RS Meet DL(68)-建模多任务学习中任务相关性的模型MMoE

原创 石晓文 小小挖掘机 2019-10-27

来自专辑

推荐系统遇上深度学习

Modeling Task Relationships in Multi-task Learning with Multi-gate Mixture-of-Experts

Jiaqi Ma[1*], Zhe Zhao[2], Xinyang Yi[2], Jilin Chen[2], Lichan Hong[2], Ed H. Chi[2]
[1]School of Information, University of Michigan, Ann Arbor [2]Google Inc.
[1]jiaqima@umich.edu [2]{zhezhao, xinyang, jilinc, lichan, edchi}@google.com

本文介绍的论文题目是：《Modeling Task Relationships in Multi-task Learning with Multi-gate Mixture-of-Experts》
论文下载地址为：https://dl.acm.org/citation.cfm?id=3220007

多任务学习最近越来越受欢迎，咱们前面也介绍过几篇阿里多任务学习的模型，不过多任务学习的效果受不同任务之间的相关性影响较大，因此本文基于Mixture-of-Experts (MoE)模型，提出了一种显式建模任务相关性的模型Multi-gate Mixture-of-Experts (MMoE)，一起来学习一下。

1、背景

近年来，深度神经网络的应用越来越广，如推荐系统。推荐系统通常需要同时优化多个目标，如电影推荐中不仅需要预测用户是否会购买，还需要预测用户对于电影的评分，在比如电商领域同时需要预测物品的点击率CTR和转化率CVR。因此，多任务学习模型成为研究领域的一大热点。

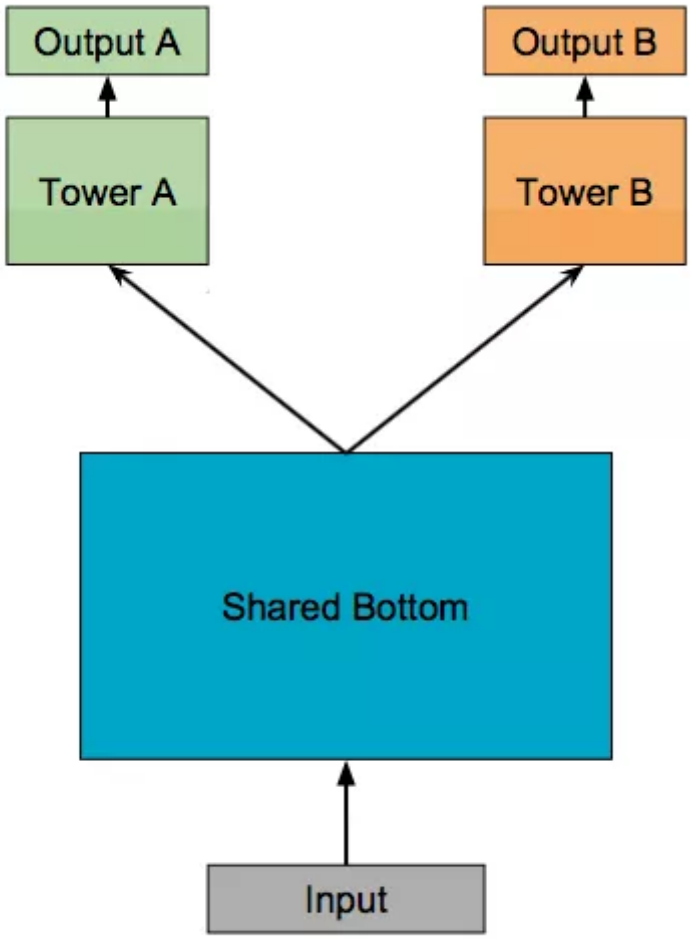
许多多任务学习模型取得了不错的效果，但是实践中多任务学习模型并不总比单任务模型效果更突出。这主要是因为不同任务之间的相关性低（如数据的分布不同等等）导致的。

是不是真的如上述所说，任务之间的相关性会影响多任务学习的效果呢，咱们先在第二节中做一个实验。

2、任务相关性实验

2.1 一般的多任务学习模型框架

一般的多任务学习模型框架如下：



(a)

