



🏠 首页 (/)    📁 技术 (https://lumingdong.cn/category/tech)

📖 随笔 (https://lumingdong.cn/category/essay)    📷 摄影 (https://lumingdong.cn/photo)



🎵 音乐 (https://lumingdong.cn/music)    🧪 LAB (https://lab.lumingdong.cn)

👤 关于 (https://lumingdong.cn/about)

🏠 首页 (https://lumingdong.cn) / 技术 (https://lumingdong.cn/category/tech) 推荐系统  
(https://lumingdong.cn/category/tech/recommender\_system) / 推荐系统中的多任务学习

# 推荐系统中的多任务学习

📄 IN : 推荐系统 (HTTPS://LUMINGDONG.CN/CATEGORY/TECH/RECOMMENDER\_SYSTEM)

🕒 2019-11-30

💬 1 评论 (HTTPS://LUMINGDONG.CN/MULTI-TASK-LEARNING-IN-RECOMMENDATION-SYSTEM.HTML#COMMENTS)

📖 17

🔥 21 千字

⚡ 53 分钟



目录



1. 多目标排序综述
2. 迁移学习
  - 2.1. 迁移学习介绍
  - 2.2. 迁移学习的基本概念
  - 2.3. 迁移学习的分类
3. 多任务学习
  - 3.1. 单任务学习和多任务学习
  - 3.2. 多任务学习的定义
  - 3.3. 为什么多任务学习是有效的
  - 3.4. 多任务学习与其他学习方法的对比
  - 3.5. 多任务学习在推荐系统中的简单应用示例
4. ESMM
  - 4.1. 传统CVR预估存在的问题
  - 4.2. 学术界当前的解决方案
  - 4.3. ESMM算法原理
  - 4.4. ESMM的性能提升
  - 4.5. ESMM代码实现
5. MMoE
  - 5.1. MMoE要解决的问题
  - 5.2. MoE神经网络结构
  - 5.3. MoE与MTL的结合
  - 5.4. MMoE的性能提升
6. 工业界案例
7. 参考资料



## 1. 多目标排序综述

在之前的🔗《推荐系统中的排序学习》(/learning-to-rank-in-recommendation-system.html)一文中，我在最后简单提到过排序学习的一大作用就是可以用于多目标排序，也顺带提到了其他解决方案，本文我们就重点介绍另外一种解决方案，多任务学习。

还是先了解一下背景，介绍一下什么是多目标排序。

**多目标排序通常是指有两个或两个以上的目标函数，目的是寻求一种排序使得所有的目标函数都达到最优或满意。** 在工业界推荐系统中，大多是基于隐式反馈来进行推荐的，用户对推荐结果的满意度通常依赖很多指标（比如，淘宝基于点击，浏览深度（停留时间），加购，收藏，购买，重复购买，好评等行为的相关指标），在不同的推荐系统、不同时期、不同的产品形态下，这些指标的重要程度或者所代表的意义也会有所不同，如何优化最终推荐列表的顺序来使得众多指标在不同的场景下近可能达到最优或者满意，这就是一个多目标排序问题。

