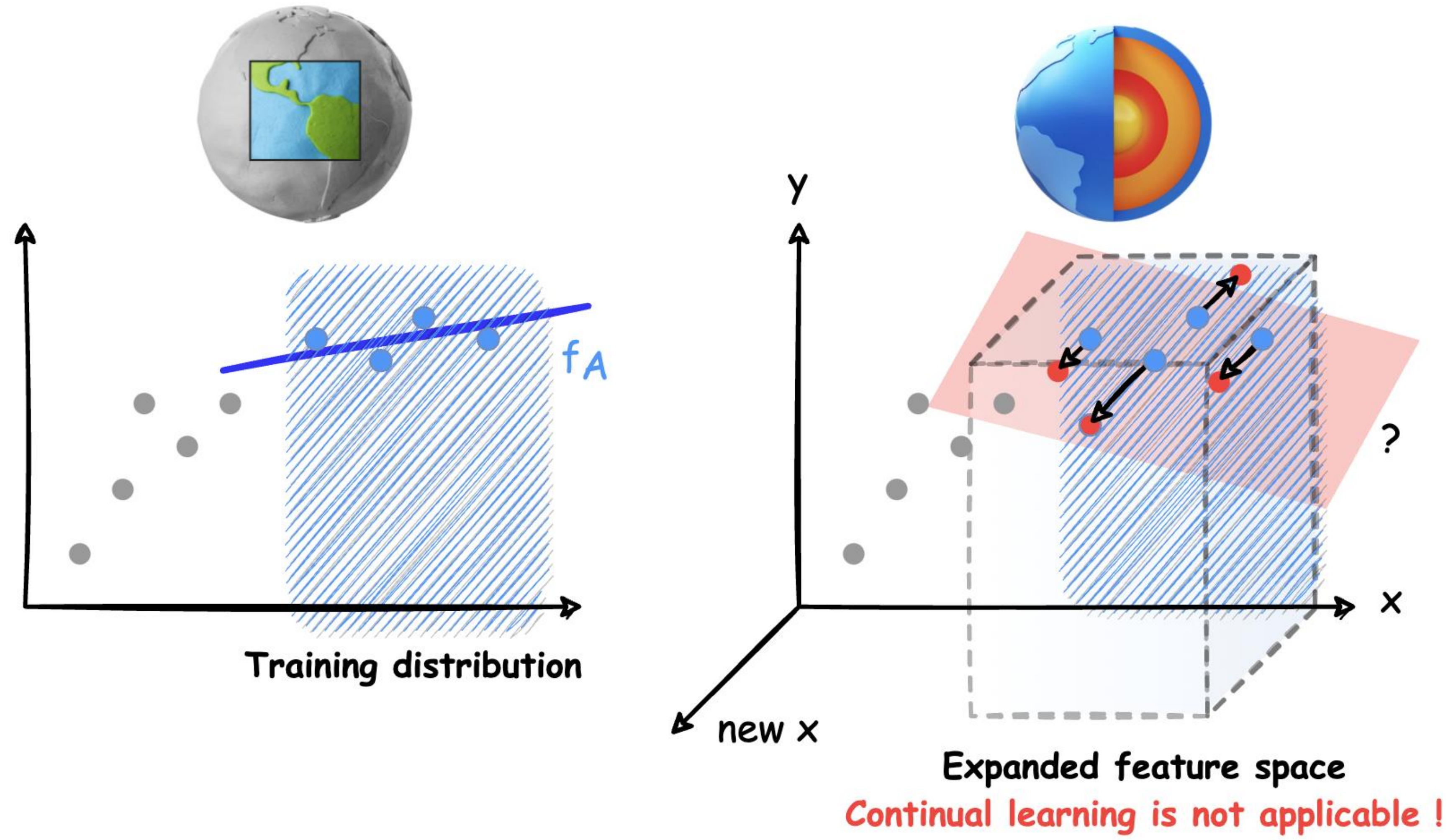
Technical Challenges

- Incremental learning or finetuning may lead to over-fitting new data



Technical Challenges

❑ **Incremental learning cannot deal with expanded feature space**



Towards Open-World Recommendation: An Inductive Model-based Collaborative Filtering Approach

International Conference on Machine Learning (ICML'21)

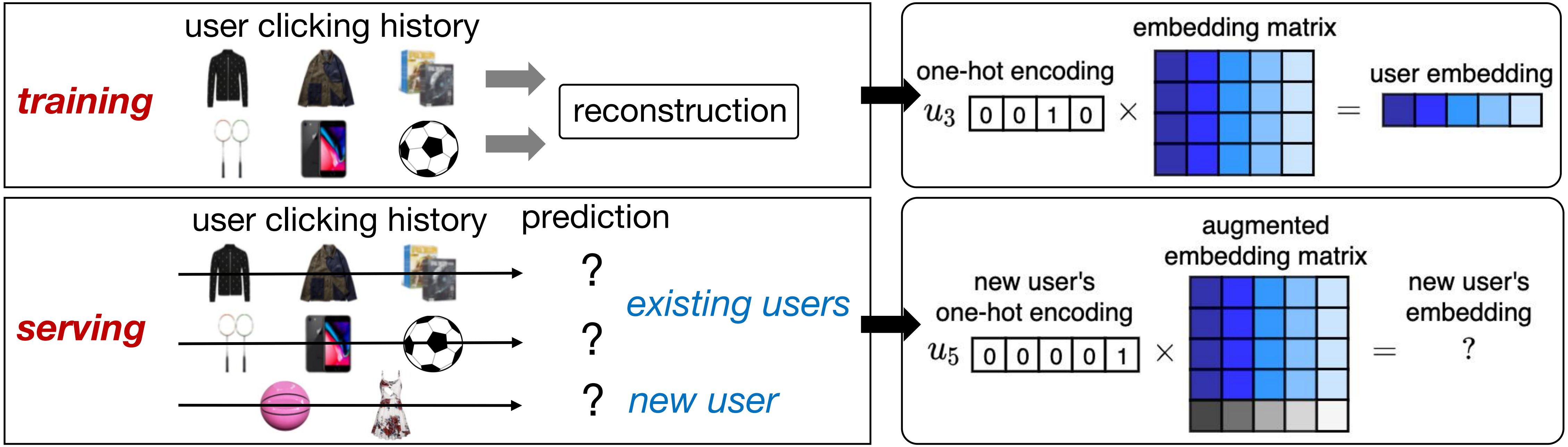
Qitian Wu, Hengrui Zhang, Xiaofeng Gao, Junchi Yan, Hongyuan Zha
Shanghai Jiao Tong University
The Chinese University of Hong Kong, Shenzhen

Background for Recommendation

- Model-based Collaborative Filtering ≈ Matric Factorization Model
- Basic idea:

  user-item rating matrix $\xrightarrow{\text{factorize}}$ { **user embeddings** / item embeddings } → reconstruction matrix

- **CF models cannot handle new unseen users in *open-world recommendation***



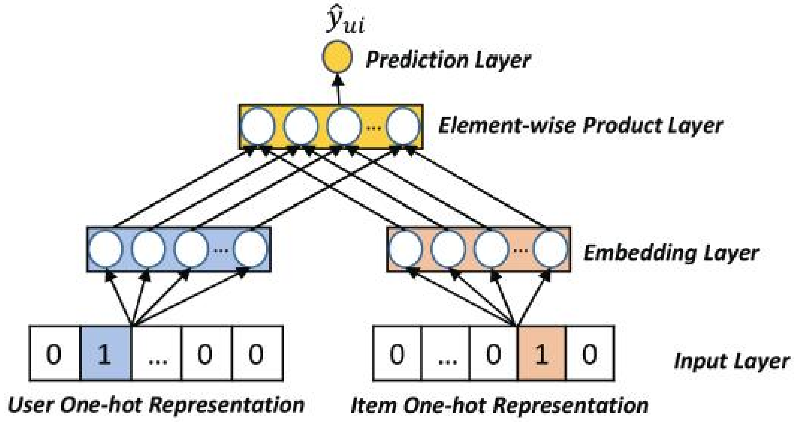
Collaborative Filtering

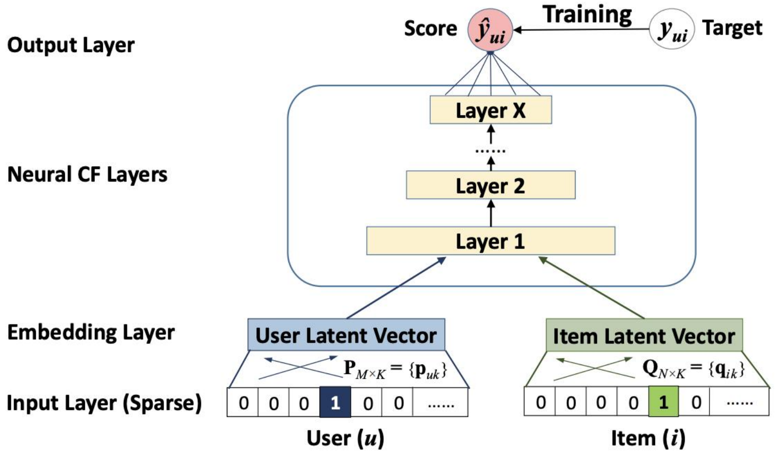
❑ Formulation of CF model for RecSys:

- a user-item interaction matrix $R = \{r_{ui}\}_{M \times N}$
- assume user latent factors $\mathbf{P} = \{\mathbf{p}_u\}_{M \times d}$
- assume item latent factors $\mathbf{Q} = \{\mathbf{q}_i\}_{N \times d}$
- consider an interaction model $\hat{r}_{ui} = f_\theta(\mathbf{p}_u, \mathbf{q}_i)$
- target objective $\mathcal{L}(\hat{R}, R) = \sum_{(u,i)} L(\hat{r}_{ui}, r_{ui})$

❑ Limitations: transductive learning

- cannot handle new unseen users
  - ◻ model retraining requires additional cost
  - ◻ retraining may also lead to over-fitting



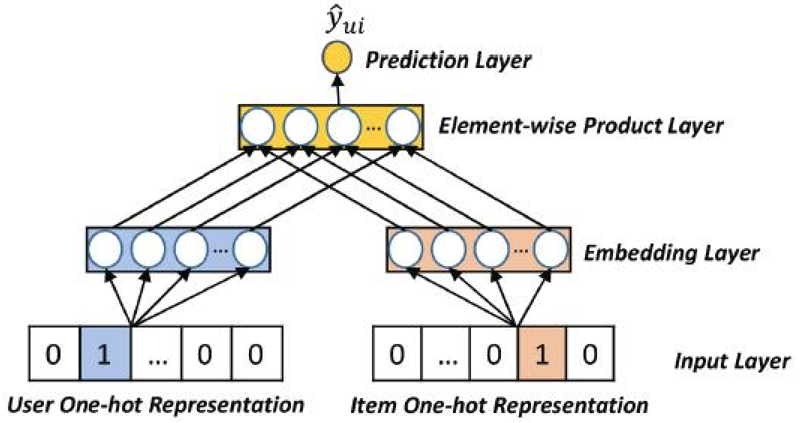


adapted from [He et al. 2017]

