Multi-Interest Learning

implicit approach：隐式地聚类user behavior提取兴趣(MIND, ComiRec)

explicit approach：构建一个兴趣池(1000~5000), 根据用户历史行为, 利用attention mechanism显式激活部分兴趣(4~8) (SINE, Octopus)

Capsule Network & Dynamic Routing

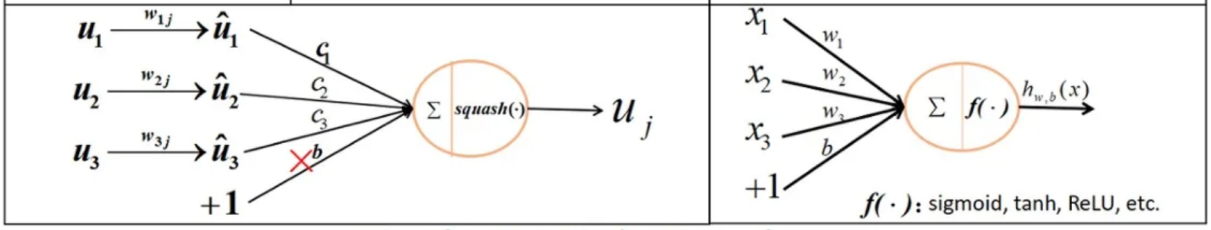
> Sabour, Sara, Nicholas Frosst, and Geoffrey E. Hinton. "Dynamic routing between capsules." *Advances in neural information processing systems* 30 (2017).

Capsule Network

类似fc layer的加权求和，但元素是向量

- 投影：$u_i = w_i v_i$，w_i learnable params
- 加权：$s = \sum_i c_i u_i$
- Squash：$v = \text{Squash}(s) = \frac{\|s\|^2}{1+\|s\|^2} \cdot \frac{s}{\|s\|}$
  - 前半部分：引入非线性，$\|s\|$接近0时整个式子接近0；$\|s\|$越大，整个式子接近1。类似Sigmoid
  - 后半部分：归一化
  - 作用：向量筛选，让短向量缩放到接近0，长向量缩放到接近1

		capsule	vs.	traditional neuron
Input from low-level neuron/capsule		vector(u_i)		scalar(x_i)
Operation	Affine Transformation	$\hat{u}_{j\|i} = W_{ij} u_i$ (Eq. 2)		–
	Weighting	$s_j = \sum_i c_{ij} \hat{u}_{j\|i}$ (Eq. 2)		$a_j = \sum_{i=1}^{3} W_i x_i + b$
	Sum			
	Non-linearity activation fun	$v_j = \frac{\|s_j\|^2}{1+\|s_j\|^2} \frac{s_j}{\|s_j\|}$ (Eq. 1)		$h_{w,b}(x) = f(a_j)$
output		vector(v_i)		scalar(h)



Capsule = New Version Neuron!
vector in, vector out VS. scalar in, scalar out

Dynamic Routing

low-level：$c_i^l \in \mathbb{R}^{N_l \times 1} \quad i = 1, \cdots, m$

high-level：$c_j^h \in \mathbb{R}^{N_h \times 1} \quad j = 1, \cdots, n$

B2I-DR (behavior to interest)