接下来我们来讨论一下为什么需要多目标排序。

因为工业界大都基于隐式反馈进行推荐，因此在评估用户满意度的时候会有不同程度的偏差：

- **Global bias**：不同目标表达不同的偏好程度



比如淘宝，购买表达的偏好程度要高于点击、浏览所表达的偏好程度；

- **Item bias**：单个目标衡量不全面

比如今日头条，仅以点击率为目标，可能会存在标题党；

比如抖音短视频，只是以视频播放完成度为目标，可能会存在设置悬念而后需要关注才能观看的视频，用户看完了可能也不满意；

还有微博，如果仅以转发分享为目标，但内容可能存在类似转发保平安，或者是拼多多之类的营销；



- **User bias**：用户表达满意度的方式不同

比如淘宝，用户表示喜欢可能会以收藏或者加购的形式，不同的人偏好也不一样；

- **综合目标收益最大化**

比如快手，要鼓励用户发布视频，该目标虽然与用户满意度不是那么相关，但对于平台的生态来说也是至关重要的，所以利用多目标可以兼顾更大的范围，使综合目标收益最大化。

那么，业界解决多目标排序问题的方案有哪些呢？

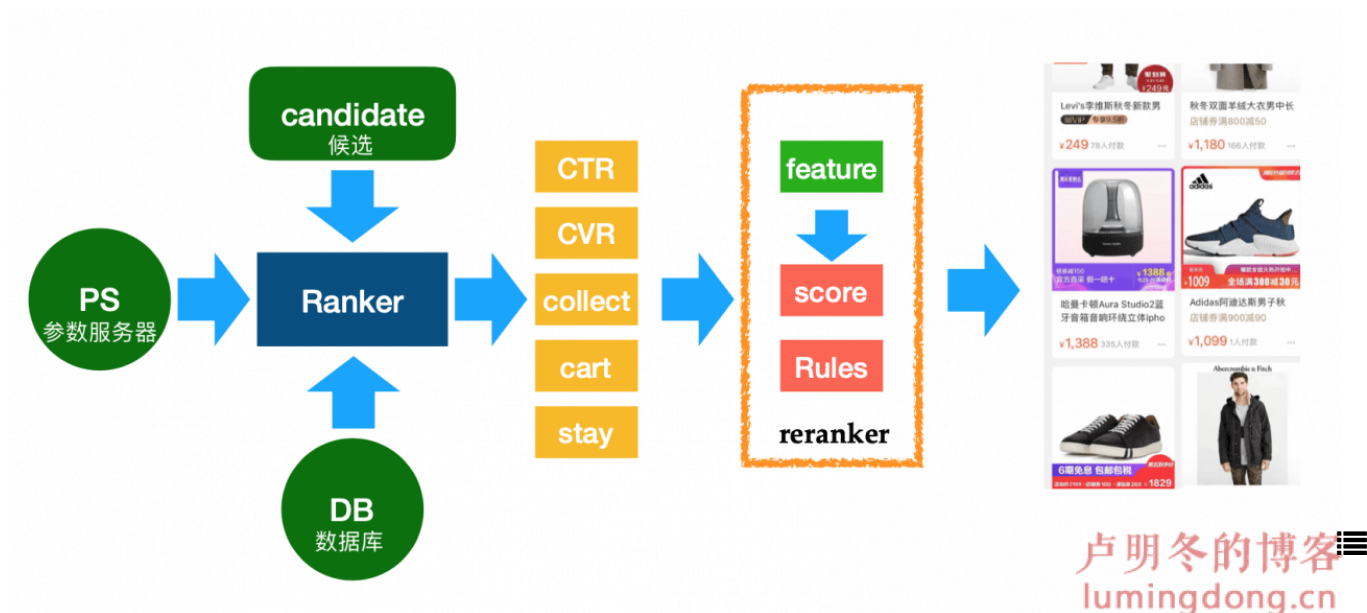
工业界解决多目标问题的方案基本有三种策略：**多模型分数融合**、**排序学习（Learning To Rank, LTR）**、**多任务学习（Multi-Task Learning, MTL）**。

### 多模型分数融合

多模型融合的方式也是比较经典传统的做法，每个目标训练一个模型，每个模型算出一个分数，然后根据自身业务的特点，通过某种方式将这些分数综合起来，计算出一个总的分数再进行排序，综合分数的计算通常会根据不同目标的重要性设定相应的参数来调节。

下面以类似淘宝的电商推荐系统为例，了解一下经典的多模型分数融合实现多目标排序的具体流程。





(<https://lumingdong.cn/wp-content/uploads/2019/11/20191110161846.png>)

上图流程图展示的是一个工业界很常用的推荐流程，包括召回（Match）、排序（Rank）、重排序（Rerank）、推荐（Recommend）过程。

当完成召回候选阶段之后，就有了可供推荐的候选 Item，然后从数据库中获取 User 和 Item 特征（特征缓存），从参数服务器获取模型参数（权重缓存），就可以进行预估。这里假定淘宝有五个预估目标，分别是点击率 CTR、购买转化率 CVR、收藏率 collect，加购率 cart、停留时长 stay，这五个目标分别对应五个模型，排序阶段的作用就是利用模型根据各自目标来给候选 Item 计算一个预估值（分数），排序阶段结束每个 Item 都会有五个不同的目标预估分数，如何用这些分数，是交给下一个流程来处理的。

重排序模块有两个主要作用，一个是融合，也就是将排序阶段传递来的多个目标分数进行融合，形成一个分数，另外一个是约束，也就是规制过滤。

分数融合的一种思路是利用一个带参数公式来实现，如下：

$$\text{score} = \text{CTR} * (\alpha + \text{CVR}) * (\beta + \text{price}) * \text{stay}^a * \text{cart}^b * \text{collect}^c * \text{rule\_wight}.$$

