



# 相似人群扩展的深度学习解决方案

SML\_21团队

*Attention, Regularization, and More*



# 算法亮点

- 纯深度学习模型，训练速度快，内存要求低
- 精细的模型设计，单模型的性能优异
- 新的网络结构，往FFM中加入了attention机制
- 极少量人工特征，使用方便，工程量小



问题回顾

模型设计

特征工程

算法性能

经验总结





问题回顾

模型设计

特征工程

算法性能

经验总结



## 数据

- 用户特征：包含基本信息、ID、兴趣、关键词、近期app等
- 广告特征：包含广告ID、类别、广告商ID等
- 训练数据：(用户, 广告, 兴趣)

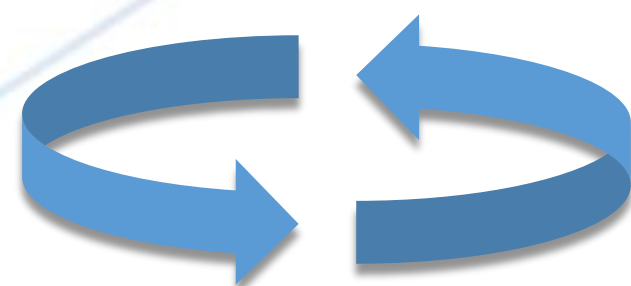
## 目标

- 预测新用户对某一广告是否感兴趣

	用户	广告	训练集	测试集
初赛	$1.1 \times 10^7$	173	$8.8 \times 10^6$	$2.3 \times 10^6$
复赛	$4.4 \times 10^7$	832	$4.6 \times 10^7$	$1.2 \times 10^7$

## 特点

- 只有离散特征
- 特征全部加密





## 挑战

- 数据量大
- 特征加密
- 特征类别多(kw1有20w+)
- 多值特征(interest等)

## 深度学习

- DeepFM/FFM
- Deep Interest Network

## 算法

### Gradient Boosting

- LightGBM
- XGBoost

## 选择

### 深度学习的优势

- 训练速度快(1 epoch)
- 可以用embedding直接学习特征表示, 无需明白特征数值具体含义
- 极少量特征工程, 节约预处理时间
- 占用内存少

- 利用attention处理多值特征
- 使用mini-batch aware正则化极大降低正则化的时间成本, 提升泛化能力
- 加入少量人工特征

## 优化



问题回顾

模型设计

特征工程

算法性能

经验总结

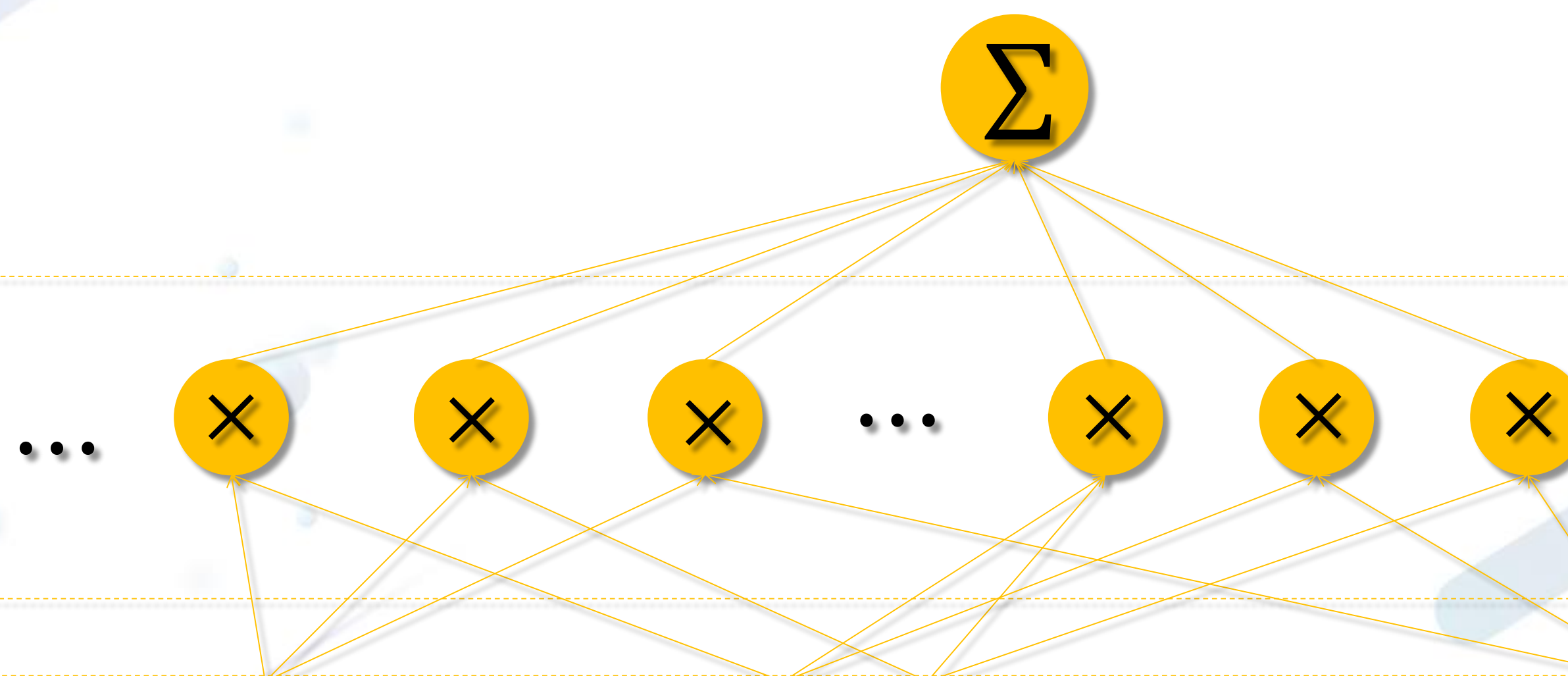




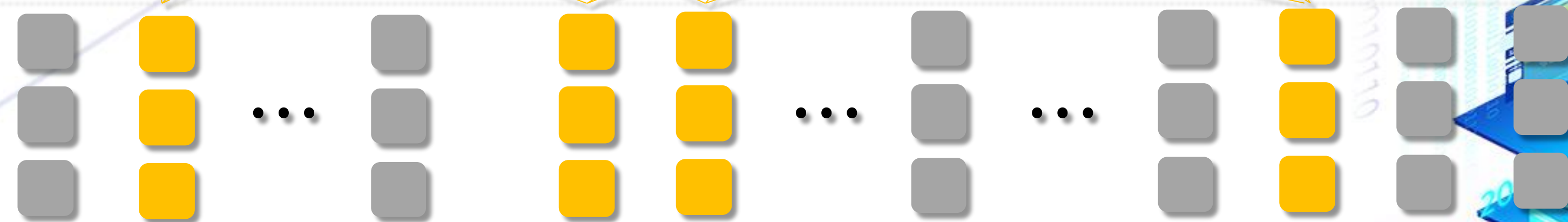
## ➤ FM

$$f = \sum_{i < j} \langle v_i, v_j \rangle x_i x_j + \sum_i w_i x_i + w_0$$

Inner Products



Dense Embedding



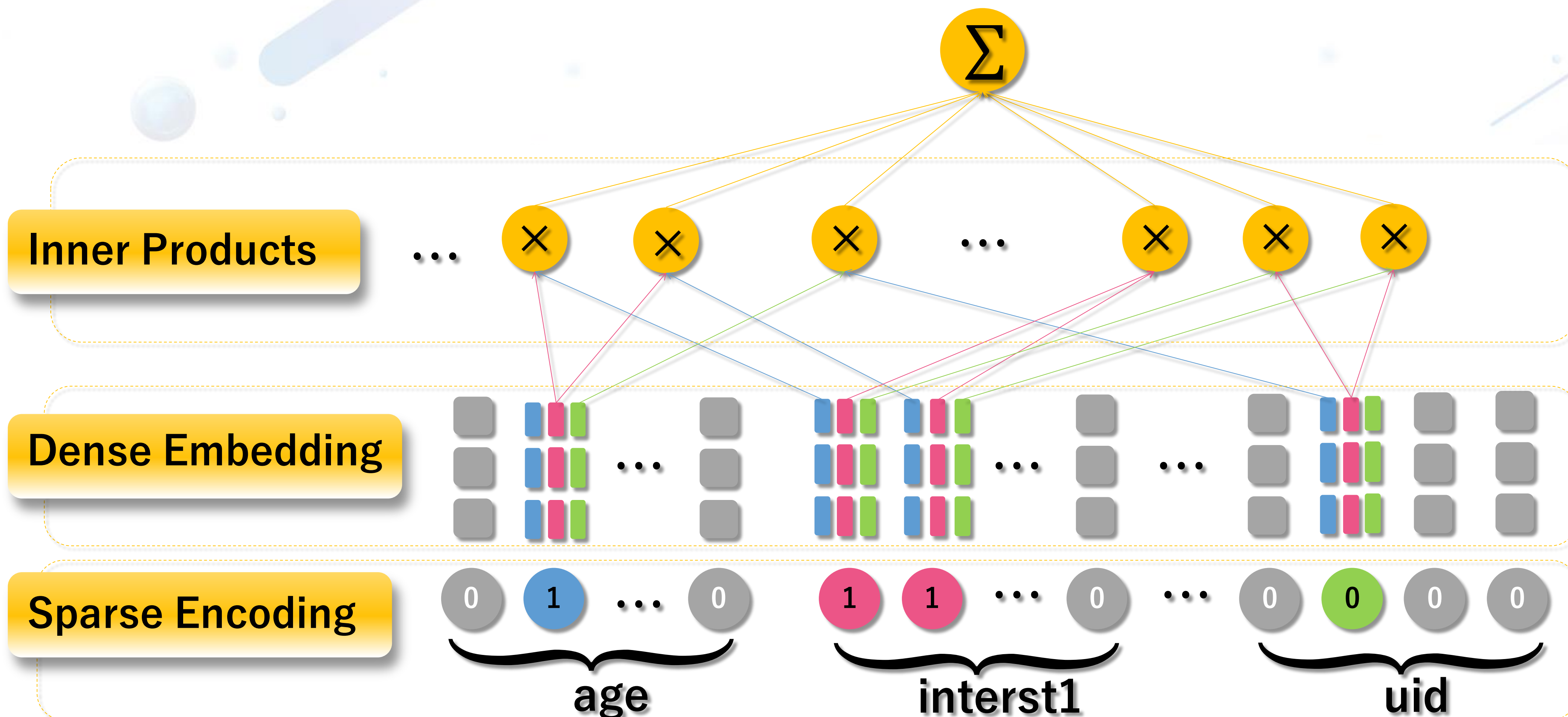
Sparse Encoding





## ➤ FFM<sup>[2]</sup>

$$f = \sum_{i < j} \langle \mathbf{v}_{i, \text{field}(j)}, \mathbf{v}_{j, \text{field}(i)} \rangle x_i x_j + \sum_i w_i x_i + w_0$$



