WWW2021

Task-adaptive Neural Process for User Cold-Start Recommendation

Xixun Lin
Institute of Information Engineering, Chinese Academy of Sciences
School of Cyber Security, University of Chinese Academy of Sciences
linxixun@iie.ac.cn

Jia Wu
Department of Computing, Macquarie University
jia.wu@mq.edu.au

Chuan Zhou*
Academy of Mathematics and Systems Science, CAS
School of Cyber Security, University of Chinese Academy of Sciences
zhouchuan@amss.ac.cn

Shirui Pan
Faculty of Information Technology, Monash University
shirui.pan@monash.edu

Yanan Cao
Institute of Information Engineering, Chinese Academy of Sciences
caoyanan@iie.ac.cn

Bin Wang
Xiaomi AI Lab, Xiaomi Inc
wangbin11@xiaomi.com

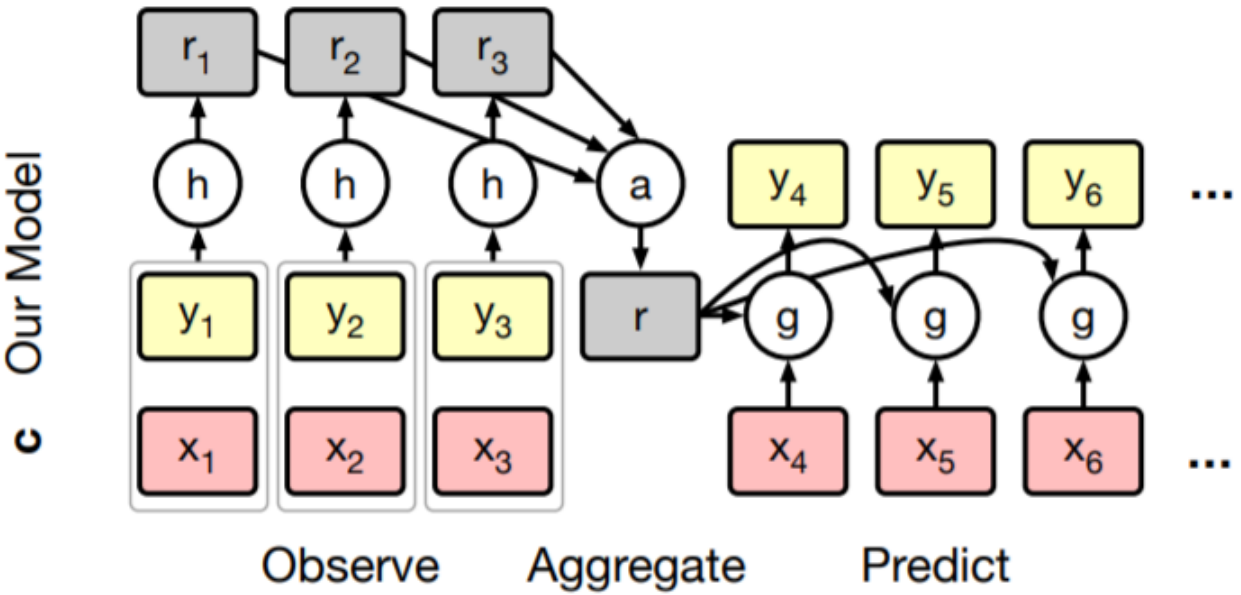
[2018ICML]Conditional Neural Processes

左边的Observe为训练集，每个样本及其对应的标签 (x_i, y_i) concat在一起，输入神经网络h得到embedding r_i ,这对应于第一步。
然后对所有的 r_i 求和,将求得的和除以总样本数N，得到r，这对应于第二步。相当于对所有的embedding求均值，这一操作称为Aggregate。