可以发现，在计算 score 时，除了各个目标以及对应的参数外，公式中还多了两个部分，一个是价格 price 以及对应的参数  $\beta$ ，另外一个是规则权重 rule\_wight。

公式中添加价格，是因为价格是影响最终满意度非常重要的一个因素，比如对于同一个物品，它的价格通常是随着营销策略浮动的，不同的价格转化率就会不同，因为在降价的时候，商品销量通常会明显升高；而对于不同物品，价格便宜的物品转化率也会更高，因为便宜的物品受众用户更广，试错成本也低，而一些比较昂贵的物品，受众用户较窄，试错成本很高，自然转化率就会较低，所以加入价格 price，可以很好的区分这些情况。

另外一个规则权重，其实是把规则过滤也放到了分数融合里面，当然有些场景下，规则过滤是单独的一个部分，这个单独的约束部分，可以称为**规则系统（Rules）**，它也是推荐系统中非常重要的一环。

商品按最终预估分排序后，会进入规则系统依次进行核对，判断是否满足规则系统。这些规则可能包含如下几种：

- ① 同一 XX N 出 1：该规则主要目的是使推荐满足多样性、惊喜性等目标，比如同一个类目只能推一个，同一个品牌只能推一个，当然，这里的 1 也可以任意合理的数值；
- ② 活动扶持：比如最近在双十一活动，那么会推荐更多参与了活动的商品；
- ③ 新品扶持：新品刚上市，相关指标预估分会很低，或者面临冷启动问题，会有相应扶持；
- ④ 低俗打压：比如快手，不能因为部分低俗视频流量高，就过多曝光，这样会影响整个产品个人的印象，因此会打压低俗，进行产品调性的控制；
- ⑤ 流量控制：为了让流量更均匀的分发到各个商品上，发掘长尾，不能给某一商品过多展示的机会，比如限制某一商品一天只能曝光 100 万次。

有了公式，那么超参数（ $\alpha$ ,  $\beta$ , a, b, c 等）如果进行学习获取？

先假设没有上面的经验公式，我们很容易想到的一种思路是类似于集成学习中一种多模型融合方法 Stacking，即将多个目标模型预估出的结果作为输入，然后用一个简单的线性回归进行线性加权融合，学习到各个目标的权重，这样我们就可以预估综合分数了。但是我们却忽略了一个重要的问题，该如何设置样本的 Label 呢？事实上，并没有一个真实的综合分数可供我们去训练学习，因此，这种办法很难真正去实现。而在工业界，更多的做法是人工调试，但如此又会带来很多问题，比如模型灵活度不够。

这种经典的多模型分数融合方法，虽然易于理解和扩展，但也存在很多问题和难点：

### 1. 部分目标数据稀疏，模型准确率低

比如相比点击率转化率样本数据稀疏；

### 2. 在线服务计算量大

ranker 模块要对 5 个模型进行计算，代价可能比较大，参数、优化、维护都会成倍增加；

### 3. 多个目标间重要性难以量化

不同的动作提供不同程度的反馈，但由于用户偏好不同，有些动作所代表的重要性很难量化，比如分享和评论在不同用户身上的重要性可能是不一样的；

#### 4. 分数融合的**超参难以学习**，两种解决思路：

- 人工标注为 Label

用显示反馈数据来学习隐式反馈的超参，但工业界收集显示反馈数据成本非常高，而且非常主观。

- 长期目标为 Label

推荐系统的一个核心目的是给企业带来长期收益，所以可以用长期目标作为 Label，比如客户留存率，但是长期目标周期长，学习模型需要很长时间来收集 Label，容易错失机会；另外就是用长期目标试错成本高，策略不好的话无法及时发现，也就无法及时更改，可能导致用户流失而不能及时有效挽回。


**综合而言，还是人工调参更加现实。**


#### 5. 规则不够智能化

比如规则 5 个只能出 1 个，但是可能客户就想买鞋，然而并没有推荐很多。传统的规则系统主要是基于全局的角度上考虑的，没有考虑到用户的上下文去分析即时需求，不够智能，可以考虑采用强化学习的方法。

前两个问题工业界解决体系比较完善，比如下面提到的另外两种方法，主要目的就是尽可能使多目标排序的预估更快更准，第三个问题偏产品设计，而最后两种并没有比较完善的解决方法。<sup>1</sup>