input：behavior embeddings e_i

output：interest capsules u_j

calculate capsules K：$K'_u = \max(1, \min(K, \log_2(|I_u|)))$

initialize routing logits $b_{ij} \sim \mathcal{N}(0, \sigma^2)$

for $k \leftarrow 1,\ r$ do

for all behavior capsule i：$w_{ij} \leftarrow \text{Softmax}(b_{ij})$

for all interest capsule j：$z_j = \sum_{i \in \mathbb{I}_u} w_{ij} S e_i$

for all interest capsule j：$u_j = \text{Squash}(z_j)$

for all behavior capsule i and interest capsule j：$b_{ij} \leftarrow u_j S e_i$

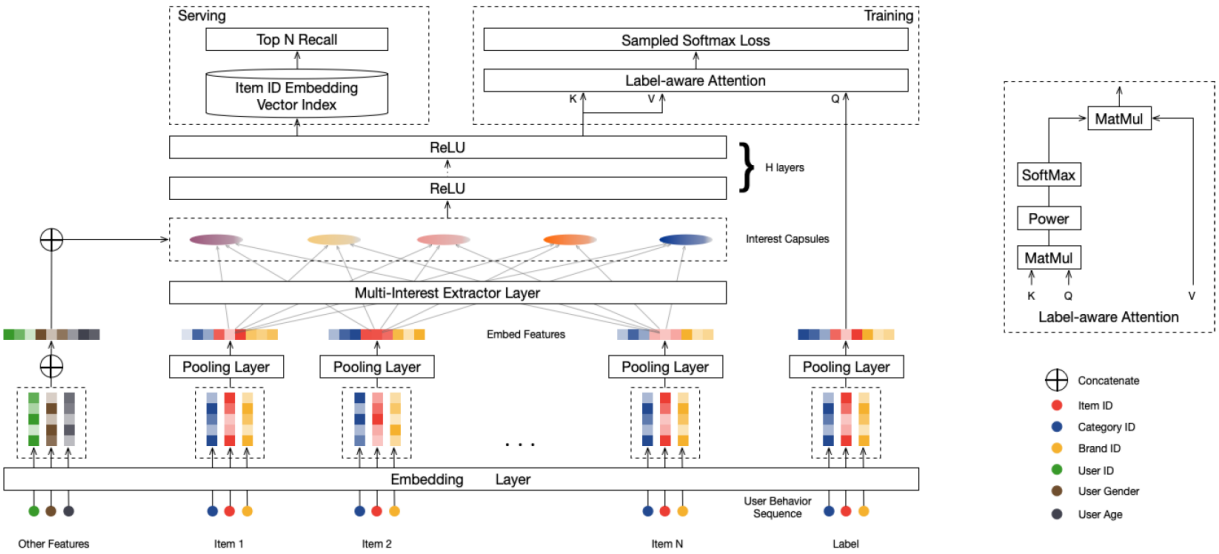
end for

return $\{u_j, j = 1, \ldots, K'_u\}$

- b_{ij}：表征behavior embedding i相对于interest capsule j的相关程度
- 该过程中，与u_j最相关的e_i，$u_j S e_i$内积越大，b_{ij}越大，在下轮迭代中更占主导。c_j会慢慢靠近更相关的e_i，远离不相关的e_i。

MIND

> Multi-Interest Network with Dynamic Routing for Recommendation @ Tmall (CIKM'19 Alibaba)



改进点：现有模型不足以学习多兴趣，用多向量学习多兴趣

学习目标：$V_u = f_{user}(\mathcal{I}_u, \mathcal{P}_u)$

- $\mathcal{I}_u$：user behavior
- $\mathcal{P}_u$：user profile

$e_i = f_{item}(\mathcal{F}_i)$，where $\mathcal{F}_i$：item label

使用：$f_{score}(V_u, e_i) = \max_k e_i^T v_u^k$

Embedding & Pooling Layer

用户特征、用户行为 -> embedding

输入：user profile、 user behavior、 label对应的target item

Multi-Interest Extractor Layer

胶囊网络的Dynamic Routing的特性对item做聚类

聚类后的k个embedding分别和user profile的embedding做concat，过一个投影->用户的多个兴趣embedding

Label-aware Attention Layer

对于target item，可以通过attention mechanism聚合用户的多个兴趣embedding，得到用户最终的embedding

$$v_u = \text{Attention}(e_i, V_u, V_u) = V_u \text{Softmax}(\text{pow}(V_u^T e_i, p))$$

其中p为超参数，p取0为平均，p取$+\infty$为one-hot

先聚合，再召回

Training

目标是v_u情况下，用户是否会和e_i交互

$$\Pr(i|u) = \Pr(e_i|v_u) = \frac{\exp(v_u^T e_i}{\sum_{j \in \mathcal{I}} \exp(v_u^T e_i)} \quad L = -\sum_{(u,i) \in \mathcal{D}} \log \Pr(i|u)$$

问题：用target label的方式训练存在训练测试不一致问题(偷看答案)