Collaborative Filtering

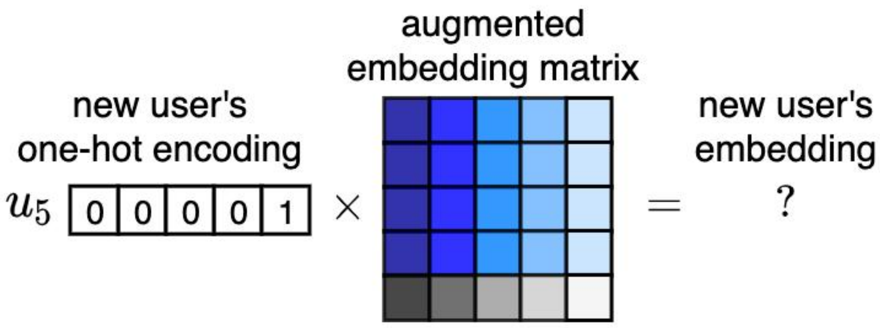
Formulation of CF model for RecSys:

- a user-item interaction matrix $R = \{r_{ui}\}_{M \times N}$
- assume user latent factors $\mathbf{P} = \{\mathbf{p}_u\}_{M \times d}$
- assume item latent factors $\mathbf{Q} = \{\mathbf{q}_i\}_{N \times d}$
- consider an interaction model $\hat{r}_{ui} = f_\theta(\mathbf{p}_u, \mathbf{q}_i)$
- target objective $\mathcal{L}(\hat{R}, R) = \sum_{(u,i)} L(\hat{r}_{ui}, r_{ui})$



Limitations: transductive learning

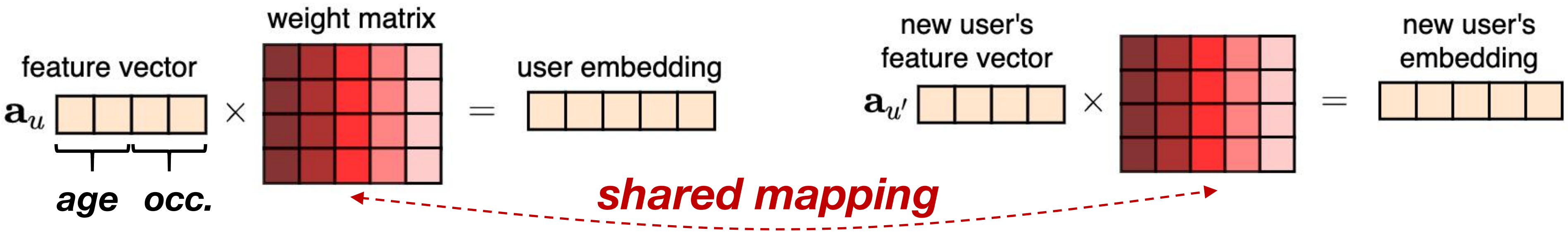
- cannot handle new unseen users
  - model retraining requires additional cost
  - retraining may also lead to over-fitting



new user's embeddings are not trained

Challenges for Inductive Learning

- Inductive learning can be achieved via shared mapping



- Expressiveness would be sacrificed with inductive learning

$$u_1 \xrightarrow{f_1} \mathbf{p}_{u_1}$$
$$u_2 \xrightarrow{f_2} \mathbf{p}_{u_2}$$

v. s.

$$\mathbf{a}_{u_1} \xrightarrow{f} \mathbf{p}_{u_1}$$
$$\mathbf{a}_{u_2} \xrightarrow{f} \mathbf{p}_{u_2}$$

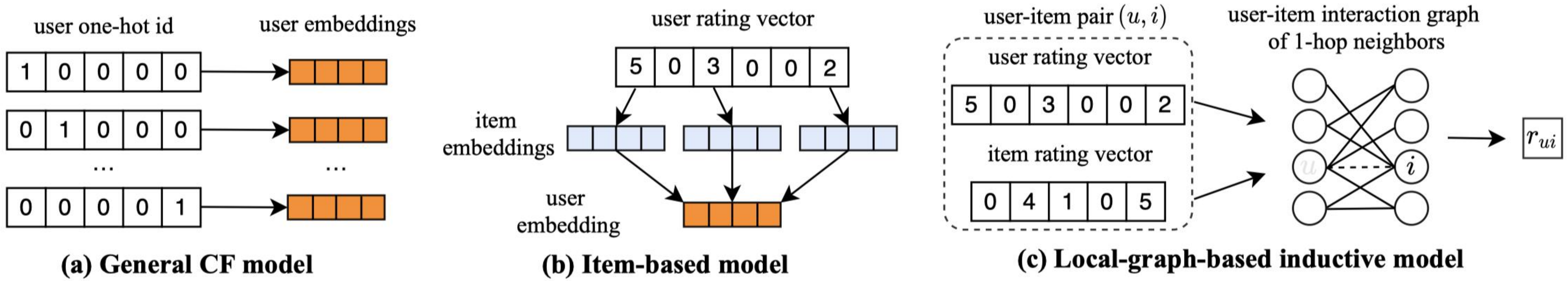
transductive learning

pros: sufficient expressiveness
cons: fail for new users

inductive learning

pros: flexible for new users
cons: limited capacity/expressiveness

Related Works and Comparison



incremental learning

pros:
1. advanced capacity
2. fast training/inference

cons:
1. bad generalization
2. over-parametrization

item-side learning

pros:
1. fewer parameters
2. enable inductive

cons:
1. limited capacity
2. user-item imbalance

index-free learning [Zhang et al. ICLR'20]

pros:
1. enable inductive
2. not require features

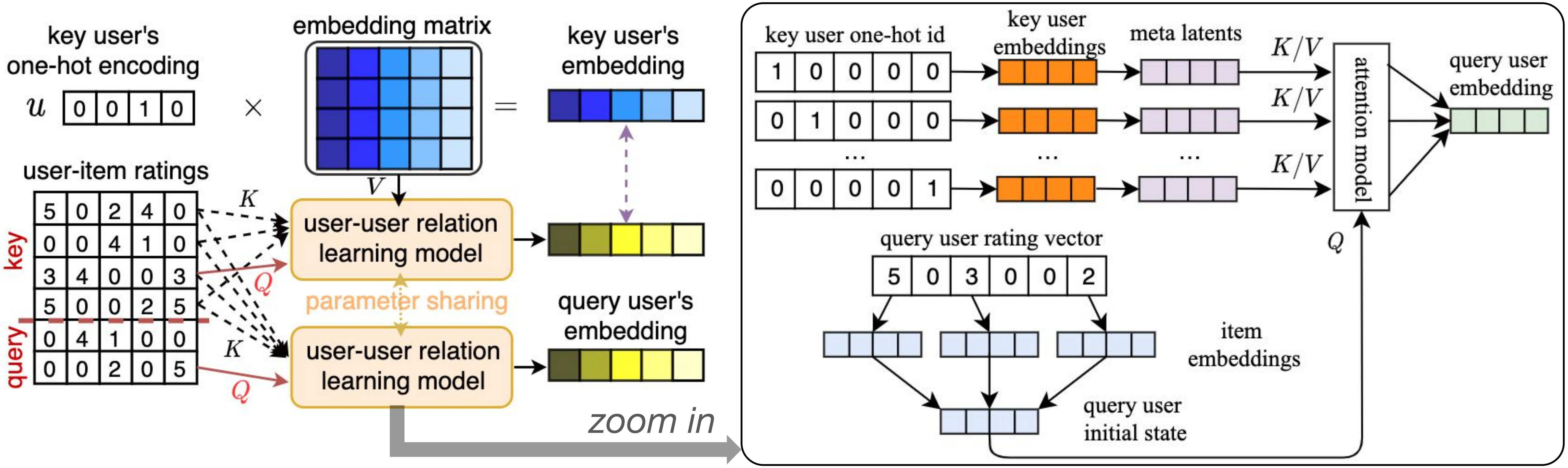
cons:
1. limited expressiveness
2. fail for implicit feedback

Our Solutions: Inductive CF Model

❑ Basic idea:

- leverage one group of users to express another
- learn a latent graph over users
- message passing from existing users to new ones

Key insight: user preferences share underlying proximity that induces latent graphs



Our Solutions: Inductive CF Model (Cont.)

- **Partition users into two groups:** $|\mathcal{U}_k| = M_k \quad |\mathcal{U}_q| = M_q$
  - Key users: transductive learning (traditional model)

model: $\mathbf{P}_k = \{\mathbf{p}_u\}_{M_k \times d} \qquad \mathbf{Q} = \{\mathbf{q}_i\}_{N \times d} \qquad \hat{r}_{ui} = f_\theta(\mathbf{p}_u, \mathbf{q}_i)$

learning: $\min_{\mathbf{P}_k, \mathbf{Q}, \theta} \mathcal{D}_{\mathcal{S}_k}(\hat{R}_k, R_k)$ **where** $R_k = \{r_{ui}\}_{M_k \times N}$

  - Query users: inductive learning (new model)

edge weights in a latent user-user graph

model: $\tilde{\mathbf{p}}_{\mathbf{u}'} = \mathbf{c}_{u'}^\top \mathbf{P}_k \qquad \boxed{c_{u'u} = \frac{\mathbf{e}^\top [\mathbf{W}_q \mathbf{d}_{u'} \oplus \mathbf{W}_k \mathbf{p}_u]}{\sum_{u_o \in \mathcal{U}_k} \mathbf{e}^\top [\mathbf{W}_q \mathbf{d}_{u'} \oplus \mathbf{W}_k \mathbf{p}_{u_0}]}}$ **where** $\mathbf{d}_{u'} = \sum_{i \in \mathcal{I}_{u'}} \mathbf{q}_i$