#### 排序学习（Learning To Rank, LTR）

首先要明白一点，多模型融合中我们通过模型计算预估值或者是综合打分，其根本目的是为了给推荐物品排序，而不是真正的打分。因此我们可以使用排序学习方法来解决多目标问题，比如 BPR 或者 LambdaMART 算法，排序学习的概念和相关算法在之前的文章  《推荐系统中的排序学习》(<https://lumingdong.cn/learning-to-rank-in-recommendation-system.html>)已经详细介绍过。

相比多模型融合中 Label 标注，排序学习模型的 Label 标注相对更容易一点，因为只关心相对关系，而不需要真实的打分数据。一种常见的标注方法是对多目标产生的物品构建 Pair，比如用户对物品  $i$  产生的购买，对物品  $j$  产生了点击，假定我们觉得购买的目标比点击的目标更 

重要，就可以让  $i >_u j$ ，其他目标以此类推。有了顺序对后，我们便可以训练排序学习模型，这样一个模型便可以融合多个目标，而不用训练多个模型。

### 排序学习解决多目标排序问题的优缺点有哪些呢？

优点：

- 直接优化排序目标，排序效果好
- 单模型融合多目标，工程代价小，serving 压力小

缺点：

- 样本数量大，训练速度慢



排序学习虽然能够减少模型数量，但是要想学习多个目标排序关系，其合起来构建的样本数量会很大（假设有  $m$  个正样本， $n$  个负样本，PointWise 是  $m + n$  个参数，PairWise 则对于单个用户可能是  $m \times n$ ），会导致训练速度非常慢；

- 有些偏序关系不容易构造

实际工业数据中并没有排序学习算法中那些完美的假设，有些目标间可能不存在偏序关系，也不存在传递性，还有一些目标所代表的重要性会因用户偏好而存在差异，这样的话，偏序关系并不容易构造；

- 多目标间的关系不易调整

举例理解，比如淘宝双十一之前一个周，目标更多是让用户加购或收藏（偏序关系：加购  $>$  购买），而双十一当天系统更加偏向于转化为目标（偏序关系：购买  $>$  加购），要知道排序关系的确定其实是从样本中体现的，如果为了重新定义目标的重要度而全部重新构造样本，这样很多流程也会随之改变，代价会非常大。

### **多任务学习（Multi-Task Learning, MTL）**

对于前面两种方法，或多或少都有些难以避免的缺陷，工业界是如何作取舍呢？在此之前，大部分公司的做法就是多模型分数融合方法，一个模型一个目标，不过只用较少的几个关键目标进行融合；也有一小部分公司直接用排序学习，用一个模型就搞定；而那些先进的大公司，通常会使用多任务学习来实现多目标排序。

多任务学习就是本节的重头戏，它也是最近两年非常热门的多目标排序的解决方案。多任务学习属于迁移学习的一种，通过共享参数，学习出多个分数，最后结合起来。典型的算法有谷歌的 **MMOE（Multi-gate Mixture-of-Experts）** 以及阿里的 **ESMM（Entire Space Multi-**

Task Model) 。

接下来，我们详细介绍多任务学习的相关知识点和推荐系统中的经典应用。

## 2. 迁移学习

多任务学习 MTL 其实是迁移学习的一种，这一小结我们先了解一下迁移学习这个体系。

### 2.1. 迁移学习介绍

在机器学习、深度学习和数据挖掘的大多数任务中，我们都会假设 training 和 inference 时，采用的数据服从相同的分布（distribution）、来源于相同的特征空间（feature space）。但在现实应用中，这个假设很难成立，往往遇到一些问题：

1. 带标记的训练样本数量有限。比如，处理 A 领域（target domain）的分类问题时，缺少足够的训练样本。同时，与 A 领域相关的 B（source domain）领域，拥有大量的训练样本，但 B 领域与 A 领域处于不同的特征空间或样本服从不同的分布。
2. 数据分布会发生变化。数据分布与时间、地点或其他动态因素相关，随着动态因素的变化，数据分布会发生变化，以前收集的数据已经过时，需要重新收集数据，重建模型。

这时，知识迁移（knowledge transfer）是一个不错的选择，即把 B 领域中的知识迁移到 A 领域中来，提高 A 领域分类效果，不需要花大量时间去标注 A 领域数据。迁移学习，作为一种新的学习范式，被提出用于解决这个问题。