应用：学到的embed可以做召回，整个模型可以做精排

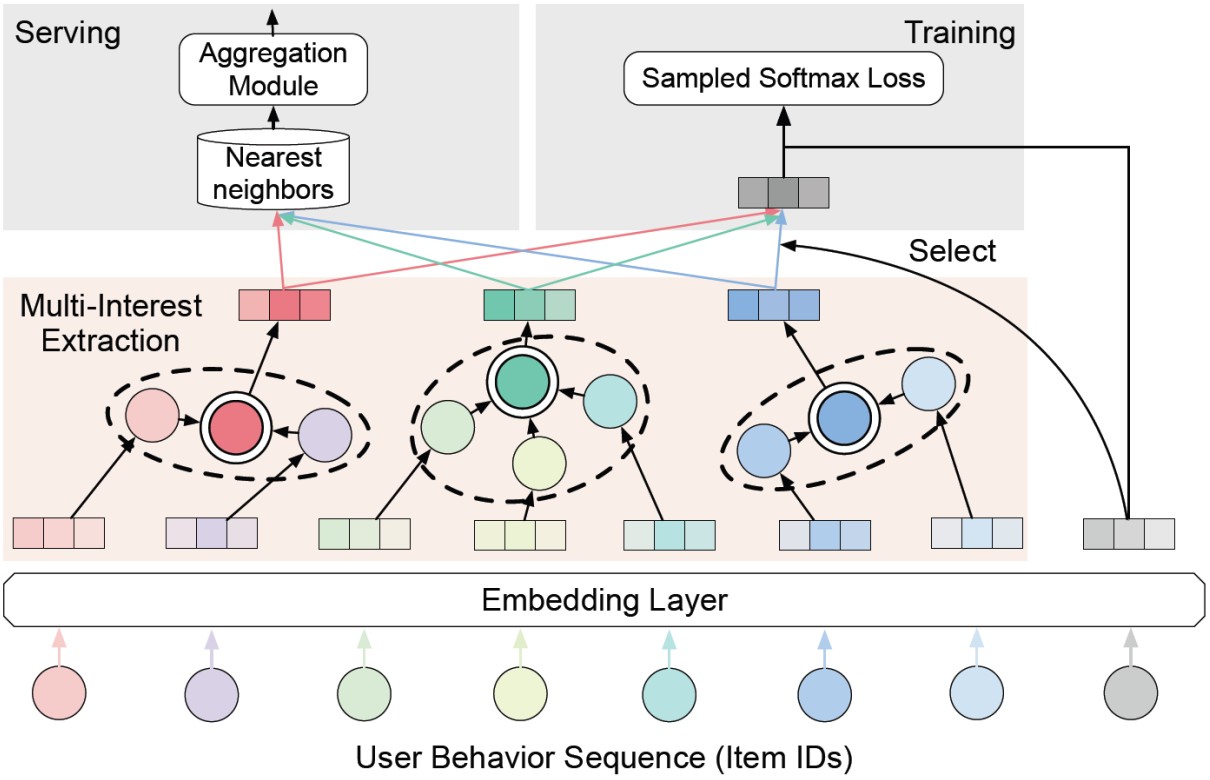
Serving

离线计算：每隔一段时间根据交互数据更新user的多个兴趣embedding并发布

线上召回：分区，用k近邻算用户向量所在的区块，再遍历区块算兴趣分，排序的topN去recall

ComiRec

> Controllable Multi-Interest Framework for Recommendation(SIGKDD'20 Alibaba)



Embedding Layer

item IDs -> item Embeddings

Multi-Interest Extraction

item Embeddings -> user interest Embeddings

- ComiRec-DR 用Dynamic Routing做Extraction，见MIND
- ComiRec-SA 用Self Attentive做Extraction

Self Attentive:

$$a = \mathrm{Softmax}(w_2^T \tanh(W_1 H))^T$$

- $H \in \mathbb{R}^{d \times n}$ user behaviors, where d: embedding length，n: user sequence length
- $w_2^{d_a \times 1}, W_1^{d_a \times d}$ learnable params
- $a^{n \times 1}$ attention weight
- H + learnable PE

since, user interest can be represented as $v_u = Ha$

multi-interest self attention：

做多次attention：$W_2^{d_a \times K}$

$$A = \mathrm{Softmax}(W_2^T \tanh(W_1 H))^T$$
$$V_u = HA$$

Aggregation Module

$$f(u,i) = \max_{1 \le k \le K} (e_i^T v_u^{(k)})$$

可以增加一个函数g表示多样性:

$$Q(u,\mathcal{S}) = \sum (i \in \mathcal{S}) f(u,i) + \lambda \sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{S}} g(i,j)$$

