Learning an Adaptive Meta Model-Generator for Incrementally Updating Recommender Systems

Danni Peng[1, 2], Sinno Jialin Pan[1], Jie Zhang[1], Anxiang Zeng[1]

[1]Nanyang Technological University, Singapore

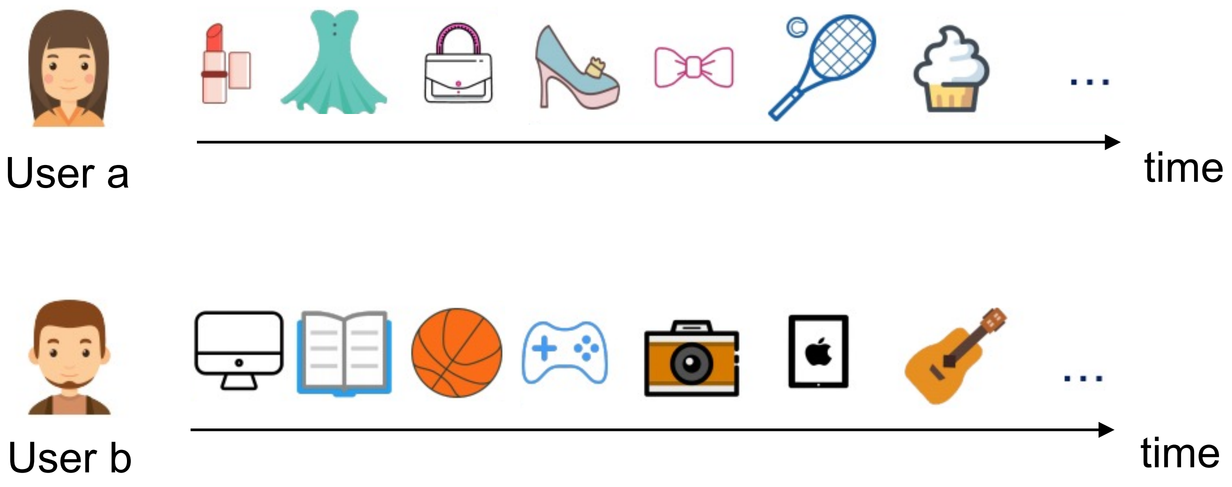
[2]Alibaba-NTU Singapore Joint Research Institute, Singapore

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Alibaba Group

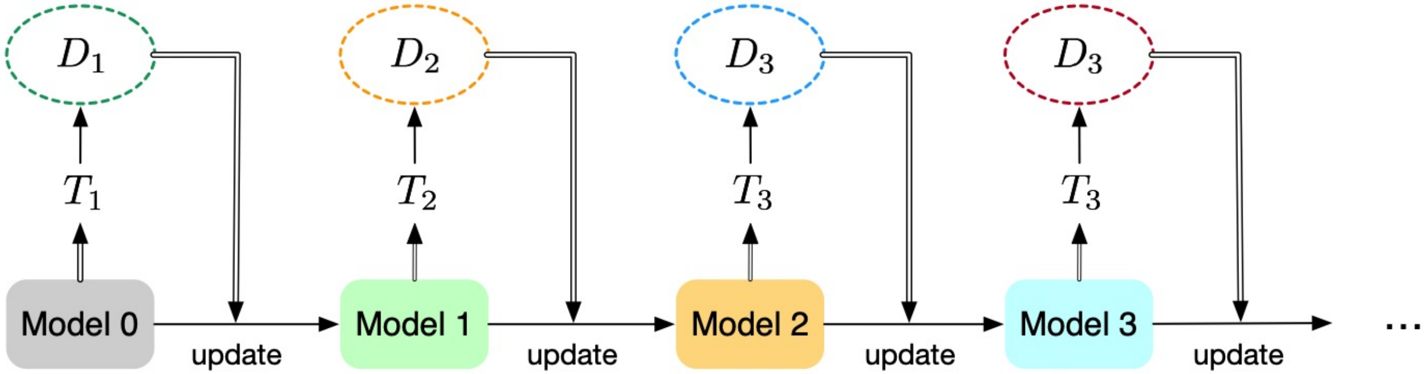
Introduction

- Why it is important to periodically update the Recommender System (RS) model?
  - To capture the latest trends (e.g., interest drift of user, change in item popularity)
  - To better serve/predict for the next period



Introduction

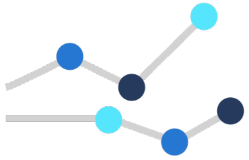
- How to update the RS model?
  - **Incremental Update** – update the model using only the newly arrived data



  - Incremental update is widely adopted in industry due to:
    a) better **training efficiency** – the amount of training data is relatively small
    b) better **prediction accuracy** – the short-term user interests can be well captured

Introduction

- The Problem of **Forgetting** for RS Incremental Update



Limited training data
from the current period

Distribution drift
across periods

Overfit to the current period
&
Fail to retain the long-term memory

- Key Objective:

  Better prediction performance for the next period

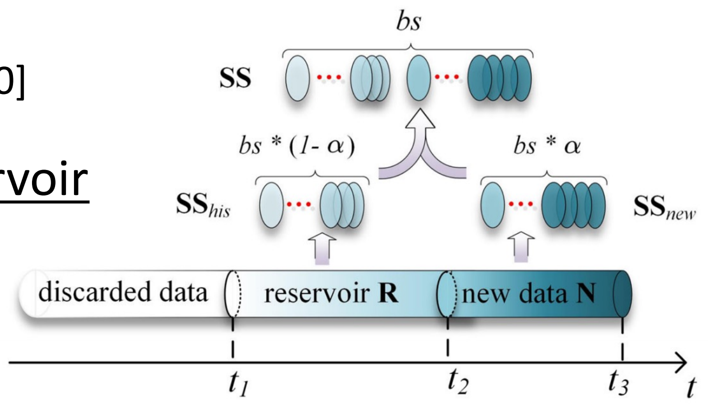
- Key Challenge:

  Extract and retain <u>past knowledge that is especially useful</u> for the next period's predictions

Related Work

- Sample-based Approach [Diaz-Aviles et al. 2012; Wang et al. 2018; Zhao et al. 2020]
  - Reuse the **past samples** for the current training by <u>maintaining a reservoir</u>
  - Key Limitation:

