DIN: 阿里点击率预估之深度兴趣网络

原创 张雨石 雨石记 9月4日

收录于话题
#推荐广告算法

10个

广告推荐算法系列文章：

- 莫比乌斯: 百度的下一代**query-ad**匹配算法
- 百度凤巢分布式层次**GPU**参数服务器架构
- DIN: 阿里点击率预估之深度兴趣网络(本篇)

本文的知识点来源于参考文献[1]，是阿里巴巴2018年在KDD上的论文。本文可以视为Attention机制在推荐系统上的应用。对Attention机制不了解的同学可以看下面的文章进行学习。

- **Transformer: Attention**的集大成者

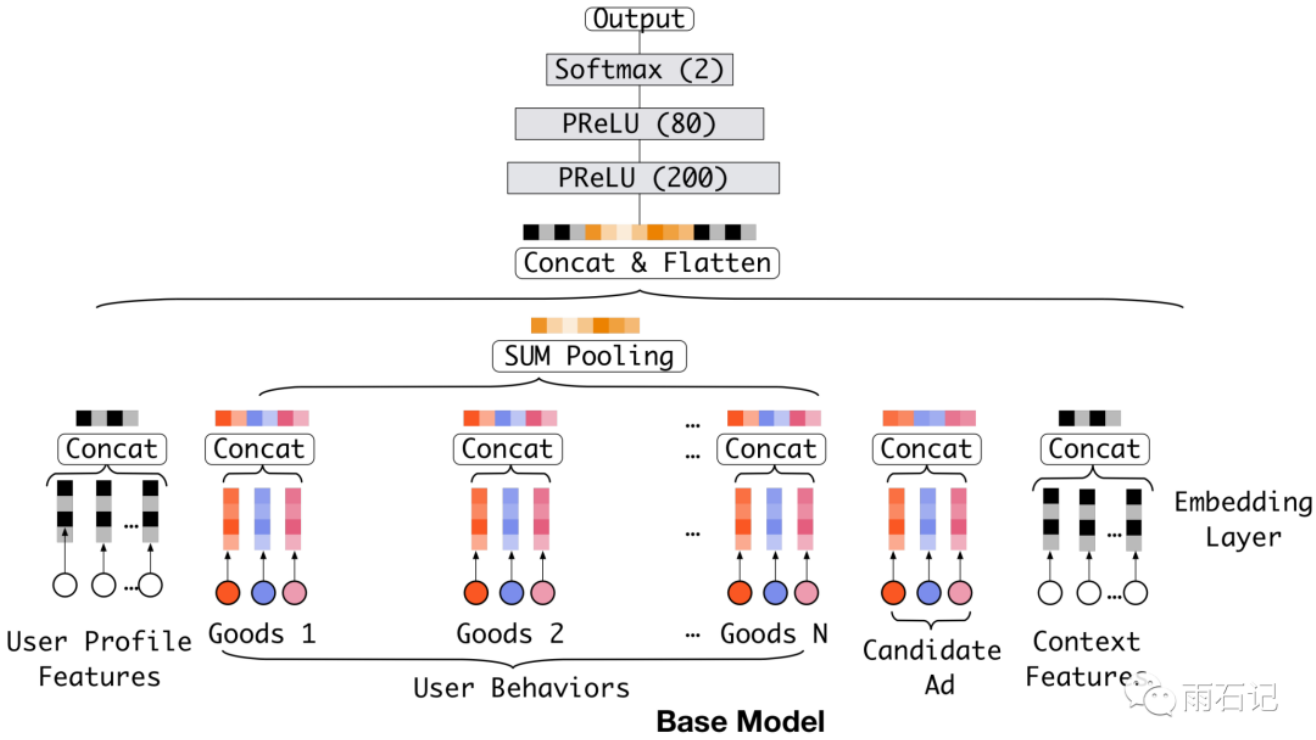
背景-推荐模型

正如我们在分布式层次**GPU**参数服务器架构所提到的，如今深度学习在推荐系统和广告点击率预估上应用广泛，普遍采用的模式是Embedding + MLP的形式。

在这种Embedding + MLP的模型下，有许许多多的特征工程上的技巧。在阿里这个场景下，可以分成这么几个大类，比如用户画像，用户的操作行为，上下文特征，广告特征等等。如下图所示。