对于不同的任务，底层的参数和网络结构是共享的，然后上层经过不同的神经网络得到对应任务的输出。假设底层输出是f(x)，那么第k个任务的输出y_k为：

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

其中h_k是第k个任务上层神经网络的参数。

2.2 任务相关性实验

接下来，我们通过一个实验来探讨任务相关性和多任务学习效果的关系。

假设模型中包含两个回归任务，而数据通过采样生成，并且规定输入相同，输出label不同。那么任务的相关性就使用label之间的皮尔逊相关系数来表示，相关系数越大，表示任务之间越相关，数据生成的过程如下：

(1) Given the input feature dimension d, we generate two orthogonal unit vectors $u_1, u_2 \in \mathbb{R}^d$, i.e.,

$$u_1^T u_2 = 0, \|u_1\|_2 = 1, \|u_2\|_2 = 1.$$

(2) Given a scale constant c and a correlation score $-1 \leq p \leq 1$, generate two weight vectors w_1, w_2 such that

$$w_1 = cu_1, \ w_2 = c\left(pu_1 + \sqrt{(1-p^2)}u_2\right). \tag{2}$$

(3) Randomly sample an input data point $x \in \mathbb{R}^d$ with each of its element from $\mathcal{N}(0, 1)$.

(4) Generate two labels y_1, y_2 for two regression tasks as follows,

$$y_1 = w_1^T x + \sum_{i=1}^{m} \sin\left(\alpha_i w_1^T x + \beta_i\right) + \epsilon_1 \tag{3}$$

$$y_2 = w_2^T x + \sum_{i=1}^{m} \sin\left(\alpha_i w_2^T x + \beta_i\right) + \epsilon_2 \tag{4}$$

where $\alpha_i, \beta_i, i = 1, 2, \ldots, m$ are given parameters that control the shape of the sinusoidal functions and $\epsilon_1, \epsilon_2 \overset{\text{i.i.d}}{\sim} \mathcal{N}(0, 0.01)$,

(5) Repeat (3) and (4) until enough data are generated.

首先，生成了两个垂直的单位向量u1和u2，并根据两个单位向量生成了模型的系数w1和w2，如上图中的第二步。w1和w2之间的cosine距离即为p，大伙可以根据cosine的计算公式得到。

随后基于正态分布的到输入数据x，而y根据下面的两个式子的到：

$$y_1 = w_1^T x + \sum_{i=1}^{m} \sin\left(\alpha_i w_1^T x + \beta_i\right) + \epsilon_1$$

$$y_2 = w_2^T x + \sum_{i=1}^{m} \sin\left(\alpha_i w_2^T x + \beta_i\right) + \epsilon_2$$

注意，这里x和y之间并非线性的关系，因为模型的第二步是多个sin函数，因此label之间的皮尔逊相关系数和参数w_1和w_2之间的cosine距离并不相等，但是呈现出一个正相关的关系，如下图：