    Individual samples can <u>hardly represent the big picture</u>

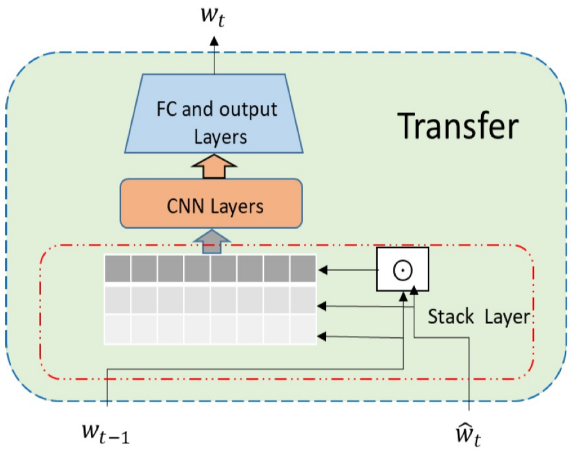


Reservoir Sampling. Figure cited from Zhao et al. 2020

- Model-based Approach [Wang et al. 2020; Mi et al. 2020; Zhang et al. 2020]
  - Extract knowledge from the **past model** via <u>distillation</u> or <u>model fusion</u>
  - Key Limitation:

    Only consider transfer between <u>two consecutive periods</u>

    The potential of <u>long-term sequential information</u> is unexplored



Model Fusion. Figure cited from Zhang et al. 2020

Problem Definition

- Conventional Incremental Update

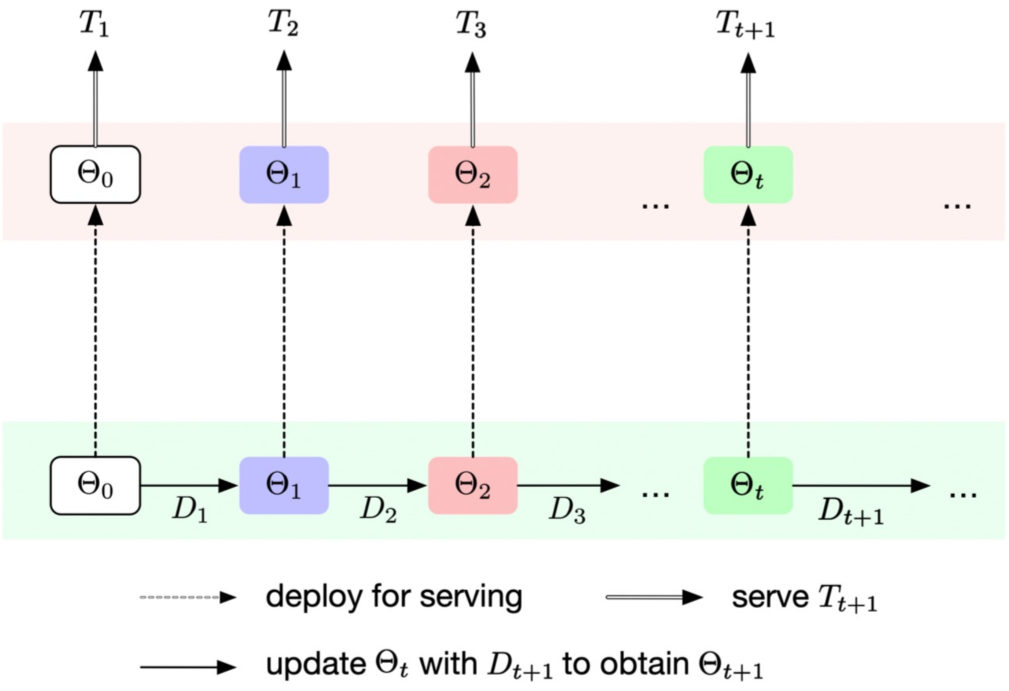
T_t – the t-th period

D_t – the dataset collected from T_t

Θ_t – the model of the t-th period, obtained by minimizing loss on D_t, initializing from Θ_{t-1}:

$$\Theta_t = \arg\min_{\Theta} \mathcal{L}(\Theta | D_t, \Theta_{t-1})$$

- Without any post-processing, the updated model Θ_t is directly deployed to serve for period T_{t+1}
- Suffer from the **overfitting** and **forgetting** issues

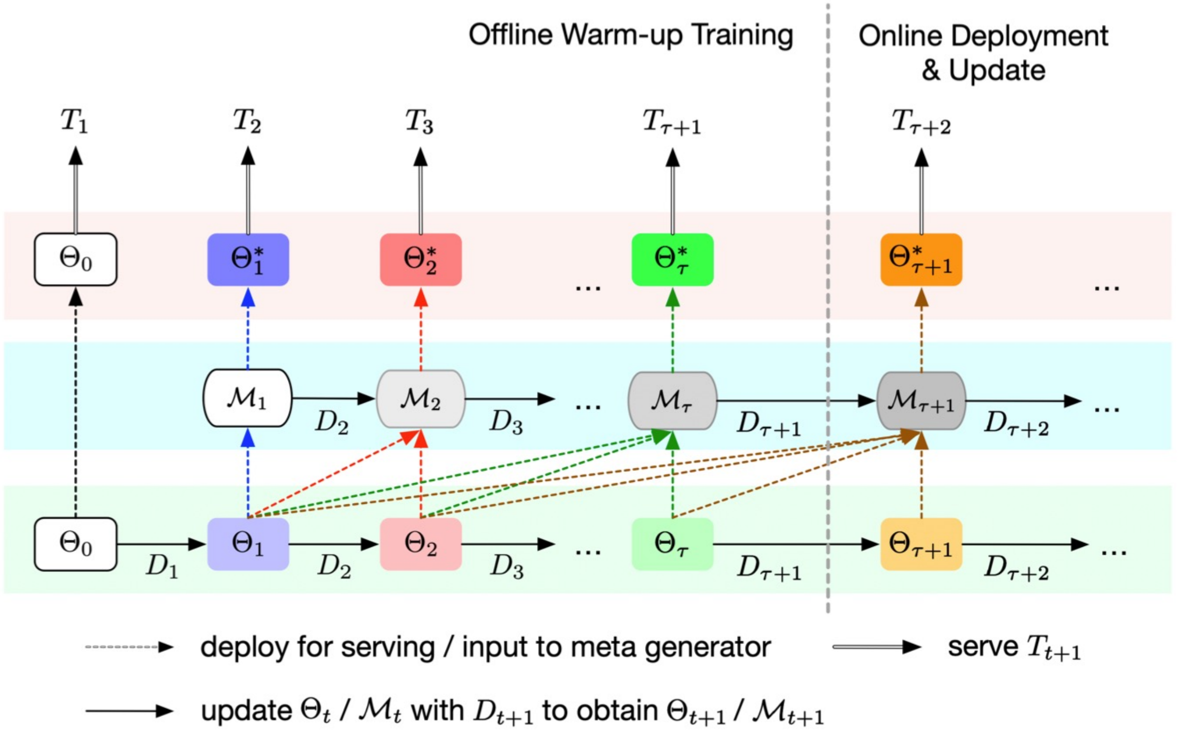


Proposed Method – ASMG Framework

- **A**daptive **S**equential **M**odel **G**eneration (ASMG) Framework

Motivation:

