0x01. 为什么要用多任务学习?

1. 方便。在推荐任务中，往往不仅要预测用户的engagement（例如CTR），还要预测用户satisfaction（例如评分、CVR、观看时长）。因为我们最终又不是只按照一个指标的维度进行排序，比如如果只按照CTR排序，那么很有可能把那些"标题党"的item排到前面，而用户看完并不喜欢、也不会转化。所以，一般都是对多个指标维度进行加权平均得到score，然后按照score排序。如果用多个模型预测多个目标，参数量会很大，而且在线上也不好维护。因此需要使用一个模型来预测多个目标，这点对工业界来说十分友好。
2. 多任务学习不仅方便，还可能效果更好。针对很多数据集比较**稀疏**的任务，比如短视频转发，大部分人看了一个短视频是不会进行转发这个操作的，这么稀疏的行为，模型是很难学好的（数据集太小，过拟合问题严重），那我们把预测用户是否转发这个稀疏的事情和用户是否click这个经常发生事情放在一起学，通过**参数共享**，一定程度上会缓解模型的过拟合，提高了模型的泛化能力。这其实是**regularization**和**transfer learning**。也可以理解为，其他任务的预测loss对于"转发"事件预测来说是**辅助loss**。从另一个角度来看，对于数据很少的新任务，这样也解决了**冷启动问题**。

0x02. 多任务学习模型

(1). Baseline -- Shared-Bottom Model

1.1 硬参数共享

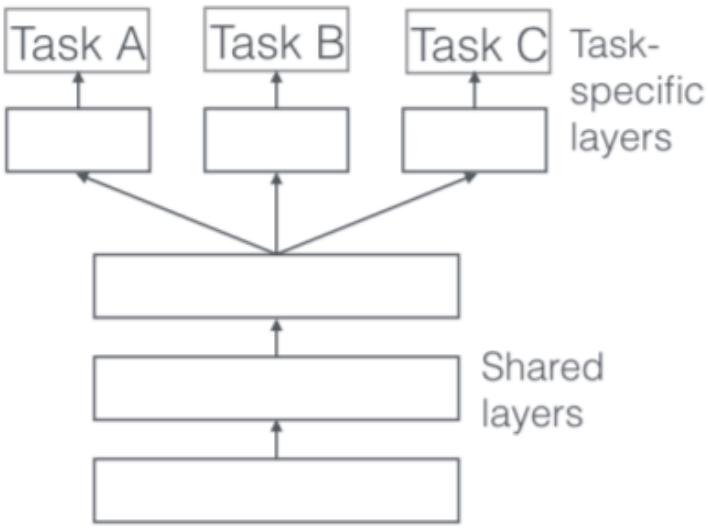


Figure 1: Hard parameter sharing for multi-task learning in deep neural networks

不同任务间共用底部的隐层。这种结构由于**全部的参数共享**可以减少过拟合的风险（原因如上所述），但是效果上受到任务**差异**（optimization conflicts caused by task differences）和数据分布差异带来的影响。