$$g(i,j) = \delta(\mathrm{CATE}(i) \neq \mathrm{CATE}(j))$$

δ是指示函数，CATE表示item的分类

Training

和目标embedding最相似的兴趣embedding会被挑选出来做sampled softmax, greedy训练更快

Serving

每个兴趣embedding找出和它最相近的topN items，然后放入aggregation模块，选出最终的topN recall

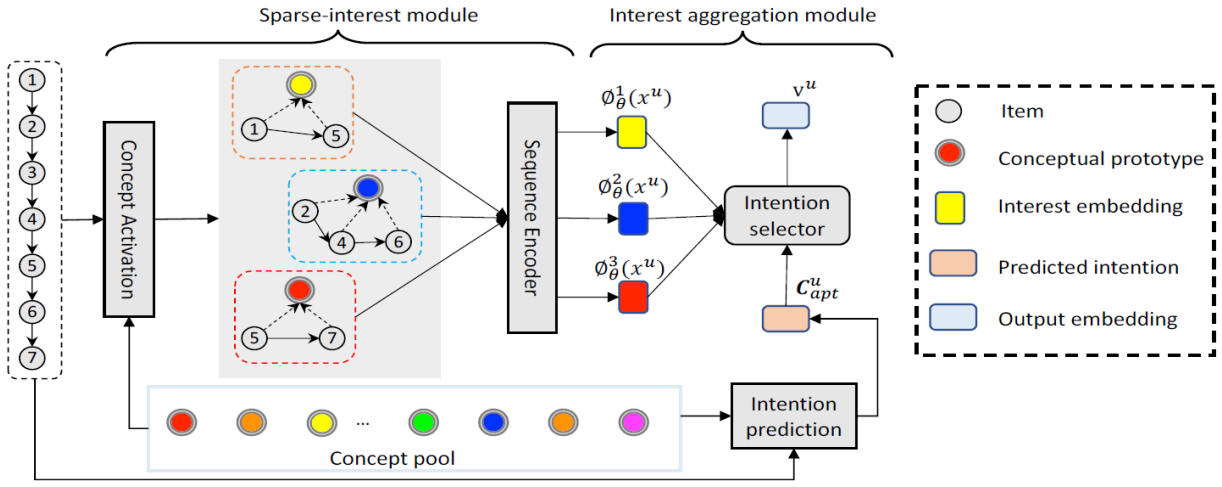
先召回，再聚合

最主要的问题：双塔独立，不好学习

解决方法：显式抽原型(SINE)，加强结合(MVKE)

SINE

> Sparse-Interest Network for Sequential Recommendation (WSDM'21 Alibaba)



利用概念原型对item进行大量聚类

一个人只和一组稀疏概念交互，需要从全量概念原型中抽取一部分

Sparse-Interest Framework

概念激活 Concept activation

根据user behavior sequence X_u，使用self-attentive生成用户概念向量表示z_u, 根据z_u和概念池的点积从L个概念池中提取topK概念向量

$$a = \text{Softmax}(\tanh(X_u W_1)W_2)$$
$$z_u = (a^T X_U)^T$$
$$s^u =< C, z_u >$$
$$idx = \text{rank}(s^u, K)$$
$$C^u = C(idx, :) \odot (\text{Sigmoid}(s^u(idx, :)1^T))$$

- $X_u \in \mathbb{R}^{n \times D}$ user behavior sequence
- $a \in \mathbb{R}^n$ attention
- $z_u \in \mathbb{R}^D$ user embedding
- $C \in \mathbb{R}^{L \times D}$ concept pool embeddings
- $C^u \in \mathbb{R}^{K \times D}$ final activated K latent concept embedding
- $< a, b >$ inner product
- $\odot$ Hadamard product (element-wise product).

Sigmoid Gate: 让离散的选择操作变得可微（可学习）

意图分配 Intention assignment

$p_{k|t}$：位于t处的item，属于兴趣k的概率(Softmax)

$$p_{k|t} = \frac{\exp(\text{LayerNorm}_1(X_t^u W_3) \cdot \text{LayerNorm}_2(C_k^u))}{\sum_{k'=1}^{K} \exp(\text{LayerNorm}_1(X_t^u W_3) \cdot \text{LayerNorm}_2(C_k'^u))}$$

其中, $C_k^u \in \mathbb{R}^D$

权重分配 Attention Weighting

$p_{t|k}$：针对兴趣k，位置t处的item的重要程度，加了PE

$$a^k = \text{Softmax}(\tanh(X^u W_{k,1})W_{k,2})^T$$

其中$a^k \in \mathbb{R}^n$

目的是估计位置t处的item对于预测用户的下一个意图至关重要的可能性

兴趣向量生成 Interest embedding generation

根据$p_{k|t}$和$p_{t|k}$计算用户的K个兴趣向量

$$\phi_\theta^k(x^{(u)}) = \text{LayerNorm}_3(\sum_{t=1}^{n} p_{k|t} \cdot p_{t|k} \cdot X_t^u)$$

其中$\phi_\theta^k(x^{(u)}) \in \mathbb{R}^D$

Interest Aggregation Module

target attention: 根据target item计算user embedding，MIND和ComiRec用的方法，存在训练测试不一致的问题。

no-target attenton: 预测用户当前的活跃兴趣生成embedding，训练测试一致。

$$\hat{X}^u = P_{k|t} C^u$$

$$C^u_{apt} = \text{LayerNorm}_4((\text{Softmax}(\tanh(\hat{X}^u W_3) W_4))^T \hat{X}_u)^T$$

$$e^u_k = \frac{\exp((C^u_{apt})^T \phi^k_\theta(x^{(u)})/\tau)}{\sum_{k'=1}^{K} \exp((C^u_{apt})^T \phi^{k'}_\theta(x^{(u)})/\tau)}$$