# ➤ DeepFM<sup>[1]</sup>

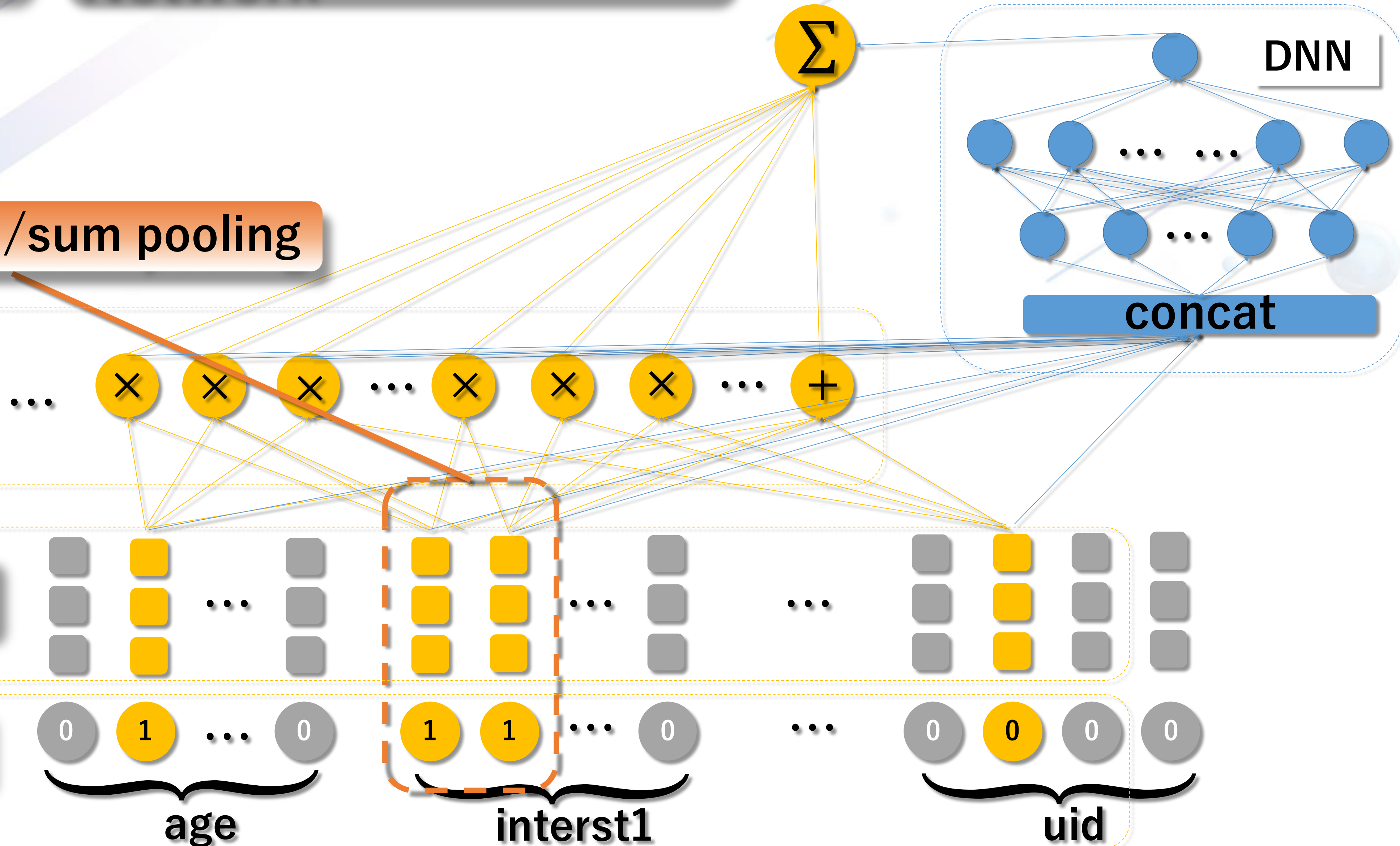
## FM + Neural Network

mean/sum pooling

Inner Products

Dense Embedding

Sparse Encoding

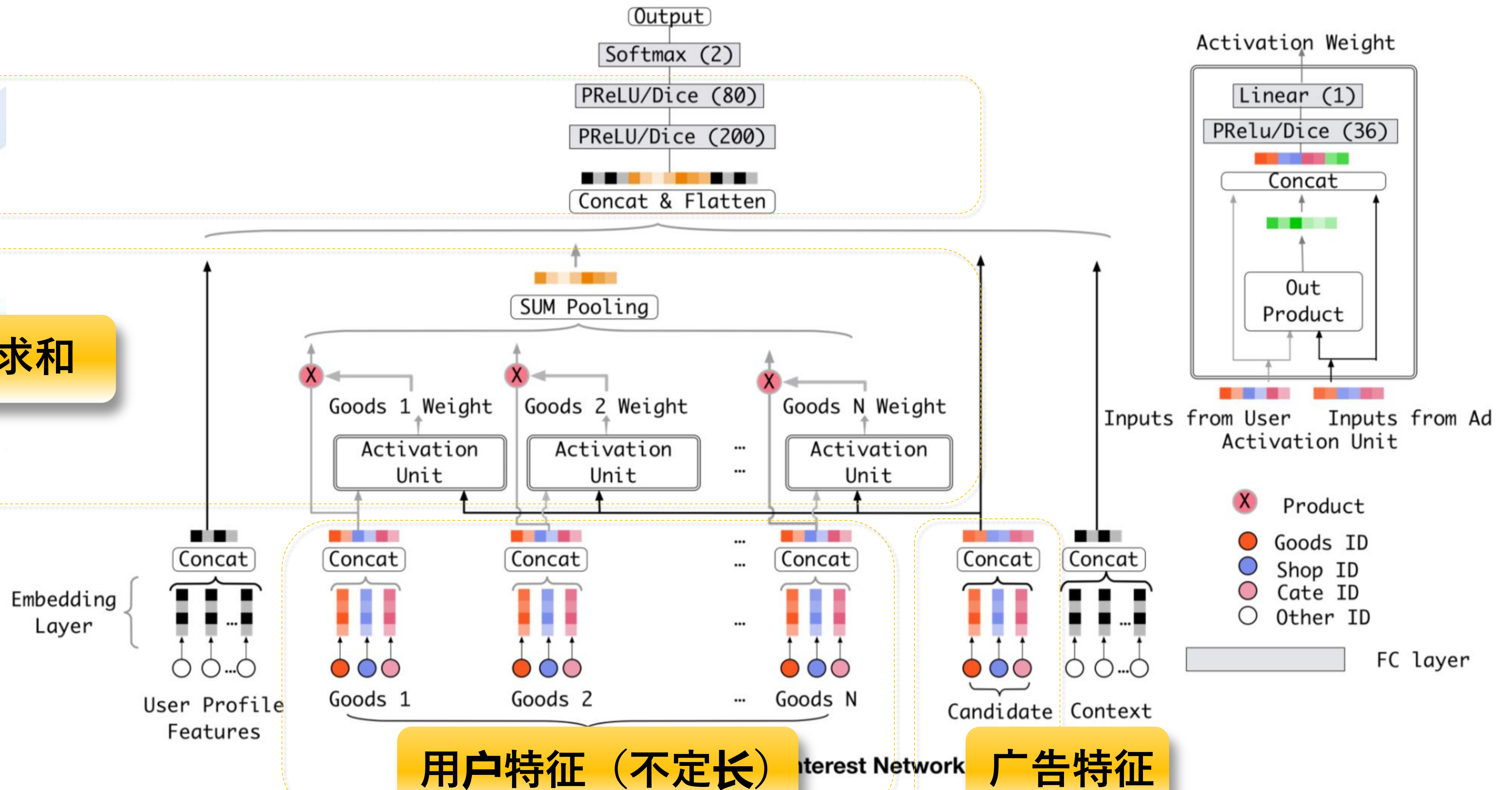




# ➤ Deep Interest Network (DIN)<sup>[3]</sup>

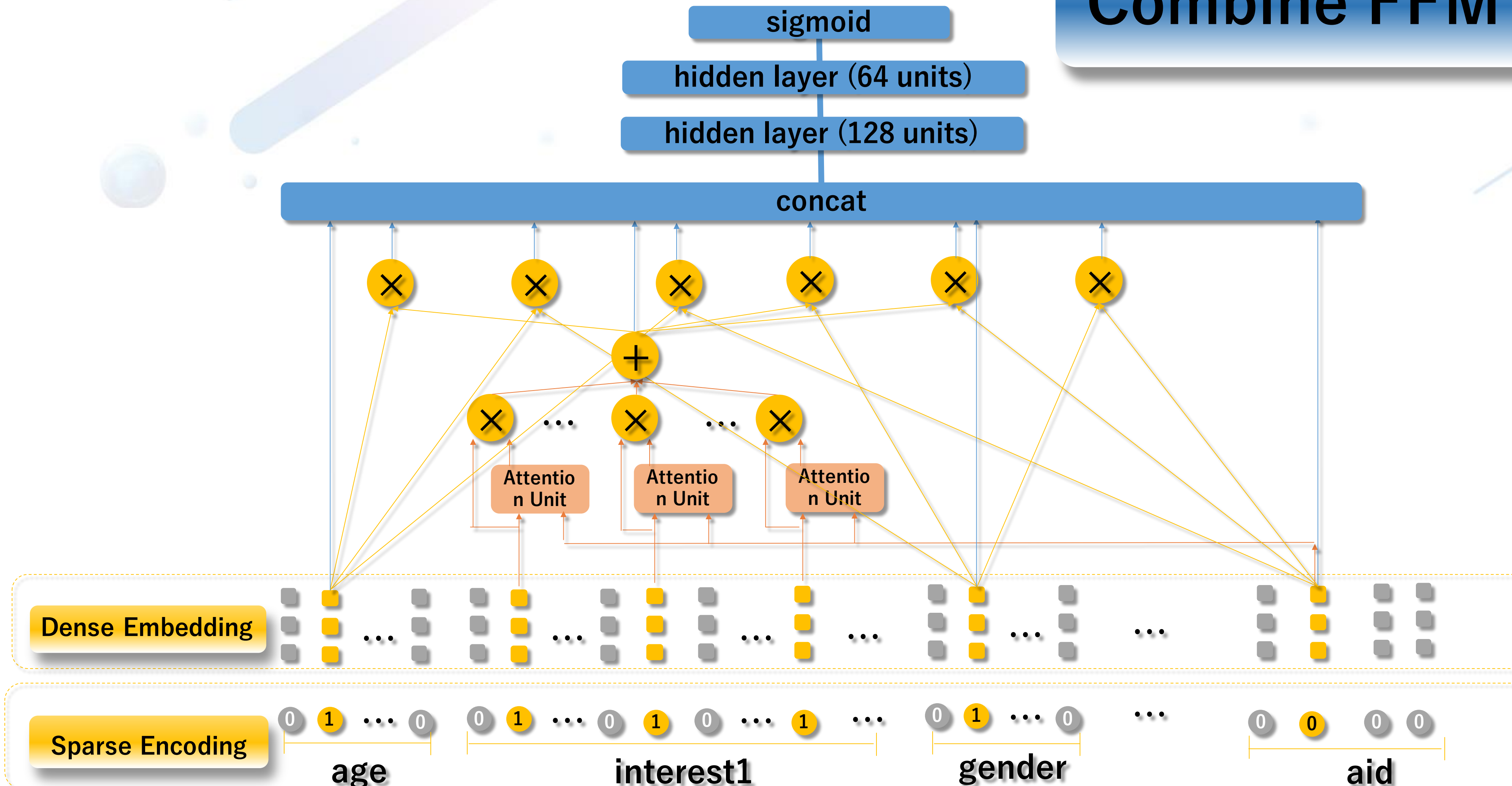
输入MLP

attention之后求和



# ➤ Our Architecture: DeepFFM + Attention<sup>[3]</sup>

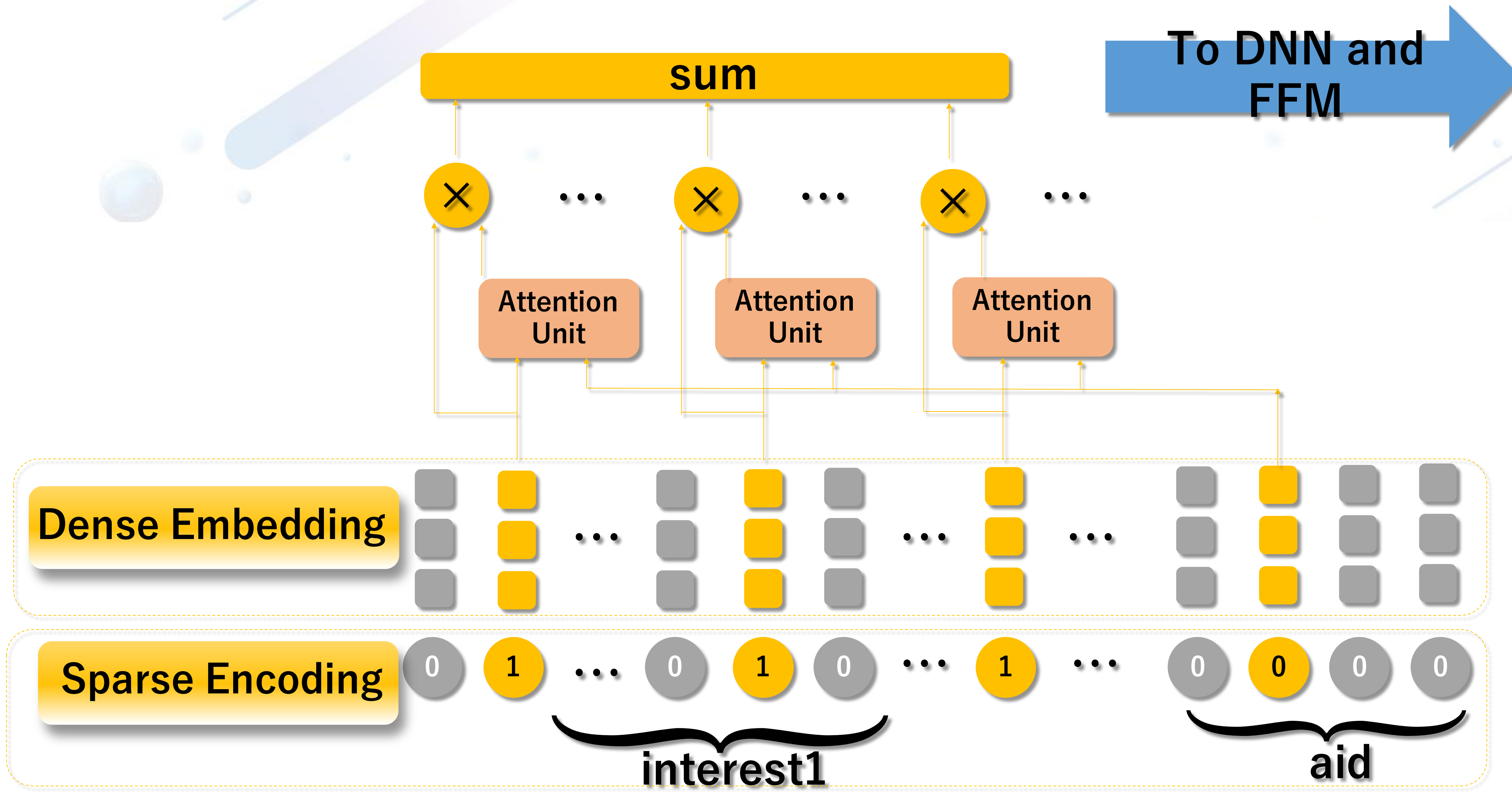
