多目标模型之SNR

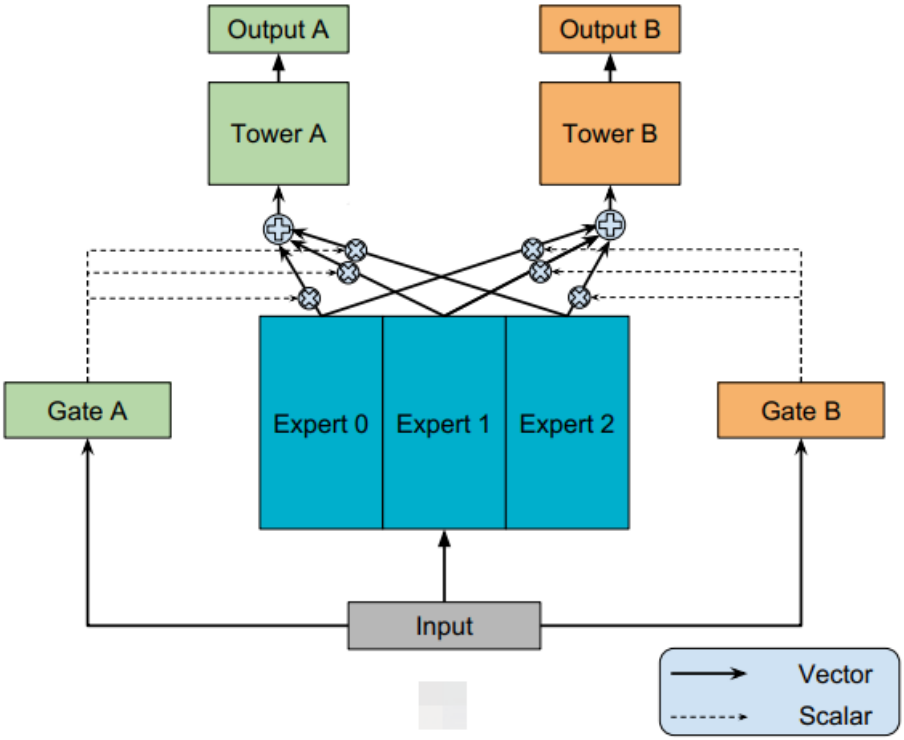
原创 mrchor 软客圈 6月24日

前言

上期我们介绍了Google推出的多目标模型——MMoE（Multi-gate Mixture-of-Experts），初步了解了可以在工业界使用的多目标模型。MMoE的出现，极大地推动了多目标模型在工业界的向前推进。然而MMoE有其自身的局限性，基于此Google又提出了新一款多目标模型——SNR（Sub-Network Routing），从一定程度上缓解了MMoE的问题，本文从模型架构，原理及实验结果等方面来介绍这款模型。

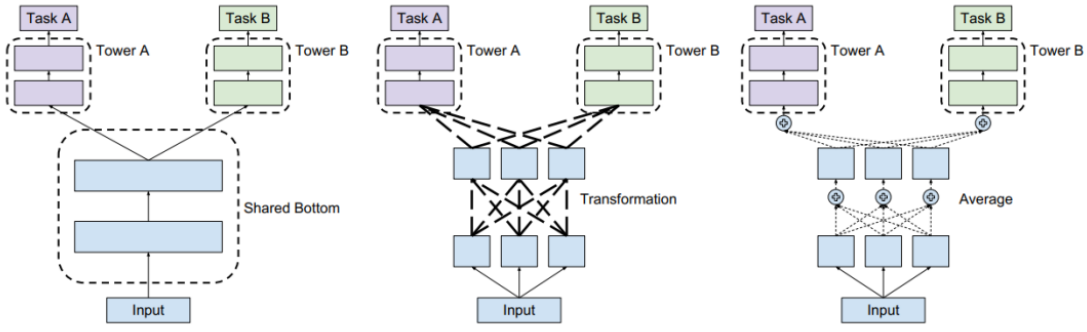
MMoE的局限性

MMoE模型从一定程度上解决了多个目标（任务）在训练过程中的相互耦合的问题，即使用门控概念（gate network）降低了因为share-layer部分带来的“特征耦合”。但其实这是不够的，因为在每一个expert内部，与其他的expert不存在联系，这导致每个expert的表达能力不是“那么强”。