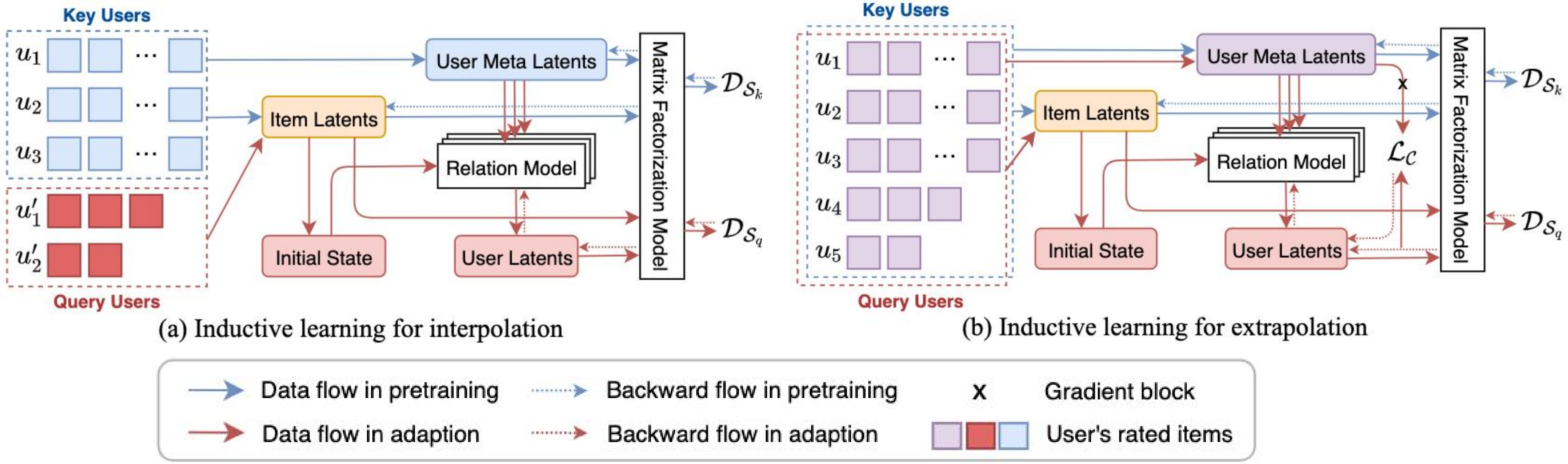
learning: $\min_{w, \theta} \mathcal{D}_{\mathcal{S}_q}(\hat{R}_q, R_q)$ **where** $R_q = \{r_{ui}\}_{M_q \times N} \qquad \hat{r}_{ui} = f_\theta(\tilde{\mathbf{p}}_{\mathbf{u}}, \mathbf{q}_i)$

objective: $\min_{w, \theta} \mathcal{D}_{\mathcal{S}_q}(\hat{R}_q, R_q) + \lambda \mathcal{L}_C(\mathbf{P}_k, \tilde{\mathbf{P}}_{\mathbf{k}}) \qquad \mathcal{L}_C(\mathbf{P}_k, \tilde{\mathbf{P}}_{\mathbf{k}}) = \frac{1}{M_q} \sum_{u \in \mathcal{U}_k} \log \frac{\exp(\mathbf{p}_u^\top \tilde{\mathbf{p}}_{\mathbf{u}})}{\sum_{u' \in \mathcal{U}_q} \exp(\mathbf{p}_u^\top \tilde{\mathbf{p}}_{\mathbf{u}'})}$

regularization: consistency between two estimated embeddings for one user

Our Solutions: Inductive CF Model (Cont.)

- Learning procedures: pretraining + adaption
- Consider two scenarios in open-world recommendation:
  - few-shot users (limited training data): pretraining on key + adaption on query
  - zero-shot users (no training data): pretraining + self-adaption on key users



Theoretical Analysis

- The model possesses the same representation capacity compared to matrix factorization
  - The only mild condition is that key users' latent factors span the latent space
- The generalization ability on new users depends on number of key users and training instances of new users

Theorem 1. *Assume Eq. (3) can achieve $\mathcal{D}_{\mathcal{S}_q}(\hat{R}_q, R_q) < \epsilon$ and the optimal $\mathbf{P}_k$ given by Eq. (1) satisfies column-full-rank, then there exists at least one solution for $\mathbf{C}$ in Eq. (2) such that $\mathcal{D}_{\mathcal{S}_q}(\hat{R}_q, R_q) < \epsilon$.*

$$\min_{\mathbf{P}_k, \mathbf{Q}, \theta} \mathcal{D}_{\mathcal{S}_k}(\hat{R}_k, R_k), \tag{1}$$

$$\min_{\mathbf{C}, \mathbf{Q}} \mathcal{D}_{\mathcal{S}_q}(\hat{R}_q, R_q), \tag{2}$$

$$\min_{\tilde{\mathbf{P}}_q, \mathbf{Q}} \mathcal{D}_{\mathcal{S}_q}(\hat{R}_q, R_q), \tag{3}$$

Theorem 2. *Assume 1) $\mathcal{D}$ is L-Lipschitz, 2) for $\forall \hat{r}_{u'i} \in \hat{R}_q$ we have $|\hat{r}_{u'i}| \leq B$, and 3) the L1-norm of $\mathbf{c}_{u'}$ is bounded by H. Then with probability at least $1 - \delta$ over the random choice of $\mathcal{S}_q \in ([M_q] \times [N])^{T_q}$, it holds that for any $\hat{R}_q$, the gap between $\mathcal{D}(\hat{R}_q, R_q)$ and $\mathcal{D}_{\mathcal{S}_q}(\hat{R}_q, R_q)$ will be bounded by*

$$O\left(2LHB\sqrt{\frac{2M_q \ln M_k}{T_q}} + \sqrt{\frac{ln(1/\delta)}{T_q}}\right). \tag{8}$$

Experiment Setup

- Dataset information:

	Dataset	# Users	#Items	# Ratings	Density	# Key/Query Users	# Training/Test Instances
explicit	Douban	3,000	3,000	0.13M	0.0152	2,131/869	80,000/20,000
	Movielens-100K	943	1,682	0.10M	0.0630	123,202/13,689	
	Movielens-1M	6,040	3,706	1.0M	0.0447	5,114/926	900,199/100,021
implicit	Amazon-Books	52,643	91,599	2.1M	0.0012	49,058/3,585	2,405,036/526,430
	Amazon-Beauty	2,944	57,289	0.08M	0.0004	780/2,164	53,464/29,440

- Evaluation Protocol:
  - Explicit dataset: random split, RMSE & NDCG metric
  - Implicit dataset: leave-last-out, AUC & NDCG metric, negative sampling
- Comparison: CF models, inductive models, feature-based models

Experiment Setup

- Implementation:
  - IDCF-NN: feedforward neural network as predictor

$$f_\theta(\mathbf{p}_u, \mathbf{q}_i) = \frac{(\mathbf{p}_u^\top \mathbf{q}_i + nn([\mathbf{p}_u \| \mathbf{q}_i \| \mathbf{p}_u \odot \mathbf{q}_i]))}{2} + b_u + b_i$$

  - IDCF-GC: graph convolution network as predictor

$$\mathbf{m}_{u,m} = ReLU\left(\frac{1}{|\mathcal{N}_{u,m}|} \sum_{i \in \mathcal{N}_{u,m}} \mathbf{W}_{q,m} \mathbf{q}_i\right)$$

$$\mathbf{n}_{i,m} = ReLU\left(\frac{1}{|\mathcal{N}_{i,m}|} \sum_{u \in \mathcal{N}_{i,m}} \mathbf{W}_{p,m} \mathbf{p}_u\right)$$

$$f(\mathbf{p}_u, \mathbf{q}_i, \{\mathbf{p}_u\}_{u \in \mathcal{N}_i}, \{\mathbf{q}_i\}_{i \in \mathcal{N}_u}) = nn'([\mathbf{p}_u \odot \mathbf{q}_i \| \mathbf{p}_u \odot \mathbf{m}_u \| \mathbf{n}_i \odot \mathbf{q}_i \| \mathbf{n}_i \odot \mathbf{m}_u]) + b_u + b_i$$

Experiment Results

❑ Comparison results for explicit feedback:

- For few-shot query users, very competitive results as inductive models and very close test performance to transductive models
- For zero-shot new users, significantly outperform SOTA inductive models

Method	Inductive	Feature	Douban				ML-100K				ML-1M			
			RMSE		NDCG		RMSE		NDCG		RMSE		NDCG	
			All	FS	All	FS	All	FS	All	FS	All	FS	All	FS
PMF	No	No	0.737	0.718	0.939	0.954	0.932	1.003	0.858	0.843	0.851	0.946	0.919	**0.940**
NNMF	No	No	0.729	**0.705**	0.939	0.952	0.925	0.987	0.895	0.878	0.848	**0.940**	0.920	0.937
GCMC	No	No	0.731	0.706	0.938	**0.956**	0.911	0.989	0.900	**0.886**	**0.837**	0.947	0.923	0.939
NIMC	Yes	Yes	0.732	0.745	0.928	0.931	1.015	1.065	0.832	0.824	0.873	0.995	0.889	0.904
BOMIC	Yes	Yes	0.735	0.747	0.923	0.925	0.931	1.001	0.828	0.815	0.847	0.953	0.905	0.924
F-EAE	Yes	No	0.738	-	-	-	0.920	-	-	-	0.860	-	-	-
IGMC	Yes	No	**0.721**	0.728	-	-	**0.905**	0.997	-	-	0.857	0.956	-	-
IDCF-NN (ours)	Yes	No	0.738	0.712	0.939	**0.956**	0.931	0.996	0.896	0.880	0.844	0.952	0.922	**0.940**
IDCF-GC (ours)	Yes	No	0.733	0.712	**0.940**	**0.956**	**0.905**	**0.981**	**0.901**	0.884	0.839	0.944	**0.924**	**0.940**

Method	Douban		ML-100K		ML-1M	
	RMSE	NDCG	RMSE	NDCG	RMSE	NDCG
NIMC	0.766	0.921	1.089	0.864	1.059	0.883
BOMIC	0.764	0.920	1.088	0.859	1.057	0.879
FISM	1.910	0.824	1.891	0.760	2.283	0.771
MultVAE	2.783	0.823	2.865	0.758	2.981	0.792
IGMC	0.743	-	1.051	-	0.997	-
IDCF-NN	0.749	**0.955**	1.078	0.877	0.994	0.941
IDCF-GC	**0.723**	**0.955**	**1.011**	**0.881**	**0.957**	**0.942**

Lower RMSE and higher NDCG are better

Experiment Results

❑ Comparison results for implicit feedback:

- For few-shot query users, achieve SOTA results
- For zero-shot new users, significantly improvement