1.2 软参数共享

与硬参数共享相对的是软参数共享：每个任务都有特定的模型、参数，参数不共享；但对模型间的参数，使用距离正则化约束，保障参数空间的相似。

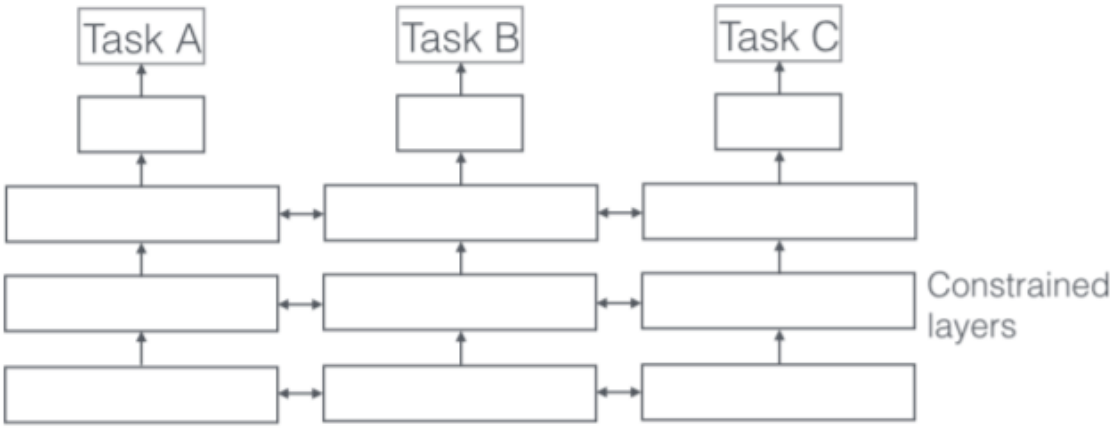


Figure 2: Soft parameter sharing for multi-task learning in deep neural networks

两个任务的参数完全不共用，但是对不同任务的参数增加**L2范数的限制**（L2-Constrained）：

L2-Constrained [15]: This method is designed for a cross-lingual problem with two tasks. In this method, parameters used for different tasks are shared softly by an L2 constraint.

Given y_k as the ground truth label for task k, $k \in 1, 2$, the prediction of task k is represented as

$$\hat{y}_k = f(x; \theta_k),$$

where θ_k are model parameters.

The objective function of this method is

$$loss = \mathbb{E}L(y_1, f(x; \theta_1)) + \mathbb{E}L(y_2, f(x; \theta_2)) + \alpha \|\theta_1 - \theta_2\|_2^2$$

where y_1, y_2 are the ground truth label for task 1 and task 2, and α is a hyper-parameter. This method models the task relatedness with the magnitude of α.

参数完全不共用，只是用L2正则增加参数限制

2个任务的参数完全不共用，但是在损失函数中加入正则项。α是两个任务的相似度，α越大，两个任务参数越趋近于一致。

和shared-bottom结构相比，这样的模型对增加了针对任务的特定参数(task-specific parameters)，在任务差异很大的情况下效果比较好。缺点就是模型增加了参数量（如果要训练k个目标，就增加k倍），所以需要更大的数据量来训练模型，而且模型更复杂并不利于在真实生产环境中实际部署使用。

(2) MMoE

论文 Modeling Task Relationships in Multi-task Learning with Multi-gate Mixture-of-Experts 中提出了一个 Multi-**gate** Mixture-of-**Experts**(MMoE)的多任务学习结构。

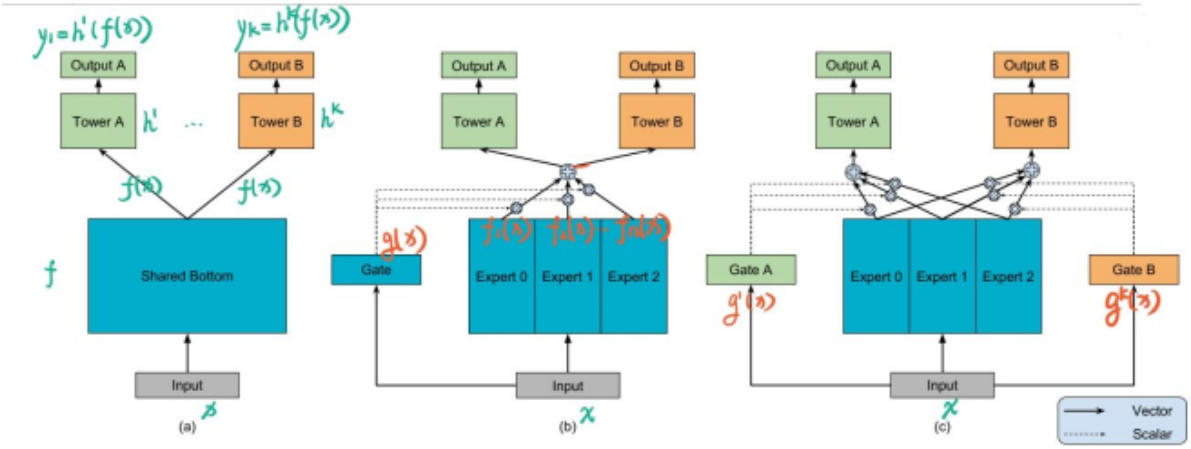


Figure 1: (a) Shared-Bottom model. (b) One-gate MoE model. (c) Multi-gate mol[illegible].