SNR的提出及需要解决的问题

为了解决上面MMoE模型的局限性，作者提出了灵活参数共享的概念，即我们不应把share layer部分作为整体的参数分享给每一个需要训练的目标，在share layer内部也需要互相共享参数，以提高表达，那么怎么做呢？作者设计了一款模型为：Sub-Network Routing（后续如无特殊标记，均以SNR代替），在share layer中的上下层进行剥离，用下层中的所有参数作为上层输入共享，此处作者设计了两种共享方式：transformation和average。



理论解释

那么，我们如何实现上述模型架构中的transformation和average呢，我们设有两个高层子网和三个底层子网，那么transformation实现公式如下：

$$\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} z_{11}W_{11} & z_{12}W_{12} & z_{13}W_{13} \\ z_{21}W_{21} & z_{22}W_{22} & z_{23}W_{23} \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}$$

其中u_1、u_2、u_3对应三个底层子网，v_1、V_2对应高层子网，W是转换矩阵，Z代表编码变量（二进制变量）。
类似的，有average实现公式如下：

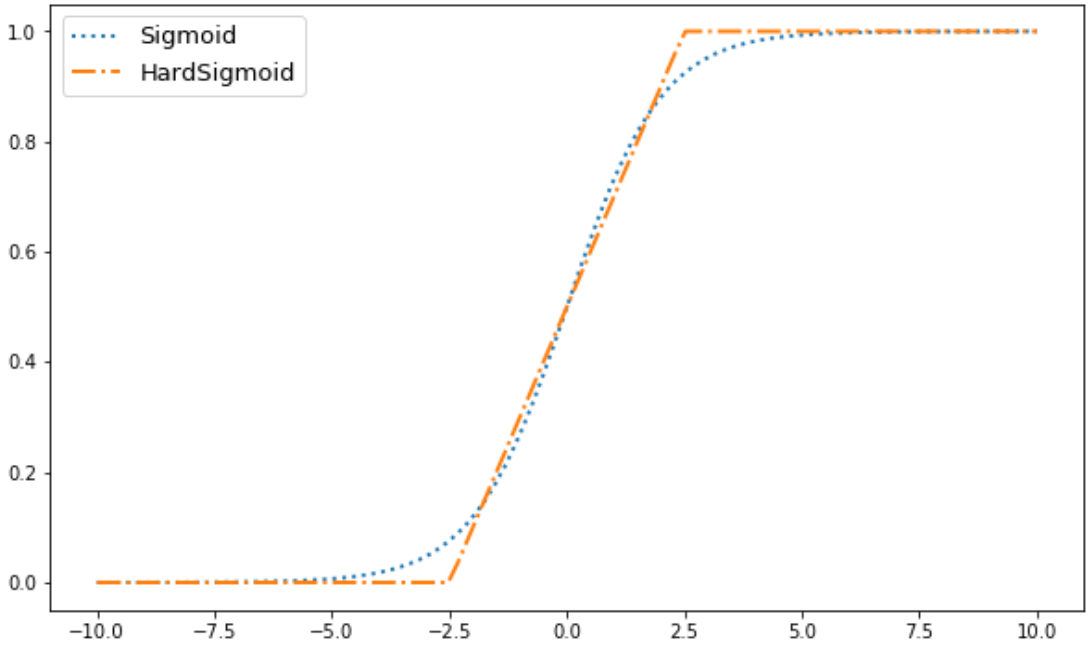
$$\begin{bmatrix} \boldsymbol{v_1} \\ \boldsymbol{v_2} \end{bmatrix} = \begin{bmatrix} z_{11}\boldsymbol{I_{11}} & z_{12}\boldsymbol{I_{12}} & z_{13}\boldsymbol{I_{13}} \\ z_{21}\boldsymbol{I_{21}} & z_{22}\boldsymbol{I_{22}} & z_{23}\boldsymbol{I_{23}} \end{bmatrix} \begin{bmatrix} \boldsymbol{u_1} \\ \boldsymbol{u_2} \\ \boldsymbol{u_3} \end{bmatrix}$$

其中Z与上述定义相一致，I是单位矩阵。
根据上述解释构造损失函数：

$$\min_{\boldsymbol{W},\boldsymbol{\pi}} \boldsymbol{E}_{\boldsymbol{z}\sim p(\boldsymbol{z};\boldsymbol{\pi})} \frac{1}{N} \sum_{i=1}^{N} L(f(\boldsymbol{x}_i;\boldsymbol{W},\boldsymbol{z}),\boldsymbol{y}_i), \qquad (1)$$