**迁移学习（Transfer Learning, TL）**是属于机器学习的一种研究领域。它专注于存储已有问题的解决模型，并将其利用在其他不同但相关问题上。比如说，用来辨识汽车的知识（或者是模型）也可以被用来提升识别卡车的能力。

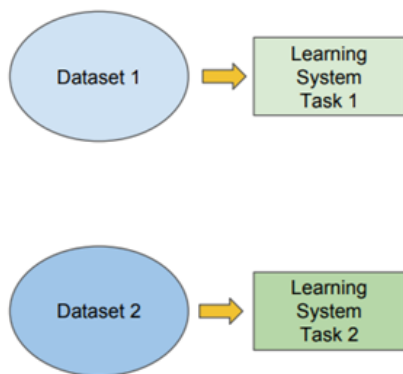


## Traditional ML

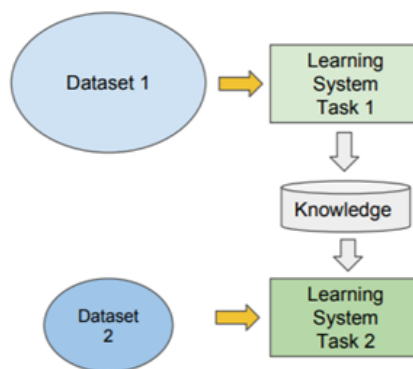
vs

## Transfer Learning

- Isolated, single task learning:
  - Knowledge is not retained or accumulated. Learning is performed w.o. considering past learned knowledge in other tasks



- Learning of a new tasks relies on the previous learned tasks:
  - Learning process can be faster, more accurate and/or need less training data



(<https://lumingdong.cn/wp-content/uploads/2019/11/20191130201008.png>)

在任务间迁移知识是人类与生具有的能力，通过完成某个任务所获得的知识，同样可用于解决其他相关的任务。传统机器学习和深度学习算法通常在设计上是独立工作的，这些算法训练用于解决特定的问题，学习中也并未保留任何可从一种模型迁移到另一种模型上的知识，一旦特征空间的分布发生变化，就需要从头开始重新构建模型。**迁移学习设计用于解决这些相互隔离的学习方式，可以利用先前训练模型中的知识（即特征，权重等）训练新的模型，实现从其它任务获取的知识去解决相关的问题，甚至可以解决诸如新任务具有较少数据等问题。**

迁移学习可以解决哪些问题？


1. **大数据与少标注的矛盾**：虽然有大量的数据，但往往都是没有标注的，无法训练机器学习模型。人工进行数据标定太耗时。
2. **大数据与弱计算的矛盾**：普通人无法拥有庞大的数据量与计算资源。因此需要借助于模型的迁移，比如图像识别中进行对训练好的模型进行 fine-tuning 完成自己的任务。
3. **普适化模型与个性化需求的矛盾**：即使是在同一个任务上，一个模型也往往难以满足每个人的个性化需求，比如特定的隐私设置。这就需要在不同人之间做模型的适配。
4. **特定应用（如冷启动）的需求**。

迁移学习使得我们能够利用前期学习任务中的知识，并将这些知识应用于新的相关任务。因此在迁移学习的过程中，必须能够回答如下三个重要的问题：

- **迁移什么 (What to transfer)**：给定一个目标领域，如何找到相对应的源领域，或者说我们需要走确定哪些知识可以从源迁移到目标。在回答这个问题时，我们会试图确定哪些知识是特定于源的，哪些知识在源和目标之间的是共同的。然后进行迁移？（源领域选择）

- **何时迁移 (When to transfer)** : 什么时候可以进行迁移, 什么时候不可以? 在某些场景中, 迁移知识可能要比改进知识更糟糕 (该问题也称为 “负迁移”)。我们的目标是通过迁移学习改善目标任务的性能或结果, 而不是降低它们。(避免负迁移)
- **如何迁移 (How to transfer)** : 如何进行迁移学习? 我们需要确定跨域或跨任务实现知识实际迁移的方法。该问题涉及如何改进现有的算法和各种技术 (设计迁移方法)

## 2.2. 迁移学习的基本概念

迁移学习中有两个非常重要的概念：**域 (domain)** 和 **任务 (task)**，在 Sinno Jialin Pan 和杨强教授发表的综述论文 “ A Survey on Transfer Learning (https://lumingdong.cn/go/uxsldr)” 中有关于域和任务的严格数学定义：