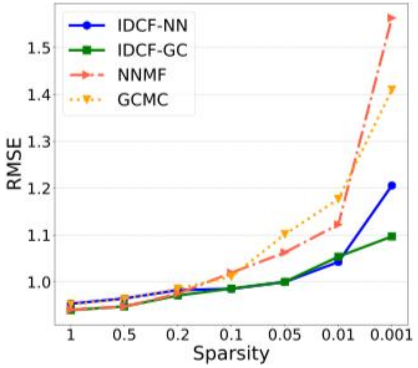
Method	Amazon-Books				Amazon-Beauty			
	AUC		NDCG		AUC		NDCG	
	Query	New	Query	New	Query	New	Query	New
PMF	0.917	-	0.888	-	0.779	-	0.769	-
NNMF	0.919	-	0.891	-	0.790	-	0.763	-
NGCF	0.916	-	0.896	-	0.793	-	0.775	-
PinSAGE	0.923	-	0.901	-	0.790	-	0.775	-
FISM	-	0.752	-	0.792	-	0.613	-	0.678
MultVAE	-	0.738	-	0.701	-	0.644	-	0.679
IDCF-NN	**0.944**	0.939	**0.928**	0.920	0.792	0.750	**0.783**	0.774
IDCF-GC	0.938	**0.946**	0.921	**0.930**	**0.801**	**0.791**	0.772	**0.791**

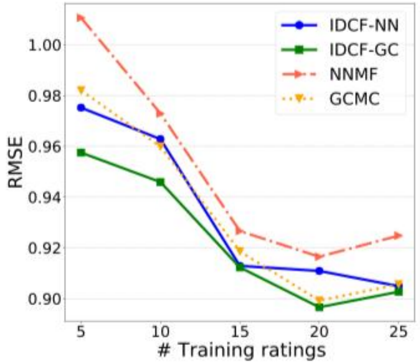
Higher AUC and higher NDCG are better

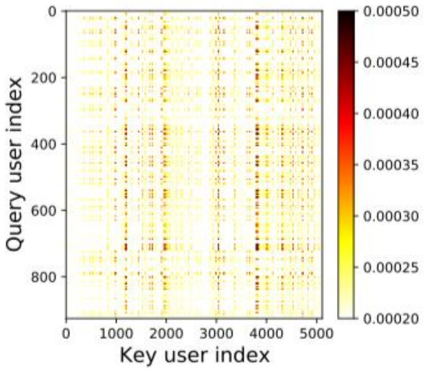
Experiment Results

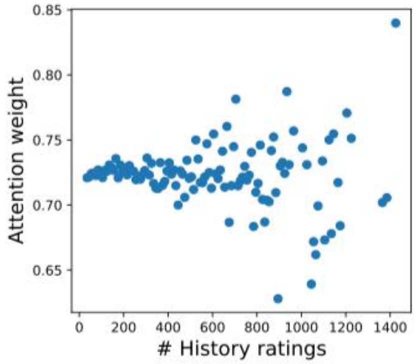
❑ Further discussions:

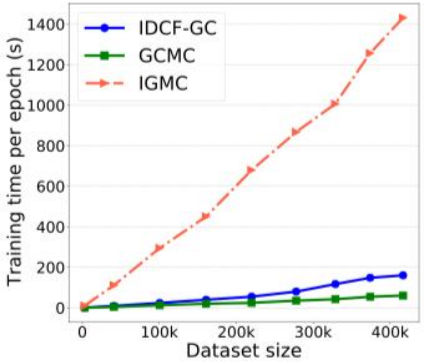
- Our model can exceed transductive models w,r,t, RMSE when users' training/historical ratings are sparse
- There exist informative key users that contribute to most of capacity. Key users with more historical ratings tend to be more important
- The training time scales linearly w.r.t. dataset size











Towards Open-World Feature Extrapolation: An Inductive Graph Learning Approach

Advances in Neural Information Processing Systems (NeurIPS'21)

Qitian Wu, Chenxiao Yang, Junchi Yan
Department of Computer Science and Engineering
Shanghai Jiao Tong University

Background for Attribute Feature Learning

- **General problem:** learn a mapping from input features to labels
  - Input data $\mathbf{x} = [x_1, x_2, \cdots, x_d]$ where x_i denotes the i-th input feature
  - Assume a prediction model $f : \mathbf{x} \rightarrow y$ and objective

$$f^* = \arg\min_f \mathbb{E}_{(\mathbf{x},y)\sim D}[l(f(\mathbf{x}), y)]$$

- **Applications**
  - Tabular data: weather/income/usage prediction, disease diagnosis…
  - Real systems: recommendation, advertisement, question answering…

Scenario 1: Predict a person's income with aga/occ/edu

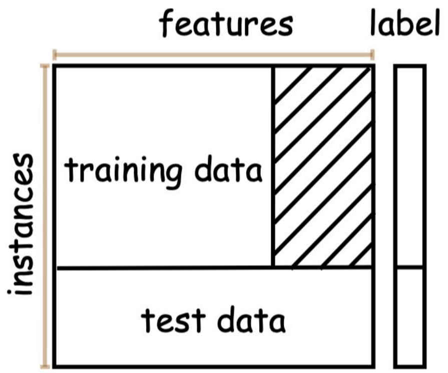
	age	occ	edu	income
o_1	x_{11}	x_{12}	x_{13}	y_1
o_2	x_{21}	x_{22}	x_{23}	y_2
o_3	x_{31}	x_{32}	x_{33}	?
$\cdots$		$\cdots$		

Scenario 2: Predict whether a user would click an item with attributes

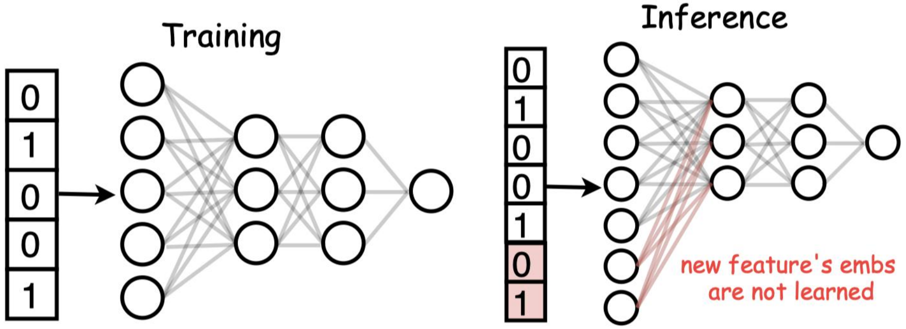
user features: age/gender...
item features: category/price...

Challenges and Limitations of Neural Networks

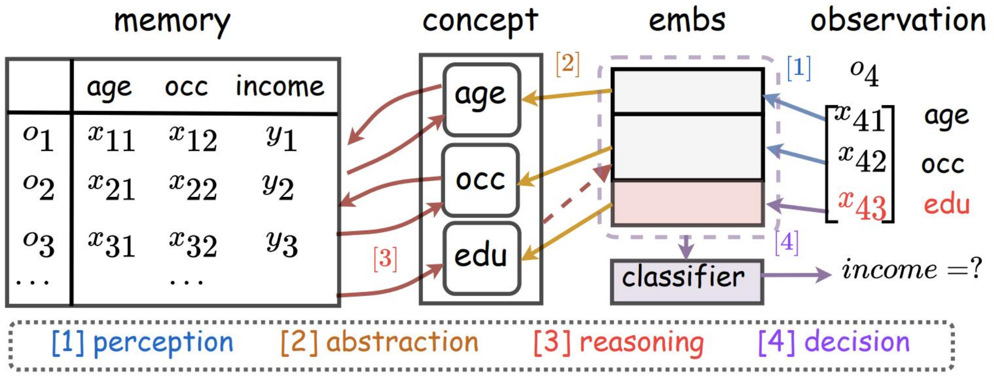
- ❑ **Challenges** for attribute feature learning
  - New features dynamically appear (unseen features in test set)
  - Scenarios: heterogeneous data sources, multi-modal data



- ❑ How can neural networks deal with new features
  - Retraining from scratch
    - ◻ Issue: time-consuming
  - Incremental learning on new features
    - ◻ Issue: over-fitting & catastrophic forgetting



- ❑ **Inductive reasoning ability**
  - Humans possess inherent ability for understanding new infromation



Problem Formulation

- Preprocessing: convert raw inputs to multi-hot vectors
  - Raw input $\mathbf{r}_i = [r_{i1}, r_{i2}, \cdots, r_{id}]$ where r_{im} denotes the m-th raw feature
  - For categorical feature: one-hot encoding representation
  - For continuous feature: first discretization then one-hot encoding

$$\mathbf{x}_i = [\mathbf{x}_i^1, \mathbf{x}_i^2, \cdots, \mathbf{x}_i^d] \quad \text{where } \mathbf{x}_i^m \text{ is a one-hot vector}$$

- Open-world feature extrapolation:

Training Set	*domain generalization* →	Test Set
Data $\{(\mathbf{x}_i, y_i)\}_{i\in I_{tr}}$		Data $\{(\mathbf{x}_{i'}, y_{i'})\}_{i'\in I_{te}}$
Feature space $\mathbf{x}_i \in \mathcal{X}_{tr} = \{0,1\}^D$	$\boxed{\mathcal{X}_{tr} \subset \mathcal{X}_{te}}$	Feature space $\mathbf{x}_{i'} \in \mathcal{X}_{te} = \{0,1\}^{D'}$
Label space $y_i \in \mathcal{Y}$		Label space $y_{i'} \in \mathcal{Y}$

- Two cases causing feature space expansion:

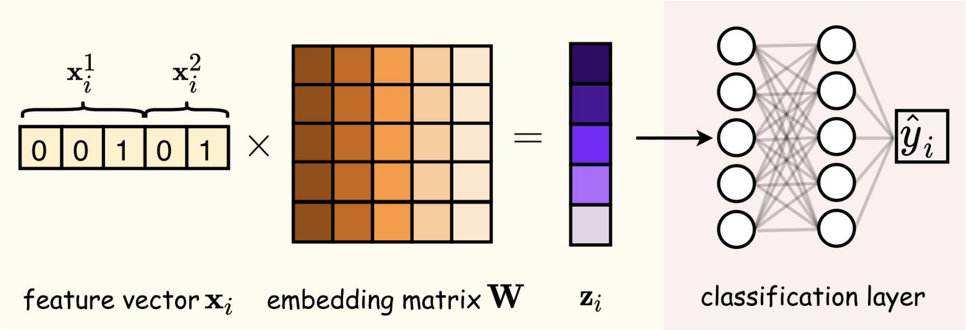
  1) new raw features come, 2) unseen feature values out of known range