然而上述的Z，作为一个需要训练的参数，由于是一个binary的变量，因此，无法直接训练，由此我们想到可以将Z看做是二项分布（伯努利分布），这样就对Z做成一个平滑变量，然后参与模型的训练。
此处的想法是将Z作为一个连续随机变量，把Z作为hard-sigmoid分布变量（跟sigmoid的区别可以看图）。

$$z = g(s) = \min(1, \max(0, s)).$$



那么，此时的损失函数变为：

$$\min_{\boldsymbol{W},\boldsymbol{\pi}} \boldsymbol{E}_{\boldsymbol{s}\sim q(\boldsymbol{s};\boldsymbol{\phi})} \frac{1}{N} \sum_{i=1}^{N} L(f(\boldsymbol{x}_i;\boldsymbol{W},g(\boldsymbol{s})),\boldsymbol{y}_i). \qquad (2)$$

为了提高模型的效率，作者此处使用了L0正则化，L0正则化能很好地对多目标模型进行参数稀疏化。

$$\boldsymbol{E}_{\boldsymbol{z}\sim p(\boldsymbol{z};\boldsymbol{\pi})} ||\boldsymbol{z}||_0 = \sum_{i=1}^{|\boldsymbol{z}|} p(z_i = 1;\pi_i).$$

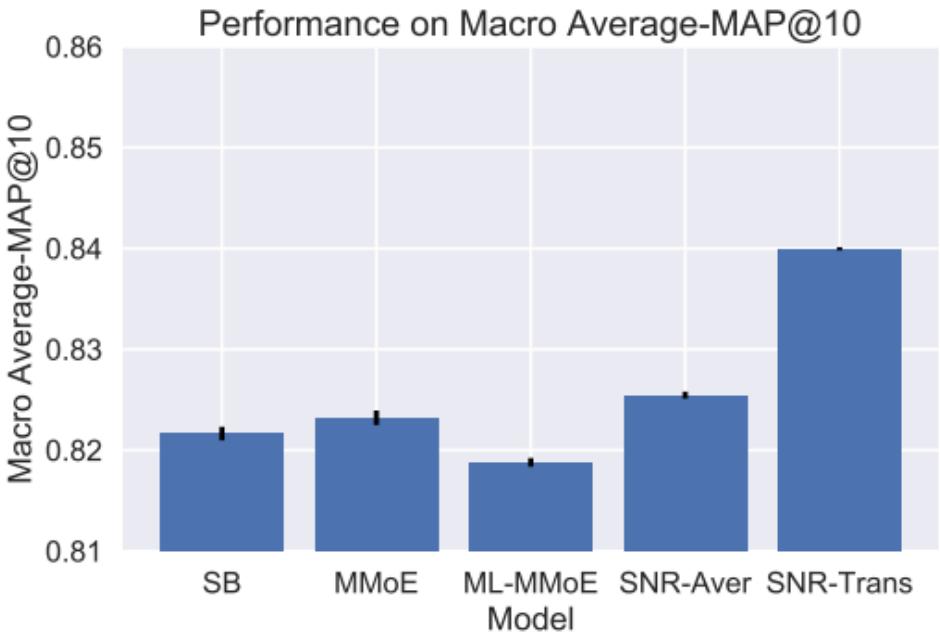
经过上述操作，最终得到的目标函数为：

$$\boldsymbol{E}_{\boldsymbol{\epsilon}\sim r(\boldsymbol{\epsilon})}\frac{1}{N}\sum_{i=1}^{N}L(f(\boldsymbol{x}_i;\boldsymbol{W},g(h(\boldsymbol{\phi},\boldsymbol{\epsilon}))),\boldsymbol{y}_i)$$