- The sequence of incrementally updated models are <u>highly representative of the respective periods</u>
- <u>Mining the temporal trends exist in this sequence</u> may help to derive a better model for the next period's serving



Proposed Method – ASMG Framework

- **A**daptive **S**equential **M**odel **G**eneration (ASMG) Framework

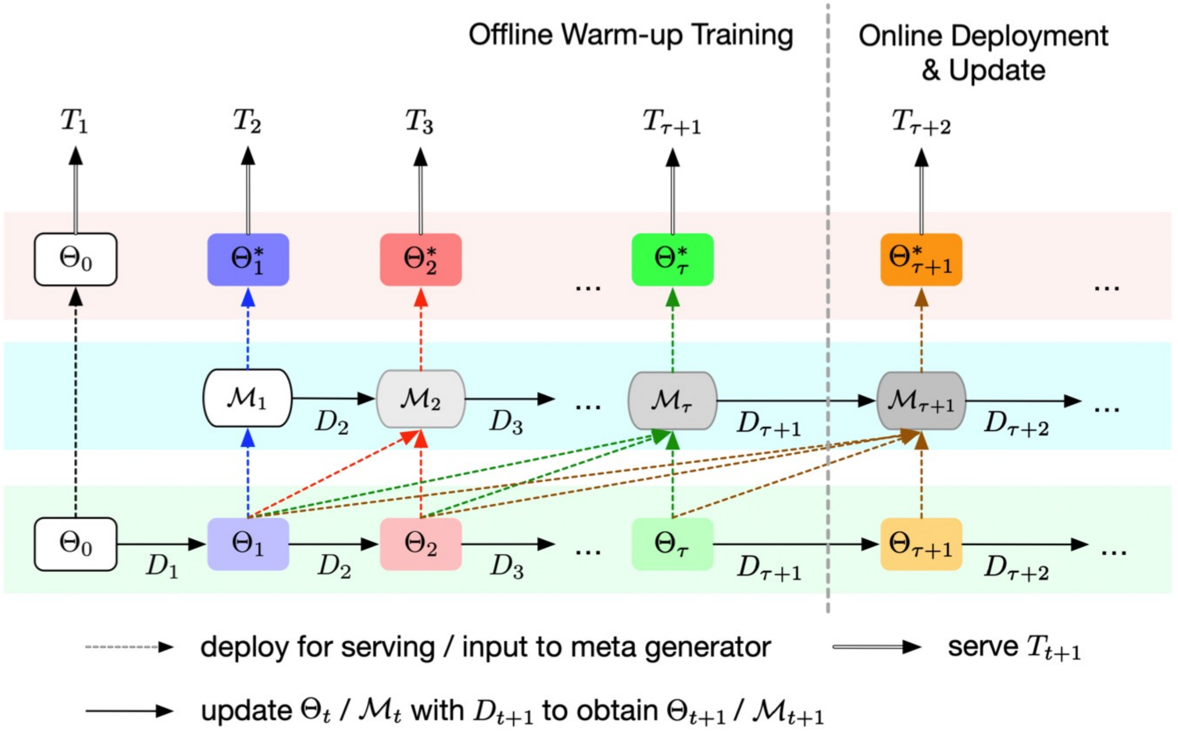
$\mathcal{M}_t$ – the meta model generator at the t-th period

$\boldsymbol{\Theta}_{1:t}$ – the sequence of incrementally updated models

Θ_t^* – the output model used to serve for T_{t+1}

The output model Θ_t^* is generated by:

$$\Theta_t^* = \mathcal{M}_t(\boldsymbol{\Theta}_{1:t})$$



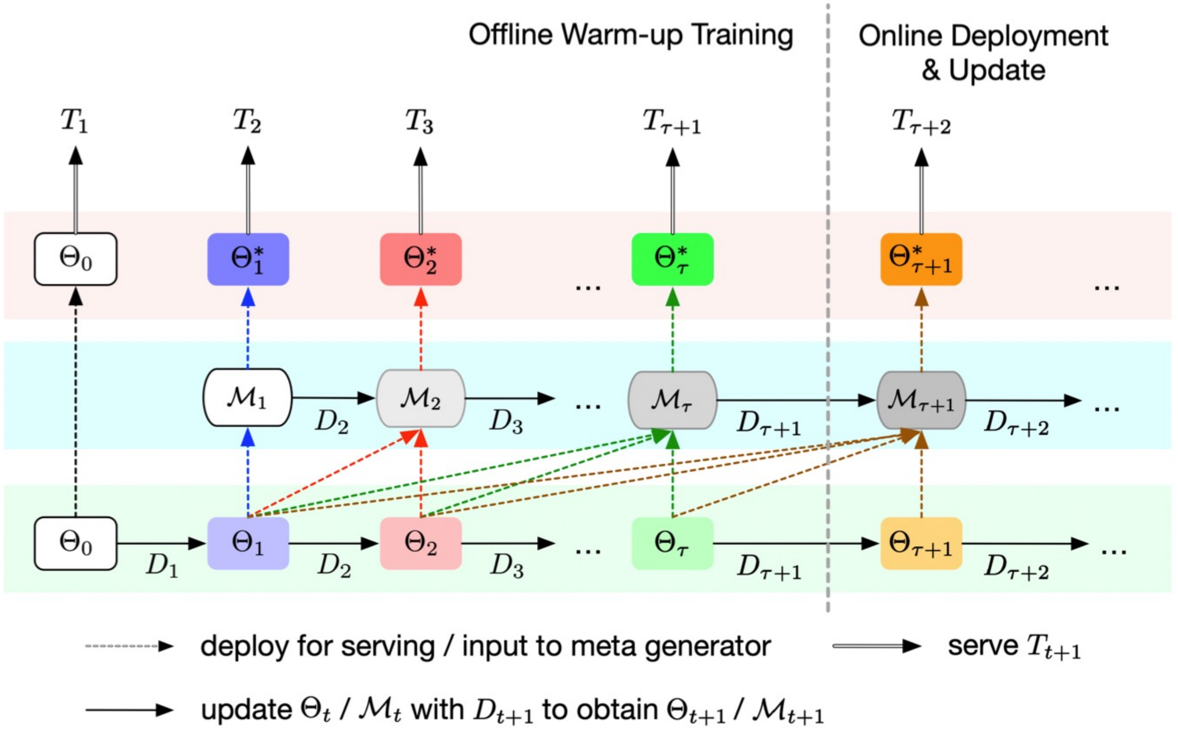
Proposed Method – ASMG Framework

- **A**daptive **S**equential **M**odel **G**eneration (ASMG) Framework

How to design and train the meta generator?