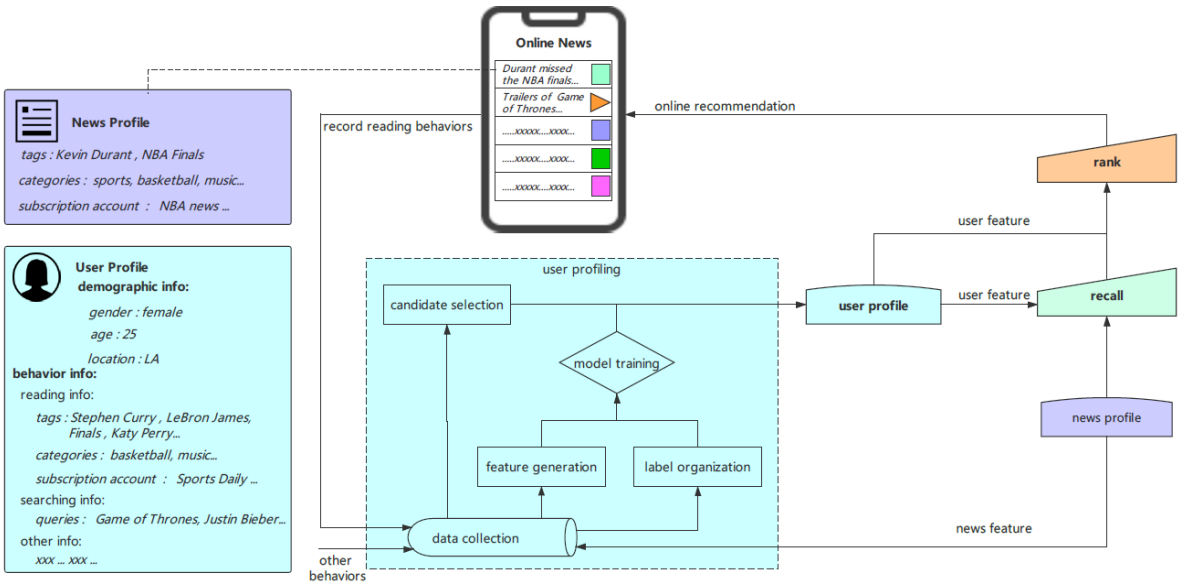
$$v_u = \sum_{k=1}^{K} e^u_k \cdot \phi^k_\theta(x^{(u)})$$

后两步: 将C_{apt}视为query, K个interest embedding 视为key和value，得到user embedding

- $X^u \in \mathbb{R}^{n \times D}$ 用户的意图序列
- $C^u_{apt} \in \mathbb{R}^D$ 预测用户当前活跃兴趣
- $e^u = [e^u_1, e^u_2, \cdots, e^u_K] \in \mathbb{R}^K$ 每个兴趣对当前活跃兴趣占的权重
- $v^u \in \mathbb{R}^D$ 用户当前兴趣向量

Learning to Build User-tag Profile in Recommendation System

> Learning to Build User-tag Profile in Recommendation System (CIKM'20 Tencent WXG)



多值特征

Feature-input layer

基础信息、历史统计信息 分别Embedding

Attention-fusion layer

自动选择有用的特征并学习不同字段内部和之间的特征之间的相互作用

- tags的pooling过程中avg和max没考虑权重，用了Multi-Head Self-Attention
- fusion过程中，两组使用的两个query是全局共享的，需要学习的参数