Key Observation 1: Permutation-Invariance

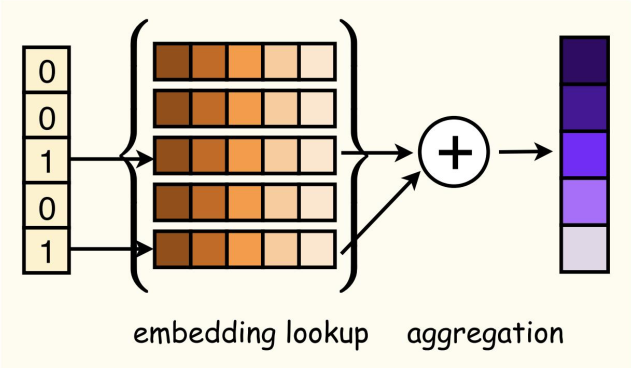
- Neural networks can be decomposed into two parts

$$\hat{y}_i = h(\mathbf{x}_i; \phi, \mathbf{W})$$

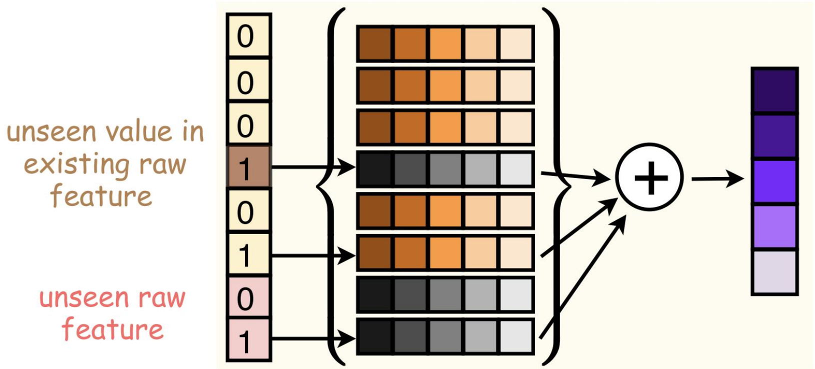
$$\Leftrightarrow \begin{cases} \mathbf{z}_i = \mathbf{W}\mathbf{x}_i \\ \hat{y}_i = \mathrm{FFN}(\mathbf{z}_i; \phi) \end{cases}$$



- Equivalent view: feature embedding look-up + embedding aggregation



Key insight:
The permutation-invariance property enables variable-length input features



Key Observation 2: Feature-Data Graph

❑ The input feature-data matrix can be treated as a **bipartite graph**

Input data matrix
$\mathbf{X}_{tr} = [\mathbf{x}_i]_{i \in I_{tr}} \in \{0,1\}^{N \times D}$

- Feature nodes $F_{tr} = \{f_j\}_{j=1}^{D}$
- Instance nodes $I_{tr} = \{o_i\}_{i=1}^{N}$
- Adjacency matrix $\mathbf{X}_{tr}$

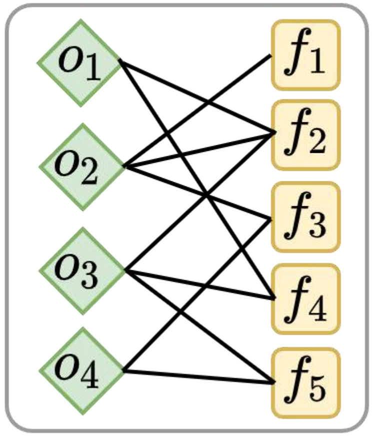
Advantage of graph representation:
1) Variable-size for features/instances
2) Missing values are allowed

Key insight:
Convert inferring embeddings for new features to inductive representation on graphs

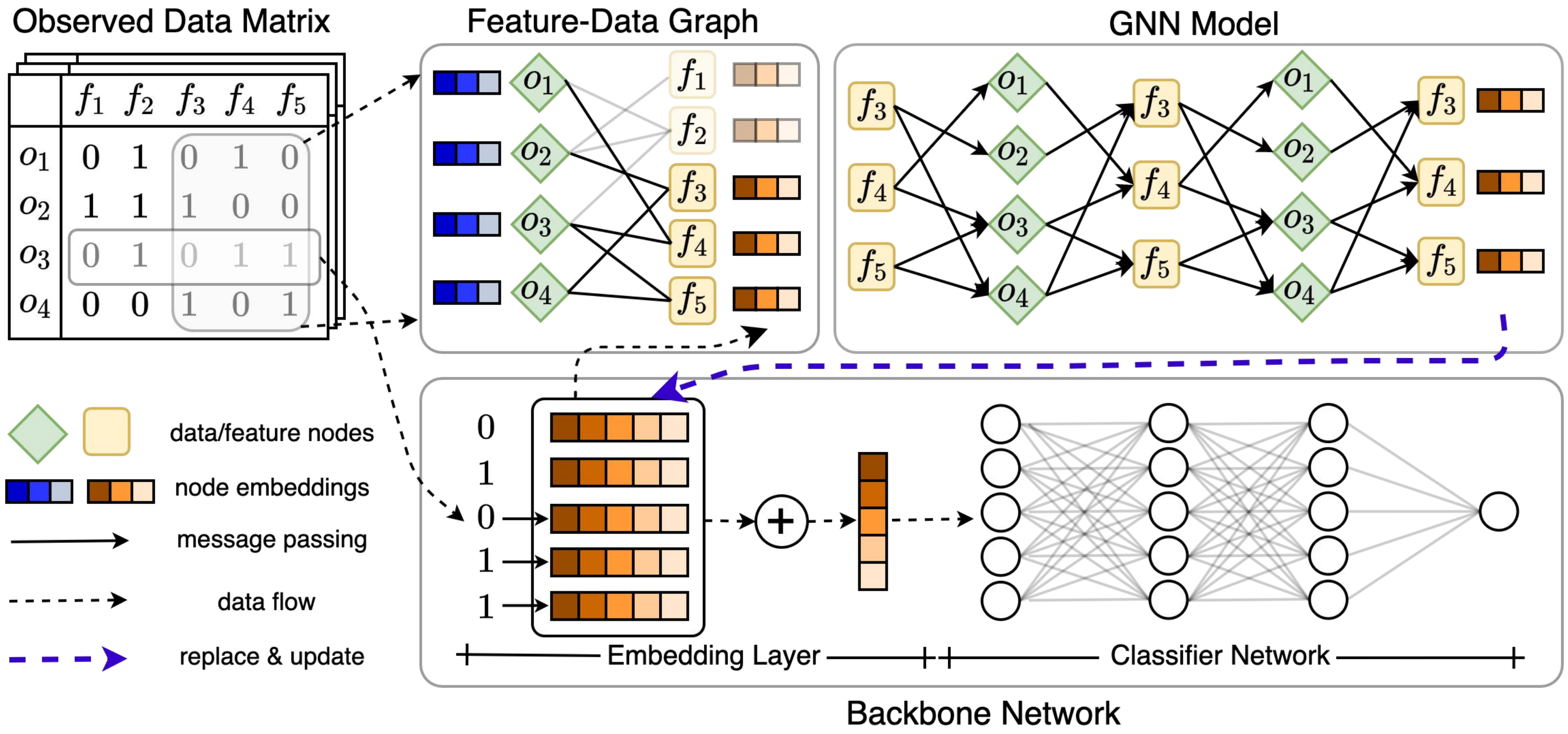
Observed Data Matrix

	f_1	f_2	f_3	f_4	f_5
o_1	0	1	0	1	0
o_2	1	1	1	0	0
o_3	0	1	0	1	1
o_4	0	0	1	0	1

Feature-Data Graph



Proposed Model Framework: FATE



- ❑ **High-level GNN:** take feature-data matrix as input and update feat. embeddings
- ❑ **Low-level backbone:** take each instance as input and output prediction

Details for Proposed Model

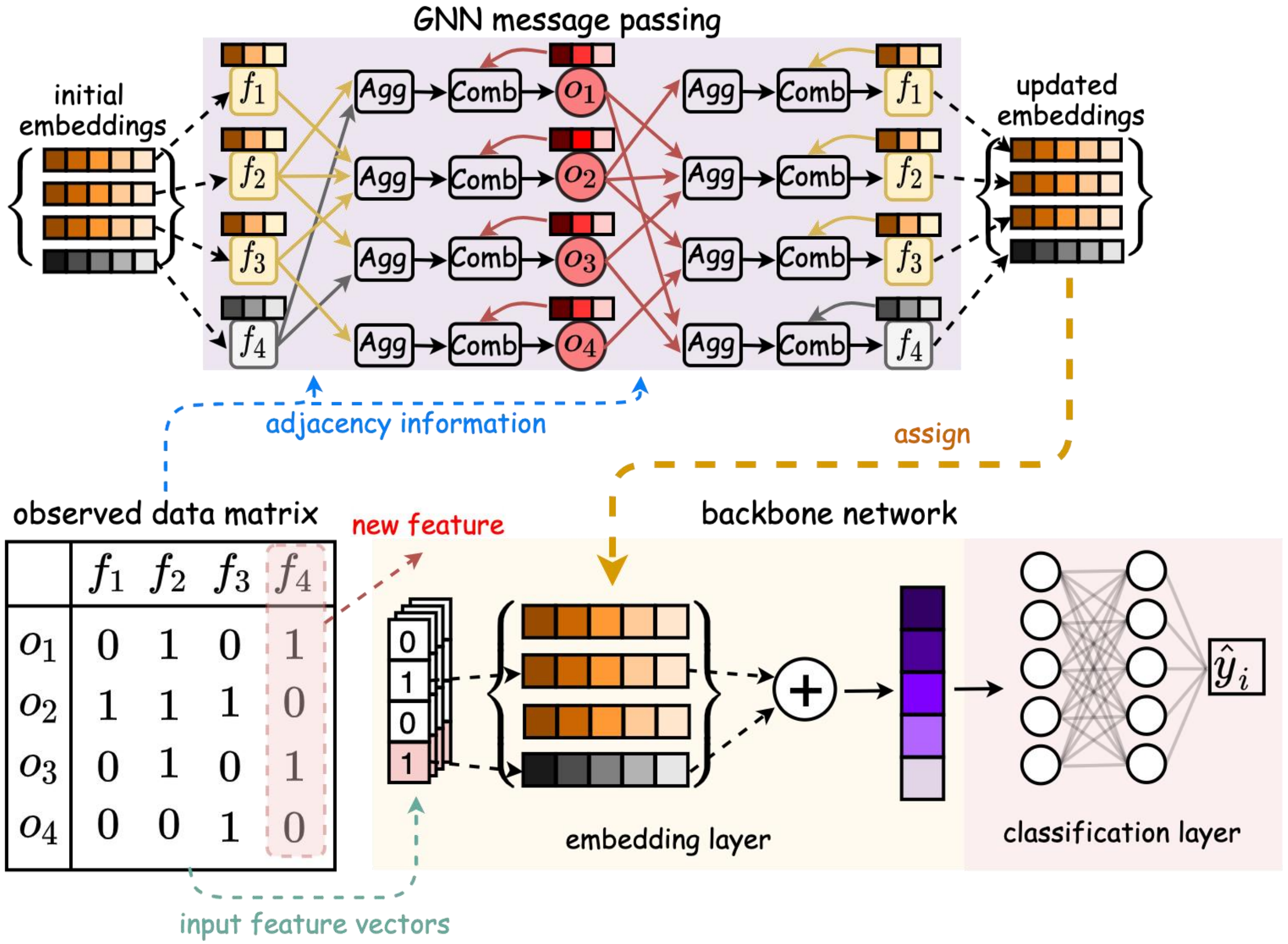
- **GNN model feedforward**
  - Feature nodes $\{\mathbf{w}_j\}_{j=1}^{D}$ (initial embeddings as $\mathbf{w}_j^{(0)}$)
  - Instance nodes $\{\mathbf{s}_i\}_{i=1}^{N}$ (initial states $\mathbf{s}_i^{(0)} = \mathbf{0}$)
  - Message passing rule:

$$\mathbf{a}_i^{(l)} = \textsc{Agg}(\{\mathbf{w}_k^{(l-1)} | \forall k, x_{ik} = 1\})$$

$$\mathbf{s}_i^{(l)} = \mathbf{P}^{(l)} \textsc{Comb}\left(\mathbf{s}_i^{(l-1)}, \mathbf{a}_i^{(l-1)}\right)$$

$$\mathbf{b}_j^{(l)} = \textsc{Agg}(\{\mathbf{s}_k^{(l-1)} | \forall k, x_{jk} = 1\})$$

$$\mathbf{w}_j^{(l)} = \mathbf{P}^{(l)} \textsc{Comb}\left(\mathbf{w}_j^{(l-1)}, \mathbf{b}_j^{(l-1)}\right)$$