To attain two desired properties:

1. Good capability of sequential modelling
   → Network Design
2. Stable performance of producing a good model for the next period's serving
   → Optimization Process

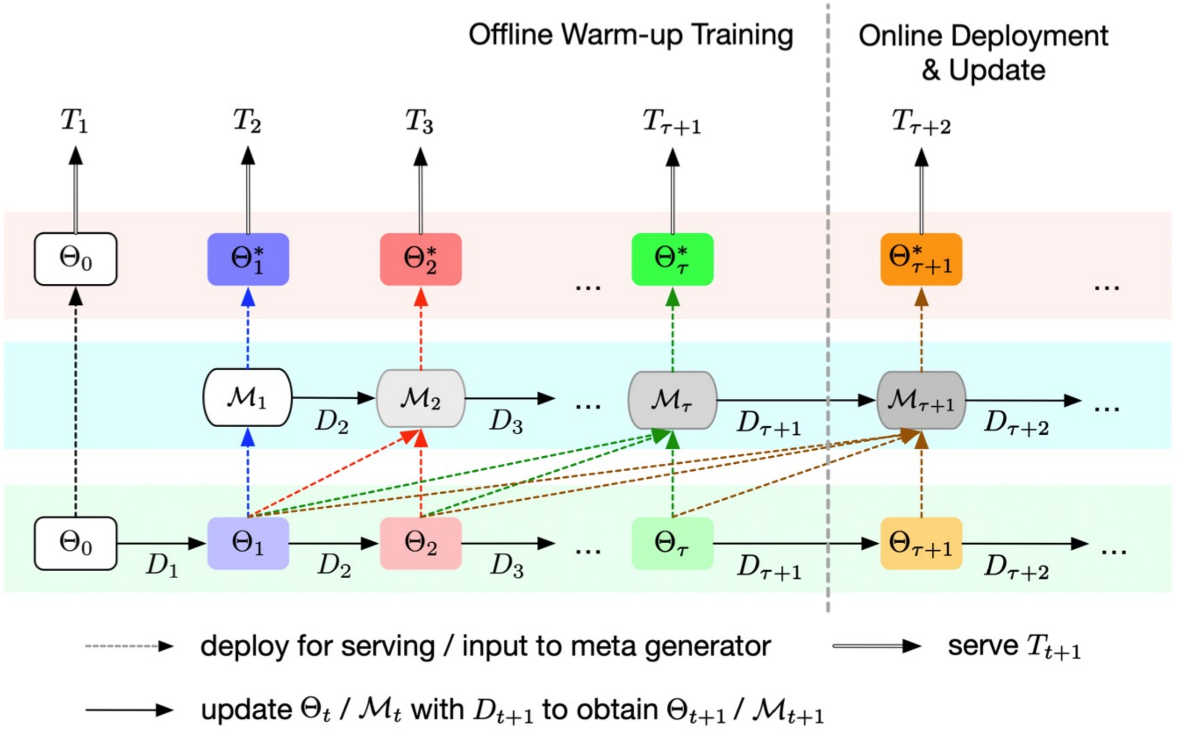


Proposed Method – ASMG Framework

- **A**daptive **S**equential **M**odel **G**eneration (ASMG) Framework

  - To extract past knowledge that is especially useful for the next period's serving
  - We update the parameters of the meta generator ω_t by optimizing the output model towards the next period's data:

$$\omega_{t+1} = \arg\min_{\omega} \mathcal{L}(\Theta_t^*|D_{t+1}, \omega_t)$$
$$= \arg\min_{\omega} \mathcal{L}(\mathcal{M}_\omega(\mathbf{\Theta}_{1:t})|D_{t+1}, \omega_t)$$



Proposed Method – GRU Meta Generator

- GRU Meta Generator – Network Design
  - At each step $i \in \{1, \dots, t\}$, the hidden state $h_i \in \mathbb{R}^d$ is computed from the previous step hidden state $h_{i-1} \in \mathbb{R}^d$ and $\theta \in \mathbb{R}$ of the current step input model $\Theta_i \in \mathbb{R}^n$:

$$
\begin{aligned}
r_i &= \sigma(W_r \cdot [h_{i-1}, \theta]), \\
z_i &= \sigma(W_z \cdot [h_{i-1}, \theta]), \\
\tilde{h}_i &= \tanh(W_{\tilde{h}} \cdot [r_i \odot h_{i-1}, \theta]), \\
h_i &= (1 - z_i) \odot h_{i-1} + z_i \odot \tilde{h}_i,
\end{aligned}
$$

  - The output parameter $\theta^* \in \mathbb{R}$ of the final serving model $\Theta_i^* \in \mathbb{R}^n$ at step i is obtained from the respective hidden state h_i via a linear transformation:

$$\theta^* = W \cdot h_i + b$$

Proposed Method – GRU Meta Generator

- GRU Meta Generator – Training Strategy 1
  - The computation time of GRUs increases linearly with sequence length
  - To improve training efficiency while preserving long-term memory, train the GRU meta generator on a **fixed-length sequence** by **continuing on a previously learned hidden state**
  - When training $\mathcal{M}_{t+1}$, we take in a sequence of k most recent models $\boldsymbol{\Theta}_{t-(k-1):t}$ as inputs, and use a previously learned hidden state h_{t-k} as the initial hidden state:

$$
\begin{aligned}
\omega_{t+1} &= \arg\min_{\omega} \mathcal{L}(\Theta_t^* | D_{t+1}, \omega_t) \\
&= \arg\min_{\omega} \mathcal{L}(\mathcal{M}_{\omega}(\boldsymbol{\Theta}_{t-(k-1):t}, h_{t-k}) | D_{t+1}, \omega_t)
\end{aligned}
$$

Proposed Method – GRU Meta Generator

- GRU Meta Generator – Training Strategy 2

  - To ensure more accurate sequential modeling, further train the meta generator **on multiple periods' data concurrently**