Cross-feature layer

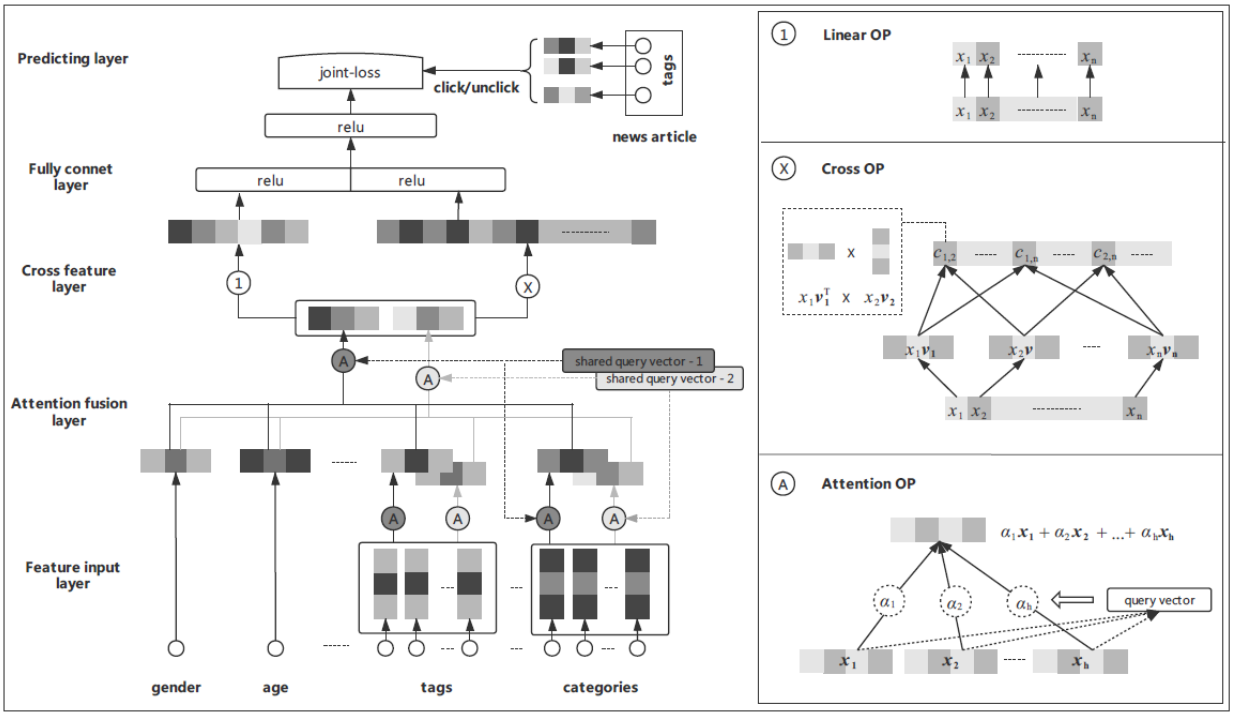
FM

Fully-connect layer

上层输出concat，过两层fc layers

Predicting layer

- motivation: 用户点击某个新闻，可能只是对其中的某个标签感兴趣
- 论文不是将点击的新闻标签集合做正例，未点击标签做负例，而是将新闻是否点击作为label
- 对某篇新闻的N个tags，转化为和用户向量u等长的向量t_i
  - $y_k = \text{Sigmoid}(\sum_{i=1}^{N} u \cdot t_i)$
  - $L = -\frac{1}{K}\sum_{k=1}^{K}(\hat{y}_k \log y_k + (1-\hat{y}_k)\log(1-y_k))$



MVKE

> Mixture of Virtual-Kernel Experts for Multi-Objective User Profile Modeling (SIGKDD'22 Tencent WXG)

认为双塔存在的问题：

- 两个塔是独立的，特征交互不足，影响建模效果
- 双塔模型很难表示用户在多主题上的多样兴趣
- 多任务学习(MMoE/ESMM)不足以应对用户多主题相关偏好的问题

Preliminaries

U个用户：偏好标签集$\mathcal{T}_u(u)$，点击集$\mathcal{C}(u)$，转化集$\mathcal{V}(u)$

$\mathcal{A}$个广告：广告标签集$\mathcal{T}_a(a)$

用户兴趣标签建模：$\mathcal{C}(u)+\mathcal{T}_a(a)\rightarrow\mathcal{T}_u^{\mathcal{C}}(u)$，预测CTR

用户意向标签建模：$\mathcal{V}(u)+\mathcal{T}_a(a)\rightarrow\mathcal{T}_u^{\mathcal{V}}(u)$，预测CVR

用户标签建模: $\mathcal{T}_u^{\mathcal{C}}(u)+\mathcal{T}_u^{\mathcal{V}}(u)\rightarrow\mathcal{T}_u(u)$