Details for Proposed Model

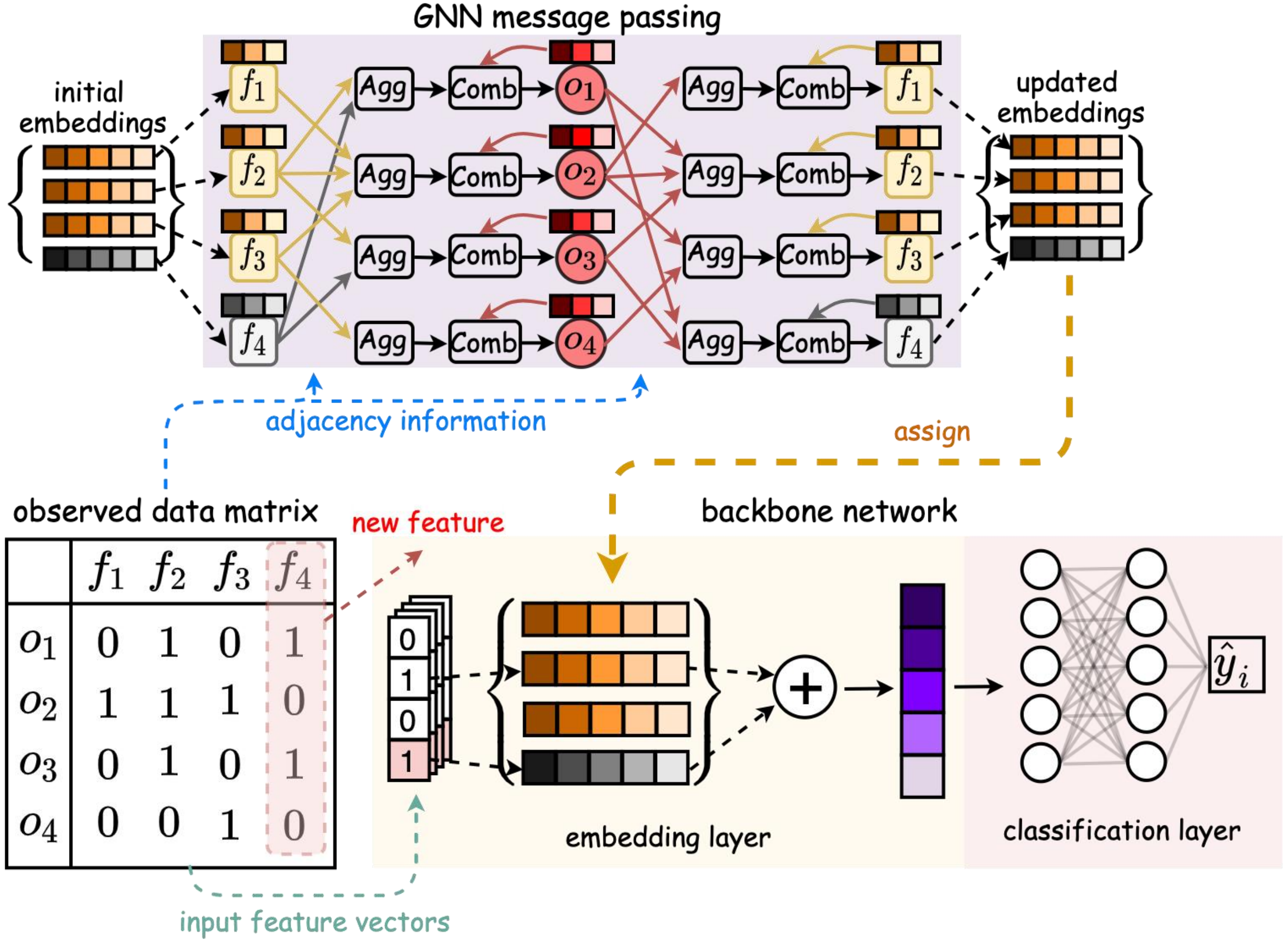
- **Entire feedforward compute**
  - Query feature embeddings
    - For old features: $\mathbf{W}$
    - For new features: set as zero
  - Updata feature embeddings

$$\hat{\mathbf{W}} = [\mathbf{w}_j^{(L)}]_{j=1}^{D} = g(\mathbf{W}, \mathbf{X}; \omega)$$

  - Assign to backbone and output predicted results

$$\hat{y}_i = h(\mathbf{x}_i; \phi, \hat{\mathbf{W}})$$

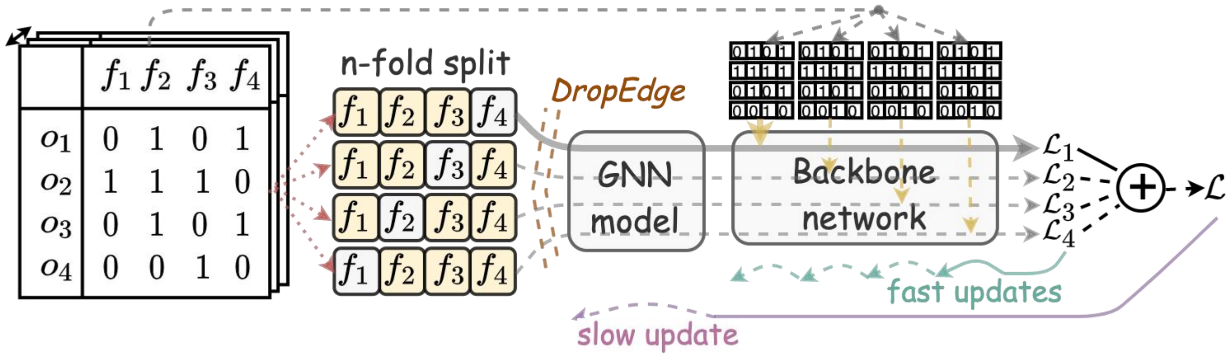
Note: 1) $\mathbf{X}$ can be either training or test data; 2) the permutation-invarance and graph representation enables arbitrarily sized $\mathbf{X}$



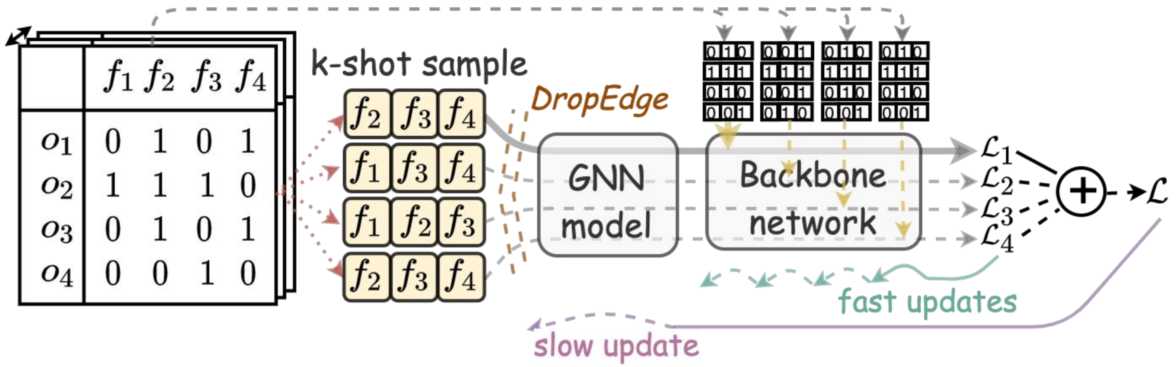
	f_1	f_2	f_3	f_4
o_1	0	1	0	1
o_2	1	1	1	0
o_3	0	1	0	1
o_4	0	0	1	0

Proposed Training Approach

- Two useful techniques for learning to extrapolate
  - Proxy training data
    - Self-supervised learning:
      n-fold splitting input features
    - inductive learning:
      k-shot sampling input features
  - Asynchronous Updates
    - Fast/slow for backbone/GNN
- DropEdge regularization
- Scaling to large systems
  - Mini-batches along the instance dimension (complexity $O(Bd)$)



(a) Self-supervised learning with n-fold splitting



(b) inductive learning with k-shot sampling

Generalization Error Analysis

- **Key aspect:** we treat input data matrix as a whole and the proposed proxy data-based training approach samples data point from

$$\mathcal{S} = \{(\mathbf{X}_1, Y_1), (\mathbf{X}_2, Y_2), \cdots, (\mathbf{X}_M, Y_M)\} \quad \text{where } M \propto \mathcal{O}\left(\frac{d!}{(d-k)!k!}\right)$$

- The empirical risk over training data

$$R_{emb}(h_\mathcal{S}) = \frac{1}{M}\sum_{m=1}^{M} \mathcal{L}(Y_m, h(\mathbf{X}_m; \psi_\mathcal{S}))$$

- The generalization error can be defined as

$$R(h_\mathcal{S}) = \mathbb{E}_{(\mathbf{X},Y)}[\mathcal{L}(Y, h(\mathbf{X}; \psi_\mathcal{S}))]$$

- We care about expected generalization gap over random sampling

$$\mathbb{E}_A[R(h_\mathcal{S}) - R_{emp}(h_\mathcal{S})]$$

Generalization Error Analysis (Cont.)