得到的输出r会作为模型g的一部分参数来对其调节。直接让r作为神经网络g的输入的一部分，和神经网络g的真正输入（测试集样本）concat在一起

$$\hat{y}_i = g(x_i, r)$$

这一模块作为Prediction，由训练集整合的信息r作为输入一部分，来调节整个神经网络g的输出，调节完毕后，便可以用函数g直接对测试样本进行预测。



Learning

- 元训练时，每个episode在一个数据集的训练集上跑一遍学习过程，得到对应的模型g，然后在测试集上测试g，通过预测的误差来更新整个“学习过程”的参数

目标函数

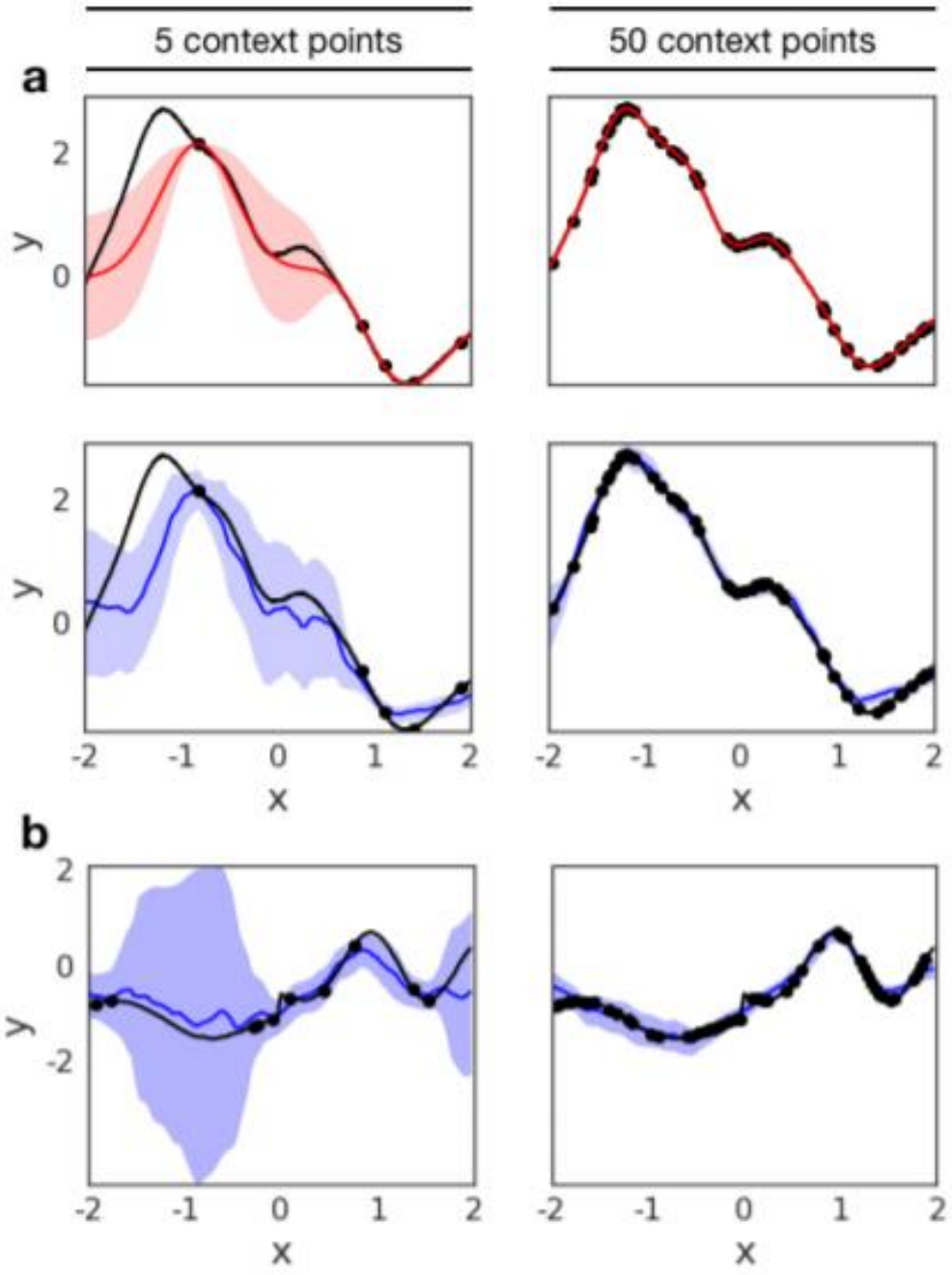
$$\sum_{D_i} \sum_{(x_j,y_j)\in D^i_{test}} \mathcal{L}(g^i_\theta(x_j), y_j)$$

$$g^i_\theta(x) = f_\theta(D^i_{train})(x)$$

$$= g_\theta(r_i, x)$$

$$= g_\theta(\frac{1}{N_i} \sum_{(x_j,y_j)\in D^i_{train}} h_\theta(x_j, y_j), x)$$

- CNP用于回归任务时，模型g的每一个预测输出是一个高斯分布，loss为真实值在该模型下的负对数似然。因此输出的分布，均值可作为预测值，方差作为不确定性估计。
- 论文中设计的每一个数据集是从一个已知的高斯过程中采样的一个函数曲线，然后随机从中取一些点作为训练集，并据此预测其他点，因此不同的采样便构成不同的数据集。
- 每个图中的黑点是训练样本，黑线为真实曲线，红或蓝线是预测曲线，带状区域为不确定性估计。第一行的图是直接用真实的用于采样的高斯过程进行预测得到的效果，是任何模型所能达到的效果上界。第二、三行为CNP的预测结果。

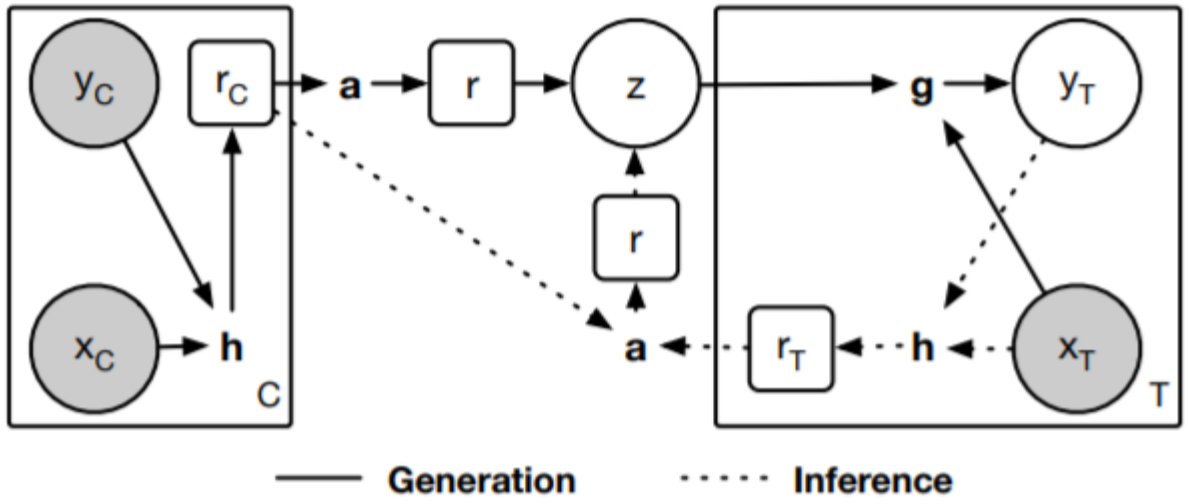


- 得到一个条件随机过程 $\hat{y_{test}} \sim N(\mu(x), \sigma^2(x)) = g_\theta^i(x_{test})$
- 每一个数据的预测为一个高斯分布，因此预测出的回归函数是有随机性的，而函数具有不确定性则称其为一个随机过程，因此CNP得到的是一个随机过程。并且由于该随机过程条件于训练集数据，因此称其为一个条件随机过程。
- **不同的点之间的预测不会相互影响**
- 分别得到预测高斯分布 $N(\mu(x_1), \sigma^2(x_1))$ 、 $N(\mu(x_2), \sigma^2(x_2))$