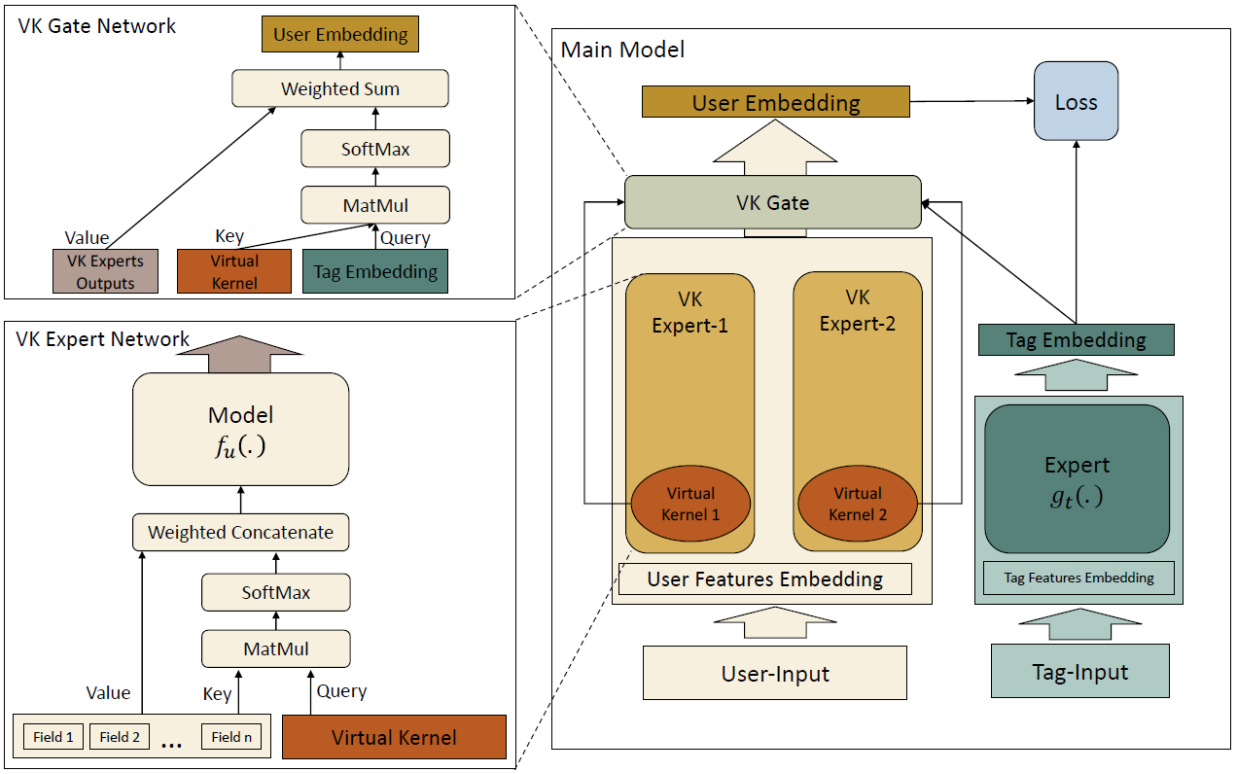
基础模型

- 用户塔：$E_{u_i}=f_u(u_i^1,u_i^2,\ldots,u_i^m;\Theta_u)$
- 标签塔：$E_{T_i}=E_{a_i}=g_t(\mathcal{T}_a(a_i);\Theta_t)$
- CTR / CVR：$p_i=\sigma(\cos(E_{u_i},E_{T_i}))$
- loss：$\mathcal{L}=\mathcal{L}_{BCE}(y,f_u(u;\Theta_u)\cdot g_t(\mathcal{T};\Theta_t))=\sum_i(y_i\log(p_i)+(1-y_i)\log(1-p_i))$

single-task MVKE

关键结构：

- VKE(Virtual-Kernel Experts) 每个VKE对应用户部分偏好，由virtual kernel表示
- VKG(Virtual-Kernel Gates) 根据不同的tag计算注意力得分，组合vke输出得到user embedding



VKE

VKE的数量是超参数，virtual-kernel是可学习变量，第k个VKE的virtual-kernel用W_{VK}^k表示

Q是virtual-kernel，K和V是用户embedding

$$Q = \sigma(W_Q^T W_{VK}^k + b_Q)$$
$$K = \sigma(W_K^T E_u + b_K)$$
$$V = \sigma(W_V^T E_u + b_V)$$
$$C_{VKE}^K = \mathrm{Softmax}(\frac{QK^T}{\sqrt{d_k}})V$$
$$E_u^k = f_u^k(C_{VKE}^k)$$

f_u^k是每个VKE独立的某种网络结构(如DeepFM, xDeepFM)，将Attention的输出做进一步变换，得到VKE的输出E_u^k

VKG

Attention拿到特定tag下的用户表示E_{u_i}

Q是tag embedding，K是virtual-kernel，V是VKE输出

$$E_{u_i} = \sum_k \mathrm{Softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