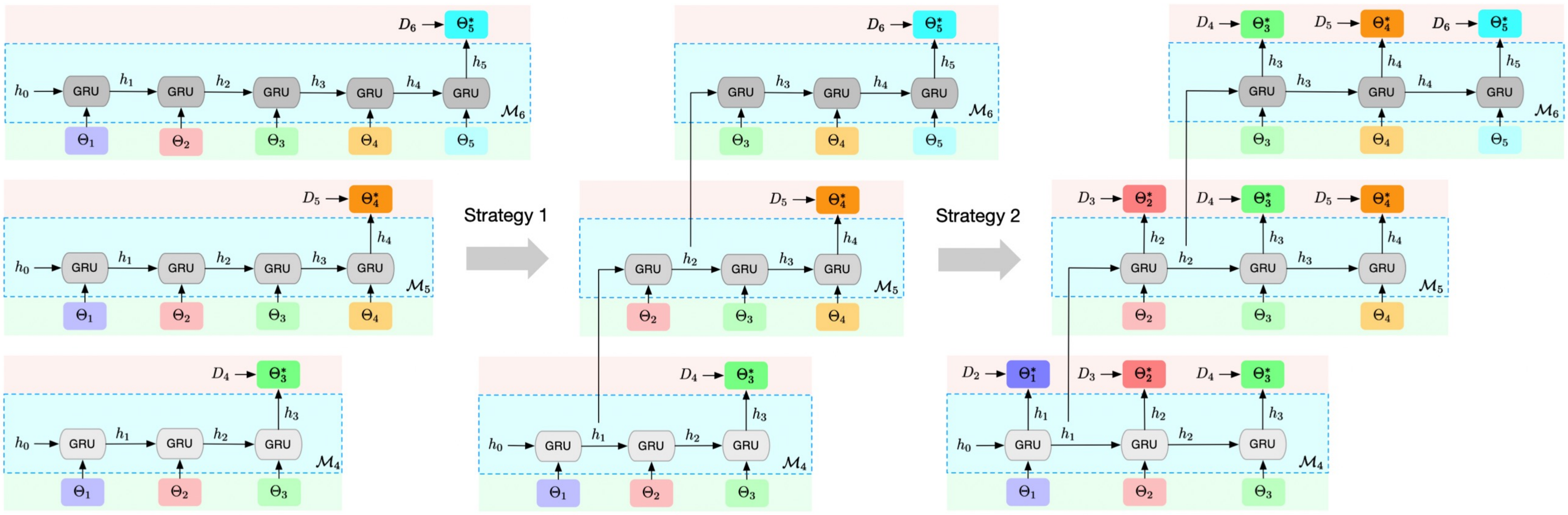
  - When trianing $\mathcal{M}_{t+1}$, instead of optimizing Θ_i^* towards D_{t+1} only, we optimize all $\{\Theta_i^*\}_{i=t-(k-1)}^{t}$ towards all $\{D_{i+1}\}_{i=t-(k-1)}^{t}$ concurrently:

$$\begin{aligned}\omega_{t+1} &= \arg\min_{\omega} \sum_{i=t-(k-1)}^{t} \lambda_i \mathcal{L}(\Theta_i^* | D_{i+1}, \omega_t) \\ &= \arg\min_{\omega} \sum_{i=t-(k-1)}^{t} \lambda_i \mathcal{L}(\mathcal{M}_\omega(\boldsymbol{\Theta}_{t-(k-1):i}, h_{t-k}) | D_{i+1}, \omega_t)\end{aligned}$$

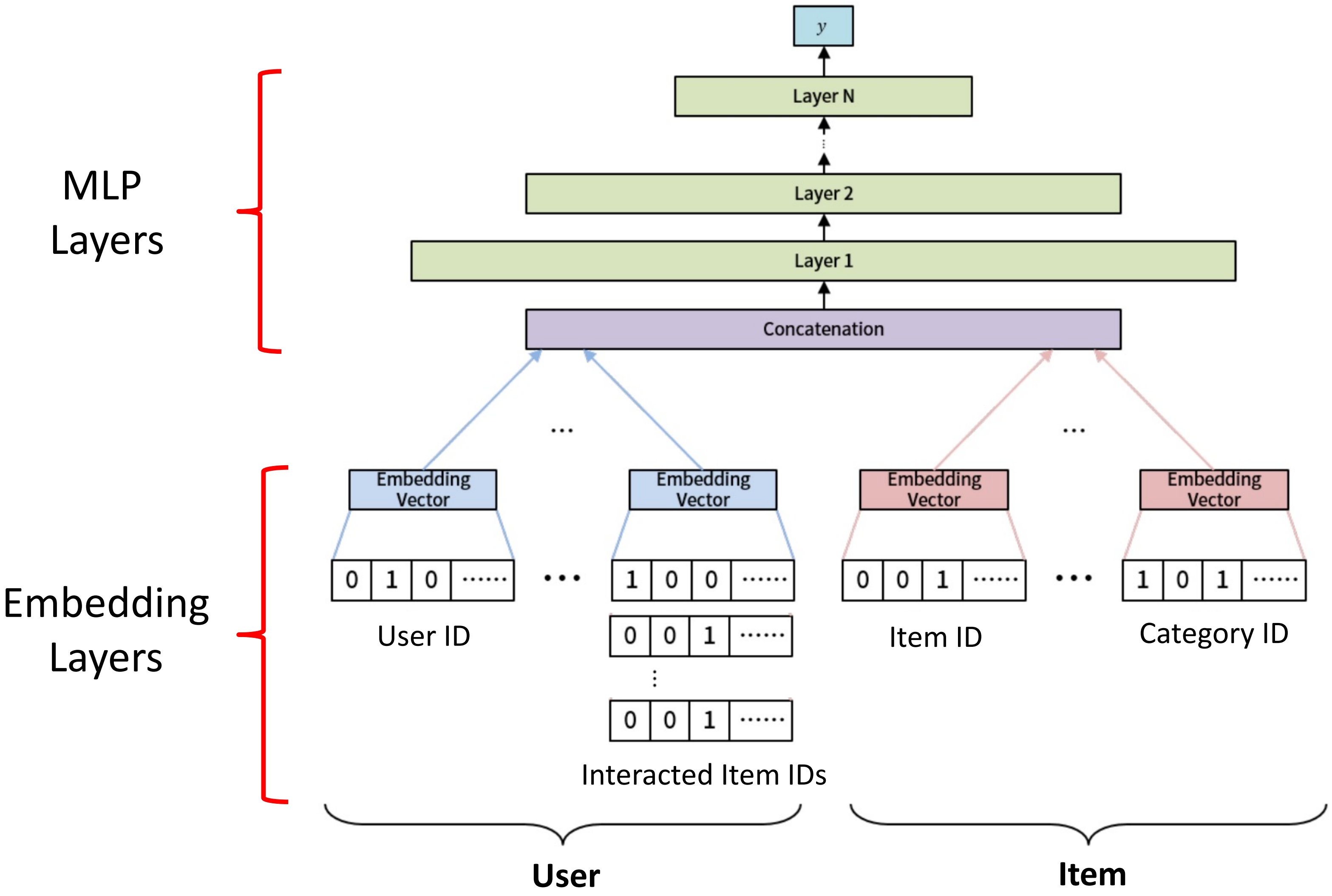
where λ_i is computed from a linear decay function $\lambda_{t-(k-j)} = \frac{j}{\sum_{j'=1}^{k} j'}$ for $j \in \{1, 2, \ldots, k\}$ to assign greater weight to the more recent data

Proposed Method – ASMG-GRU

ASMG-GRUsingle **ASMG-GRUmulti**



Proposed Method – Instantiate on Embedding&MLP Base Model



Experiments – Settings

- Datasets

	Dataset	Users	Items	Categories	Samples
Public datasets	Tmall	49,986	43,571	634	6,094,202
	Sobazaar	17,126	24,785	-	1,685,320
	MovieLens	43,181	51,142	20	6,840,091
Industrial dataset	Lazada	10,000	43,878	3,860	6,659,828