Shared-bottom, OMoE, MMoE

文章提出的模型MMoE目的就是相对于shared-bottom结构不明显增加模型参数 的要求下捕捉任务的**不同** 。其核心思想是将shared-bottom网络中的函数 f 替换成 MoE 层，如上图c所示，形式化表达为：

$$f^k(x) = \sum_{i=1}^{n} g^k(x)_i f_i(x)$$

$$y_k = h^k(f^k(x))$$

其中门控网络 $g^k(x) = softmax(W_{gk}x)$ ，**输入就是input feature，输出是所有experts上的权重。**其实这个门控很像**attention**，针对不同的任务分配给experts以不同的权重。

一方面，因为gating networks通常是轻量级的，而且expert network是所有任务共用，所以相对于上文提到的软参数共享方法有参数量上的优势；

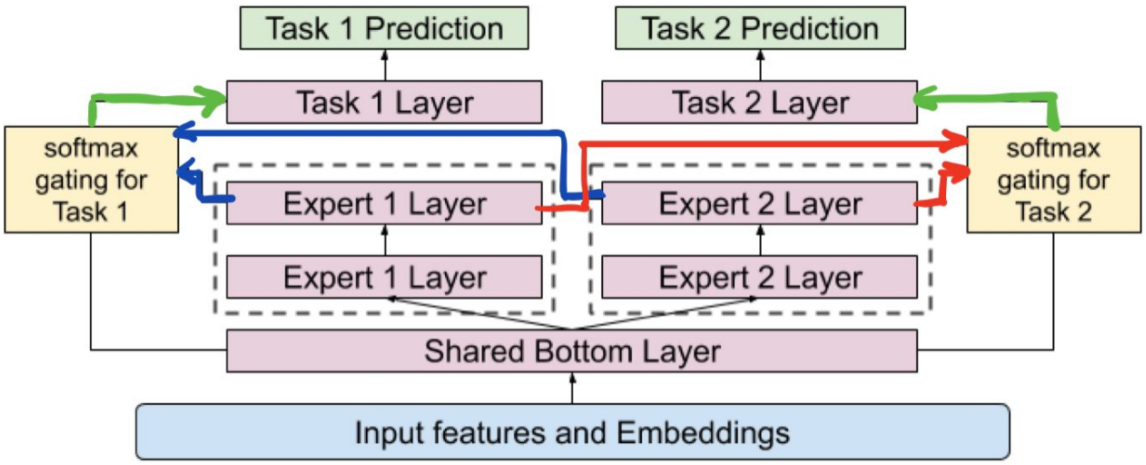
另一方面，相对于所有任务公用一个gate的方法One-gate MoE model(OMOE)，这里MMoE中每个任务使用不同的gating networks（即**不同的attention net**），从而学习到对于不同的任务、experts的不同权重，因此模型考虑到了捕捉到任务的相关性和**区别**。因此在模型的效果上优于上文提到的硬参数共享的方法。实际上，如果任务相关度很低，**则OMoE的效果相对于MMoE明显下降**，说明MMoE中的multi-gate的结构对于任务差异带来的**冲突**有一定的缓解作用。其实MMoE就是对不同的任务，都训练一个不同的attention net，用来赋予expert不同的权重。

MMoE在Youtube推荐场景下的实践

论文：Recommending What Video to Watch Next: A Multitask Ranking System，这篇主要是在商业推荐上用了MMoE,以及提出了shallow tower解决position bias的方法。