创新

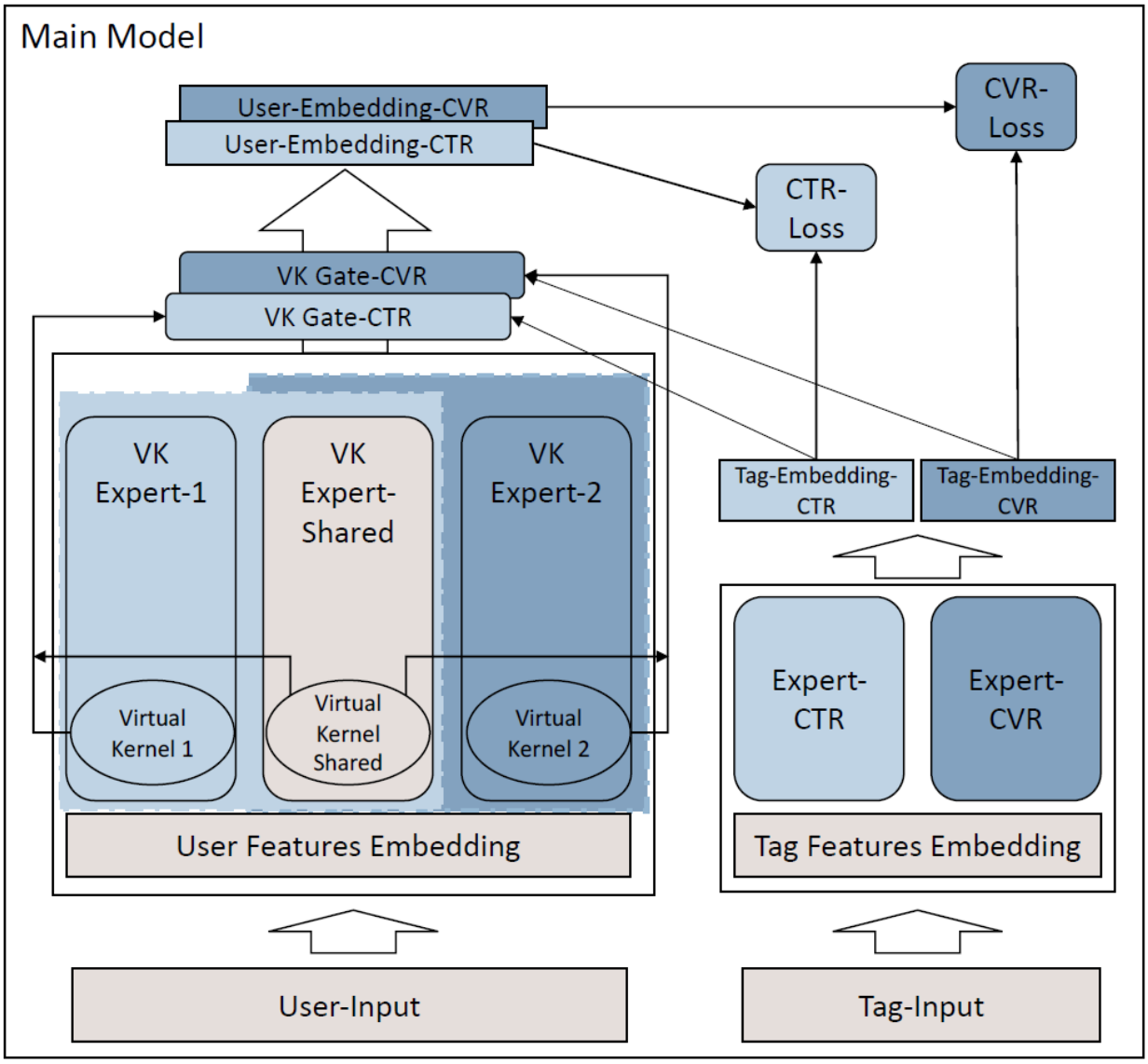
- virtual-kernel被用于VKE和VKG中，作为连接用户塔和标签塔的桥梁，让用户特征和标签特征更好交互
- VKE表示用户隐式偏好
- VKG对VKEs输出进行聚合得到用户表示

multi-task MVKE

设置更多VKEs：

- 每个任务用不同的VKEs子集
- 每个VKE可以用于不同的任务
- 同时设定每个任务独有的VKE和共享的VKE，既保证了模型的差异性（specialization），又保证了模型的泛化能力（generalization）。
- 不同任务的输出由VKEs决定，共享的VKE起到了不同任务间相互补充的作用。与此同时，为了保证不同任务训练的差异性（difference of training），每个任务至少有一个独立的VKE
- 对于有序列依赖关系的任务（如曝光-点击-转化）来说，越靠后的任务可以包含更多服务于上游任务的VKE，用于获取更多的上游任务的有用信息。如共有5个VKE，前三个用于CTR任务，而2到5用于CVR任务。

$$\mathcal{L}_{MTL} = \mathcal{L}_{ctr} + \mathcal{L}_{cvr}$$


Main Model
User-Embedding-CVR
User-Embedding-CTR
CVR-Loss
CTR-Loss
VK Gate-CVR
VK Gate-CTR
VK Expert-1
VK Expert-Shared
VK Expert-2
Tag-Embedding-CTR
Tag-Embedding-CVR
Virtual Kernel 1
Virtual Kernel Shared
Virtual Kernel 2
Expert-CTR
Expert-CVR
User Features Embedding
Tag Features Embedding
User-Input
Tag-Input