- Baselines

IU – Conventional incremental update
BU-w – Batch update using the most recent w periods of data
SPMF – A sample-based approach
IncCTR – A model-basd approach that applies knowledge distillation across two consecutive models
SML – A model based approach that learns to fuse two consecutive models
SML-MF – SML instantiated on Matrix Factorization (MF) base model
ASMG-linear – A meta generator that linearly combines the sequence of models, i.e., $\Theta_t^* = \sum_{i=t-(k-1)}^{t} \alpha_i \Theta_i$
ASMG-GRU – Our proposed method

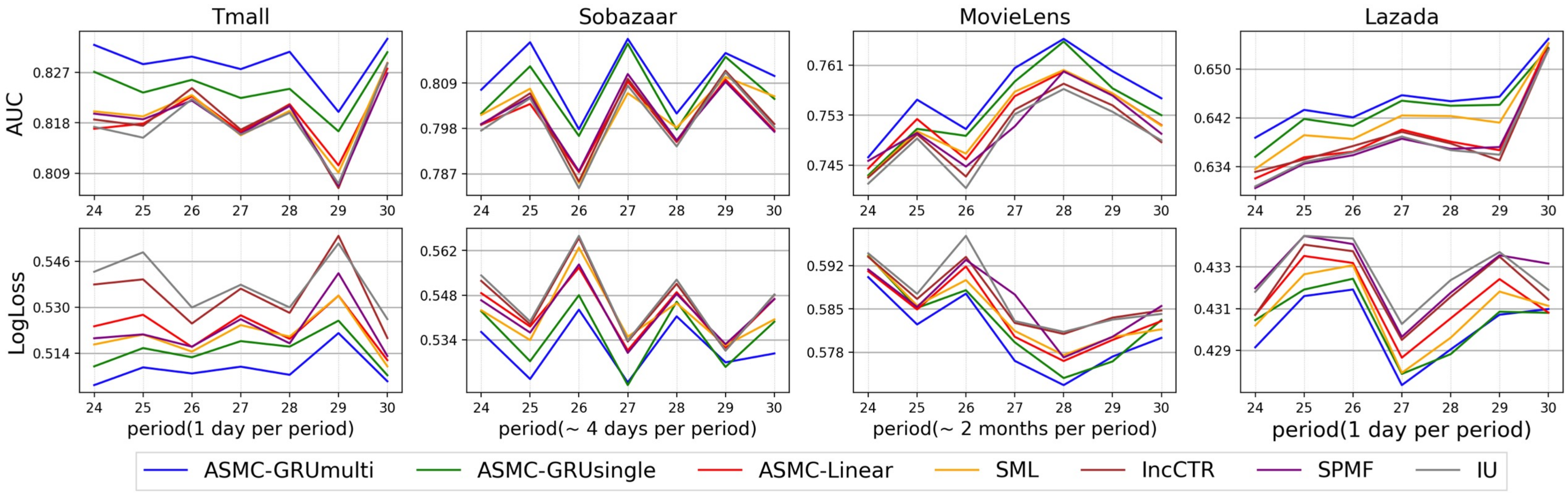
Experiments – Comparison with Baselines

Method	Tmall		Sobazaar		MovieLens		Lazada	
	AUC ↑	imp%	AUC ↑	imp%	AUC ↑	imp%	AUC ↑	imp%
IU	0.8180 ± 0.0007	-	0.7998 ± 0.0007	-	0.7494 ± 0.0002	-	0.6381 ± 0.0001	-
BU-3	0.8107 ± 0.0009	-0.89%	0.7913 ± 0.0009	-1.06%	0.7379 ± 0.0003	-1.53%	0.6332 ± 0.0002	-0.77%
BU-5	0.8002 ± 0.0009	-2.18%	0.7824 ± 0.0012	-2.18%	0.7280 ± 0.0005	-2.86%	0.6287 ± 0.0004	-1.47%
BU-7	0.7938 ± 0.0005	-2.96%	0.7781 ± 0.0007	-2.71%	0.7212 ± 0.0003	-3.76%	0.6203 ± 0.0002	-2.79%
SPMF	0.8187 ± 0.0006	0.09%	0.8007 ± 0.0004	0.11%	0.7511 ± 0.0002	0.23%	0.6381 ± 0.0002	0.00%
IncCTR	0.8190 ± 0.0007	0.12%	0.8009 ± 0.0006	0.14%	0.7502 ± 0.0003	0.11%	0.6388 ± 0.0003	0.11%
SML	0.8194 ± 0.0007	0.17%	0.8021 ± 0.0012	0.29%	0.7522 ± 0.0009	0.37%	0.6416 ± 0.0011	0.55%
SML-MF	0.7628 ± 0.0013	-6.75%	0.7782 ± 0.0017	-2.70%	0.7242 ± 0.0012	-3.36%	0.6100 ± 0.0016	-4.40%
ASMG-Linear	0.8190 ± 0.0006	0.12%	0.8002 ± 0.0008	0.05%	0.7524 ± 0.0002	0.40%	0.6390 ± 0.0001	0.14%
ASMG-GRUsingle	0.8241 ± 0.0010	0.75%	0.8055 ± 0.0017	0.71%	0.7539 ± 0.0009	0.60%	0.6439 ± 0.0005	0.91%
ASMG-GRUmulti	**0.8289 ± 0.0010**	1.33%	**0.8108 ± 0.0017**	1.38%	**0.7564 ± 0.0009**	0.93%	**0.6452 ± 0.0005**	1.11%
	LogLoss ↓	imp%	LogLoss ↓	imp%	LogLoss ↓	imp%	LogLoss ↓	imp%
IU	0.5382 ± 0.0011	-	0.5466 ± 0.0017	-	0.5871 ± 0.0002	-	0.4327 ± 0.0001	-
BU-3	0.5518 ± 0.0009	-2.53%	0.5536 ± 0.0013	-1.28%	0.5949 ± 0.0004	-1.33%	0.4342 ± 0.0001	-0.35%
BU-5	0.5615 ± 0.0015	-4.33%	0.5653 ± 0.0009	-3.42%	0.6056 ± 0.0007	-3.15%	0.4361 ± 0.0003	-0.79%
BU-7	0.5732 ± 0.0012	-6.50%	0.5783 ± 0.0017	-5.80%	0.6105 ± 0.0004	-3.99%	0.4379 ± 0.0002	-1.20%
SPMF	0.5220 ± 0.0007	3.01%	0.5425 ± 0.0005	0.75%	0.5857 ± 0.0001	0.24%	0.4327 ± 0.0001	0.00%
IncCTR	0.5344 ± 0.0010	0.71%	0.5458 ± 0.0007	0.15%	0.5865 ± 0.0003	0.10%	0.4321 ± 0.0001	0.14%
SML	0.5198 ± 0.0009	3.42%	0.5418 ± 0.0017	0.88%	0.5843 ± 0.0008	0.48%	0.4309 ± 0.0003	0.42%
SML-MF	0.5822 ± 0.0019	-8.18%	0.5713 ± 0.0021	-4.52%	0.6106 ± 0.0014	-4.00%	0.4390 ± 0.0005	-1.46%
ASMG-Linear	0.5226 ± 0.0012	2.90%	0.5430 ± 0.0013	0.66%	0.5840 ± 0.0002	0.53%	0.4314 ± 0.0000	0.30%
ASMG-GRUsingle	0.5154 ± 0.0018	4.24%	0.5370 ± 0.0017	1.76%	0.5828 ± 0.0011	0.73%	0.4304 ± 0.0003	0.53%
ASMG-GRUmulti	**0.5079 ± 0.0018**	5.63%	**0.5309 ± 0.0017**	2.87%	**0.5806 ± 0.0011**	1.11%	**0.4299 ± 0.0003**	0.65%