- Theorem. Assume the loss function is bounded by $l(y_i, \hat{y}_i) \leq \lambda$. For a learning algorithm trained on data $\{\mathbf{X}_{tr}, Y_{tr}\}$ with T iterations of SGD updates, with probability at least $1-\delta$, we have

$$\mathbb{E}_A[R(h_\mathcal{S}) - R_{emp}(h_\mathcal{S})] \leq \mathcal{O}(\frac{d^T}{M}) + \left(\mathcal{O}(\frac{d^T}{M^2} + \lambda)\sqrt{\frac{\log(1/\delta)}{2M}}\right)$$

where $M \propto \mathcal{O}\left(\frac{d!}{(d-k)!k!}\right)$ and k denotes the size of sampled features

Note: 1) The generalization gap depends on the number of raw features, i.e. d

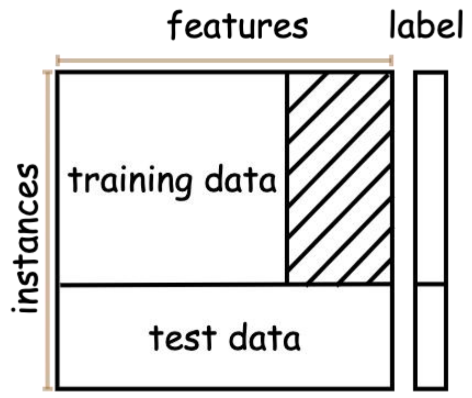
2) The size M is determined by the configuration of proxy training data.

(If there is more randomness, then M would be larger)

Is larger M always better? No! larger variance and larger optimization error

Experiments on UCI Datasets

Dataset	Domain	#Instances	#Raw Feat.	Cardinality	#0-1 Feat.	#Class
Gene	Life	3190	60	4∼6	287	3
Protein	Life	1080	80	2∼8	743	8
Robot	Computer	5456	24	9	237	4
Drive	Computer	58509	49	9	378	11
Calls	Life	7195	10	4∼10	219	10
Github	Social	37700	-	∼	4006	2



- **Evaluation:** training on observed features and testing on all features
  - Instance-level: random split all the instances into training/validation/test data
  - Feature-level: random split all the features into observed/unobserved features
- **Baselines/Competitors:**
  - Base (use observed features for tr/te), Oracle (use all features for tr)
  - Simple extrapolation approaches: Avg, KNN, Mean pooling
  - Incremental learning (first train on observed feat, then retrain on unobserved)
- **Implementation:** 3-layer NN as backbone, 4-layer GNN

Experiments on UCI Datasets

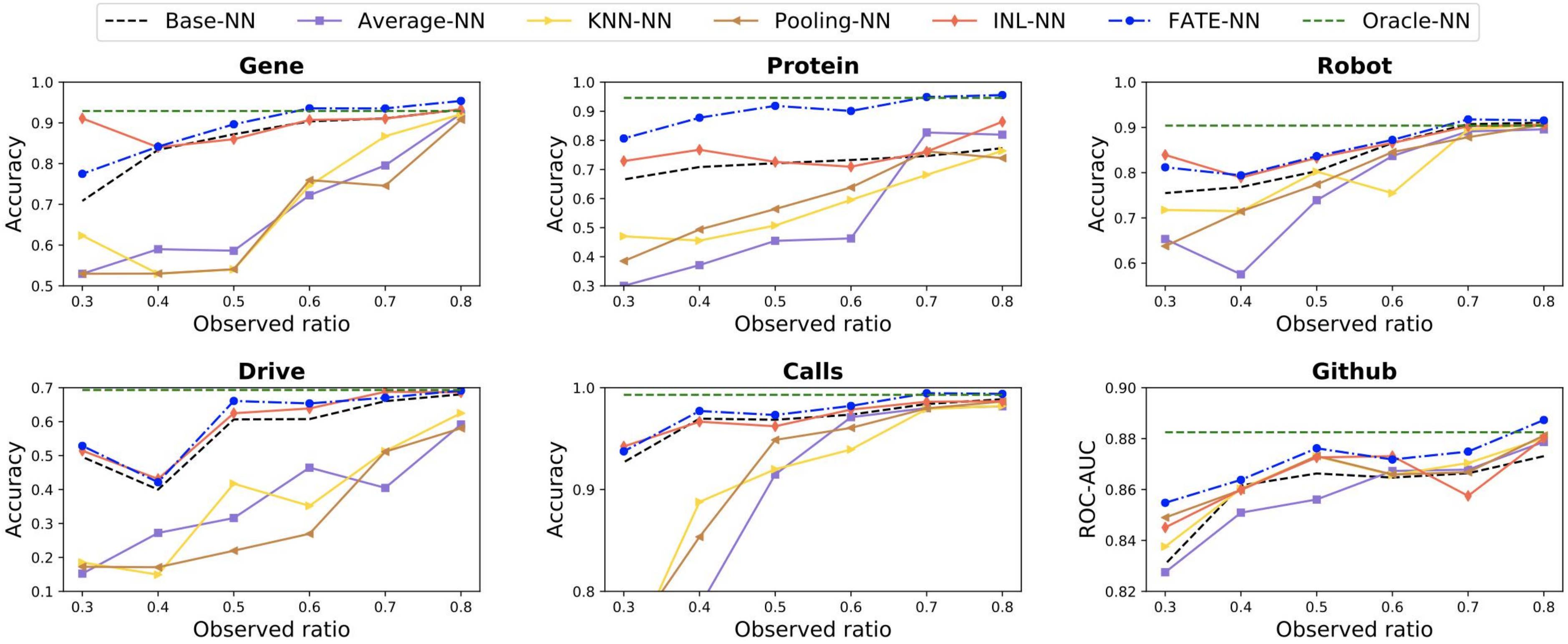


Figure. Accuracy/ROC-AUC results w.r.t. different ratios for observed features

Resutls: 1) FATE (ours) yields 7.3% higher acc. and 1.3% higher AUC than Base

2) FATE achieves very close performance to Oracle (using all features)

2) FATE produces 29.8% higher acc. than baselines Avg, KNN, Pooling

3) FATE even outperforms INL in most cases with averagely 4.1% impv.

Experiments on UCI Datasets

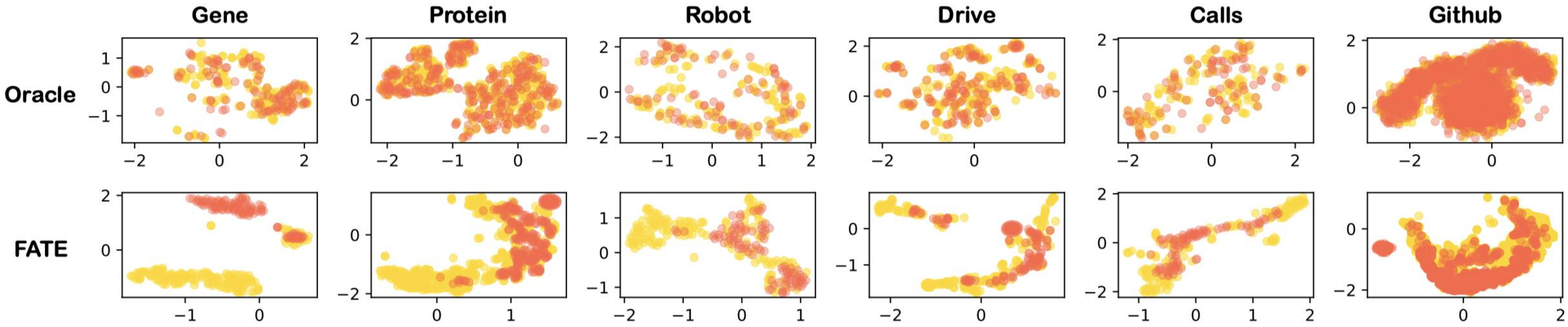


Figure. T-SNE visualization of feature embeddings produced by FATE (ours) and Oracle. Red for observed features and yellow for unobserved ones.

Key insights: 1) FATE's produced embeddings for observed/unobserved features have dissimilar distributions compared to Oracle

➡ *FATE manages to extract some informative knowledge from new features*

2) The embeddings of FATE form some particular structures

➡ *FATE could further capture feature-level relations through GNN interaction*

Experiments on Advertisement Click Prediction

- **Scenario:** click-through rate (CTR) prediction for online advertisement
  - Goal: predict whether a user would click on a displayed ad. (binary classification)
  - Input: attribute features for users/ads
    - Typical features: device id, site id, app id, ad category, app category, etc.
    - The ID features have massive values which induces large feature dimensions

Dataset	Domain	#Instances	#Raw Feat.	Cardinality	#0-1 Feat.	#Class
Avazu	Ad.	40,428,967	22	5~1611749	2,018,025	2
Criteo	Ad.	45,840,617	39	5~541311	2,647,481	2

- **Evaluation:** chronologically split all the instances into 10-fold
  - Use first subset for training, second for validation and the remaining for test
  - ~1.3M/~0.4M/~0.8M exclusive features in training/validation/test data in Criteo
- **Implementation:** 3-layer NN/DeepFM as backbones

Experiments on Advertisement Click Prediction

Table. ROC-AUC results for eight test sets (T1 - T8) on Avazu and Criteo

Dataset	Backbone	Model	T1	T2	T3	T4	T5	T6	T7	T8	Overall
Avazu	NN	Base	0.666	0.680	0.691	0.694	0.699	0.703	0.705	0.705	0.693 ± 0.012
		Pooling	0.655	0.671	0.683	0.683	0.689	0.694	0.697	0.697	0.684 ± 0.011
		FATE	**0.689**	**0.699**	**0.708**	**0.710**	**0.715**	**0.720**	**0.721**	**0.721**	$\mathbf{0.710} \pm 0.010$
	DeepFM	Base	0.675	0.684	0.694	0.697	0.699	0.706	0.708	0.706	0.697 ± 0.009
		Pooling	0.666	0.676	0.685	0.685	0.688	0.693	0.694	0.694	0.685 ± 0.009
		FATE	**0.692**	**0.702**	**0.711**	**0.714**	**0.718**	**0.722**	**0.724**	**0.724**	$\mathbf{0.713} \pm 0.010$
Criteo	NN	Base	0.761	0.761	0.763	0.763	0.765	0.766	0.766	0.766	0.764 ± 0.002
		Pooling	0.761	0.762	0.764	0.763	0.766	0.767	0.768	0.768	0.765 ± 0.001
		FATE	**0.770**	**0.769**	**0.771**	**0.772**	**0.773**	**0.774**	**0.774**	**0.774**	$\mathbf{0.772} \pm 0.001$
	DeepFM	Base	0.772	0.771	0.772	0.772	0.774	0.774	0.774	0.774	0.773 ± 0.001
		Pooling	0.772	0.772	0.773	0.774	0.776	0.776	0.776	0.776	0.774 ± 0.002
		FATE	**0.781**	**0.780**	**0.782**	**0.782**	**0.784**	**0.784**	**0.784**	**0.784**	$\mathbf{0.783} \pm 0.001$

Resutls: FATE achieves significantly improvements over Base/Pooling with different backbones (NN and DeepFM[1])

FATE can extrapolate for unseen features in dynamic data

[1] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He. Deepfm: A factorization-machine based neural network for CTR prediction. In International Joint Conference on Artificial Intelligence, 2017.