Category	Feature Group	Dimemsionality	Type	#Nonzero Ids per Instance
User Profile Features	gender	2	one-hot	1
	age_level	~ 10	one-hot	1
	...	...	...	...
User Behavior Features	visited_goods_ids	~ 10^9	multi-hot	~ 10^3
	visited_shop_ids	~ 10^7	multi-hot	~ 10^3
	visited_cate_ids	~ 10^4	multi-hot	~ 10^2
Ad Features	goods_id	~ 10^7	one-hot	1
	shop_id	~ 10^5	one-hot	1
	cate_id	~ 10^4	one-hot	1
	...	...	...	...
Context Features	pid	~ 10	one-hot	1
	time	~ 10	one-hot	1
	...	...	...	...

而对应的模型结构则如下图



可以看到，不同种类的特征在形成向量后是拼接起来的。

用户兴趣多元化

上述模型结构和特征工程已经能达到一个较好的结果了，但是要想精益求精，还是需要对业务有更加深刻的了解。而在阿里的这个场景下，那就是用户购物需求的多元化。在上面的模型中，用户的行为被压缩到了一个特征向量中，就相当于是把所有的兴趣爱好的信息做了平均。但这样做是不精准的。

比如说，一个女性游泳爱好者，可能既会关注包包，又会关注游泳类产品。那么她在浏览包包的时候，对游泳类产品的兴趣其实是与CTR的估计相当无关的事情。

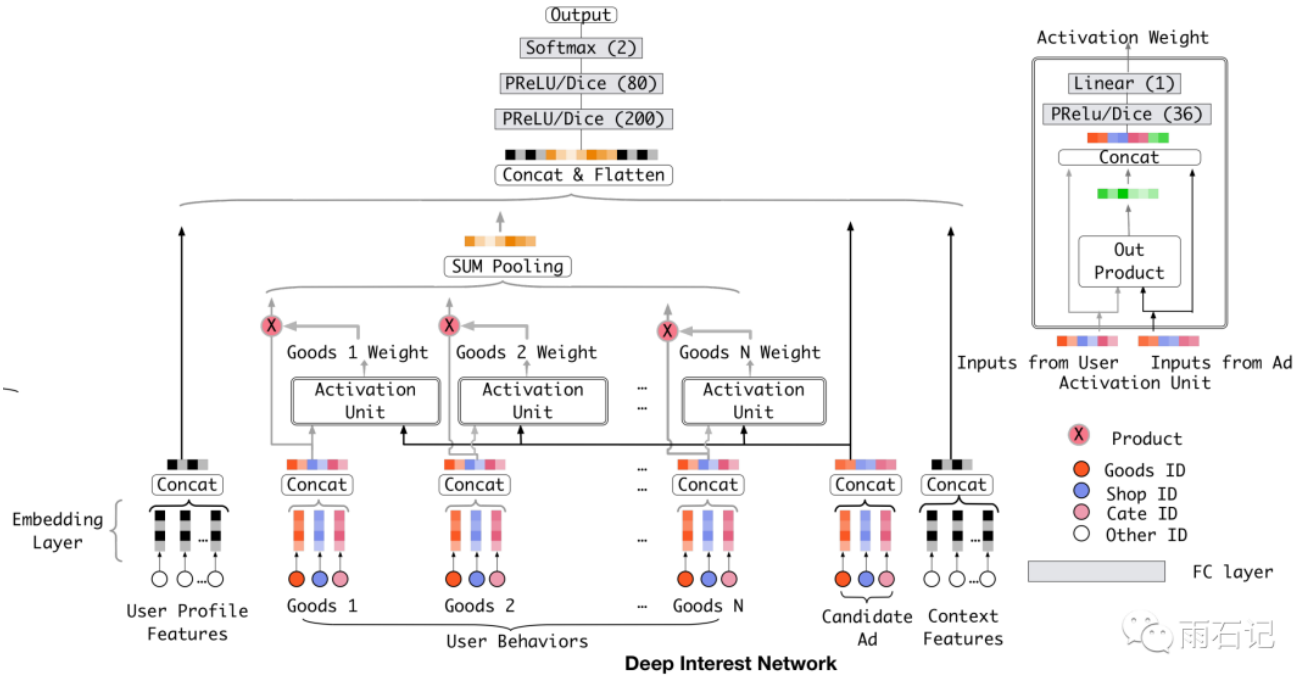
那么如何解决这个问题呢？解决的方法就是对于一个<用户，商品>对来说，不同的商品，要去触发用户不同的兴趣点才合理。这样的操作在NLP问题中其实是非常常见的，比如翻译问题，目标语言句子上的不同位置的词语，对应的是源语言句子上的词语也是不同的，这种对应关系被Attention所解决。