Combine FFM and DIN





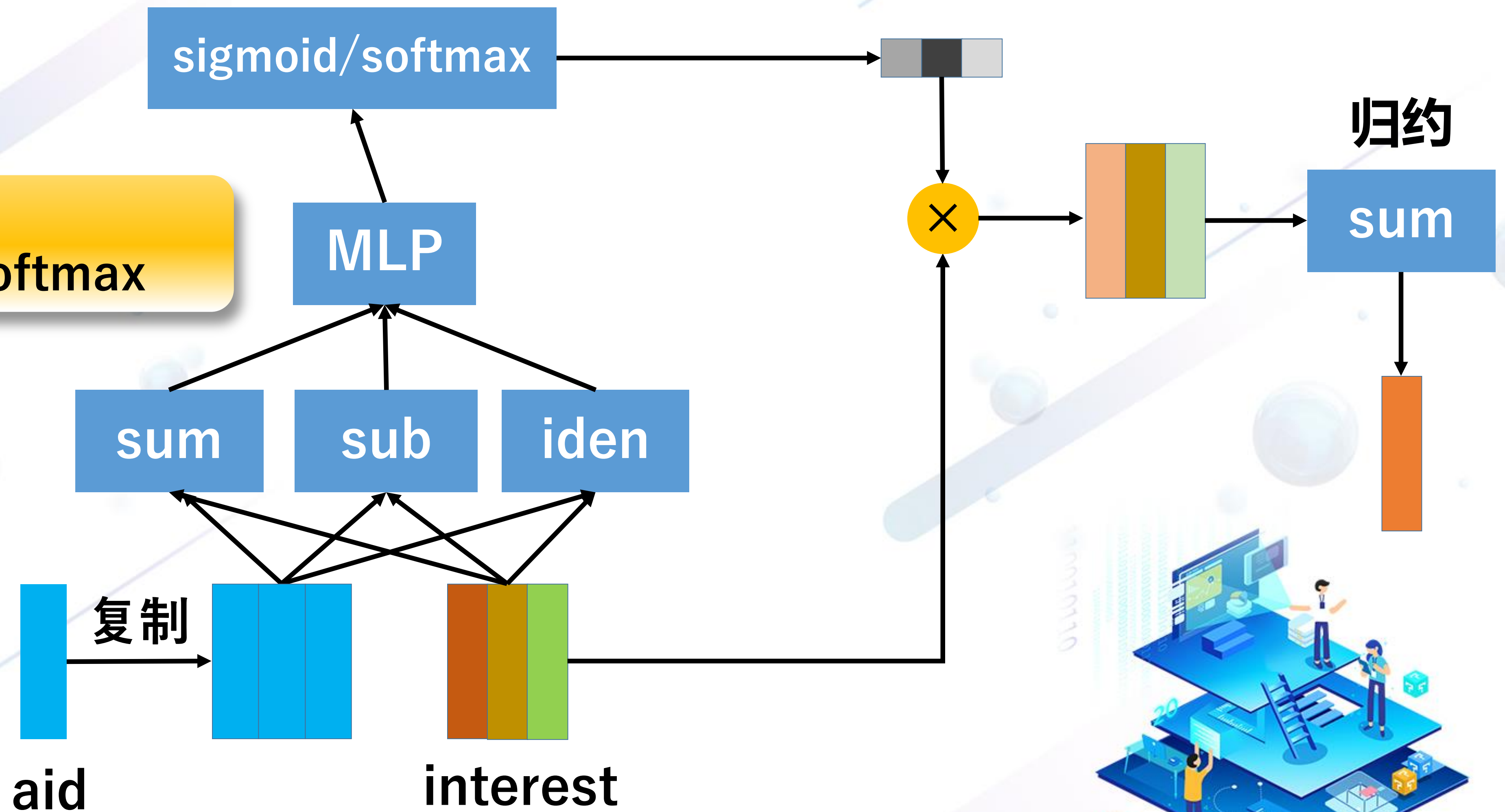
# ➤ DeepFFM + Attention<sup>[3]</sup>

Combine FFM and DIN



## ➤ Details of Attention

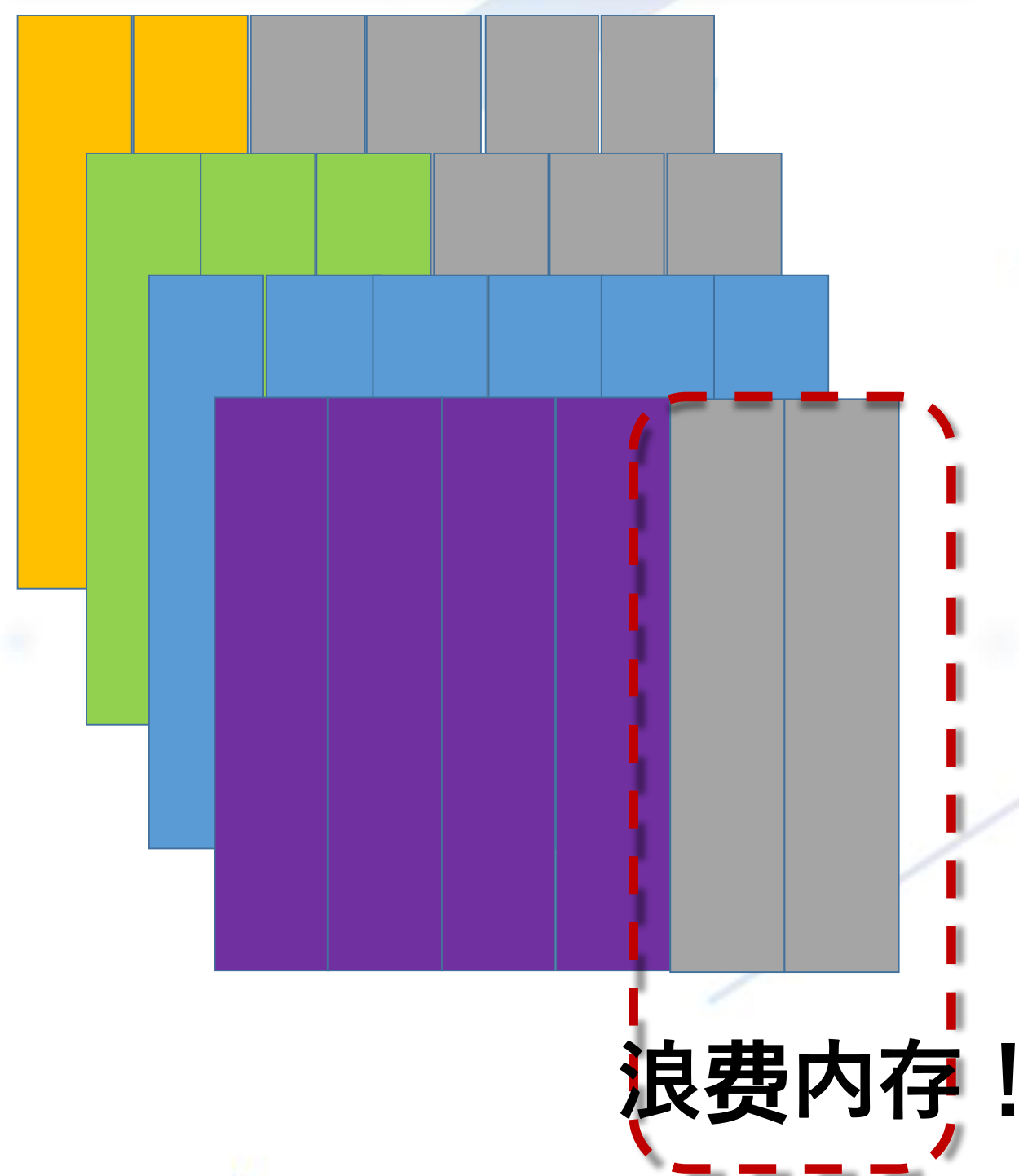
实践中  
sigmoid  $\geq$  softmax



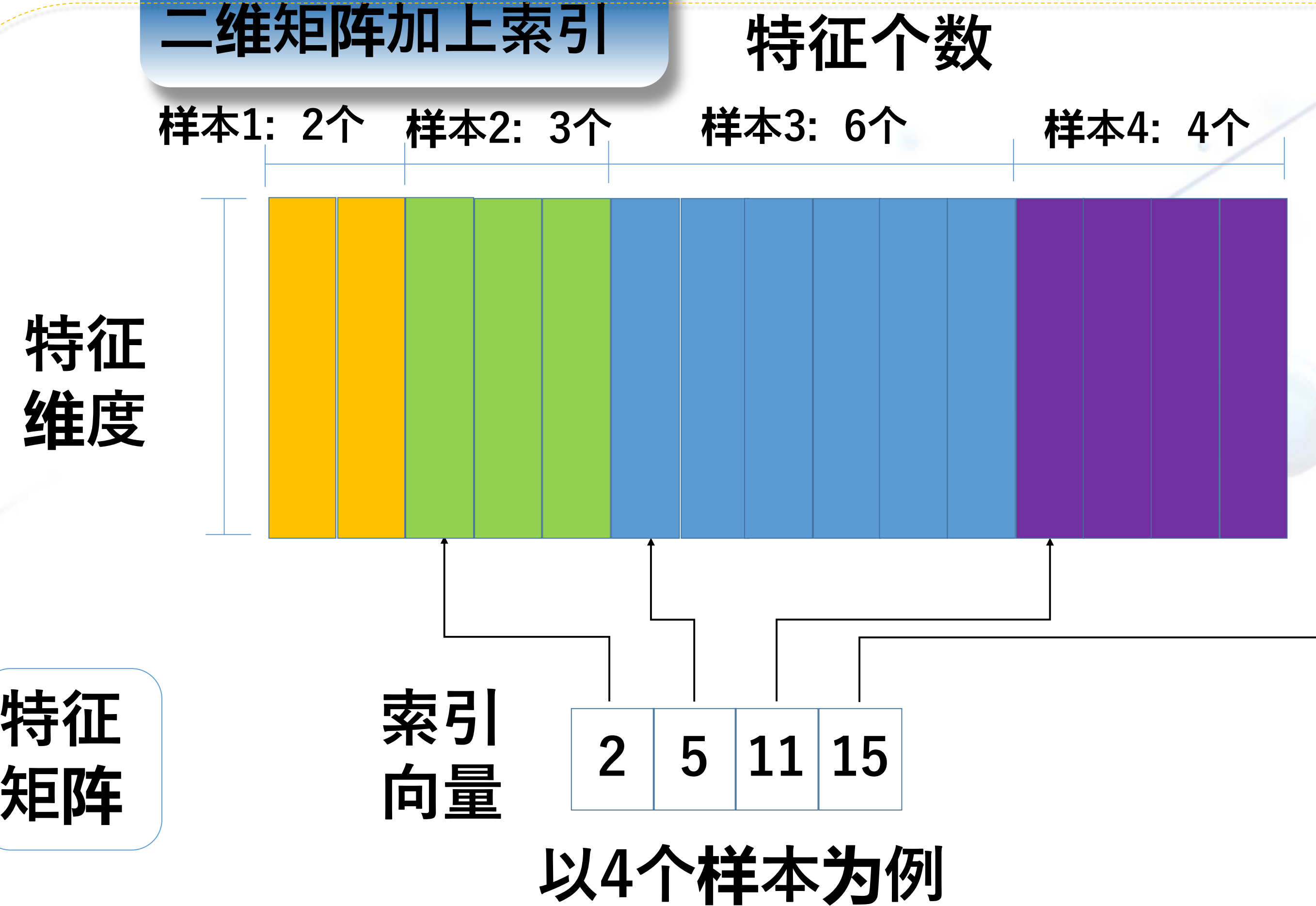


## ➤ 不定长个数的特征的处理

一般的方式: padding  
存储成三维tensor

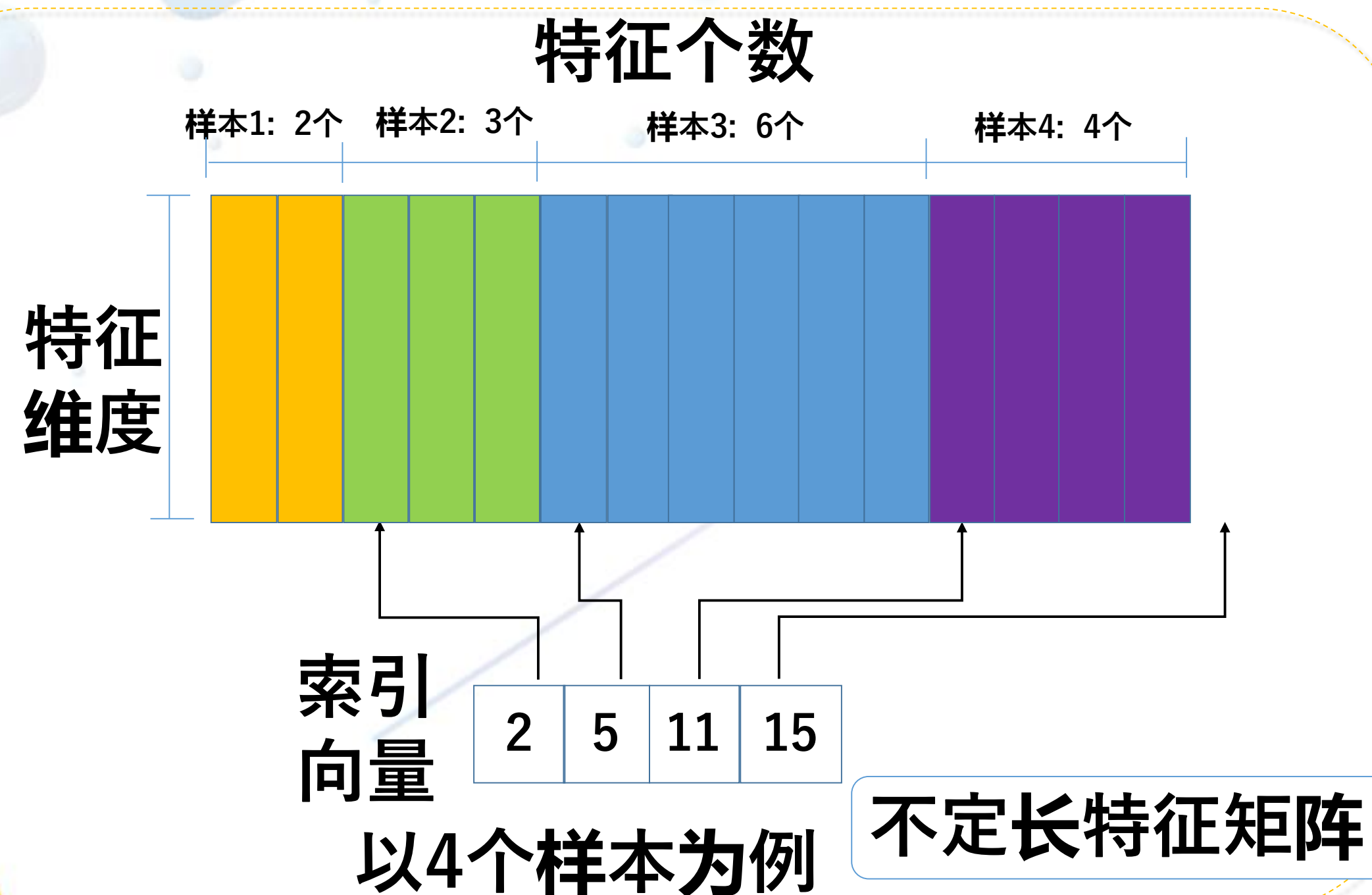


我们的方法:  
二维矩阵加上索引



## ➤ 不定长个数的特征的处理

### 归约与复制



归约