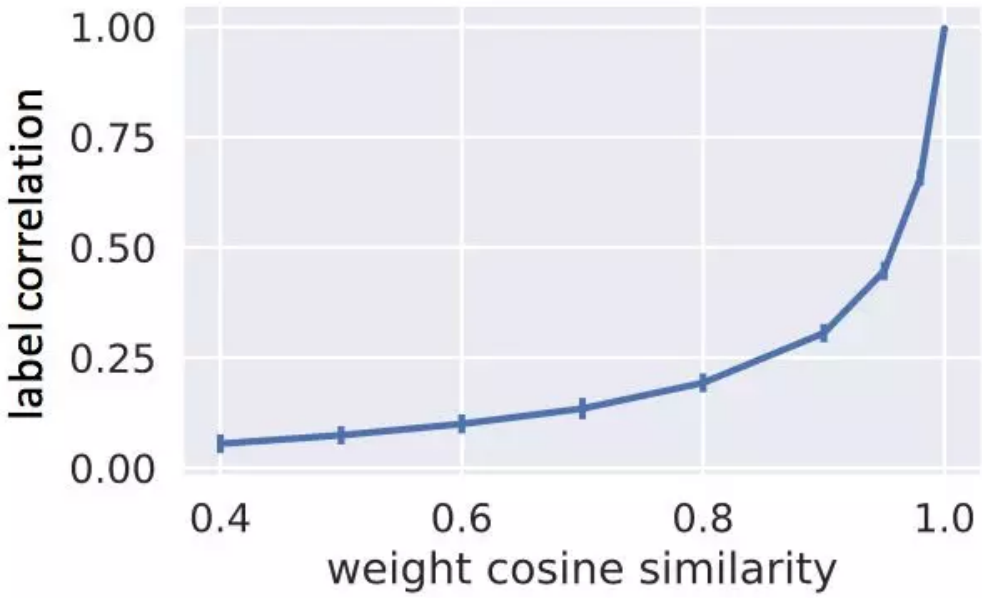


Figure 2: Label Pearson correlation v.s. weight cosine similarity (task correlation). X-axis shows the cosine similarities of weight vectors. Y-axis is the resulting Pearson correlation between the labels. For each weight cosine similarity, we generate 10k data points with two labels and calculate the Pearson correlation between these two labels. We repeat this process and plot the average with the error bar indicating 2 standard deviations among the 100 trials.