类似的，在这篇论文中，这个问题被兴趣网络解决。

深度兴趣网络

针对上述问题，提出了深度兴趣网络，Deep Interest Network，简称DIN。

DIN的核心idea很直观，模型结构如下图，在这个结构中，可以看到，候选广告需要去跟用户行为中的每一个商品去做权重的计算，然后用权重去做加权平均得到用户针对这个候选广告的兴趣向量。权重和加权，这就是Attention。



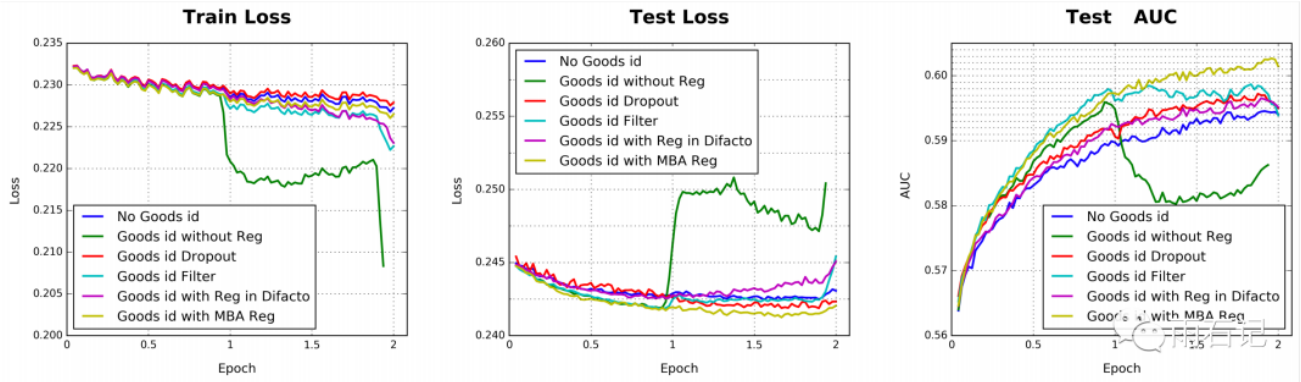
举一个例子，如果用户的行为历史中有两类，衣服和电子产品，其中衣服占90%，电子产品占10%，那么给两个产品T恤和iPhone，那么计算得到的用户对T恤的兴趣很可能大于用户对iPhone的兴趣。之所以说很可能而不是肯定，是因为商品还要本身的性质，比如衣服可以换的很频繁，但iPhone不是，衣服的盈利远远不如iPhone的利润大等等。所以在计算attention的时候，还会有很多特征需要挖掘来计算得到可靠的权重。

大家注意到，这里用户行为历史中的操作是并行计算权重的。其实可以通过循环神经网络来把时间因素考虑进来。论文中尝试过，但是没有提升，可能的原因是兴趣本身就是共存的，时间的前后顺序影响不大。

训练技巧

模型结构本身并不复杂，但相对于阿里巴巴的业务量而言，这个模型的训练是非常难的，因为模型中的用户和候选广告都是以亿计的，而特征又是极其稀疏的。

比如，当上面第一张图中的goods_ids的特征在6亿的时候，如果没有正则化，那么模型在训练一个epoch后在训练集上的loss会迅速下降，导致在测试集上过拟合。如下图中的红线所示：



而如果采用传统的L2或者L1正则化又是不可能的，因为传统方法需要在所有非0的参数上进行计算，而对于这个问题来说，每次训练都在数以亿计的参数上去做正则化是不可行的。

论文提出了一种近似的办法，即Mini-batch aware的正则化，这种正则化的方法只考虑了在一个mini-batch中出现了的特征所对应的参数。因为稀疏特征的众多，网络中大部分的参数都分布在embedding层，论文以embedding层为例来讲解了正则化的操作。如下所示：

$$L_2(\mathbf{W}) = \|\mathbf{W}\|_2^2 = \sum_{j=1}^{K} \|\boldsymbol{w}_j\|_2^2 = \sum_{(\boldsymbol{x},y)\in\mathcal{S}} \sum_{j=1}^{K} \frac{I(\boldsymbol{x}_j \neq 0)}{n_j} \|\boldsymbol{w}_j\|_2^2,$$

