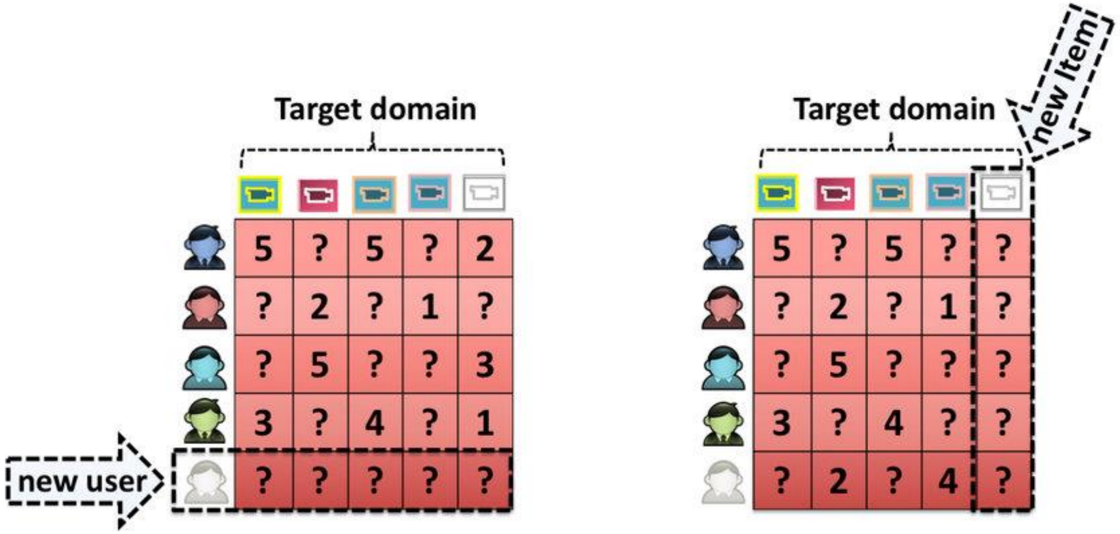
A Review of Cold Start Problem in Recommendation System

Cold Start Problem

- Cold-start (CS) problem: a problem to generate recommendations for a fresh user or to recommend a new item.
- New users, new items, new community (not our focus).
- CS problem, pure CS problem.

pure CS problem: a sub-task of the CS problem, where users have just started to iterate with the systems without any personal information associated.



Categories of Solutions

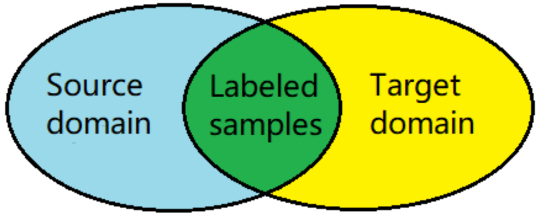
- Integrate side information:
- 1. demographic features; 2. product descriptions; 3. cross-domain recommendation.

- Group recommendation: new users/items can be related to the group they belong to
- Examples: PreHash, AGRRE, HIM

- Meta-learning: learning-to-learn process.
- Examples: MeLU, MAMO

Cross-Domain Recommendation

- SSCDR: Semi-Supervised Learning for Cross-Domain Recommendation to Cold-Start Users, 2019.
- Find a mapping $f_\theta(u_i^s) = u_i^t$, where u and v are the embedding vectors for users and items, respectively, and OU is the overlapping users set.



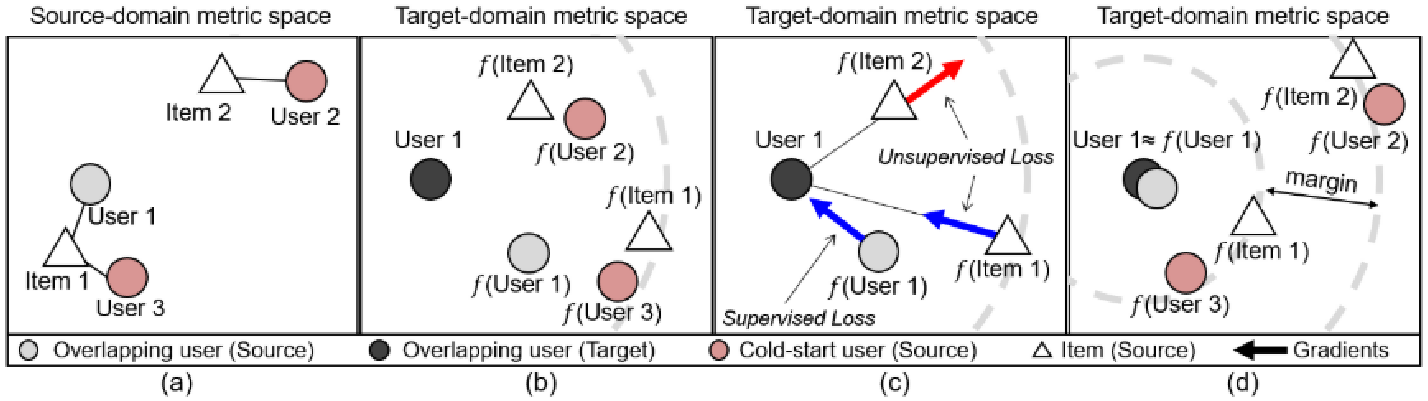
$$\min_\theta \mathcal{L}_S + \lambda \cdot \mathcal{L}_U$$

$$s.t. \quad ||f_\theta(\mathbf{u}_*)||^2 \leq 1 \text{ and } ||f_\theta(\mathbf{v}_*)||^2 \leq 1.$$

- Supervised loss + Unsupervised loss (negative samples):