复制



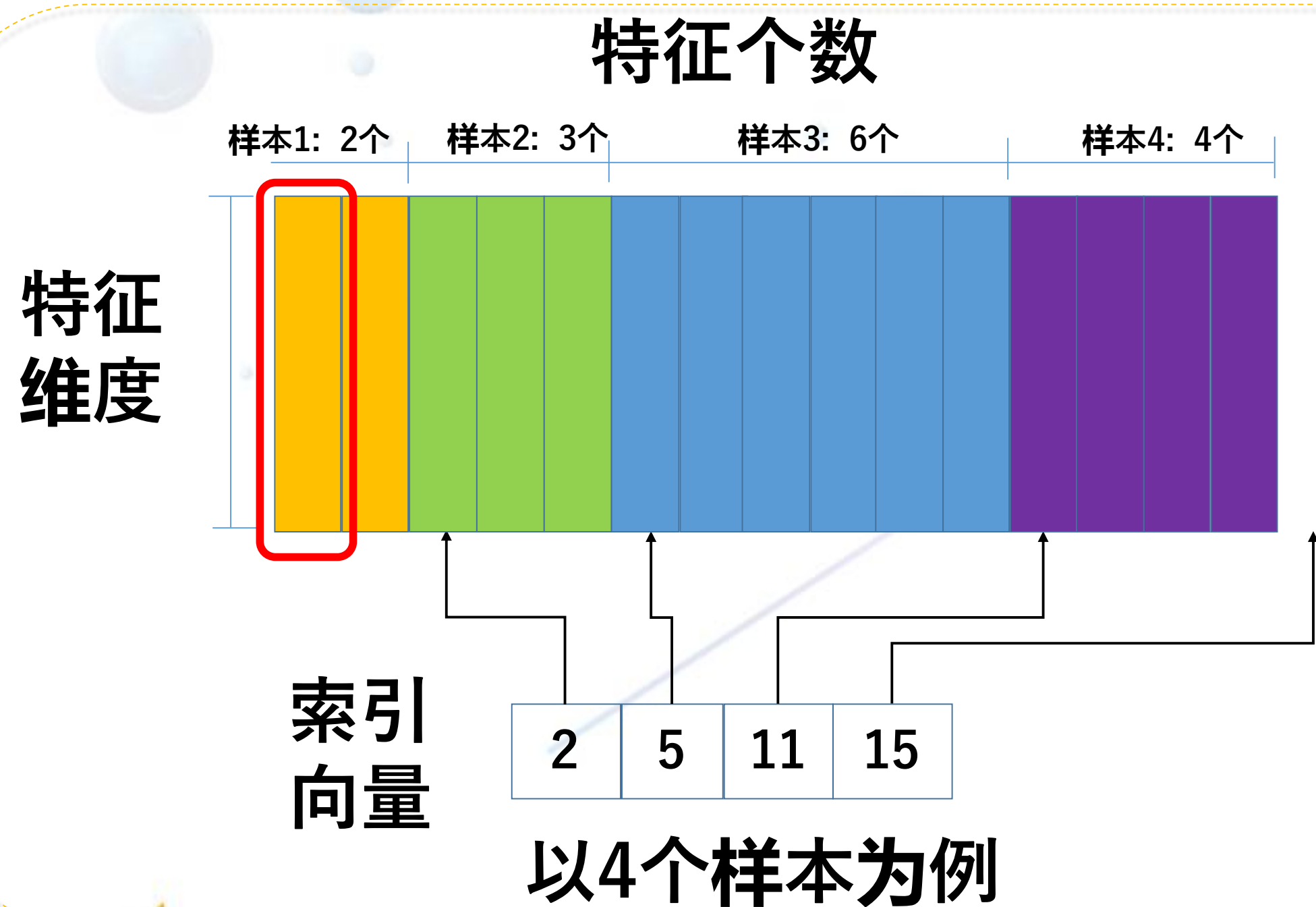
定长特征矩阵



## ➤ 不定长个数的特征的处理

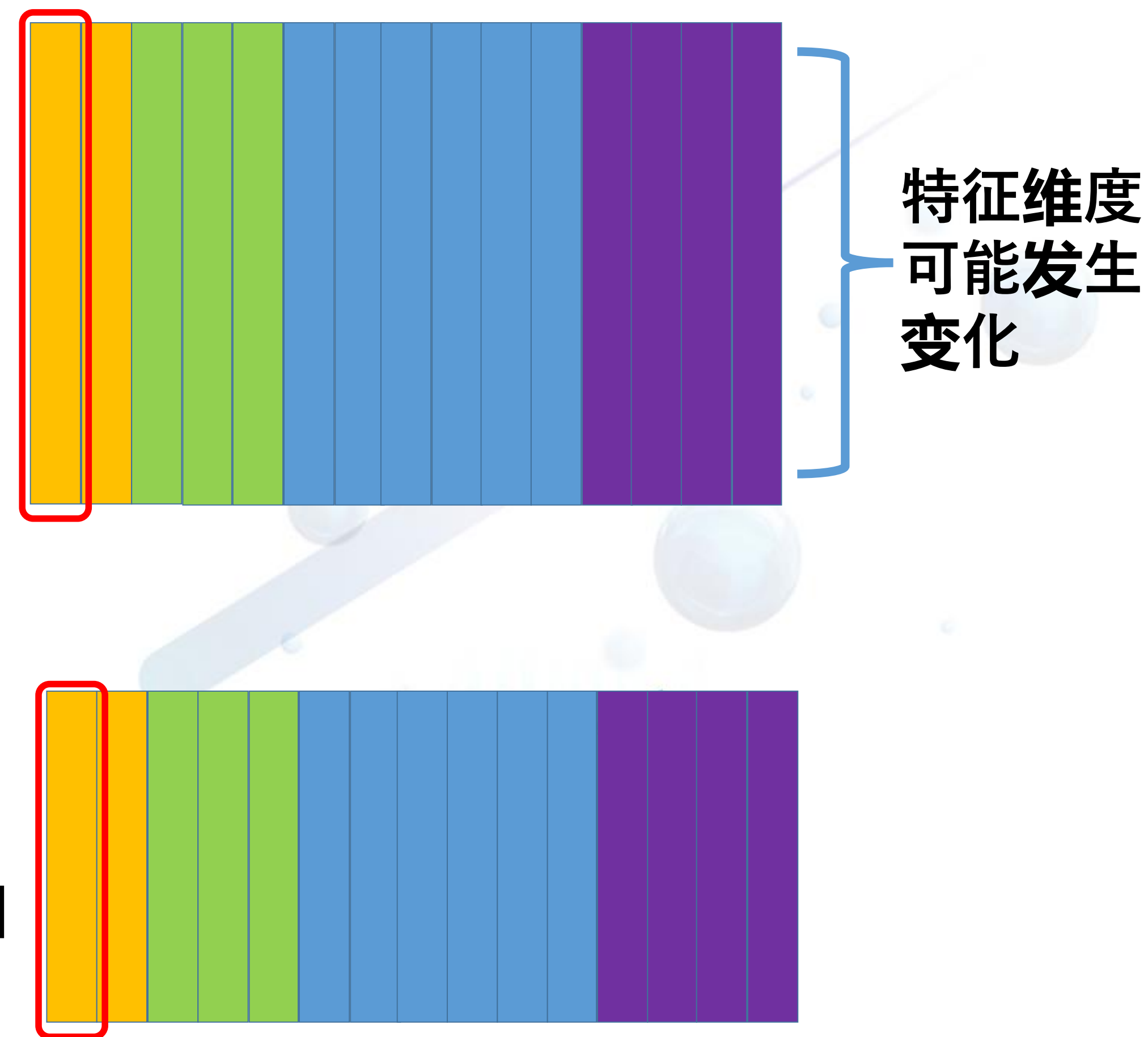
### 特征操作

和普通神经网络一样，可以高效并行



全连接

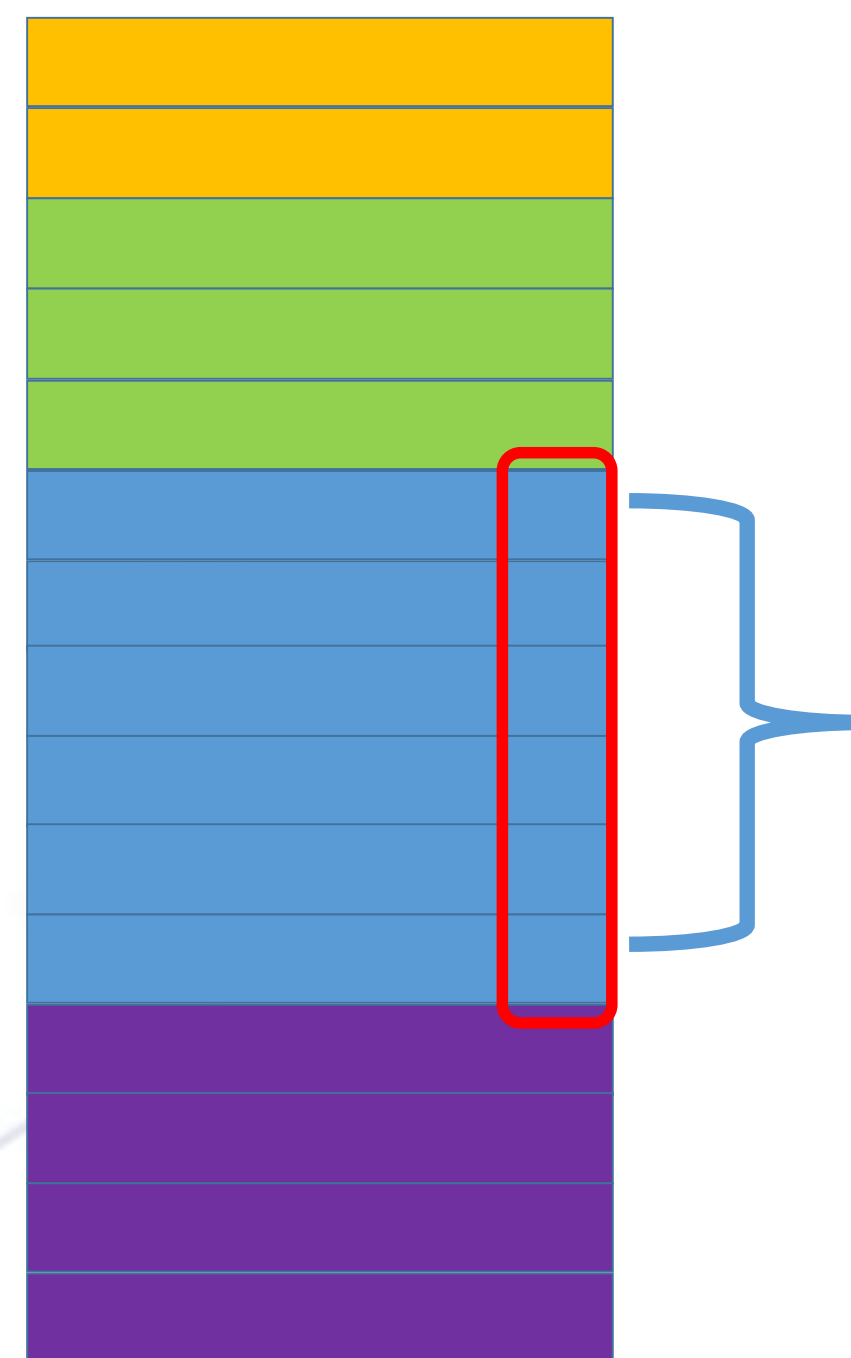
ReLU, sigmoid



## ➤ 不定长个数的特征的处理

### 样本操作

基本可以由前三个组合操作得到



对一个样本内不同特征做softmax: 需要一个对样本内所有特征的求和过程

$$\frac{\exp(f_i)}{\sum_i \exp(f_i)}$$





## ➤ 不定长个数的特征的处理

### 操作总结

#### 归约(Reduce)

求和

求最大