上图公式中表明了在embedding层上只计算mini-batch上用到的特征所对应参数的L2正则化的方法，其中$I(x_j \neq 0)$是指示器来表明x_j特征是否存在，nj表示所有样本中x_j不为0的样本数。w_j代表特征j的embedding参数。

然后这个公式可以化简：

$$L_2(\mathbf{W}) = \sum_{j=1}^{K} \sum_{m=1}^{B} \sum_{(\boldsymbol{x},y)\in\mathcal{B}_m} \frac{I(\boldsymbol{x}_j \neq 0)}{n_j} \|\boldsymbol{w}_j\|_2^2,$$

再近似

$$L_2(\mathbf{W}) \approx \sum_{j=1}^{K} \sum_{m=1}^{B} \frac{\alpha_{mj}}{n_j} \|\boldsymbol{w}_j\|_2^2.$$

其中

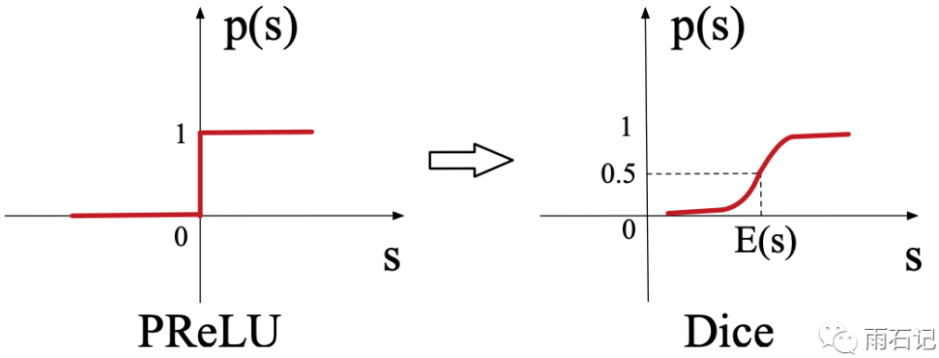
$$\alpha_{mj} = \max_{(x,y)\in\mathcal{B}_m} I(x_j \neq 0)$$

表示的是特征j在mini-batch Bm中至少出现过一次。

这样，经过正则化的梯度就可以计算出来：

$$w_j \leftarrow w_j - \eta \left[\frac{1}{|\mathcal{B}_m|} \sum_{(x,y)\in\mathcal{B}_m} \frac{\partial L(p(x),y)}{\partial w_j} + \lambda \frac{\alpha_{mj}}{n_j} w_j \right],$$

在激活函数上，论文提出了一种叫做Dice的激活函数，是PRELU的泛化版本。两种激活函数图示如下：



其中，论文上的PRelu的图画错了，其公式如下：

$$f(s) = \begin{cases} s & \text{if } s > 0 \\ \alpha s & \text{if } s \leq 0. \end{cases} = p(s)\cdot s + (1-p(s))\cdot \alpha s,$$

基于这个公式，大家可以自行画出正确的图。

而Dice的公式如下：

$$f(s) = p(s)\cdot s + (1-p(s))\cdot \alpha s, \quad p(s) = \frac{1}{1+e^{-\frac{s-E[s]}{\sqrt{Var[s]+\epsilon}}}}$$