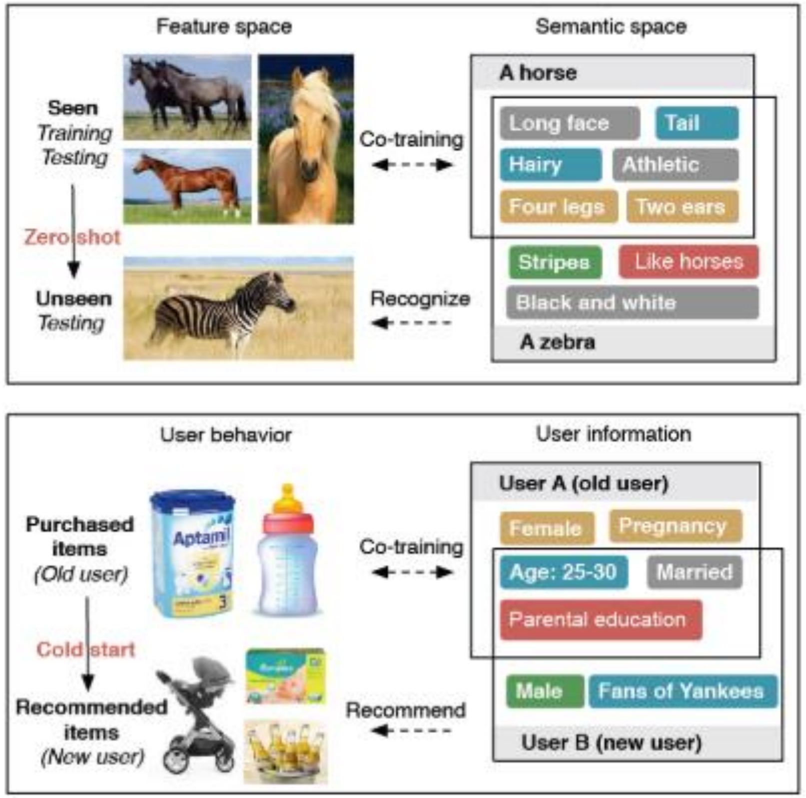
$$\mathcal{L}_S = \sum_{i \in \mathcal{OU}} d\left(f_\theta\left(\mathbf{u}_i^s\right), \mathbf{u}_i^t\right),$$

$$\mathcal{L}_U = \sum_{i \in \mathcal{OU}} \sum_{j \in \mathcal{NI}_i^s} \sum_{k \notin \mathcal{NI}_i^s} \left[m + d\left(f_\theta(\mathbf{v}_j^s), \mathbf{u}_i^t\right) - d\left(f_\theta(\mathbf{v}_k^s), \mathbf{u}_i^t\right)\right]_+,$$



Zero-Shot Learning

- From Zero-Shot Learning to Cold-Start Recommendation, 2019
- Zero-shot learning (top) and cold-start (bottom)



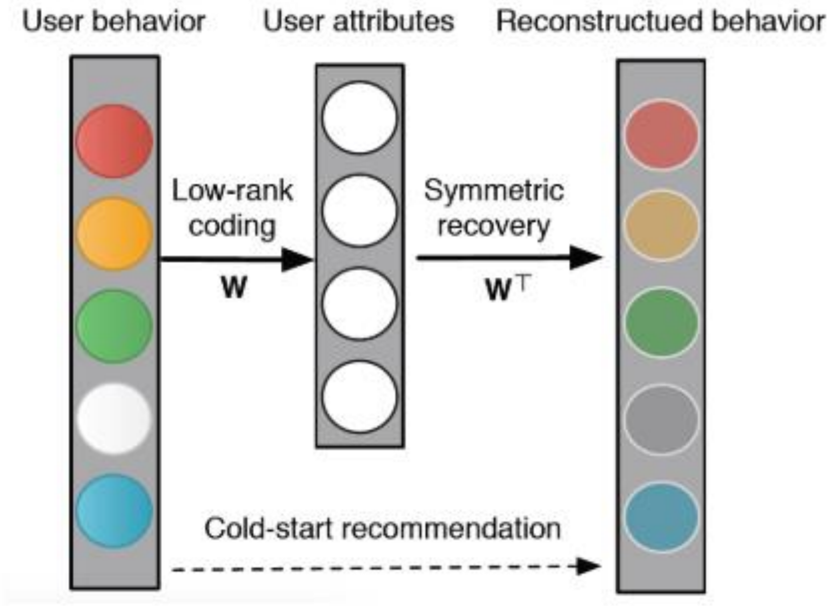
Zero-Shot Learning

- From Zero-Shot Learning to Cold-Start Recommendation, 2019
- X: user behavior
- S: user attributes, like personal information and social network data

$$\min_{\mathbf{W}} \|\mathbf{X} - \mathbf{W}^\top \mathbf{W}\mathbf{X}\|_F^2 + \beta \text{rank}(\mathbf{W}), \quad s.t.\ \mathbf{W}\mathbf{X} = \mathbf{S},$$

- New user: $X_{new} = W^T S_{new}$

- Here a linear auto-encoder is used. Maybe neural auto-encoder will perform better.



Multi-Feature Fusion

- MFDCF: Multi-Feature Discrete Collaborative Filtering for Fast Cold-Start Recommendation, 2020.
- Multi-Feature Fusion:
- Denote user's multiple content features as $X^{(m)}|_{m=1}^{M}$, where $X^{(m)} = \left[x_1^{(m)}, \cdots, x_n^{(m)}\right] \in R^{d_m \times n}$.
- Map $X^{(m)}|_{m=1}^{M}$into a multi-feature representation $H \in R^{r \times n}$ with low-rank constraint on W.

$$\min_{\mu \in \Delta_M, \mathbf{W}^{(m)}, \mathbf{H}} \sum_{m=1}^{M} \frac{1}{\mu^{(m)}} ||\boldsymbol{H} - \boldsymbol{W}^{(m)} \boldsymbol{X}^{(m)}||_F^2 + \gamma rank(\boldsymbol{W}^{(m)})$$