- 能够捕捉相关性

[ICML Workshop2018]Neural Processes

- CNP中的“学习过程”是一个从训练集映射到向量r的神经网络，然后用r来 调整网络g的参数。那么我们可以直接给r加上不确定性，使其变为一个隐变量z，在取不同值时会得到不同的 g，从而由z的变化来影响所有数据预测的结果，这样就间接促成了预测之间的相关性。
- 假设z服从高斯分布，直接在CNP得到r后，再补充一个神经网络，输入为r，输出得到z的均值和方差：$z \sim N(\mu(r), \sigma^2(r))$



(b) Computational diagram

每次预测时，从这个高斯分布中采样一个z，和CNP一样送入g网络和数据concat在一起进行预测。

- 目标函数是让条件随机过程的似然函数值最大

$$\max \frac{1}{N}\sum_{i}^{N} log(p(D_{test}^{i}|D_{train}^{i}))$$

$$log(p(D_{test}^{i}|D_{train}^{i}))$$

$$= log(\int p(z|D_{trian}^{i})p(D_{test}^{i}|z)dz)$$

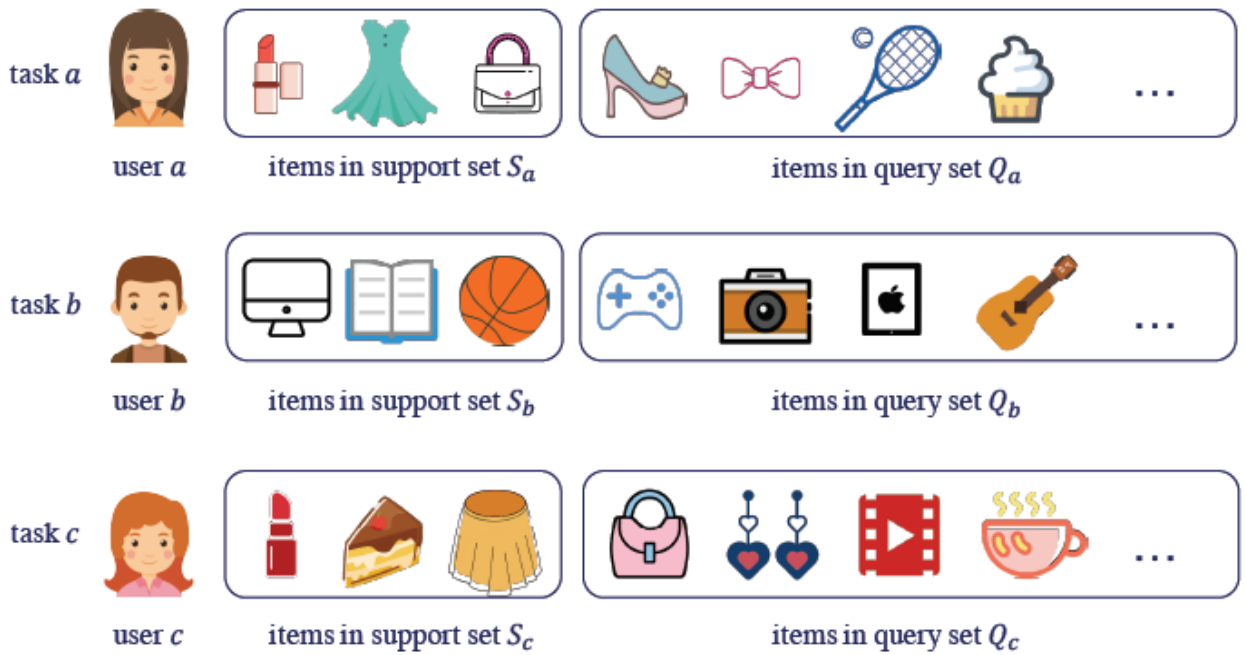
$$log(p(D_{test}^{i}|D_{train}^{i}))$$

$$\geq E_{p(z|D^{i})}[logp(D_{test}^{i}|z)] - KL[p(z|D^{i})||p(z|D_{train}^{i})]$$

$$p(z|D_{test}^{i}, D_{train}^{i}) = p(z|D^{i})$$

$$p(z|D^{i}) \approx p_{\theta}(z|D^{i})$$

- 第一项，用整个数据集作为条件得到z的分布，从中采样一个z，送入g中得到所有测试集数据的预测分布，将标签代入得到此对数似然期望的点估计。
- 第二项，分别以整个数据集作为条件，以及只是训练集作为条件得到两个z的分布，并求其KL距离。



$$p(y_{i,1:N_i}|x_{i,1:N_i}) = \int p(z_i) \prod_{j=1}^{N_i} p(y_{i,j}|x_{i,j}, z_i) dz_i.$$

$$\log p(y_{i,1:N_i}|x_{i,1:N_i}) = \log \int p(z_i, y_{i,1:N_i}|x_{i,1:N_i}) dz_i$$

$$= \log \int p(z_i, y_{i,1:N_i}|x_{i,1:N_i}) \frac{q(z_i|\tau_i)}{q(z_i|\tau_i)} dz_i$$