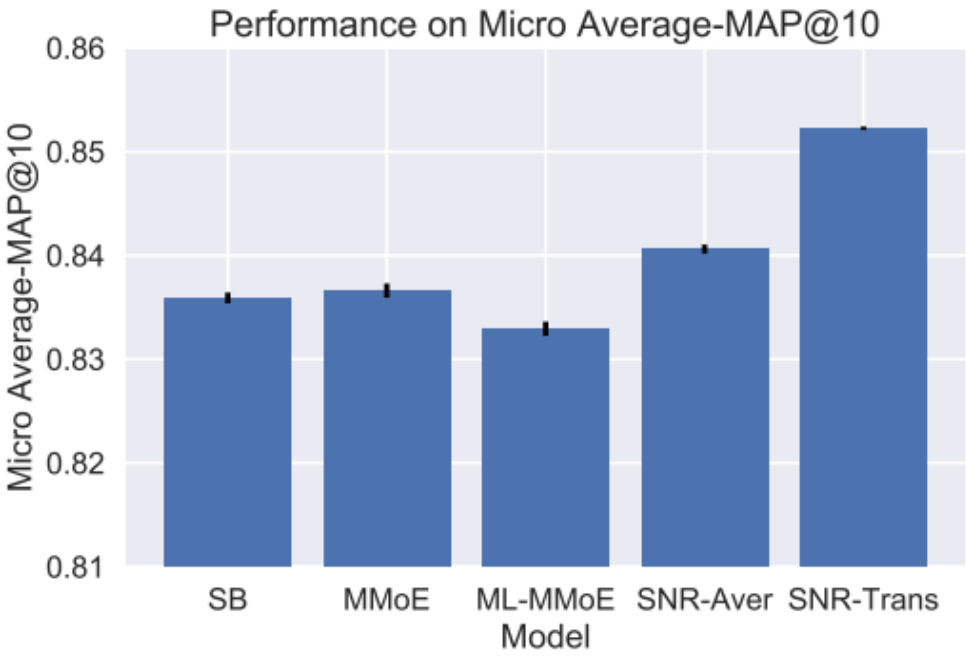
$$+\lambda\sum_{j=1}^{|\boldsymbol{z}|}1-Q(s_j<0;\phi_j),$$

实验对照

为验证模型的有效性，作者在YouTube8M进行了相应的实验，以下是实验结果对比：

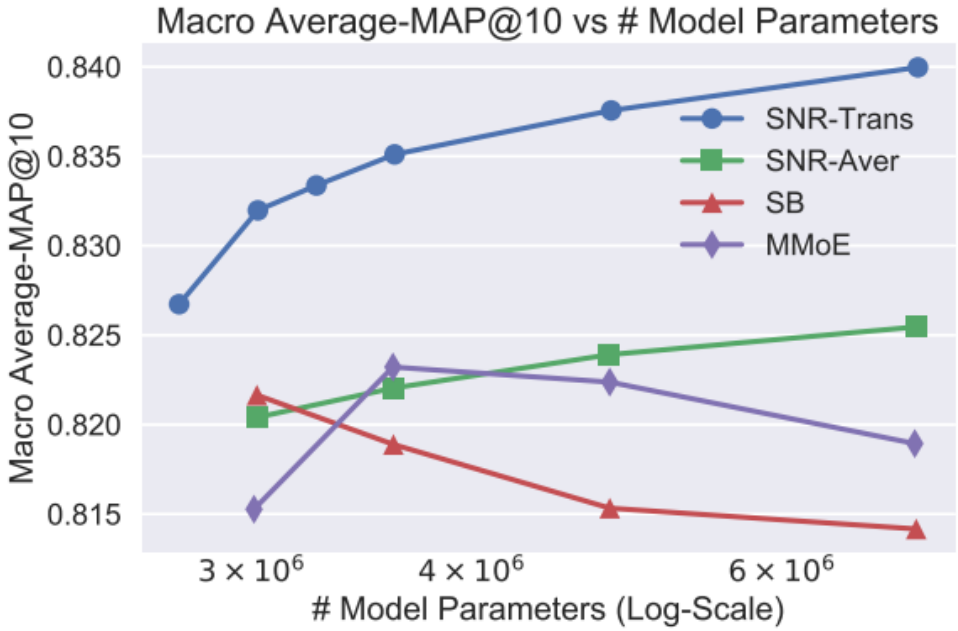


(a) Macro Average-MAP@10



(b) Micro Average-MAP@10

由上述实验对照图可以看出了SNR-Trans和SNR-Aver在准确率均超越了相关最新的多目标模型。
随后，作者又进行了参数个数对模型效果影响的实验，以下是结果，可以发现，随着参数个数的增加，SNR-Trans的性能在逐步提升，SNR-Aver的性能也在提升，但是效果不明显。



总结

SNR是继MMoE后，Google祭出的又一个多目标模型大招，其在同等参数个数下的性能超过了MMoE，但是笔者认为，SNR虽然是一个很好用的模型，但是其设计复杂度已经远超MMoE，因此带来的调参是一个巨大的问题，可能不适合理论基础一般的算法工程师拿来实践。虽然后期作者实验了在参数增加的条件下，SNR的模型能够稳定提升性能，但是在实际工业界，我们需要在模型复杂度与算力之间做一个平衡，即不用一味地为了提升效果，而增加模型的参数个数，因此这就要求我们能否设计一款，既能超越MMoE又不至于持续增加参数个数的模型，希望未来有大佬能解决此问题，毕竟我们的原则还是“奥卡姆剃刀”嘛~