求平均

#### 复制(Replicate)

#### 特征操作

全连接

ReLU

sigmoid

#### 样本操作

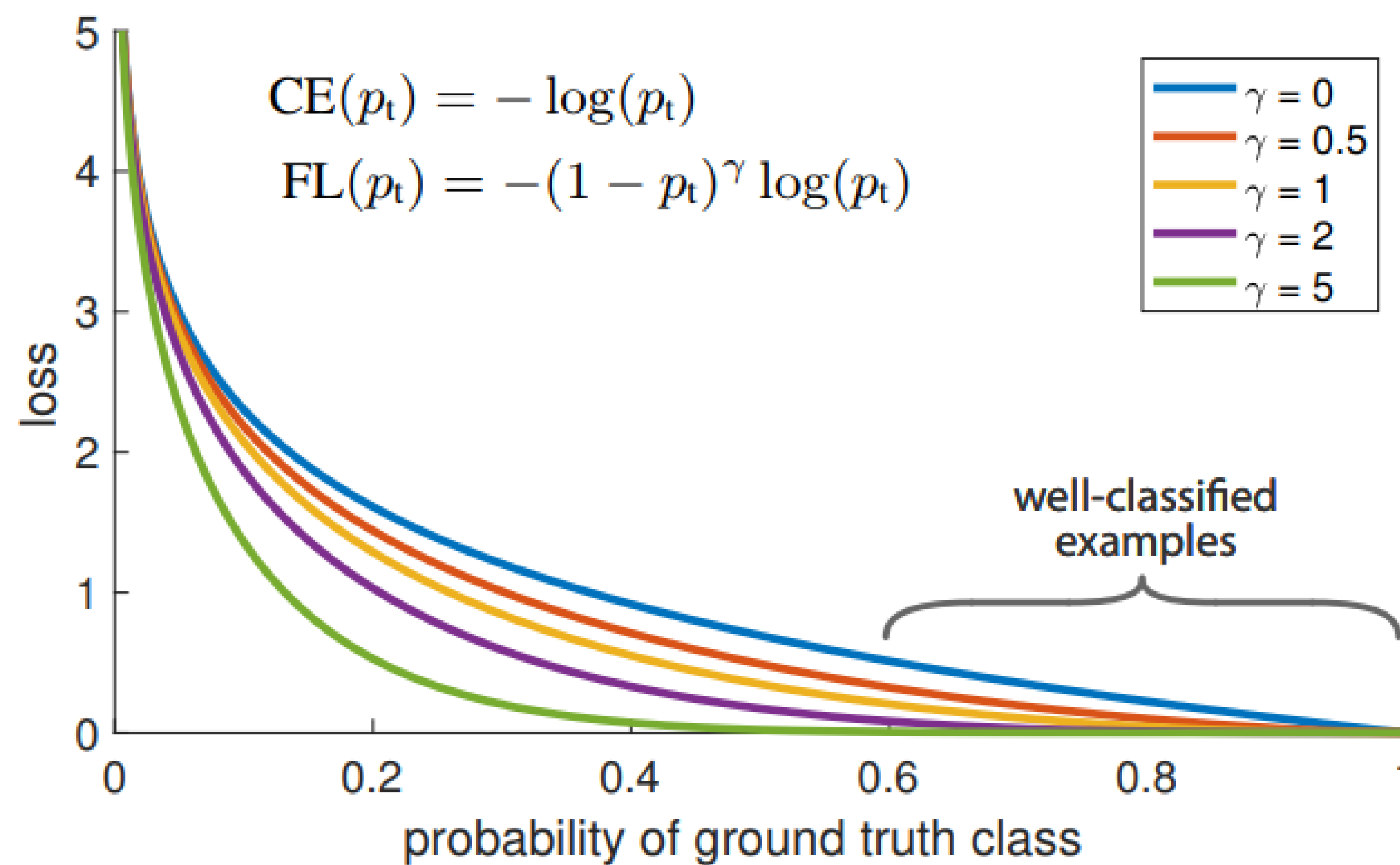
样本内softmax

样本内归一化



## ➤ Focal Loss

- 正负样本比例约为1:20
- 降低简单的负样本的权重



Cross entropy loss:  $CE(p_t) = -\log(p_t)$

Focal loss:  $CE(p_t) = -(1 - p_t)^\gamma \log(p_t)$

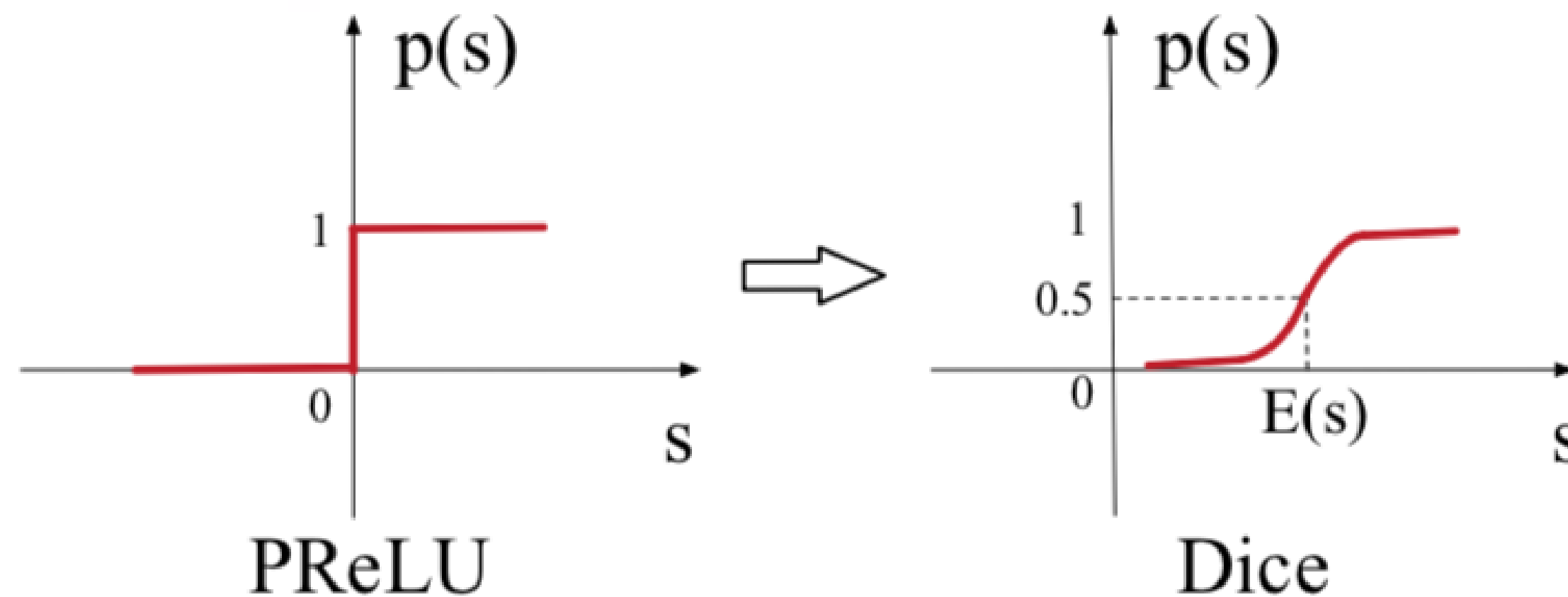
$p_t$ 为预测实际类别的概率





## ➤ Dice激励函数<sup>[3]</sup>

➤ PReLU:  $f(x) = \begin{cases} x, x > 0 \\ ax, x \leq 0 \end{cases} = p(x)x + (1 - p(x))ax$  其中  $p(x) = \begin{cases} 1, x > 0 \\ 0, x \leq 0 \end{cases}$



➤ Dice:  $f(x) = p(x)x + (1 - p(x))ax$  其中  $p(x) = \frac{1}{1 + \exp\left(-\frac{s - E[s]}{\sqrt{\text{var}[s]} + \epsilon}\right)}$

## ➤ Mini-Batch Aware Regularization<sup>[3]</sup>

- Embedding参数量大(例如kw1有200,000+), 需要正则化
- 希望特征数值越稀有,  $w_j$ 约束越强
- 只考虑每个mini-batch中使用过的embedding参数