Multi-Feature Fusion

- Collaborative Filtering: get hash codes (embedding) for user and item
- user-item rating matrix $S \in R^{n \times m}$; $b_i \in \{\pm 1\}^r$ denote the binary hash codes for i-th user; $d_j \in \{\pm 1\}^r$ denote the binary hash codes for j-th item;
- $B = [b_1, \cdots, b_n]$; $D = [d_1, \cdots, d_m]$

$$\min_{\boldsymbol{B},\boldsymbol{D}} ||\boldsymbol{S} - \boldsymbol{B}^\top \boldsymbol{D}||_F^2 \quad s.t. \boldsymbol{B} \in \{\pm 1\}^{r \times n}, \boldsymbol{D} \in \{\pm 1\}^{r \times m}$$

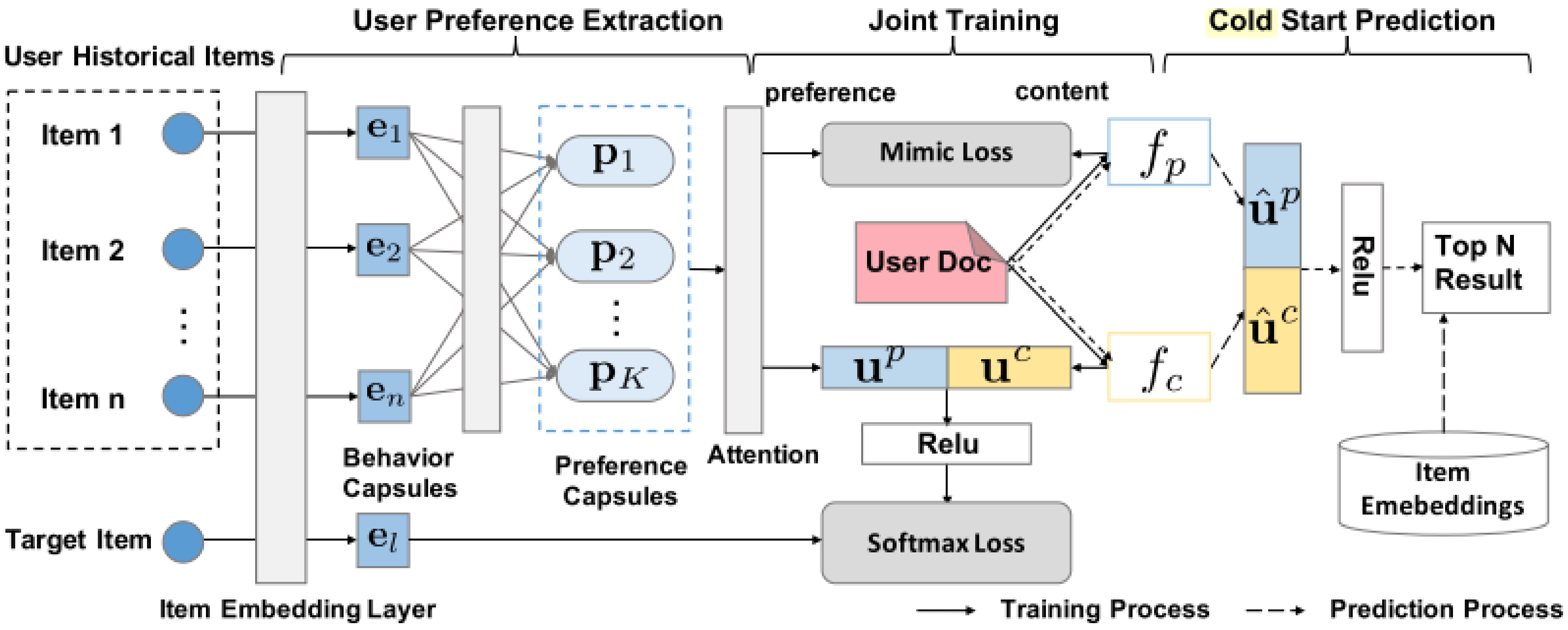
$$\Rightarrow$$

$$\min_{\boldsymbol{B},\boldsymbol{D},\boldsymbol{R}} ||\boldsymbol{S} - \boldsymbol{H}^\top \boldsymbol{R}^\top \boldsymbol{D}||_F^2 + \beta ||\boldsymbol{B} - \boldsymbol{R}\boldsymbol{H}||_F^2 \quad s.t.\ \boldsymbol{B} \in \{\pm 1\}^{r \times n}, \boldsymbol{D} \in \{\pm 1\}^{r \times m}, \boldsymbol{R}^\top \boldsymbol{R} = \boldsymbol{I}_r$$

- For new user, we use $X^{(m)}$ to infer B.
- Linear function can be replaced by more complex function like neural network.
- $W^{(m)} X^{(m)} \rightarrow f(X^{(m)})$; $B^T D \rightarrow f(B, D)$
- The fusion process can be replaced by mixture-of-experts model.

Capsule Network

- Joint Training Capsule Network for Cold Start Recommendation, 2020.



Group recommendation

- Preference Hash (PreHash): Beyond User Embedding Matrix: Learning to Hash for Modeling Large-Scale Users in Recommendation, 2020.