Scalability Test for Large Datasets

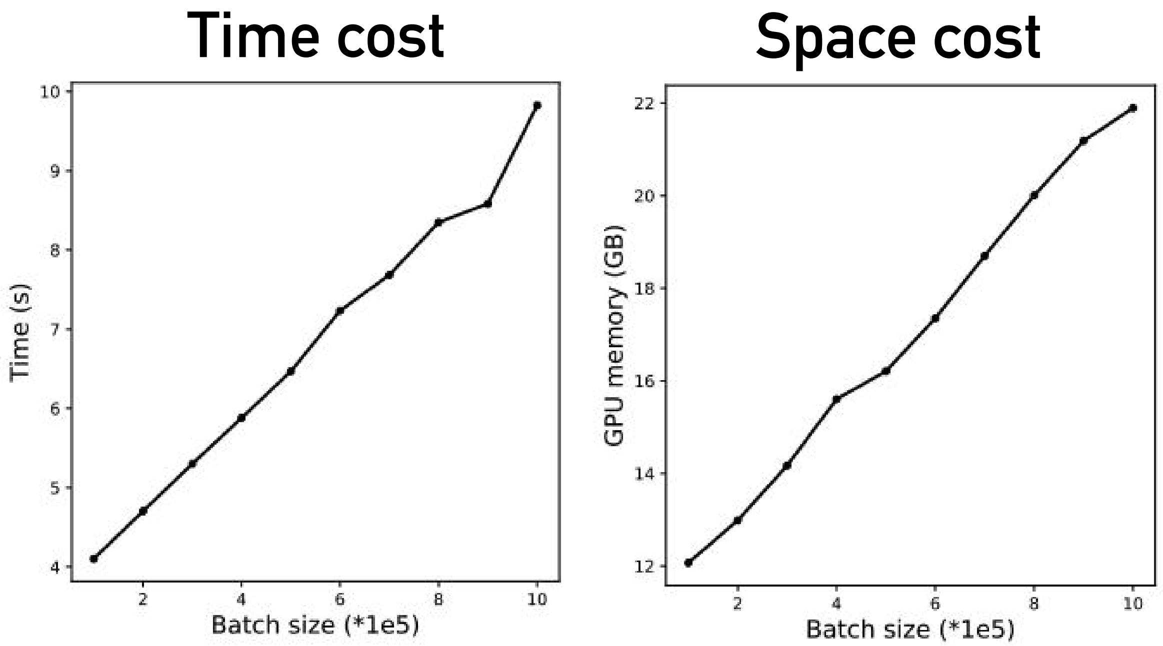


Figure 1. Scalability w.r.t. batch sizes

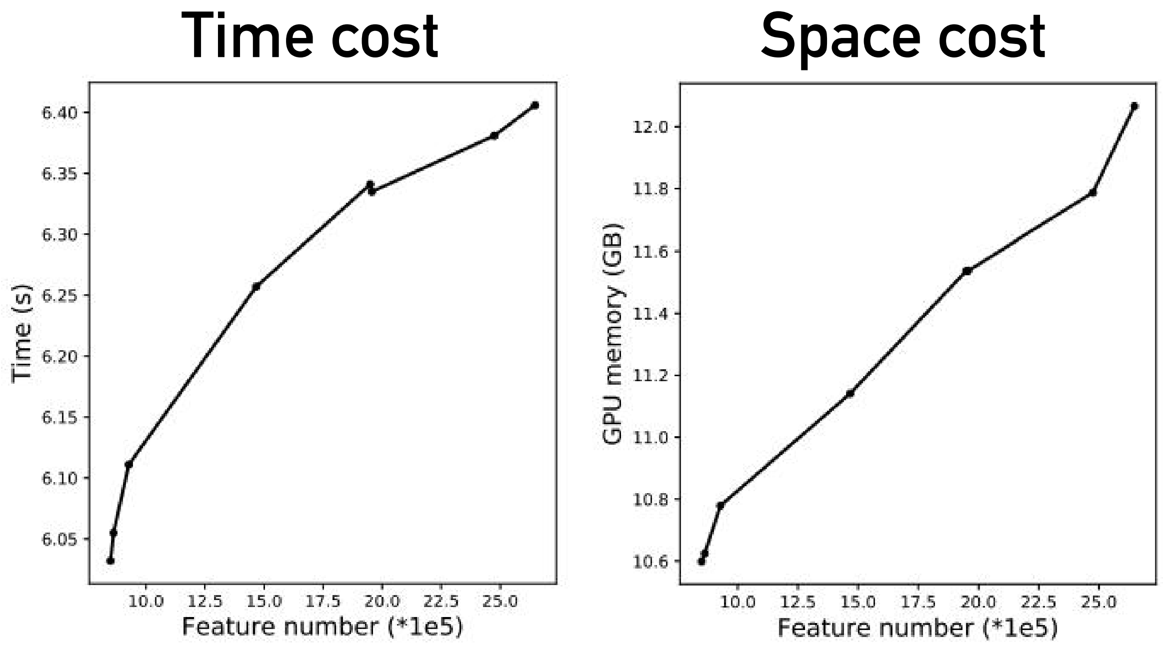


Figure 2. Scalability w.r.t. feature numbers

Resutls: FATE yields linear time/space scalability w.r.t. data and feature sizes

Promising for larger datasets and real systems

The feature-data graph representation and GNN learning induces complexity $O(Bd)$

Comparison with Other Learning Problems

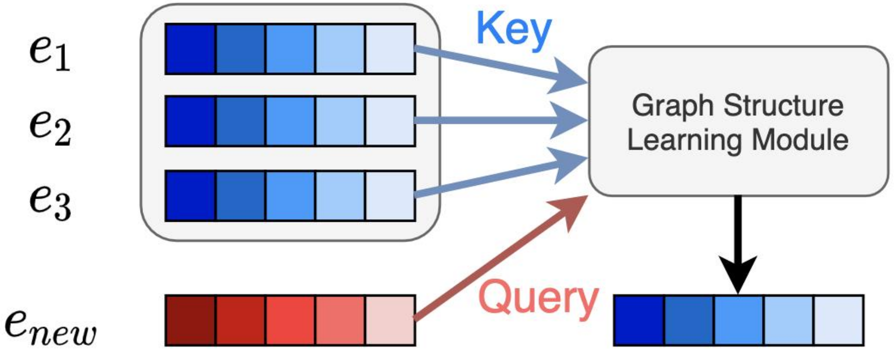
- **Domain Adaption:** source domain → target domain
  - Our differences: 1) same label distribution , 2) one task with different input feature space
- **Continual Learning:** data 1 → data 2 → data 3 → data 4 → …
  - Our differences: 1) one-piece data, 2) no further re-training, 3) one task for learning
- **Open-Set Learning:** training label space → new label class
  - Our differences: 1) same label space, 2) different input feature space
- **Zero-Shot Learning:** observed entities → unseen entities
  - Our differences: 1) no extra side information, 2) different feature space

Conclusions

❑ The main ideas of *open-world recommendation* [ICML'21]:

1) partition entities into two groups

2) learn a latent graph among entities and compute new entities' embeddings using those of existing ones



❑ Potential applications:

- For entity representation extrapolation, e.g. in knowledge graphs
- Transferring embeddings from well-trained concepts to long-tail ones

❑ Future works:

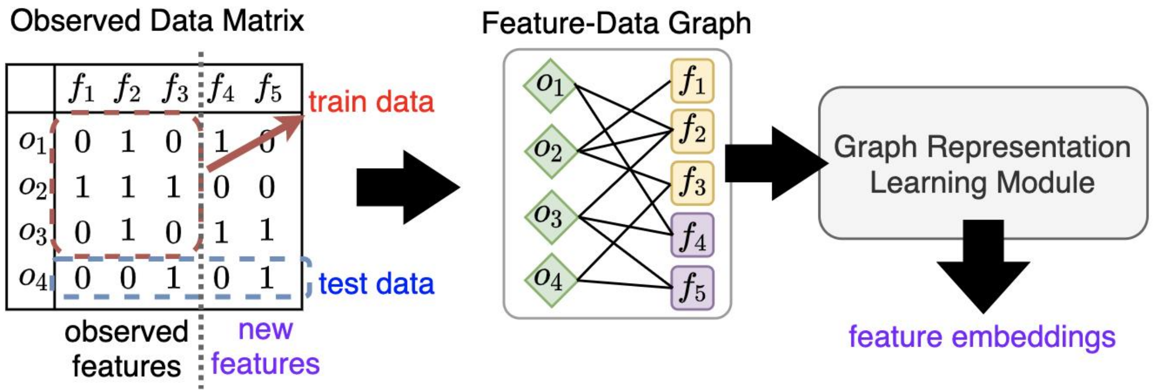
- More expressive architecture, e.g. flow model
- Further theoretical understanding for capacity and generalization

Conclusions

- The main ideas of *open-world feature extrapolation* [NeurIPS'21]:

  1) instance-feature matrix as a graph

  2) convert feature embedding learning to graph representation learning (extrapolation via message passing)



- Potential applications:
  - New attribute features for question answering and reasoning (NLP)
  - Information from new sensors for robot learning and decisions (Robot)
  - Novel drugs or combination for healthcare treatment (Life Science/Healthcare)
  - Extra annotation features for image learning and understanding (Vision)

References

[1] Qitian Wu, Chenxiao Yang, Junchi Yan, Towards Open-World Feature Extrapolation: An Inductive Graph Learning Approach, Advances in Neural Information Processing Systems (NeurIPS'21)

[2] Qitian Wu, Hengrui Zhang, Xiaofeng Gao, Junchi Yan, Hongyuan Zha, Towards Open-World Recommendation: An Inductive Model-Based Collaborative Filtering Approach. International Conference on Machine Learning (ICML'21)

[3] Hengrui Zhang, Qitian Wu, Junchi Yan, David Wipf, Philip S Yu, From Canonical Correlation Analysis to Self-supervised Graph Neural Networks, Advances in Neural Information Processing Systems (NeurIPS'21)

Thanks for listening!

contact: echo740@sjtu.edu.cn