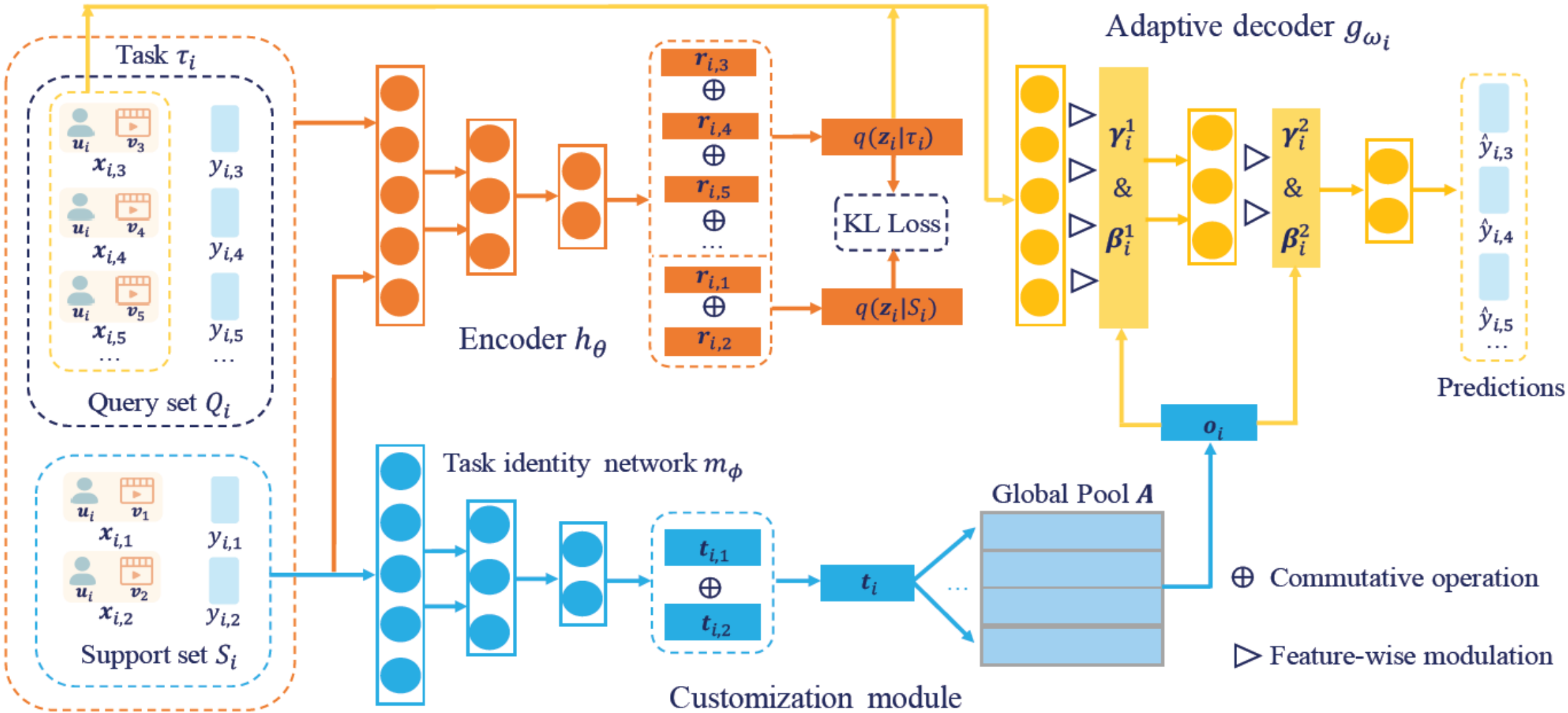
$$\geq \mathbb{E}_{q(z_i|\tau_i)} \left[\log \frac{p(z_i, y_{i,1:N_i}|x_{i,1:N_i})}{q(z_i|\tau_i)} \right]$$

$$= \mathbb{E}_{q(z_i|\tau_i)} \left[\log \frac{p(z_i) p(y_{i,1:N_i}|x_{i,1:N_i}, z_i)}{q(z_i|\tau_i)} \right]$$

$$= \mathbb{E}_{q(z_i|\tau_i)} \left[\sum_{j=1}^{N_i} \log p(y_{i,j}|x_{i,j}, z_i) + \log \frac{p(z_i)}{q(z_i|\tau_i)} \right].$$

$$\log p(y_{i,1:N_{Q_i}}|x_{i,1:N_{Q_i}}, S_i)$$

$$\geq \mathbb{E}_{q(z_i|\tau_i)} \left[\sum_{j=1}^{N_{Q_i}} \log p(y_{i,j}|x_{i,j}, z_i) + \log \frac{q(z_i|S_i)}{q(z_i|\tau_i)} \right].$$


Task τ_i
u_i v_3 $x_{i,3}$ $y_{i,3}$
u_i v_4 $x_{i,4}$ $y_{i,4}$
u_i v_5 $x_{i,5}$ $y_{i,5}$
Query set Q_i
u_i v_1 $x_{i,1}$ $y_{i,1}$
u_i v_2 $x_{i,2}$ $y_{i,2}$
Support set S_i
Encoder h_θ
$r_{i,3}$ $r_{i,4}$ $r_{i,5}$ $r_{i,1}$ $r_{i,2}$
$q(z_i|\tau_i)$
$q(z_i|S_i)$
KL Loss
Adaptive decoder g_{ω_i}
γ_i^1 & β_i^1
γ_i^2 & β_i^2
o_i
$\hat{y}_{i,3}$ $\hat{y}_{i,4}$ $\hat{y}_{i,5}$
Predictions
Task identity network m_ϕ
$t_{i,1}$ $t_{i,2}$
t_i
Global Pool A
Customization module
⊕ Commutative operation
▷ Feature-wise modulation