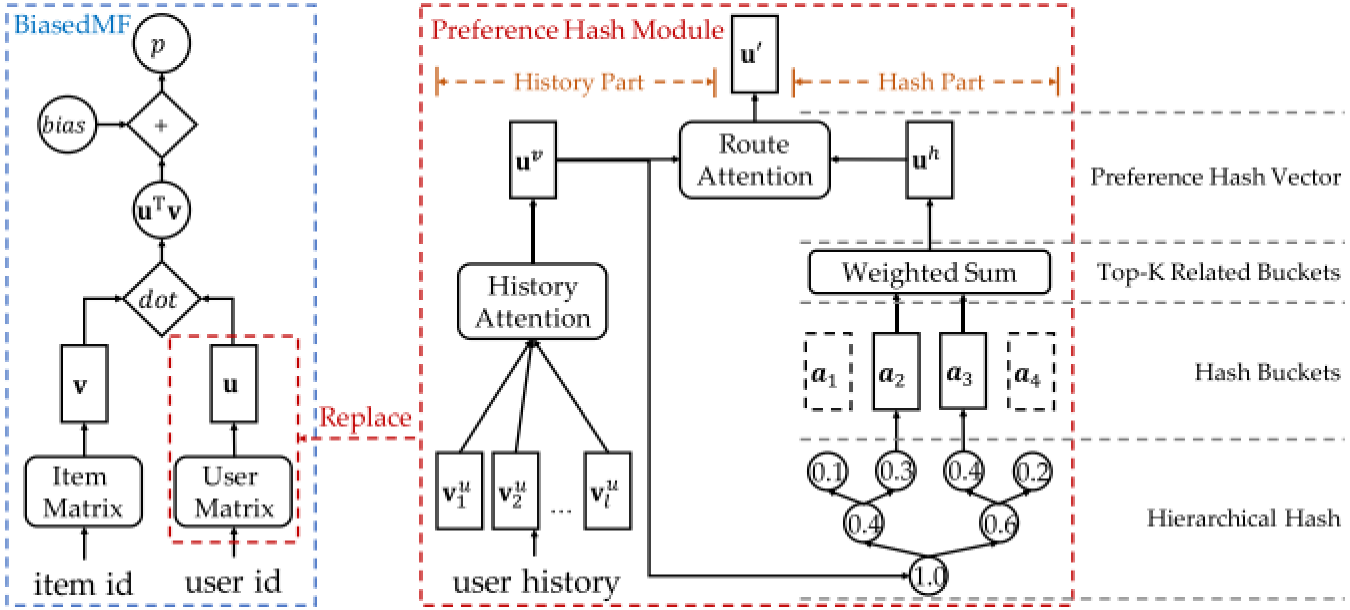


Figure 1: An example of BiasedMF$_{PreHash}$ (PreHash enhanced BiasedMF).

Group recommendation

- In PreHash, the hash vector for new user is a weighted sum of top K related buckets.

$$\mathbf{u}^h = \frac{1}{\sum_{j=1}^{K} r_{t_j}} \sum_{j=1}^{K} r_{t_j} \mathbf{a}_{t_j}$$

- Potential cons: the selection of K; if K is small, u^h may be sensitive to whether a_{t_j} is among the top K or not.

- Prob PreHash:
- 1. A automatic clustering during the process (mixture Gaussian process) →

  for new user: $u^h \sim \sum_{j=1}^{K} w_j \, N(\mu_j, \Sigma_j)$
- 2. User history can be divided into several sessions like long and short term history.
- 3. We can use capsule network to better the embeddings.

Group recommendation

- AGRRE: Attentive Group Recommendation, 2018.

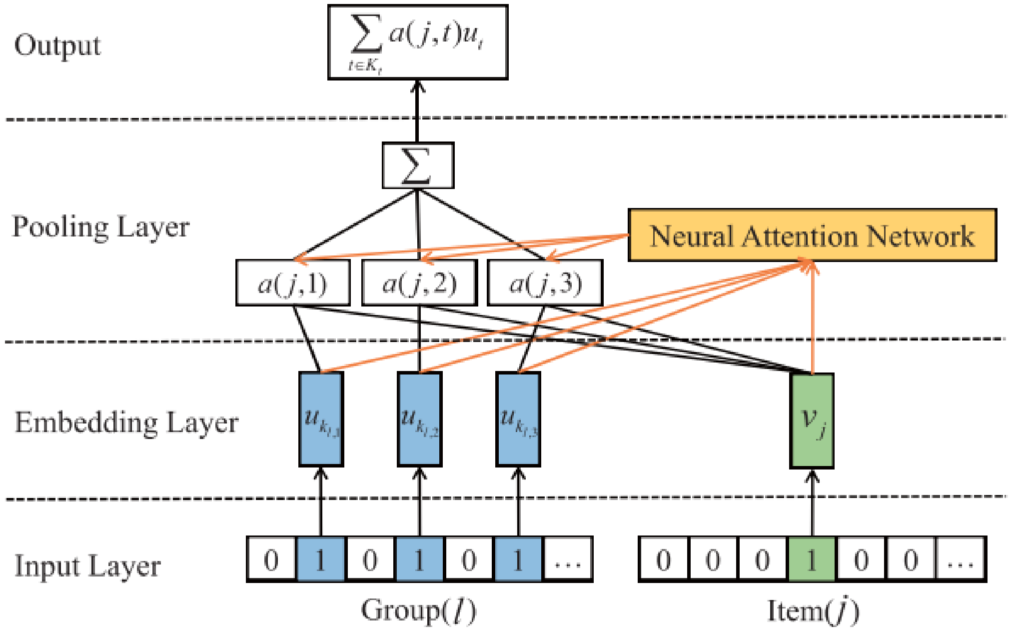


Figure 2: Illustration of the user embedding aggregation component based on neural attention network.