Average performance over 7 test periods

- Performance drops as window size increases.
- Model-based approach performs better than the sample-based approach.
- GRU meta generator design is better than its linear counterpart in terms of modelling the sequential patterns.

Experiments – Comparison with Baselines

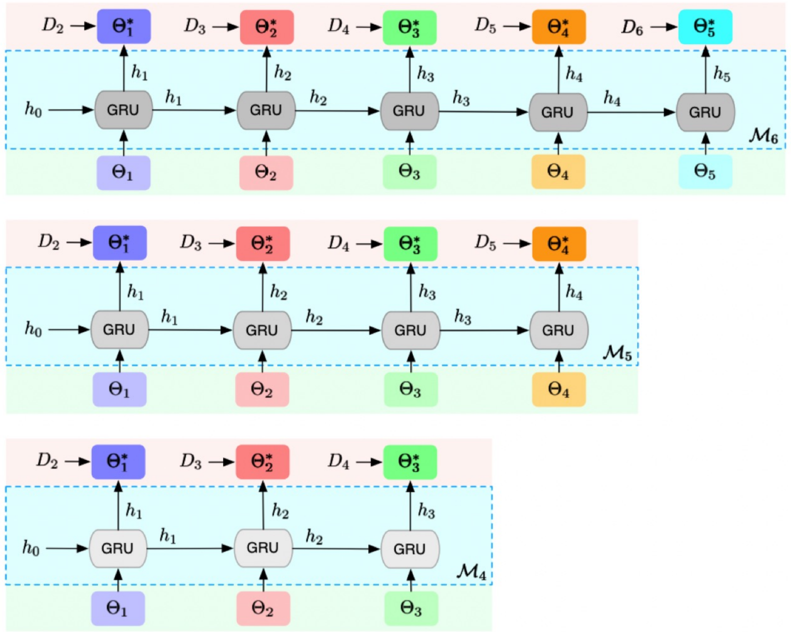


Disentangled performance over 7 test periods

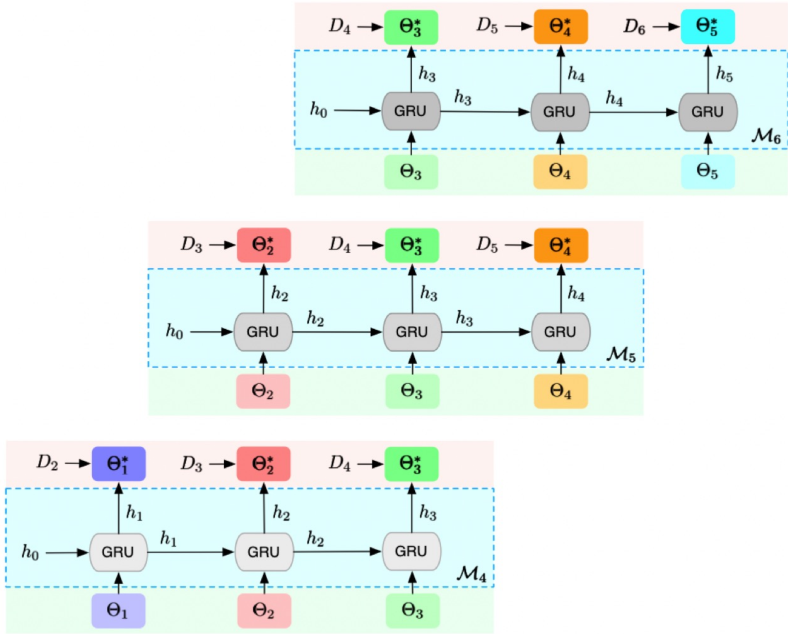
Experiments – Ablation Study

- Variants of ASMG-GRU

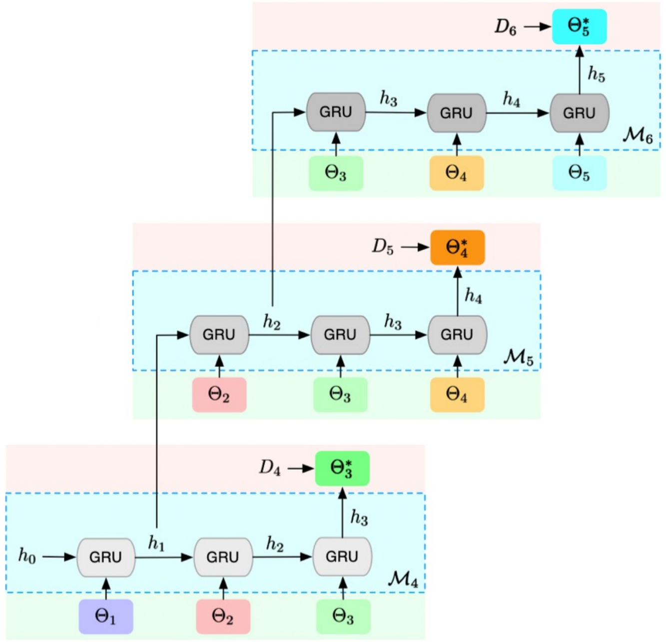
ASMG-GRUfull



ASMG-GRUzero



ASMG-GRUsingle



Experiments – Ablation Study

- Prediction Performance

Variant	Tmall		Sobazaar		MovieLens		Lazada	
	AUC ↑	LogLoss ↓	AUC ↑	LogLoss ↓	AUC ↑	LogLoss ↓	AUC ↑	LogLoss ↓
ASMG-GRUfull	0.8267	0.5108	0.8083	0.5323	**0.7565**	0.5811	0.6452	0.4299
ASMG-GRUzero	0.8224	0.5162	0.8079	0.5343	0.7550	0.5818	0.6440	0.4303
ASMG-GRUunif	0.8284	0.5102	0.8091	0.5324	0.7563	0.5811	0.6449	0.4300
ASMG-GRUsingle	0.8241	0.5154	0.8055	0.5370	0.7539	0.5828	0.6439	0.4304
ASMG-GRUmulti	**0.8289**	**0.5079**	**0.8108**	**0.5309**	0.7564	**0.5806**	**0.6452**	**0.4299**