Embedding Layers

$$u_i = \left[E_1 c_{i,1} | ... | E_n c_{i,n}\right], \qquad \text{content}$$

$$u_i = \sigma(W_2 \sigma(W_1 e_i + b_1) + b_2), \qquad \text{No content}$$

Encoder

generate the variational approximations $q(z_i|S_i)$ and $q(z_i|\tau_i)$ respectively.

$$r_{i,j} = h^l(h^{l-1}(\cdots h^1([u_i|v_j|y_{i,j}]))),$$

$$r_i = r_{i,1} \oplus r_{i,2} \oplus \ \ldots r_{i,N_{i-1}} \oplus r_{i,N_i},$$

$$z_i \sim \mathcal{N}(\mu_i, \mathrm{diag}(\sigma_i^2))$$

$$r_i = \mathrm{ReLU}(W_s r_i),$$
$$\mu_i = W_\mu r_i, \ \log\sigma_i = W_\sigma r_i,$$
$$z_i = \mu_i + \epsilon \odot \sigma_i, \ \epsilon \sim \mathcal{N}(0, I),$$

Customization Module

$$t_{i,j} = m^l(m^{l-1}(\cdots m^1([u_i|v_j|y_{i,j}]))).$$

The global pool $A = [a_1, ..., a_k] \in R^{d \times k}$ soft cluster centroids

We use the Student's t-distribution as a kernel to measure the normalized similarity be·tween t_i and a_j as follows

$$c_{i,j} = \frac{(1 + ||t_i - a_j||^2/\alpha)^{-\frac{\alpha+1}{2}}}{\sum_{j'}(1 + ||t_i - a_{j'}||^2/\alpha)^{-\frac{\alpha+1}{2}}},$$

where α is the degree of freedom of the Student's t-distribution.

final task embedding o_i $o_i = \sigma(W_o(t_i + Ac_i^T))$,

The normalized assignments of all training tasks construct a assignment matrix $C = [c_1, ..., c_{|\mathcal{T}^{tr}|}] \in R^{|\mathcal{T}^{tr}| \times k}$. As suggested by [49], we use an unsupervised clustering loss $\mathcal{L}_u$ with the guidance of an auxiliary clustering target distribution D. $\mathcal{L}_u$ is defined as a KL divergence loss between C and D:

$$\mathcal{L}_u = \mathrm{KL}(D||C) = \sum_i \sum_j D_{i,j} \log \frac{D_{i,j}}{C_{i,j}}. \quad (13)$$

where the clustering target distribution D can be defined as follows,

$$D_{i,j} = \frac{(C_{i,j})^2/\sum_i C_{i,j}}{\sum_{j'}(C_{i,j'})^2/\sum_i C_{i,j'}}. \quad (14)$$

$$\log p(y_{i,1:N_{Q_i}} | x_{i,1:N_{Q_i}}, S_i)$$
$$\geq \mathbb{E}_{q(z_i|\tau_i)}\Big[\sum_{j=1}^{N_{Q_i}} \log p(y_{i,j}|x_{i,j}, z_i) + \log \frac{q(z_i|S_i)}{q(z_i|\tau_i)}\Big].$$

Adaptive Decoder

The original decoder g_ω in Eq.(4) is used to learn the conditional likelihood $p(y|x,z)$, which is a global predictor shared by all tasks. In this section, TaNP introduces an adaptive decoder g_{ω_i} in a parameter-efficient manner. We describe two candidate modulation strategies to generate the model parameters of adaptive decoder via using the final task embedding $\boldsymbol{o}_i$. The first variant is Feature-wise Linear Modulation (FiLM) [37]. Based on this basic modulation, we propose the second modulation strategy, i.e., Gating-FiLM.

FiLM.

$$\boldsymbol{\gamma}_i^l = \tanh(\boldsymbol{W}_\gamma^l \boldsymbol{o}_i), \quad \boldsymbol{\beta}_i^l = \tanh(\boldsymbol{W}_\beta^l \boldsymbol{o}_i),$$
$$\boldsymbol{g}_{i,j}^{l+1} = \mathrm{ReLU}(\boldsymbol{\gamma}_i^l \odot (\boldsymbol{W}_\omega^l \boldsymbol{g}_{i,j}^l + \boldsymbol{b}_\omega^l) + \boldsymbol{\beta}_i^l),$$
$$\boldsymbol{g}_{i,j}^1 = [\boldsymbol{u}_i | \boldsymbol{v}_j | \boldsymbol{z}_i].$$

Gating-FiLM.

$$\boldsymbol{\gamma}_i^l = \tanh(\boldsymbol{W}_\gamma^l \boldsymbol{o}_i), \ \boldsymbol{\beta}_i^l = \tanh(\boldsymbol{W}_\beta^l \boldsymbol{o}_i),$$
$$\boldsymbol{\eta}_i^l = \tanh(\boldsymbol{W}_\eta^l \boldsymbol{o}_i), \ \boldsymbol{\delta}_i^l = \sigma(\boldsymbol{W}_\delta^l \boldsymbol{o}_i),$$
$$\boldsymbol{\gamma}_i^l = \boldsymbol{\gamma}_i^l \odot \boldsymbol{\delta}_i^l + \boldsymbol{\eta}_i^l \odot (1 - \boldsymbol{\delta}_i^l),$$
$$\boldsymbol{\beta}_i^l = \boldsymbol{\beta}_i^l \odot \boldsymbol{\delta}_i^l + \boldsymbol{\eta}_i^l \odot (1 - \boldsymbol{\delta}_i^l),$$
$$\boldsymbol{g}_{i,j}^{l+1} = \mathrm{ReLU}(\boldsymbol{\gamma}_i^l \odot (\boldsymbol{W}_\omega^l \boldsymbol{g}_{i,j}^l + \boldsymbol{b}_\omega^l) + \boldsymbol{\beta}_i^l),$$