因此，本文中使用参数的cosine距离来近似表示任务之间的相关性。

2.3 实验结果

基于上述数据生成过程以及任务相关性的表示方法，分别测试任务相关性在0.5、0.9和1时的多任务学习模型的效果，如下图：

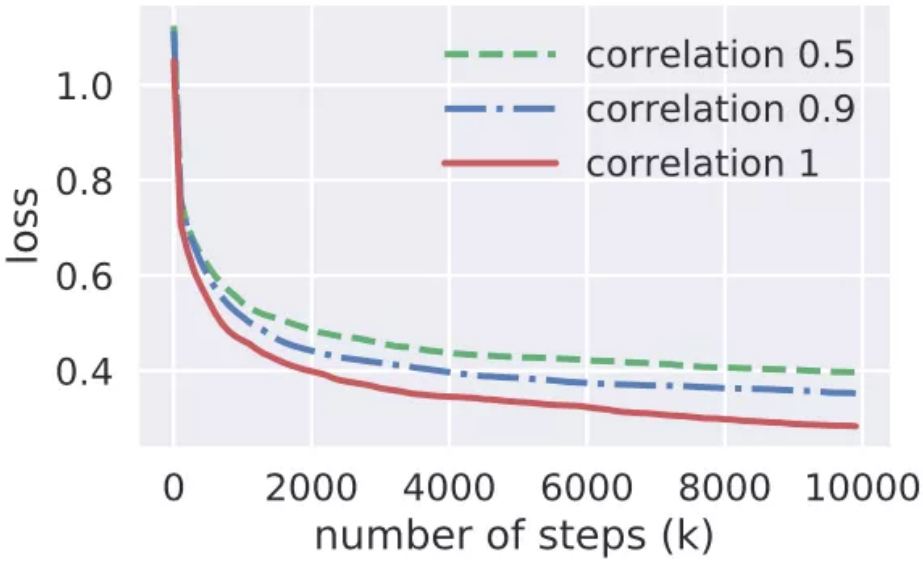


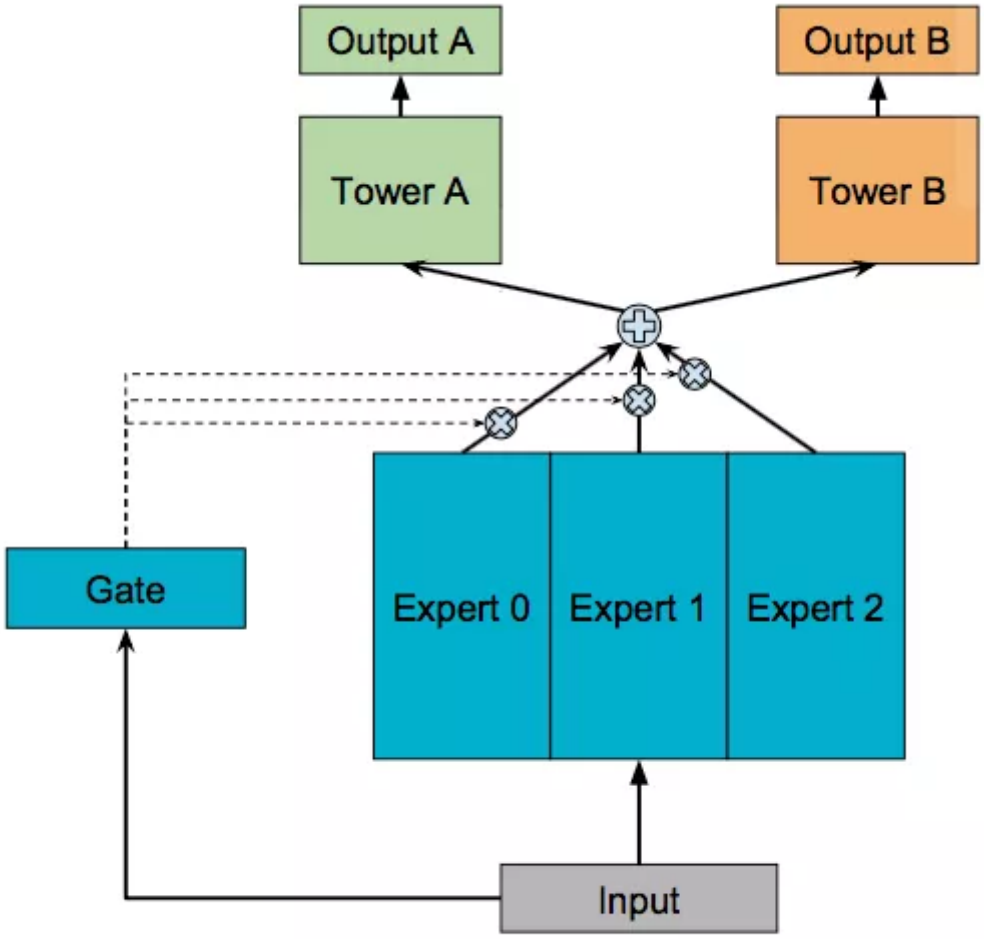
Figure 3: Performance of the Shared-Bottom model on synthetic data with different task correlation. Tasks with task correlation 1 means the two tasks have the same weight vectors but independent noises. X-axis is the number of training steps. Y-axis is the average loss of 200 independent runs.

可以看到的是，随着任务相关性的提升，模型的loss越小，效果越好，从而印证了前面的猜想。

3、MMoE模型

3.1 MoE模型

先来看一下Mixture-of-Experts (MoE)模型（文中后面称作 One-gate Mixture-of-Experts (OMoE)），如下图所示：



(b)