Average performance over 7 test periods

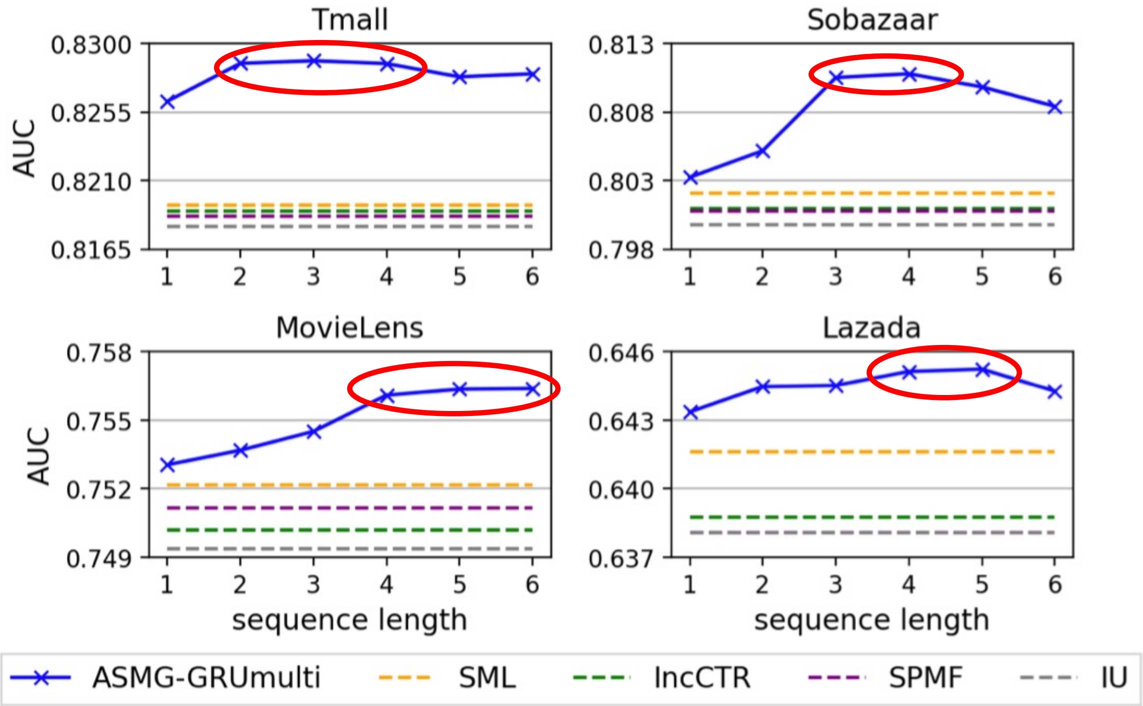
- Computation Efficiency

	Tmall	Sobazaar	MovieLens	Lazada
ASMG-GRUfull	93.6	23.1	59.3	42.6
ASMG-GRUmulti	13.8	3.4	8.7	6.2

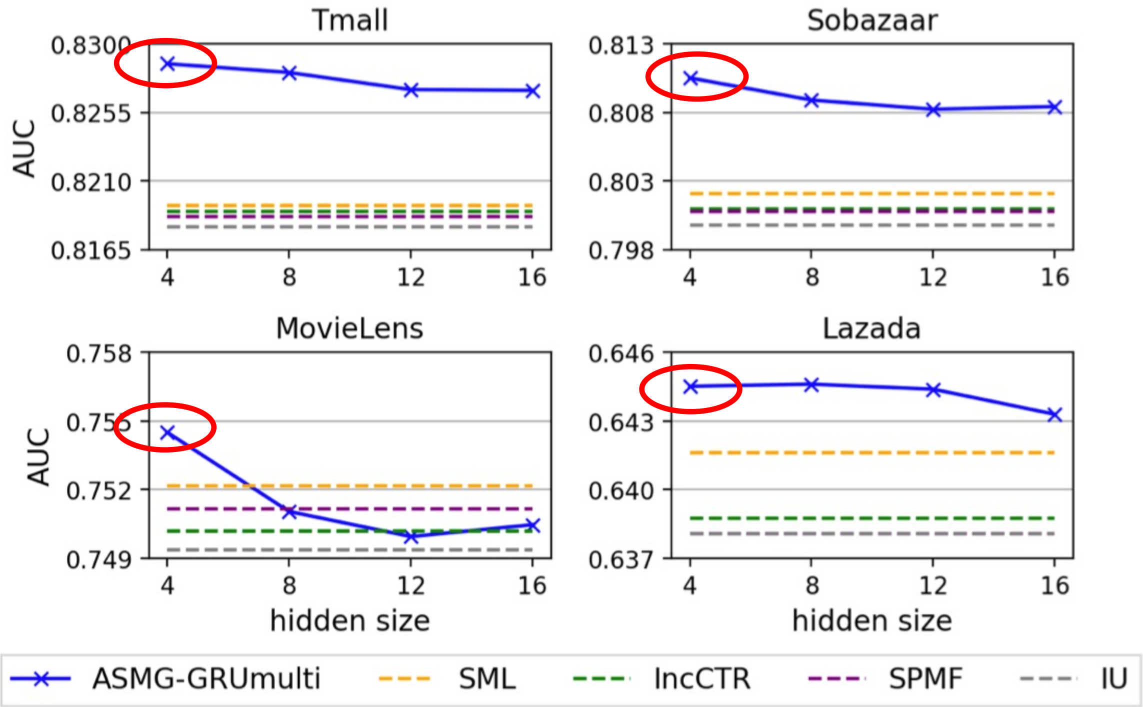
Training time (in minutes) of GRU meta generator at the last test period

Experiments – Sensitivity Analysis

- Effect of Input Sequence Length



- Effect of GRU Hidden Size



Conclusion

- Propose an **ASMG framework** to generate a better serving model by encoding a sequence of historical models.
- Introduce a **GRU-based meta generator** capable of capturing the sequential pattens.
- Further develop **two training strategies** to improve the computation efficiency and sequential modelling ability of the GRU meta generator.
- Conduct model updating experiments on three public datasets and one industrial dataset from Lazada.

Thank you for your attention!

Contact me: danni001@ntu.edu.sg