Group recommendation

- AGRRE: Attentive Group Recommendation , 2018.

$$\mathbf{g}_l(j) = \underbrace{\sum_{t \in \mathcal{K}_l} \alpha(j,t)\mathbf{u}_t}_{\text{user embedding aggregation}} + \underbrace{\mathbf{q}_l}_{\text{group preference embedding}}$$

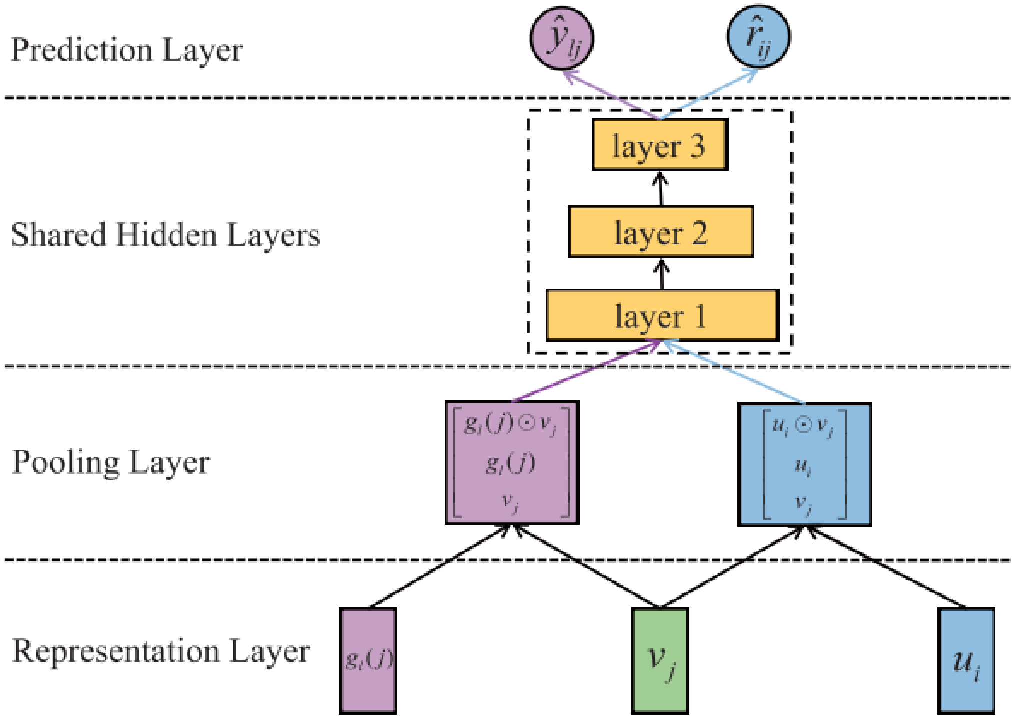
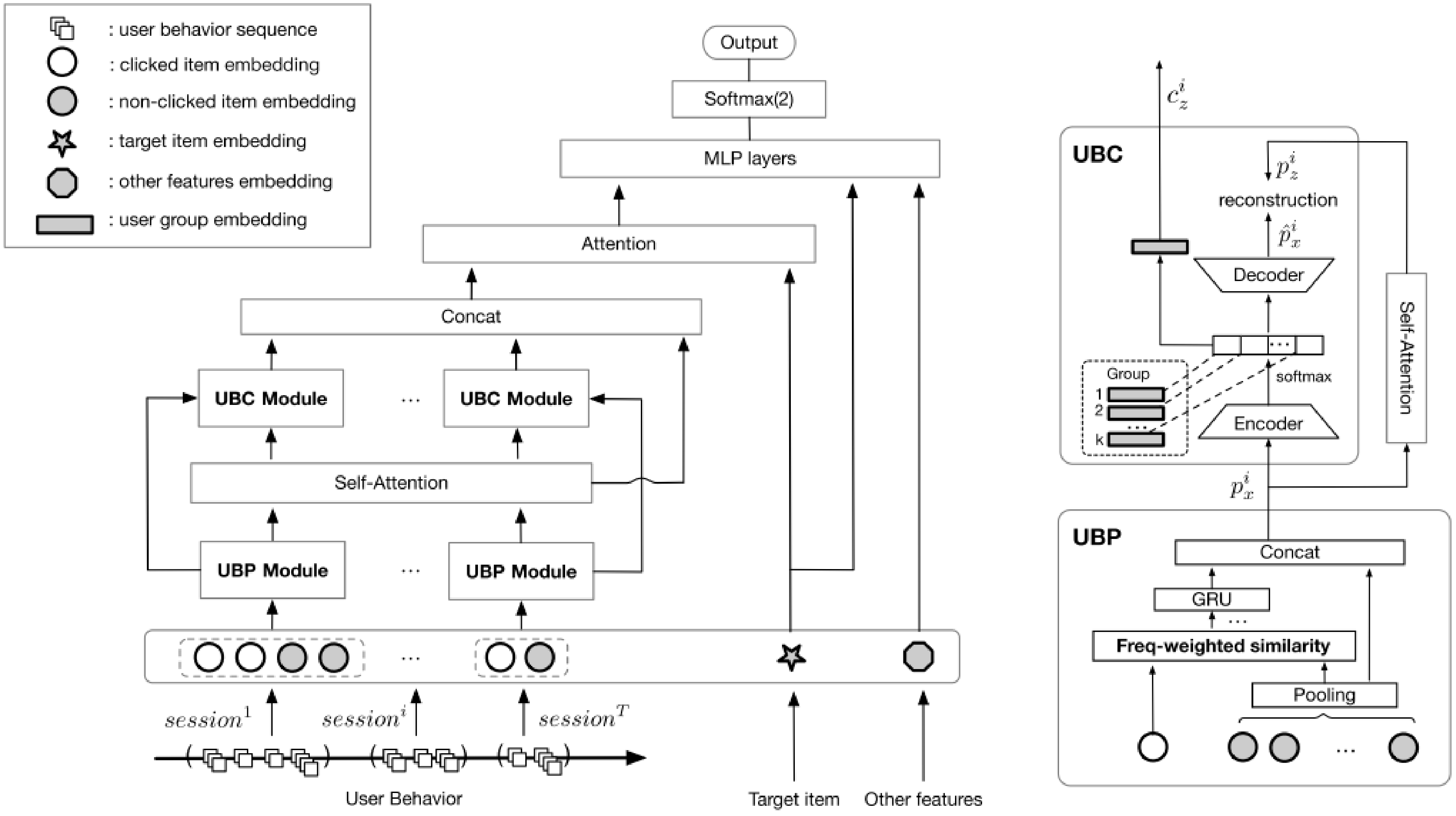


Figure 3: Illustration of interaction learning based on NCF.