Loss Function

In our model, the likelihood term in Eq.(4) is reformulated as a regression-based loss function:

$$\begin{aligned}\mathcal{L}_{r,i} &= -\mathbb{E}_{q(z_i|\tau_i)}\log p(y_{i,1:N_{Q_i}}|x_{i,1:N_{Q_i}}, z_i) \\ &\propto \frac{1}{N_{Q_i}}\sum_{j=1}^{N_{Q_i}}(y_{i,j}-\hat{y}_{i,j})^2,\end{aligned} \tag{17}$$

where $\hat{y}_{i,j}$ is the final output of $g_{\omega_i}(x_{i,j}, z_i, \boldsymbol{o}_i)$. For implicit feedback data, $\mathcal{L}_{r,i}$ is defined as a binary cross-entropy loss:

$$\begin{aligned}\mathcal{L}_{r,i} &= -\mathbb{E}_{q(z_i|\tau_i)}\log p(y_{i,1:N_{Q_i}}|x_{i,1:N_{Q_i}}, z_i) \\ &\propto -\frac{1}{N_{Q_i}}\sum_{j=1}^{N_{Q_i}} y_{i,j}\log(\hat{y}_{i,j}) + (1-y_{i,j})\log(1-\hat{y}_{i,j}).\end{aligned} \tag{18}$$

The training loss of TaNP is defined as:

$$\mathcal{L} = \frac{1}{|\mathcal{T}^{tr}|}\sum_{i=1}^{|\mathcal{T}^{tr}|}(\mathcal{L}_{r,i}+\mathcal{L}_{c,i}) + \lambda\mathcal{L}_u, \tag{19}$$

where $\mathcal{L}_{c,i} = \mathrm{KL}(q(z_i|\tau_i)||q(z_i|S_i))$ that can be also considered as a regularization term to approximate the condition of consistency, and λ is a hyper-parameter which is selected between 0 and 1.

Algorithm 1 The training procedure of TaNP.

Input: Training user set U^{tr}; Item set V; User and item side-information (optional); Training task set $\mathcal{T}^{tr}$; Hyper-parameters: d, l, k, α, λ.

Output: Parameters in embedding layer; h_θ; m_ϕ; $\boldsymbol{A}$; g_{ω_i}.

```
Initialize all model parameters.
while not convergence do
    for τ_i ∈ T^tr do
        Construct S_i and Q_i from τ_i.
        Generate q(z_i|τ_i) via h_θ in Eq.(9).
        Generate task embedding o_i via m_φ and A in Eq.(10)-(12).
        Predictions on Q_i via adaptive decoder g_{ω_i}, z_i and o_i in
        Eq.(15) or Eq.(16).
        Generate q(z_i|S_i) via h_θ in Eq.(9).
        Calculate prediction loss L_{r,i} in Eq.(17) or Eq.(18).
        Calculate regularization loss L_{c,i} in Eq.(19).
    end for
    Calculate clustering loss L_u in Eq.(13) and the total loss L
    in Eq.(19).
    Update model parameters by Adam optimizer.
end while
```

Datasets	Users	Items	Ratings	Type	Content
MovieLens-1M	6,040	3,706	1,000,209	explicit	yes
Last.FM	1,872	3,846	42,346	implicit	no
Gowalla	2692	27,237	134,476	implicit	no