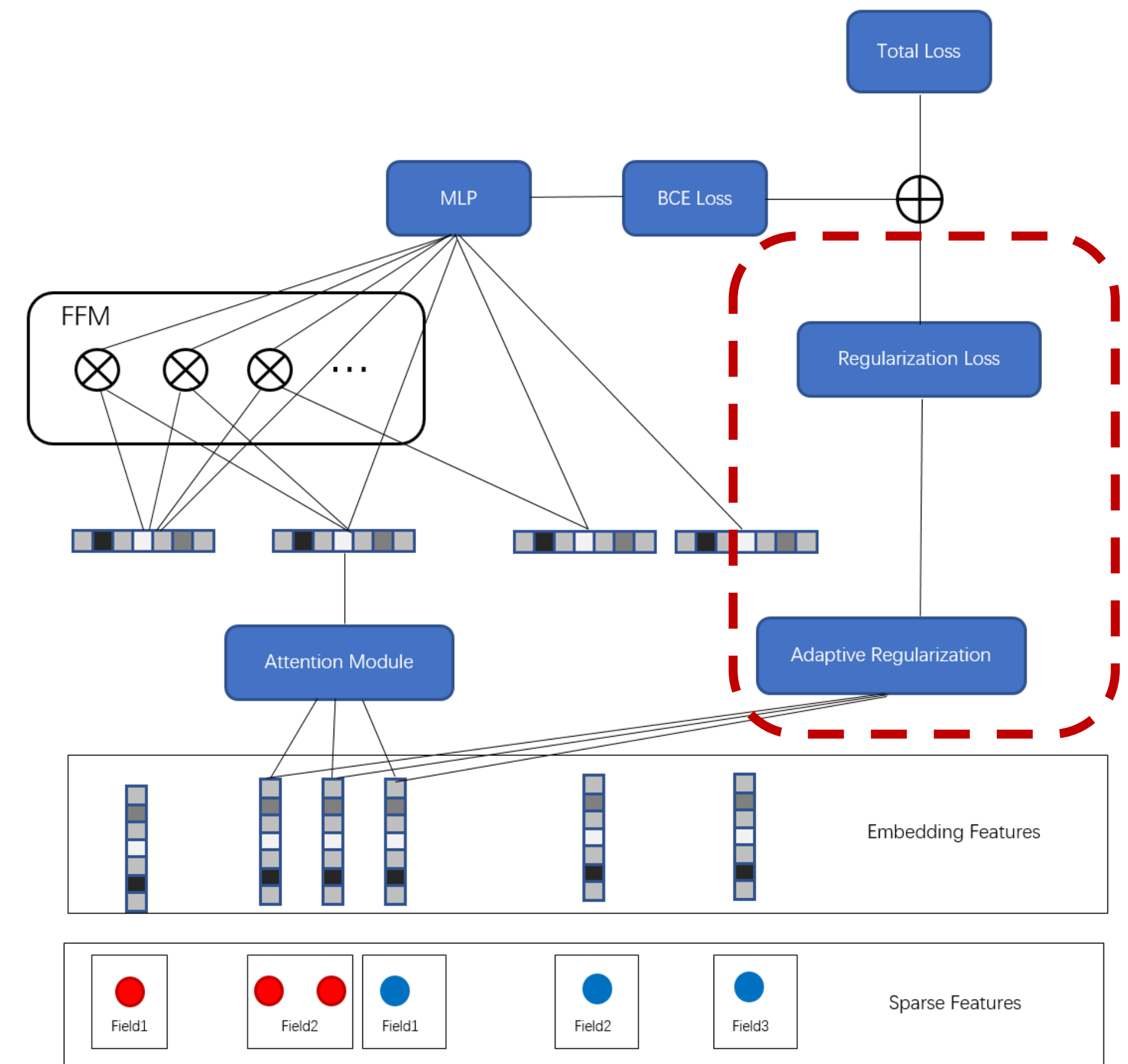
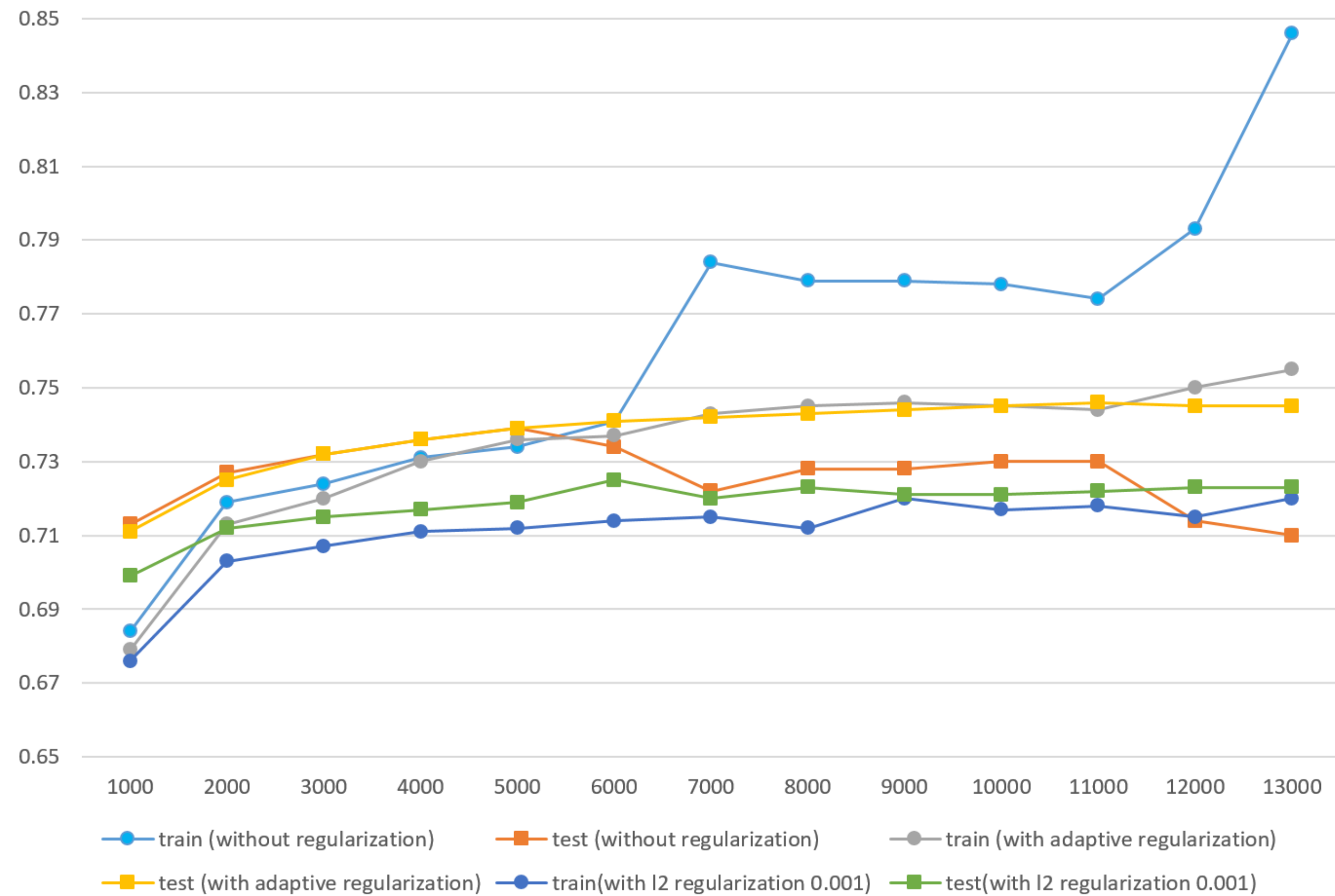
$$L_2(W) = \sum_{j=1}^K \sum_{m=1}^B \frac{\alpha_{mj}}{n_j} \|w_j\|_2^2$$

- $B$ : mini-batch 的数目
- $\alpha_{mj}$ : 特征的值  $j$  是否在第  $m$  个 batch 中出现
- $n_j$ : 特征的值  $j$  在整个数据集中出现的次数
- $w_j$ :  $j$  对应的 embedding 向量