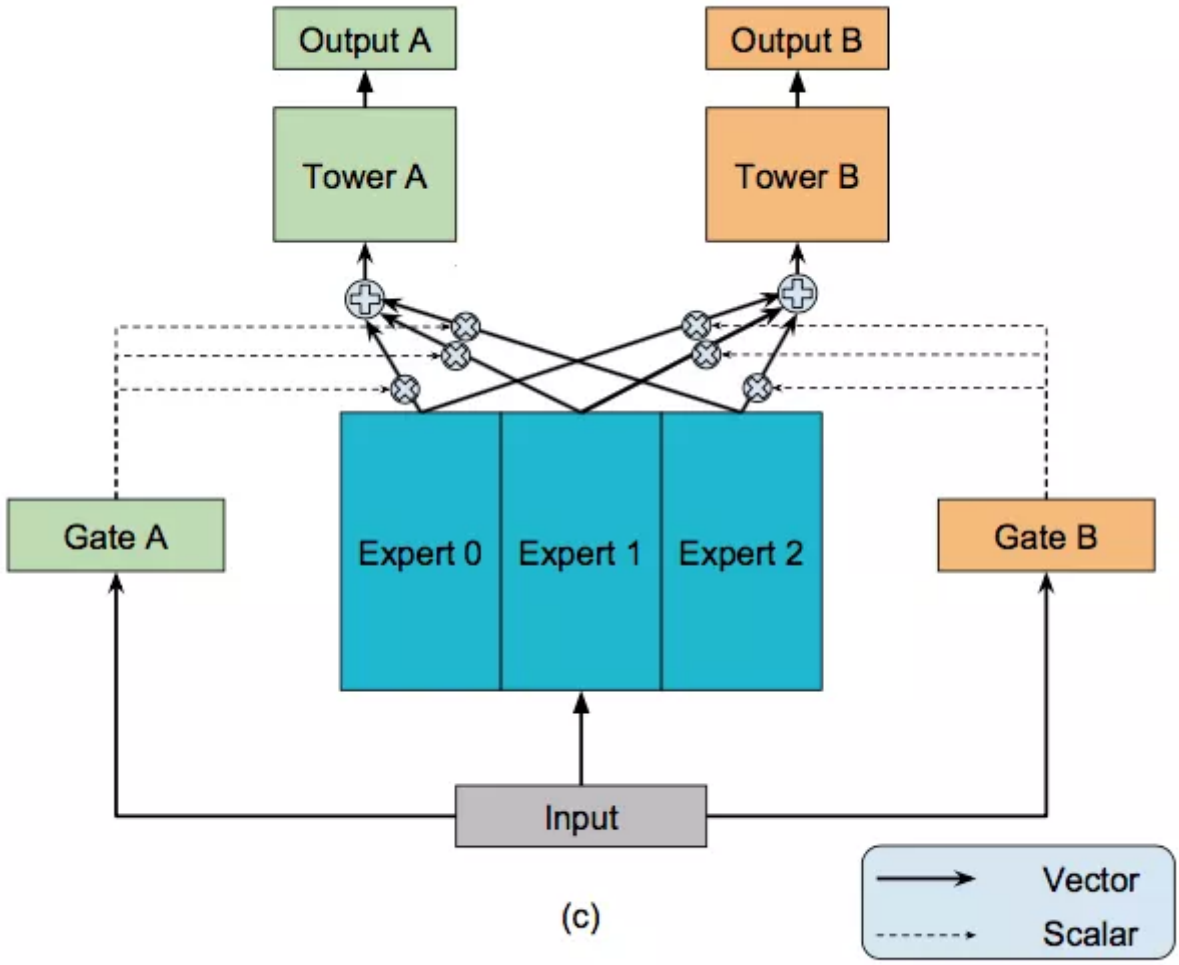
可以看到，相较于一般的多任务学习框架，共享的底层分为了多个expert，同时设置了一个Gate，使不同的数据可以多样化的使用共享层。此时共享层的输出可以表示为：

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

其中f_i代表第i个expert的输出，g_i代表第第i个expert对应的权重，是基于输入数据得到的，计算公式为$g(x) = \text{softmax}(W_g x)$。

3.2 MMoE模型

相较于MoE模型，Multi-gate Mixture-of-Experts (MMoE)模型为每一个task设置了一个gate，使不同的任务和不同的数据可以多样化的使用共享层，模型结构如下：



此时每个任务的共享层的输出不同，第k个任务的共享层输出计算公式如下：

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

$$g^k(x) = \text{softmax}(W_{gk}x),$$

随后每个任务对应的共享层输出，经过多层全连接神经网络得到每个任务的输出：

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

从直观上考虑，如果两个任务并不十分相关，那么经过Gate之后，二者得到的权重系数会差别比较大，从而可以利用部分expert网络输出的信息，近似于多个单任务学习模型。如果两个任务紧密相关，那么经过Gate得到的权重分布应该相差不多，类似于一般的多任务学习框架。