本文的场景是根据seed video预测用户下一个观看的video，属于i2i问题。文中的优化目标大体分为两类，一类是engagement目标，包括点击、观看时长、完播率等，表示用户的参与度；第二类是satisfaction目标，例如评分或差评，表示用户的满意度。这其中既有分类任务(e.g. clicked)也有回归任务(e.g. 观看时长、评分)。从文中实验来看，总共包括7个任务，这些任务或者是**递进/依赖**的关系，例如只有观看之后才会打分；或者是冲突的关系，如点了之后发现不喜欢（虽然engagement高但是satisfaction低）。MMoE比较适合这种多个任务之间联系不紧密、甚至冲突的场景。

为了更efficient，降低参数量，先使用一个shared bottom，把input layer的维度降下来，然后再接不同的expert：



(b) Multi-gate Mixture-of-Expert Model with one shared bottom layer and separate hidden layers for two tasks.

完整的模型结构如下图所示。模型对每一个目标都做预估，分类问题就用cross entropy loss学习，回归问题可就是square loss。最后用**融合公式**来平衡用户交互和满意度指标（将不同任务的计算得分做加权平均），用这个score大小来对召回的结果进行排序。这个权重需要人工手动来调整。所以，我们做多目标优化的意义就是，不仅要关注CTR等engagement metric，还要去关注用户的满意程度，才能够让用户用的开心、保持推荐系统的健康生态。

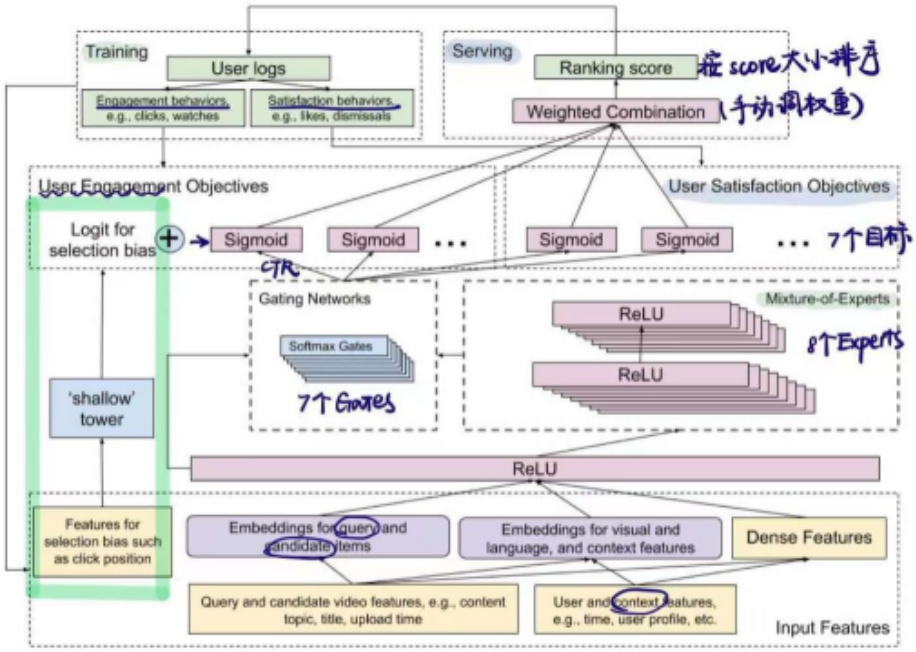


Figure 1: Model architecture of our proposed ranking system. It consumes user logs as training data, builds Multi-gate Mixture-of-Experts layers to predict two categories of user behaviors, i.e., engagement and satisfaction. It corrects ranking selection bias with a side-tower. On top, multiple predictions are combined into a final ranking score.

这篇文章中的测试结果对比也很引人深思。在对比结果的时候，是对比**相同FLOPs**条件下的测试结果。这是为了在参数计算量一样的前提下比较模型的效果，看不同的模型对线上的engagement和satisfaction指标的影响。

Model Architecture	Number of Multiplications	Engagement Metric	Satisfaction Metric
Shared-Bottom	3.7M	/	/
Shared-Bottom	6.1M	+0.1%	+ 1.89%
MMoE (4 experts)	3.7M	+0.20%	+ 1.22%
MMoE (8 Experts)	6.1M	+0.45%	+ 3.07%