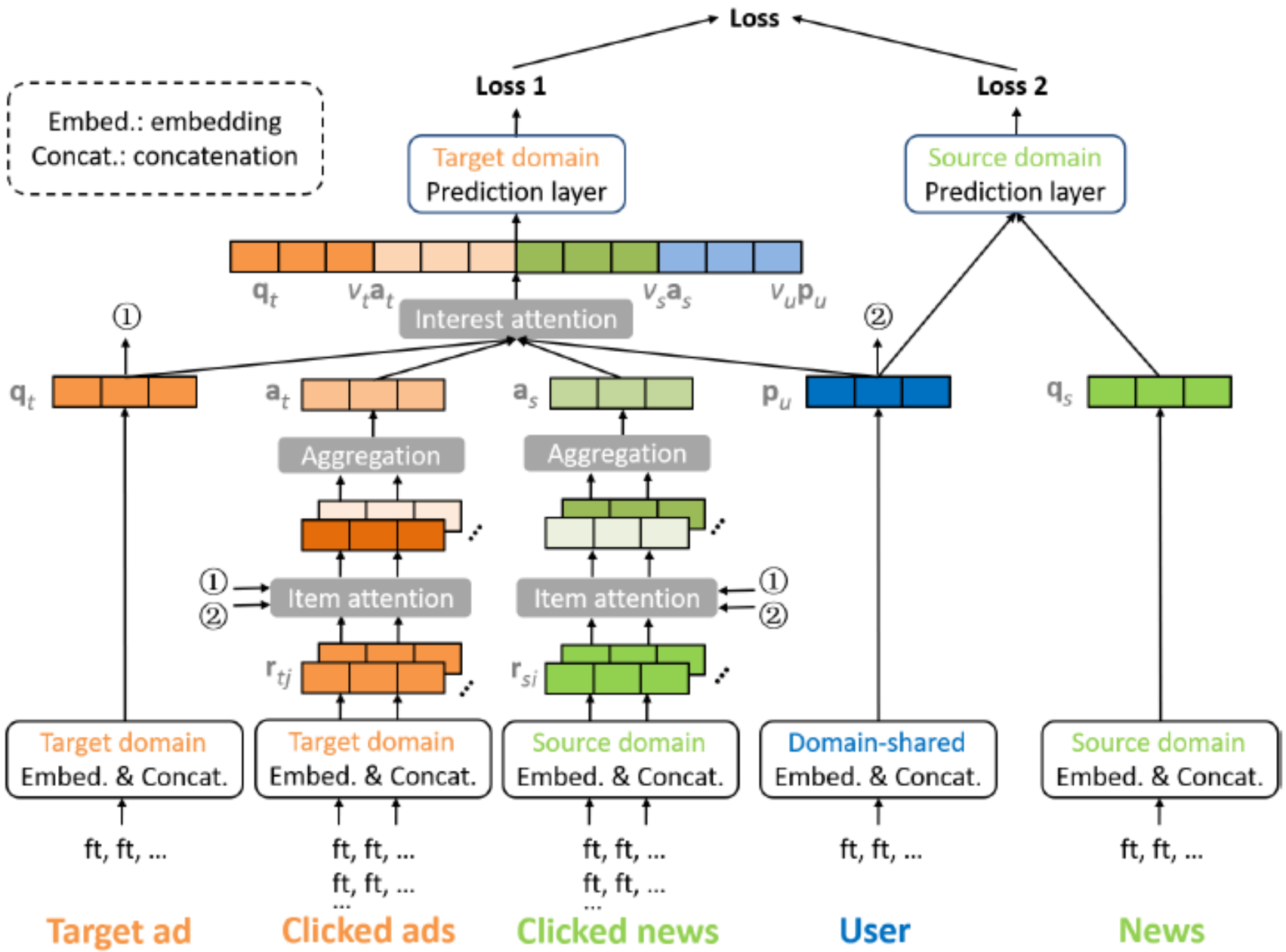
Model	P@5	NDCG@5	MAP@5	P@7	NDCG@7	MAP@7	P@10	NDCG@10	MAP@10
PPR	49.36	66.62	37.86	51.25	67.27	38.12	54.30	68.27	41.20
NeuMF	48.27	65.43	36.65	51.24	66.55	37.90	55.32	68.19	41.43
DropoutNet	51.77	69.34	41.82	53.67	70.83	43.81	57.34	72.02	46.59
MeLU	55.99	73.08	46.79	57.34	73.18	48.45	61.05	74.04	49.02
MetaCS	55.43	71.69	44.85	56.89	72.05	44.94	59.78	72.86	47.52
MetaHIN	57.65	73.43	47.40	58.67	73.95	48.75	61.18	74.50	49.99
MAMO	57.69	73.24	47.72	58.42	74.03	49.62	61.51	74.41	50.06
TaNP (w/o tm)	58.03	73.76	47.79	58.90	73.89	48.37	61.29	74.44	49.73
TaNP (FiLM)	59.76	74.97	49.08	60.45	75.22	49.76	62.78	75.48	51.12
TaNP (Gating-FiLM)	60.12	75.00	49.12	60.29	75.34	50.79	62.66	75.53	51.56

CIKM2020

MiNet: Mixed Interest Network for Cross-Domain Click-Through Rate Prediction

Wentao Ouyang, Xiuwu Zhang, Lei Zhao, Jinmei Luo, Yu Zhang, Heng Zou, Zhaojie Liu, Yanlong Du
Intelligent Marketing Platform, Alibaba Group
{maiwei.oywt,xiuwu.zxw,zhaolei.zl,cathy.jm,zy107620,zouheng.zh,zhaojie.lzj,yanlong.dyl}@alibaba-inc.com

Figure 1: Illustration of r



Long-term Interest across Domains

For example, if user u has features "UID = u123, City = BJ, Gender = male, OS = ios", we have

$$\mathbf{p}_u = [\mathbf{e}_{\text{u123}} \| \mathbf{e}_{\text{BJ}} \| \mathbf{e}_{\text{male}} \| \mathbf{e}_{\text{ios}}],$$

Short-term Interest from the Source Domain

$$\mathbf{a}_s = \sum_i \alpha_i \mathbf{r}_{si},$$

$$\alpha_i = \frac{\exp(\tilde{\alpha}_i)}{\sum_{i'} \exp(\tilde{\alpha}_{i'})},$$

$$\tilde{\alpha}_i = \mathbf{h}_s^T ReLU\left(\mathbf{W}_s \mathbf{r}_{si}\right),$$

However, Eq. (3) only considers each piece of clicked news $\mathbf{r}_{si}$ alone. It does not capture the relationship between a piece of clicked news and the target ad. Moreover, Eq. (3) is not tailored to the target user as well.

加性模型	$s(\mathbf{x}_i, \mathbf{q}) = \mathbf{v}^{\mathrm{T}} \tanh(W\mathbf{x}_i + U\mathbf{q}),$
点积模型	$s(\mathbf{x}_i, \mathbf{q}) = \mathbf{x}_i^{\mathrm{T}}\mathbf{q},$
缩放点积模型	$s(\mathbf{x}_i, \mathbf{q}) = \frac{\mathbf{x}_i^{\mathrm{T}}\mathbf{q}}{\sqrt{d}},$
双线性模型	$s(\mathbf{x}_i, \mathbf{q}) = \mathbf{x}_i^{\mathrm{T}} W \mathbf{q},$

Item-level Attention.

$$\tilde{\alpha}_i = \mathbf{h}_s^T ReLU\left(\mathbf{W}_s[\mathbf{r}_{si} \| \mathbf{q}_t \| \mathbf{p}_u \| \mathbf{M}\mathbf{r}_{si} \odot \mathbf{q}_t]\right),$$

The **clicked news** $\mathbf{r}_{si} \in \mathbb{R}^{D_s}$ in the source domain.
The **target ad** $\mathbf{q}_t \in \mathbb{R}^{D_t}$ in the target domain.
The **target user** $\mathbf{p}_u \in \mathbb{R}^{D_u}$.