Dice是PRelu的泛化版本，当均值为0方差为1的时候，两者是等价的，之所以要改成这个形式，是为了要使激活函数适应数据分布。

实验

采用了三个数据集

- Amazon dataset
- MovieLens Dataset
- Alibaba Dataset

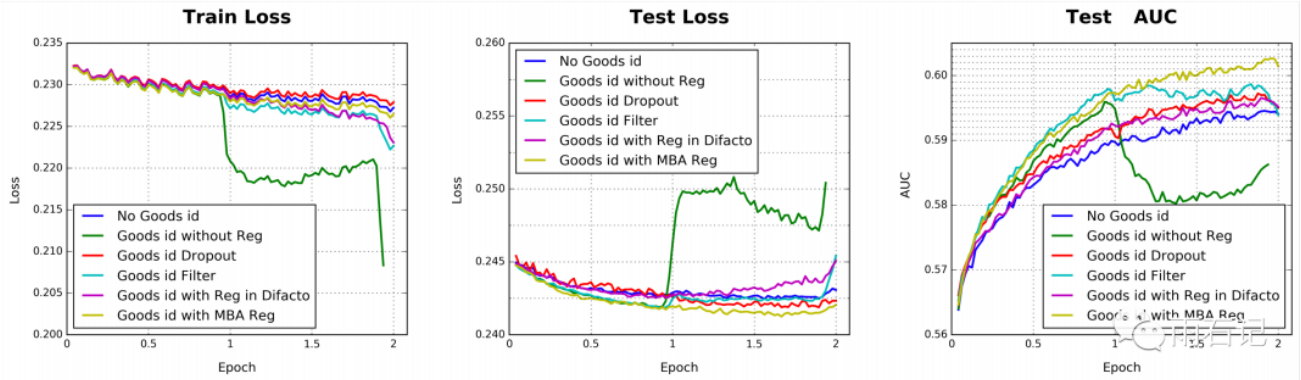
前两者是公开数据集。

不同的方法在前两个数据集上的对比，可以看到带来的相对提升还是很高的，达到了2%和6.8%。

Model	MovieLens.		Amazon(Electro).	
	AUC	RelaImpr	AUC	RelaImpr
LR	0.7263	-1.61%	0.7742	-24.34%
BaseModel	0.7300	0.00%	0.8624	0.00%
Wide&Deep	0.7304	0.17%	0.8637	0.36%
PNN	0.7321	0.91%	0.8679	1.52%
DeepFM	0.7324	1.04%	0.8683	1.63%
DIN	**0.7337**	**1.61%**	**0.8818**	**5.35%**
DIN with Dice[a]	**0.7348**	**2.09%**	**0.8871**	**6.82%**

[a] Other lines except LR use PReLU as activation function.

正则化方法的对比，



可以看到，即便在BaseModel上，正则化方法也有效。

Regularization	AUC	RelaImpr
Without goods_ids feature and Reg.	0.5940	0.00%
With goods_ids feature without Reg.	0.5959	2.02%
With goods_ids feature and Dropout Reg.	0.5970	3.19%
With goods_ids feature and Filter Reg.	0.5983	4.57%
With goods_ids feature and Difacto Reg.	0.5954	1.49%
With goods_ids feature and MBA. Reg.	**0.6031**	**9.68%**

在阿里巴巴的数据集上，带来了11.65%的提升，又是财富在发光。

Model	AUC	RelaImpr
LR	0.5738	- 23.92%
BaseModel[a,b]	0.5970	0.00%
Wide&Deep[a,b]	0.5977	0.72%
PNN[a,b]	0.5983	1.34%
DeepFM[a,b]	0.5993	2.37%
DIN Model[a,b]	**0.6029**	**6.08%**
DIN with MBA Reg.[a]	**0.6060**	**9.28%**
DIN with Dice [b]	**0.6044**	**7.63%**
DIN with MBA Reg. and Dice	**0.6083**	**11.65%**

[a] These lines are trained with PReLU as the activation function.
[b] These lines are trained with dropout regularization.

有了attention之后，推荐结果也变得可解释了，下图是一个可视化效果图，反应了当前商品和用户历史行为中的商品的权重。

思考

> 勤思考，多提问是每个Engineer的良好品德。

- 随着时间的流逝，用户操作行为越来越长，在超长序列上建模会遇到性能问题，如何解决？

答案后续发布在其他文章里，欢迎关注公众号【雨石记】。

参考文献

- [1]. Zhou, Guorui, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. "Deep interest network for click-through rate prediction." In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 1059-1068. 2018.

收录于话题 #推荐广告算法·10个

上一篇
基于Delaunay图的快速最大内积搜索算法

下一篇
分布式层次GPU参数服务器架构

喜欢此内容的人还喜欢

阿里副总裁"人设"翻车：30岁成AI顶尖科学家，但我很懒
纯洁的微笑

阿里定向广告最新突破：面向下一代的粗排排序系统COLD
DataFunTalk

城市大脑助力文明创城，浪潮VS阿里，玩法有何不同？
胖头陀