4、实验结果

先回顾上面介绍的三种多任务学习的架构：

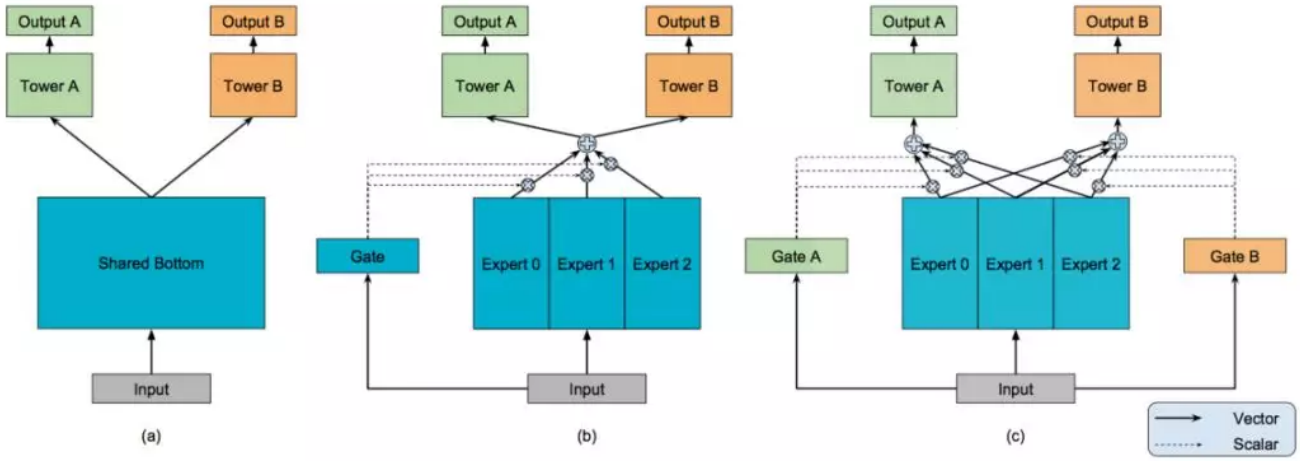


Figure 1: (a) Shared-Bottom model. (b) One-gate MoE model. (c) Multi-gate MoE model.

实验分为三部分：人工合成数据集（即本文第二部分所介绍的人工生成的数据集）、UCI census-income dataset和Large-scale Content Recommendation

4.1 人工合成数据集-实验结果

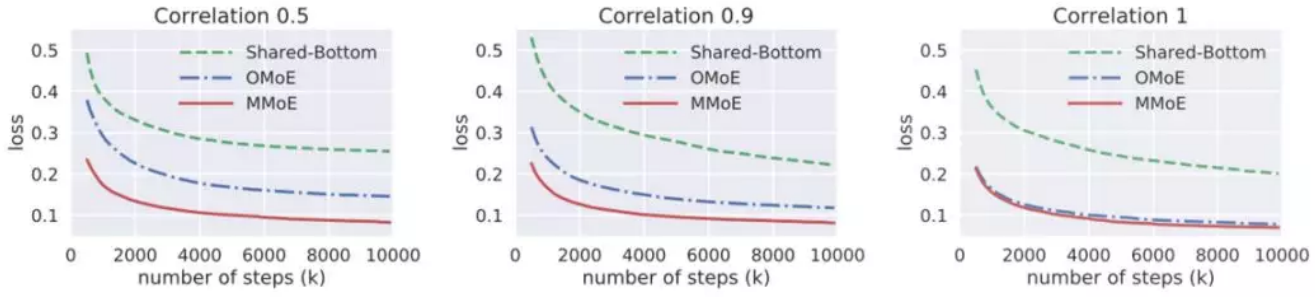


Figure 4: Average performance of MMoE, OMoE, and Shared-Bottom on synthetic data with different correlations.

4.2 UCI census-income dataset-实验结果

这块文中介绍了几种多任务学习的模式，这里就不过多介绍了。

Table 1: Performance on the first group of UCI Census-income dataset.

Group 1	AUC/Income		AUC/Marital Stat	
	best	mean	w/ best income	mean
Single-Task	0.9398	0.9337	**0.9933**	0.9922
Shared-Bottom	0.9361	0.9295	0.9915	0.9921
L2-Constrained	0.9389	0.9359	0.9922	0.9918
Cross-Stitch	0.9406	**0.9361**	0.9917	0.9922
Tensor-Factorization	0.7460	0.6765	0.8175	0.8412
OMoE	0.9387	0.9319	0.9928	0.9923
MMoE	**0.9410**	0.9359	0.9926	**0.9927**

4.3 Large-scale Content Recommendation-实验结果

Table 2: Performance on the second group of UCI Census-income dataset.

Group 2	AUC/Education		AUC/Marital Stat	
	best	mean	w/ best education	mean
Single-Task	0.8843	0.8792	**0.9933**	0.9922
Shared-Bottom	0.8836	0.8813	0.9927	0.9917
L2-Constrained	0.8855	0.8823	0.9923	0.9918
Cross-Stitch	0.8855	0.8819	0.9919	0.9921
Tensor-Factorization	0.7367	0.7256	0.7453	0.7497
OMoE	0.8852	0.8813	0.9915	0.9912
MMoE	**0.8860**	**0.8826**	0.9932	**0.9924**