Group recommendation

- HIM: Hybrid Interest Modeling for Long-tailed Users, 2020.
- User Behavior Pyramid (UBP) module: to capture the fine-grained personalized interest.
- User Behavior Clustering (UBC) module: to learn latent user interest groups and capture coarse-grained semi-personalized interest.
- Group initialization information is not needed here.

Group recommendation (HIM)



(a) Framework of HIM

(b) Modeling in each session

Meta-learning (learn to learn)

- Traditionally, given $D = \{(x_i, y_i)\}$, we learn $\hat{y} = f_\theta(x)$. This usually works when we have lots of data.
- Can we learn new concepts and skills fast with a few training examples?
- Meta-learning: trained over a variety of learning tasks and optimized for the best performance on a distribution of tasks.
- L is a task sample, S^L is a support set, B^L is a prediction set.
- Maximize:

$$\mathbb{E}_{L\subset\mathcal{L}}[\mathbb{E}_{S^L\subset\mathcal{D},B^L\subset\mathcal{D}}[\sum_{(\mathbf{x},y)\in B^L} P_{g_\phi(\theta,S^L)}(y|\mathbf{x})]]$$

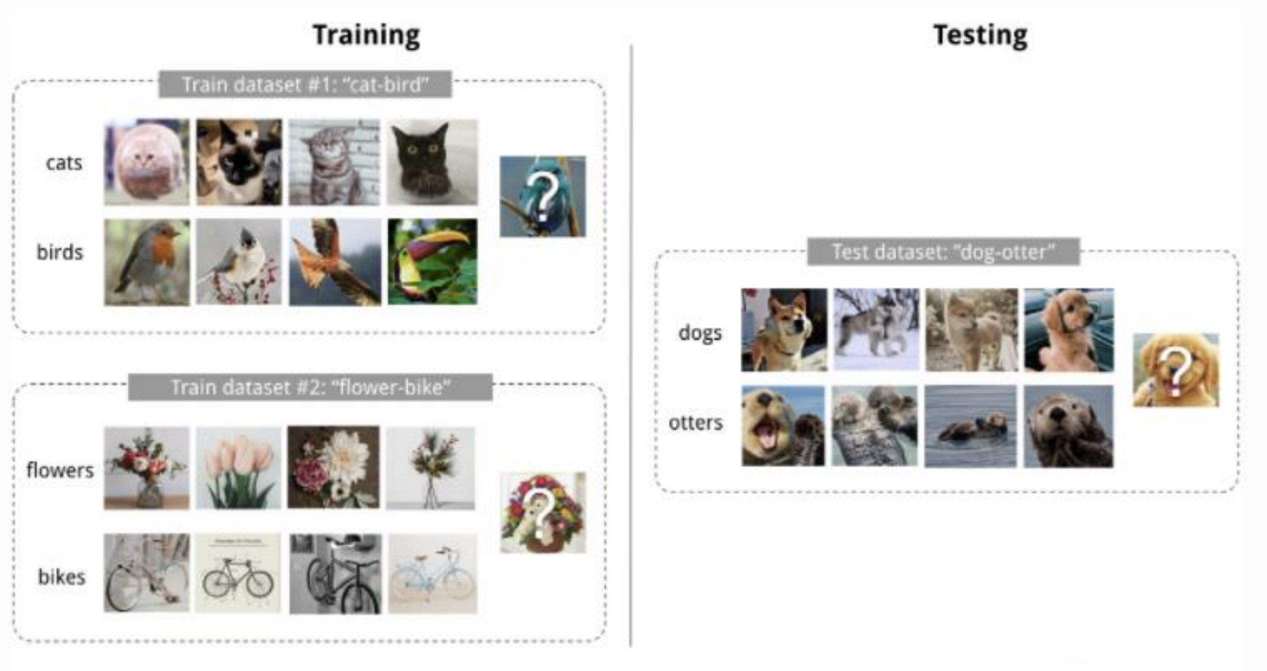
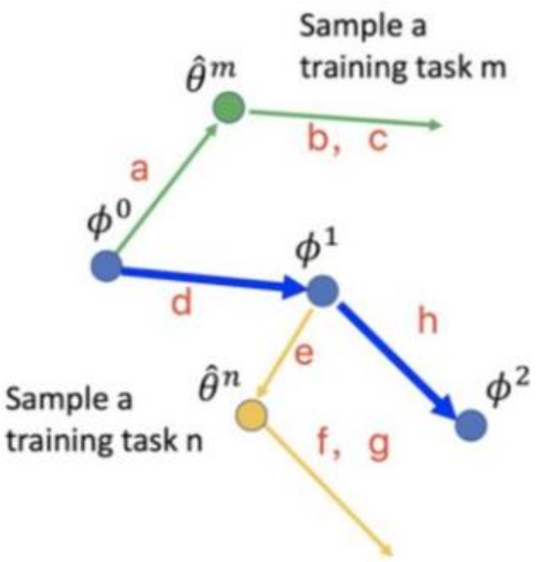


Fig. 1. An example of 4-shot 2-class image classification. (Image thumbnails are from Pinterest)

Meta-learning (optimization-based)

- Optimize the model parameters explicitly for fast learning (good initialization)
- MAML, short for Model-Agnostic Meta-Learning, is compatible with model that learns through gradient descent.
- Local update (ϕ^0 *to* $\hat{\theta}^m$): sample several tasks.
- Global update (ϕ^0 *to* ϕ^1): based on query set.