The **transferred interaction** $\mathbf{M}\mathbf{r}_{si} \odot \mathbf{q}_t \in \mathbb{R}^{D_t}$ between the clicked news and the target ad. $\odot$ is the element-wise product operator. $\mathbf{M}$ is a transfer matrix that transfers $\mathbf{r}_{si} \in \mathbb{R}^{D_s}$ in the source domain to $\mathbf{M}\mathbf{r}_{si} \in \mathbb{R}^{D_t}$ in the target domain such that $\mathbf{M}\mathbf{r}_{si}$ can be compared with the target ad $\mathbf{q}_t$.

Short-term Interest in the Target Domain

$$\mathbf{a}_t = \sum_j \beta_j \mathbf{r}_{tj},\ \beta_j = \frac{\exp(\tilde{\beta}_j)}{\sum_{j'} \exp(\tilde{\beta}_{j'})},$$

$$\tilde{\beta}_j = \mathbf{h}_t^T ReLU\left(\mathbf{W}_t[\mathbf{r}_{tj} \| \mathbf{q}_t \| \mathbf{p}_u \| \mathbf{r}_{tj} \odot \mathbf{q}_t]\right),$$

The **clicked ad** $\mathbf{r}_{tj} \in \mathbb{R}^{D_t}$ in the target domain.
The **target ad** $\mathbf{q}_t \in \mathbb{R}^{D_t}$ in the target domain.
The **target user** $\mathbf{p}_u \in \mathbb{R}^{D_u}$.
The **interaction** $\mathbf{r}_{tj} \odot \mathbf{q}_t \in \mathbb{R}^{D_t}$ between the clicked ad and the target ad. Because they are in the same domain, no transfer matrix is needed.

Interest-Level Attention

$$\mathbf{m}_t \triangleq [\mathbf{q}_t \| v_u \mathbf{p}_u \| v_s \mathbf{a}_s \| v_t \mathbf{a}_t],$$

$$v_u = \exp\left(\mathbf{g}_u^T ReLU\left(\mathbf{V}_u[\mathbf{q}_t \| \mathbf{p}_u \| \mathbf{a}_s \| \mathbf{a}_t]\right) + b_u\right),$$

$$v_s = \exp\left(\mathbf{g}_s^T ReLU\left(\mathbf{V}_s[\mathbf{q}_t \| \mathbf{p}_u \| \mathbf{a}_s \| \mathbf{a}_t]\right) + b_s\right),$$

$$v_t = \exp\left(\mathbf{g}_t^T ReLU\left(\mathbf{V}_t[\mathbf{q}_t \| \mathbf{p}_u \| \mathbf{a}_s \| \mathbf{a}_t]\right) + b_t\right),$$

Prediction

transformation [10]. Formally, the FC layers are defined as follows:

$$\mathbf{z}_1 = ReLU(\mathbf{W}_1\mathbf{m}_t + \mathbf{b}_1),\ \mathbf{z}_2 = ReLU(\mathbf{W}_2\mathbf{z}_1 + \mathbf{b}_2),\ \cdots$$

$$\mathbf{z}_L = ReLU(\mathbf{W}_L\mathbf{z}_{L-1} + \mathbf{b}_L),$$

$$\hat{y}_t = \frac{1}{1 + \exp[-(\mathbf{w}^T\mathbf{z}_L + b)]},$$

Model Learning

$$loss_t = -\frac{1}{|\mathbb{Y}_t|}\sum_{y_t \in \mathbb{Y}_t}[y_t \log \hat{y}_t + (1 - y_t)\log(1 - \hat{y}_t)],$$

$$loss = loss_t + \gamma loss_s,$$

To facilitate the learning of long-term user interest $\mathbf{p}_u$, we also create an input vector for the source domain as $\mathbf{m}_s \triangleq [\mathbf{q}_s \| \mathbf{p}_u]$, where $\mathbf{q}_s \in \mathbb{R}^{D_s}$ is the concatenation of the feature embedding vectors for the target news. Similarly, we let $\mathbf{m}_s$ go through several FC layers and an output layer (with their own parameters). Finally, we obtain the predicted CTR $\hat{y}_s$ of the target news.

		Source: News	Target: Ad
Company	# Fields	User: 5, News: 18	User: 5, Ad: 13
	# Unique fts.	10,868,554	
	# Ini. train insts.	53,776,761	11,947,267
	# Val. insts.	8,834,570	1,995,980
	# Test insts.	8,525,115	1,889,092
	Max/Avg. # clicked (*) per target ad	25 / 7.37	5 / 1.10

		Source: Book	Target: Movie
Amazon	# Fields	User: 1, Book: 5	User: 1 , Movie: 5
	# Shared users	20,479	
	# Unique fts.	841,927	
	# Ini. train insts.	794,048	328,005
	# Val. insts.	20,479	20,479
	# Test insts.	20,479	20,479
	Max/Avg. # clicked (*) per target movie	20 / 9.82	10 / 6.69

		Company		Amazon	
Model		AUC	Logloss	AUC	Logloss
Single-domain	LR	0.6678	0.5147	0.7173	0.4977
	FM	0.6713	0.5133	0.7380	0.4483
	DNN	0.7167	0.4884	0.7688	0.4397
	Wide&Deep	0.7178	0.4879	0.7699	0.4389
	DeepFM	0.7149	0.4898	0.7689	0.4406
	DeepMP	0.7215	0.4860	0.7714	0.4382
	DIN	0.7241	0.4837	0.7704	0.4393
	DSTN	0.7268	0.4822	0.7720	0.4296
Cross-domain	CCCFNet	0.6967	0.5162	0.7518	0.4470
	MV-DNN	0.7184	0.4875	0.7814	0.4298
	MLP++	0.7192	0.4878	0.7813	0.4306
	CoNet	0.7175	0.4882	0.7791	0.4389
	MiNet	**0.7326***	**0.4784***	**0.7855***	**0.4254***