# ➤ Mini-Batch Aware Regularization<sup>[3]</sup>

AUC vs Training Batches

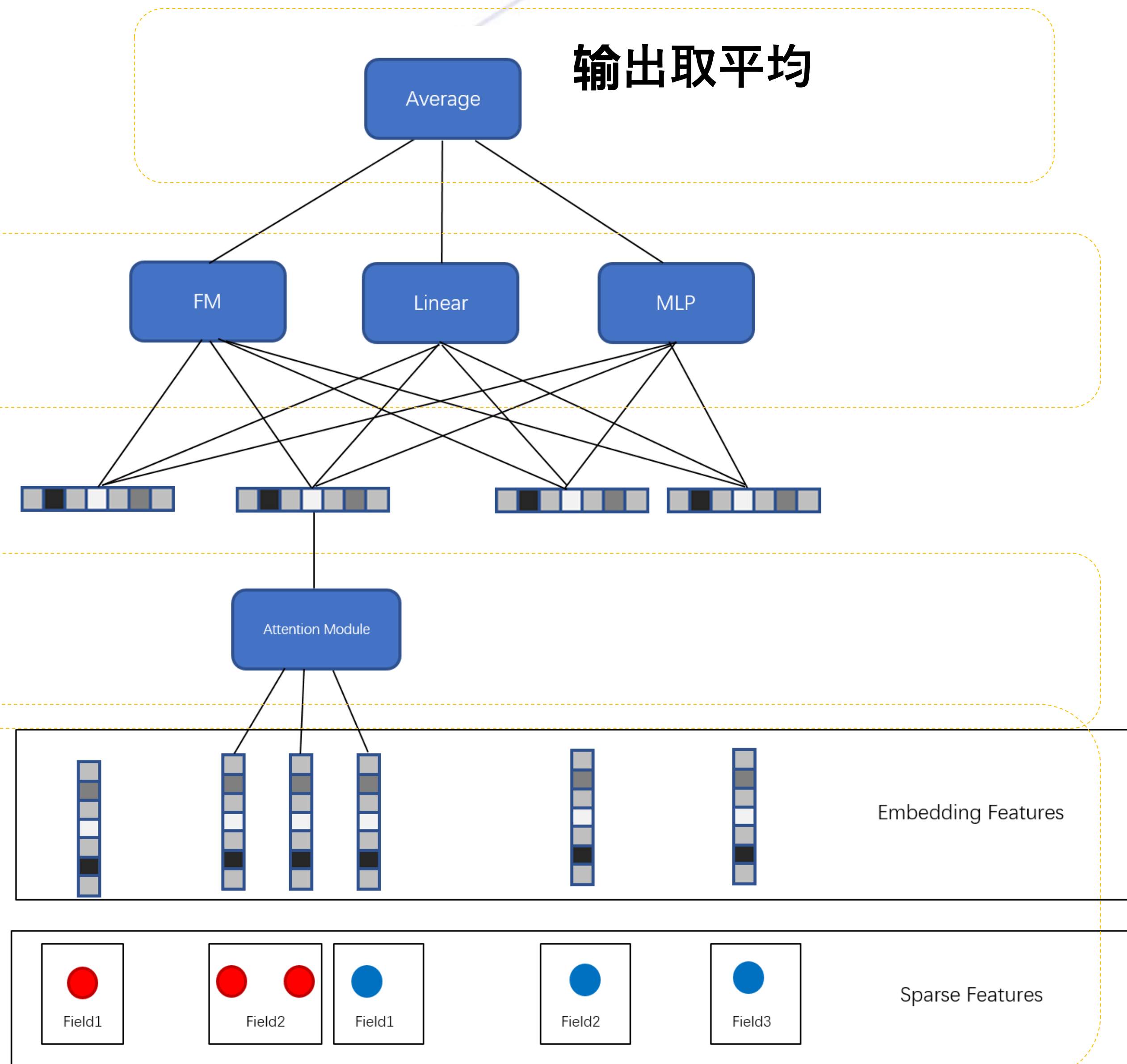


# ➤ DeepFM + Attention<sup>[3]</sup>

同时输入到FM、线性模型、MLP中

Attention

获取embedding向量





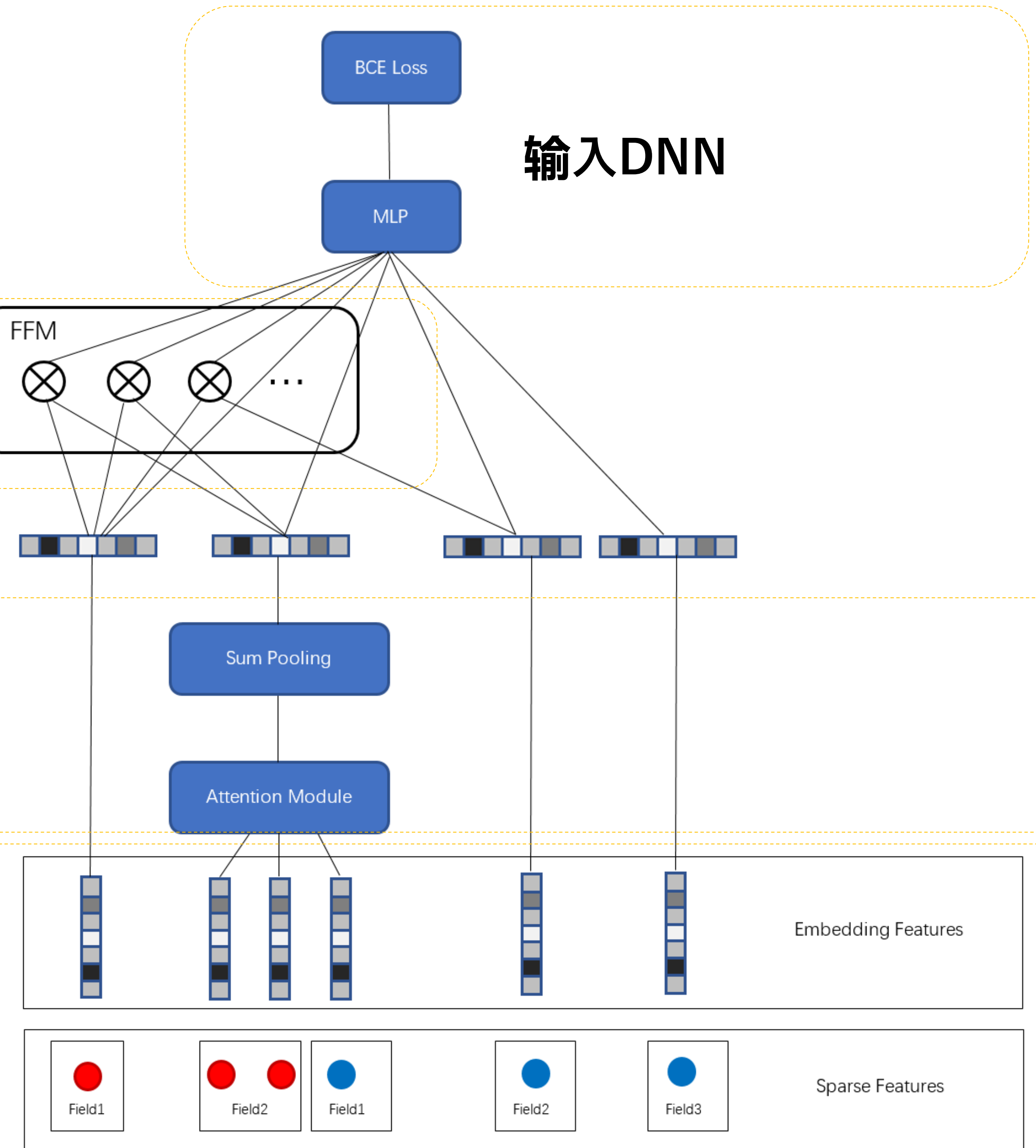
## ➤ 将FFM输入到DNN中

输入FFM

Attention

获取embedding向量

输入DNN



## ➤ Summation VS Concatenation

每个模块，如FFM可以看作是单纯的特征提取器，摆脱了原来的数学上解释，从而可以变得更加灵活

Sum/Avg

Sub-Model  
1

Sub-Model  
2

Sub-Model  
3

VS

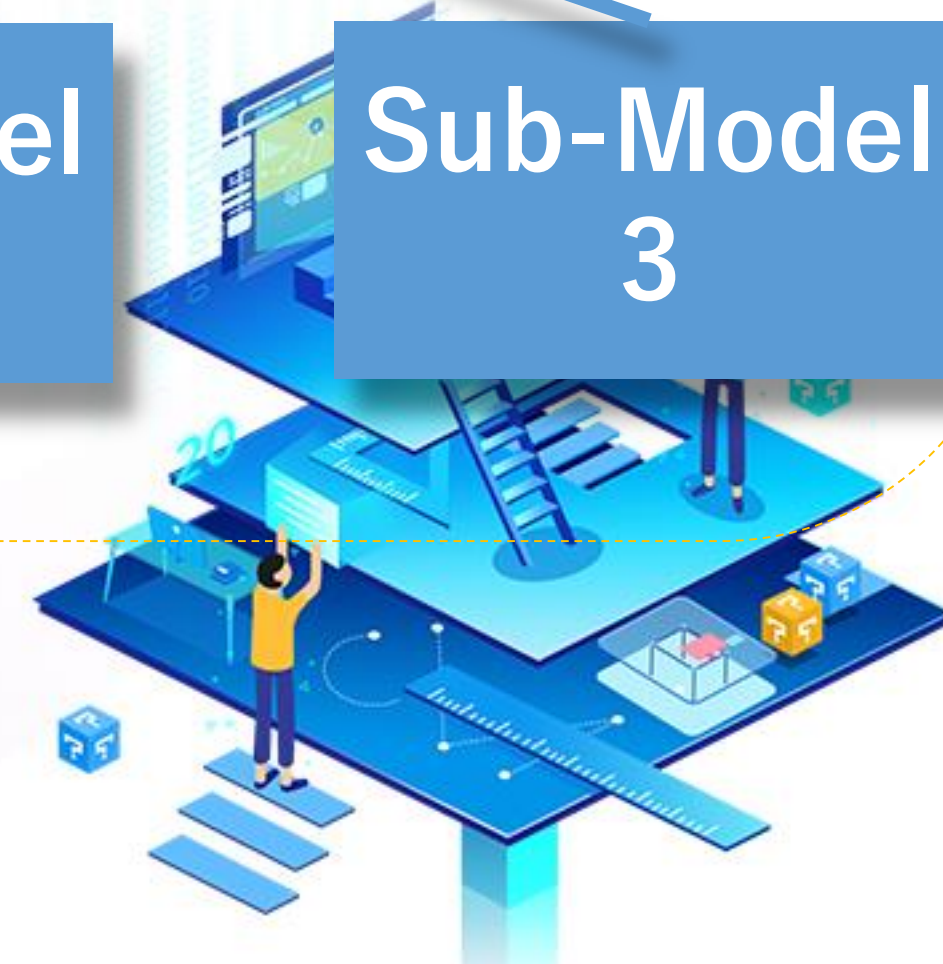
MLP

Concat

Sub-Model  
1

Sub-Model  
2

Sub-Model  
3





### ➤ 样本权重

- 正负样本赋予不同权值，我们正样本权值维10，负样本为1

### ➤ 去低频特征

- 将出现次数小于20的特征数值去掉

### ➤ Batch-Norm

- 用在全连接层和ReLU之间
- 第一层神经网络的输入也做了batch-norm

### ➤ 模型融合

- 训练了13个模型
- 不同的随机种子
- 将预测概率取平均



问题回顾

模型设计

特征工程

算法性能

经验总结





➤ 从历史数据中挖掘特征

➤ 基本思路：记录下用户喜欢和讨厌的广告特征

train

uid	aid	label	ad features
u1	a1	1	{1,2,3}
u1	a2	0	{1,3,5}
u1	a3	1	{2,3,4,5}
u1	a4	0	{0,1,2,5}

test

u1	a5	na	na
----	----	----	----

uid	pos aid	neg aid	pos ads' features	neg ads' features
u1	a1,a3	a2,a4	{1,2,3}+{2,3,4,5}	{1,3,5}+{0,1,2,5}

uid	pos aid	neg aid	pos ads' features	neg ads' features
u1	a1,a3	a2,a4	{1,2,3}+{2,3,4,5}	{1,3,5}+{0,1,2,5}

uid	pos aid	neg aid	pos ads' features	neg ads' features
u1	a3	a2,a4	{2,3,4,5}	{0,1,2,3,5}

➤ 统计每个训练样本的特征时剔除该样本本身（防止过拟合）

## ➤ 从历史数据中挖掘特征

### ➤ 基本思路：记录下用户喜欢和讨厌的广告特征

	uid	aid	label	ad features		uid	aid	pos aid	neg aid	pos ads' features	neg ads' features
train	u1	a1	1	{1,2,3}		u1	a1	a3	a2,a4	{2,3,4,5}	{0,1,2,3,5}
	u1	a2	0	{1,3,5}		u1	a2	a1,a3	a4	{1,2,3,4,5}	{0,1,2,5}
	u1	a3	1	{2,3,4,5}		u1	a3	a1	a2,a4	{1,2,3}	{0,1,2,3,5}
	u1	a4	0	{0,1,2,5}		u1	a4	a1,a3	a2	{1,2,3,4,5}	{1,3,5}
test	u1	a5	na	na		u1	a5	a1,a3	a2,a4	{1,2,3,4,5}	{0,1,2,3,5}

### ➤ 统计每个训练样本的特征时剔除该样本本身（防止过拟合）



问题回顾

模型设计

特征工程

算法性能

经验总结



## ➤ 各种优化初赛阶段验证集上的效果

Model	AUC
baseline1 (MLP)	0.724
baseline2 (MLP)	0.727
Deep FM	0.732
add orthogonal initialization	0.736
add focal loss	0.737
add low-frequency filter	0.738
add attention and adaptive regularization	0.745
add learning rate decay	0.748
add BN	0.749
train on more data	0.751
LightGBM add cross features	0.743
LightGBM ensemble	0.748
LightGBM and NN	0.751
bagging	0.755
Deep Interest Neural FFM	0.752






## ➤ 各种优化复赛阶段验证集上的效果

取复赛训练集的1/10做验证集

SETTINGS	AUC
submission	0.7659
submission + focal loss	0.7661
submission + regularization	0.7660
submission - dice	0.7678
submission - attention	0.7628
submission - hand crafted feature	0.7514



## ➤ FM VS FFM



	FM	FFM
Train Epochs	>1 epochs	1 epoch
Attention	提升显著	提升显著
Focal Loss	略有提升	效果不明显
Dice	略有提升	有反效果
Adaptive Regularization	提升显著	效果不明显





## ➤ 运行时间与内存消耗

提交的版本:

- GPU 1080ti单卡 11G显存 (6000 samples/Batch)
- 8进程IO + chunk size(200 Batches) 150G内存 (数据直接读入内存)
- 2小时可以完成训练和测试 (0.52s/Batch)

除去Attention的版本:

- 显存可以降到5G

使用更少的IO进程, 更小的chunk size:

- 4进程IO + chunk size(50 Batches) 100G内存 (数据直接读入内存)
- 3.5小时可以完成训练和测试 (0.96s/Batch)



问题回顾

模型设计

特征工程

算法性能

经验总结





## ➤ 技巧与经验

- 嵌入层参数正交初始化
- FFM 嵌入维度小一点够用，我们采用的是6维
- 尽可能大的Batch Size，我们采用的是6000
- 映射空值到特殊值
- 限制嵌入向量的范数
- 使用gradient norm clipping



## ➤ 实现上可以提升的地方

- 增量式地统计特征
- 数据的读取方式：可以使用数据库，文件指针或者切割文件等等方式读入数据。我们现在的做法是直接把所有文件直接读进内存，加上python多进程支持的不是很好，对内存的需求非常大，提交的版本需要150G
- 模型集成的方法：比较简单的可以考虑在特征的子集上训练不同的模型，这样还可以顺带节省显存
- 可以直接在底层实现FFM的两两点积
- 使用try catch或者一定的调度策略来应对不定长特征导致的显存不固定的问题
- 用于attention的子网络的权重对于不同特征可以共享



## ➤ 主要参考文献

- [1] H. Guo, R. Tang, Y. Ye, Z. Li, and X. He.  
Deepfm: A factorization-machine based neural network for ctr prediction.  
arXiv preprint [arXiv:1703.04247](https://arxiv.org/abs/1703.04247), 2017.
- [2] Y. Juan, Y. Zhuang, W.-S. Chin, and C.-J. Lin.  
Field-aware factorization machines for ctr prediction.  
In Proceedings of the 10th ACM Conference on Recommender Systems, pages 43–50. ACM, 2016.
- [3] G. Zhou, C. Song, X. Zhu, X. Ma, Y. Yan, X. Dai, H. Zhu, J. Jin, H. Li, and K. Gai.  
Deep interest network for click-through rate prediction.  
arXiv preprint [arXiv:1706.06978](https://arxiv.org/abs/1706.06978), 2017.







Thanks