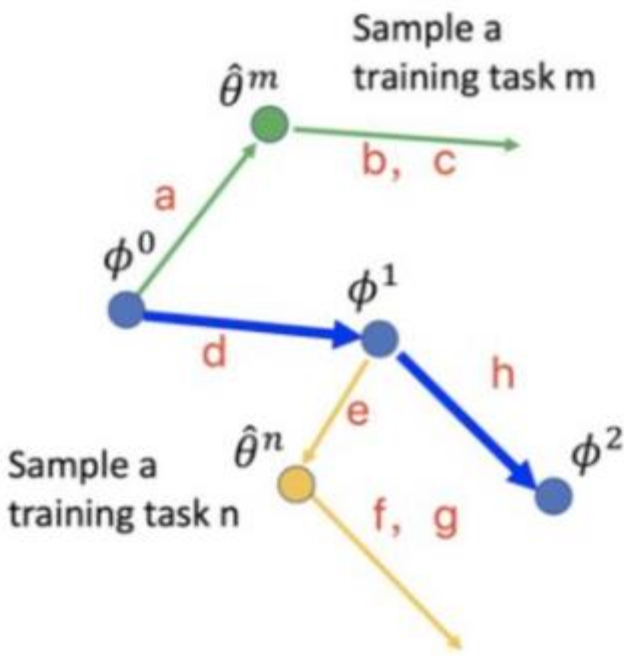


Algorithm 1 Model-Agnostic Meta-Learning

Require: $p(\mathcal{T})$: distribution over tasks
Require: α, β: step size hyperparameters
randomly initialize θ
while not done **do**
  Sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
  for all $\mathcal{T}_i$ **do**
    Evaluate $\nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$ with respect to K examples
    Compute adapted parameters with gradient descent: $\theta_i' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)$
  end for Note: the meta-update is using different set of data.
  Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}(f_{\theta_i'})$
end while

Meta-learning

- MeLU: Meta-Learned User Preference Estimator for Cold-Start Recommendation, 2019.
- MeLU can estimate new user’s preferences with a few consumed items (good initialization for the parameters).
- Each user can be regarded as a task.

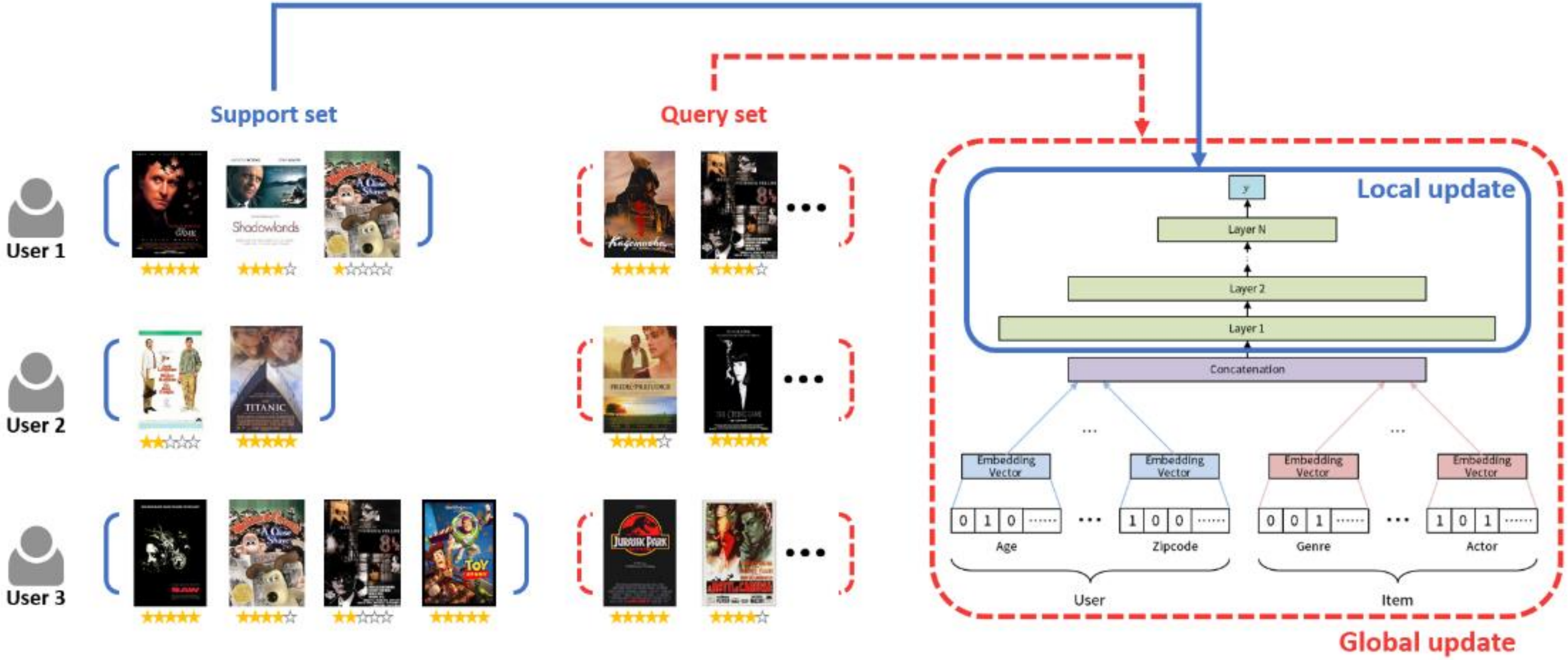


- Items with large gradient are useful for identifying user preferences.
- We recommend not only products that users like but also products that help us understand users. (balance between exploitation and exploration)

  → Bayesian optimization

Meta-learning

- MeLU: Meta-Learned User Preference Estimator for Cold-Start Recommendation



Meta-learning

- MeLU: Meta-Learned User Preference Estimator for Cold-Start Recommendation

Algorithm 1 Model-Agnostic Meta-Learning for User Preference Estimator

Require: α, β: step size hyperparameters

randomly initialize θ_1 (parameters in Eqs. 1 and 2)

randomly initialize θ_2 (parameters in Eq. 3)

while not converge **do**

  sample batch of users $B \sim p(\mathcal{B})$

  for user i in B **do**

    set $\theta_2^i = \theta_2$

    evaluate $\nabla_{\theta_2^i} \mathcal{L}_i(f_{\theta_1, \theta_2^i})$

    local update $\theta_2^i \leftarrow \theta_2^i - \alpha \nabla_{\theta_2^i} \mathcal{L}'_i(f_{\theta_1, \theta_2^i})$

  end for

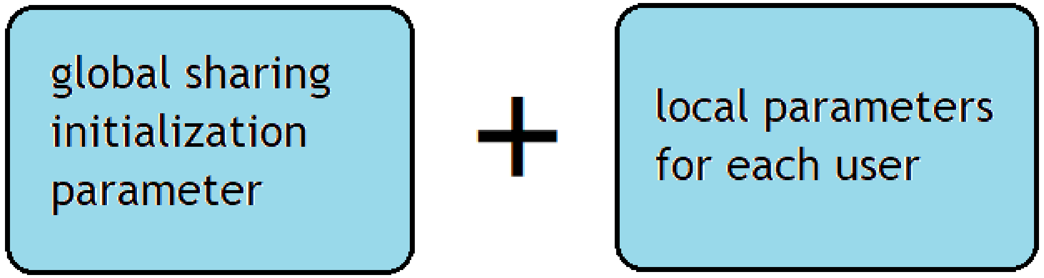
  global update $\theta_1 \leftarrow \theta_1 - \beta \sum_{i \in B} \nabla_{\theta_1} \mathcal{L}'_i(f_{\theta_1, \theta_2^i})$

  $\theta_2 \leftarrow \theta_2 - \beta \sum_{i \in B} \nabla_{\theta_2} \mathcal{L}'_i(f_{\theta_1, \theta_2^i})$

end while

Meta-learning

- MAMO: Memory-Augmented Meta-Optimization for Cold-start Recommendation, 2020
- Global sharing parameter may lead the model into local optima for some users



- 1. Feature-specific memory matric: guide the model with personalized parameter initialization.
- 2. Task-specific memory matric: guide the model fast predicting the user preference.

Meta-learning

- MAMO: Memory-Augmented Meta-Optimization for Cold-start Recommendation
- 1. user profile memory M_P + user embedding memory M_U: providing personalized bias term
- $(P_U, M_U, M_P) \rightarrow b_u$
- $\theta_u \leftarrow \phi_u \ - \tau b_u$
- b_u: guiding the global parameter to fast adapt to the case of user u
- 2. task-specific memory $M_{U,I}$
- We may learn PreHash in a meta-learning scheme.

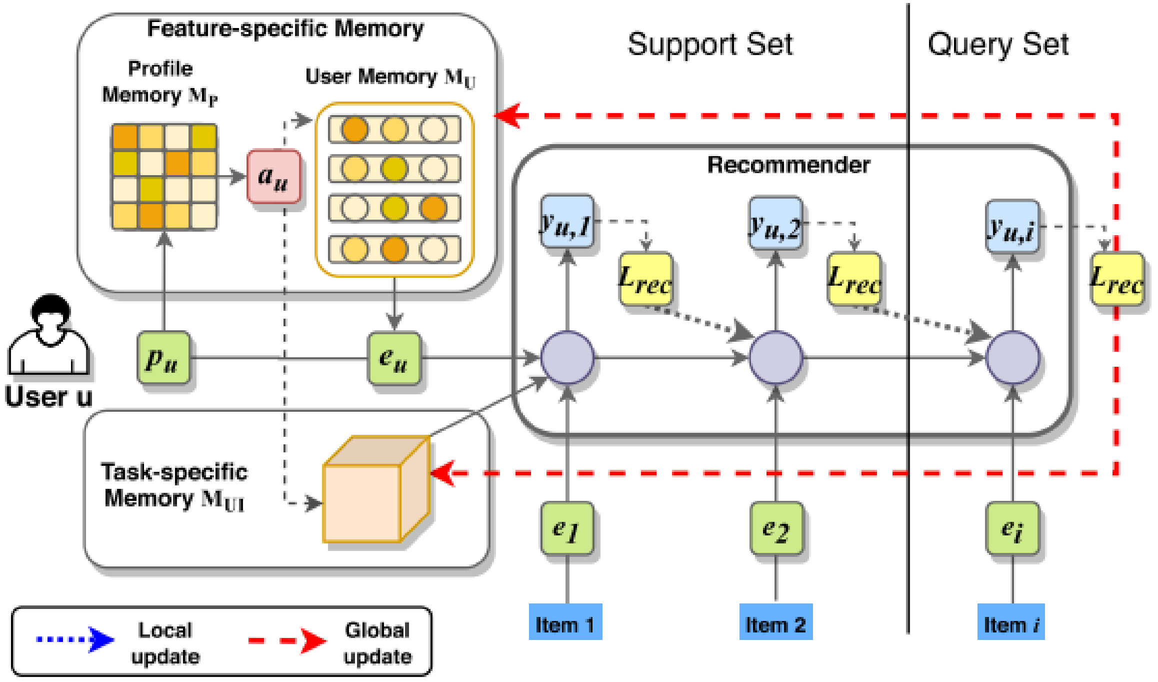


Figure 1: The training phase of MAMO

Pure CS problem

- Pure CS problem: users without historical data
- Knowledge RSs: collect user information using small questionnaires in the first user interaction.
- Social Filtering RSs: external information about users, such as social, demographic and/or personal data.
- Non-personalized RSs: exploit global information about items and users to provide recommendations
- (popularity, recency and positive ratings → max-coverage and category-exploration)