- **域 (domain)** : 一个域  $D$  定义为由特征空间  $\mathcal{X}$  和边缘概率  $P(X)$  组成双元素元组, 可以形式化表示为  $D = \{\mathcal{X}, P(X)\}$ , 其中  $X = \{x_1, x_2, \dots, x_n\}, x_i \in \mathcal{X}$ .
    - 特征空间  $\mathcal{X}$  代表了所有可能特征向量取值
    - 边缘概率分布  $P(X)$  代表了某种特定的采样, 例如  $\mathcal{X}$  是一个二维空间,  $P(X)$  为过原点的一条直线。
- 边缘分布函数:** 如果二维随机变量  $X, Y$  的分布函数  $F(x, y)$  为已知, 那么随机变量  $x, y$  的分布函数  $F_x(x)$  和  $F_y(y)$  可由  $F(x, y)$  求得。则  $F_x(x)$  和  $F_y(y)$  为分布函数  $F(x, y)$  的边缘分布函数。
- **任务 (task)** : 在给定的域  $D = \{\mathcal{X}, P(X)\}$  中, 一个任务  $T$  定义为由标签空间  $\mathcal{Y}$  和预测函数  $f(\cdot)$  组成的双元素元组, 预测函数是基于输入的特征向量和标签学习而来的, 它通常写为条件概率分布  $P(Y|X)$ 。可以形式化表示为  $T = \{\mathcal{Y}, P(Y|X)\} = \{\mathcal{Y}, f(\cdot)\}$ , 其中  $Y = \{y_1, y_2, \dots, y_n\}, y_i \in \mathcal{Y}$ .
    - 预测函数  $f(\cdot)$  并不是已知的, 不过可以通过特征向量和标签对  $(x_i, y_i)$  学习而来, 其中  $x_i \in \mathcal{X}, y_i \in \mathcal{Y}$ 。

论文中用域和任务的概念来严格的定义了迁移学习：

给定了源域  $D_S$  和源任务  $T_S$  以及目标域  $D_T$  和目标任务  $T_T$ , 那么迁移学习的标就是通过源域  $D_S$  和源任务  $T_S$  中获得的一些知识来使我们能够学习到目标域  $D_T$  中的目标条件概率分布  $P(Y_T|X_T)$ , 从而提升目标任务  $T_T$  中预测函数  $f_T(\cdot)$  预测能力的一种算法, 其中  $D_S \neq D_T$  或  $T_S \neq T_T$ 。通常, 假定目标域中标记样本的数量要远远小于源域中标记样本的数量。



从更加容易理解的角度去解释域和任务的概念：

- **域 (Domain):** 数据特征和特征分布组成，是学习的主体
  - **源域 (Source domain):** 已有知识的域
  - **目标域 (Target domain):** 要进行学习的域
- **任务 (Task):** 由目标函数和学习结果组成，是学习的结果

根据域、任务和数据的可用性不同，形成了不同的迁移学习策略和技术路线，而关于“域” (domain) 和 “任务” (task) 二者间的差异，我们需要更进一步去理解：



域的不同有两种可能的场景：

- **特征空间不同 ( $\mathcal{X}_S \neq \mathcal{X}_T$ )**

例如文本分类任务中，中文文本和英文文本的特征空间不同；

图像识别任务中，人脸图片和鸟类图片的特征空间不同；

- **边缘概率分布不同 ( $P(X_S) \neq P(X_T)$ )**

例如文本分类任务中，文本都是中文语言的特征空间，但讨论的是不同的主题，如政治和娱乐；

图片识别任务中，图片都是鸟类的特征空间，但一个是在城市中拍到的鸟，一个是在大自然中拍到的。

任务的不同也有两种可能的场景：

- **标签空间不同 ( $\mathcal{Y}_S \neq \mathcal{Y}_T$ )**

例如在文本分类任务中，一个标签是新闻类别标签，一个标签是文本情感标签；

人脸识别任务中，一个标签是性别，一个标签是人名。

- **条件概率不同 ( $P(Y_S|X_S) \neq P(Y_T|X_T)$ )**

例如源和目标数据类别分布不均衡。这种情况非常普遍，可用过采样 (over-sampling)、欠采样 (under-sampling)、SMOTE 等方法进行处理。



一般标签不同，条件概率分布也会不同，因为很少会出现两个不同的任务有不同的标签空间而条件概率分布完全相同的情况。

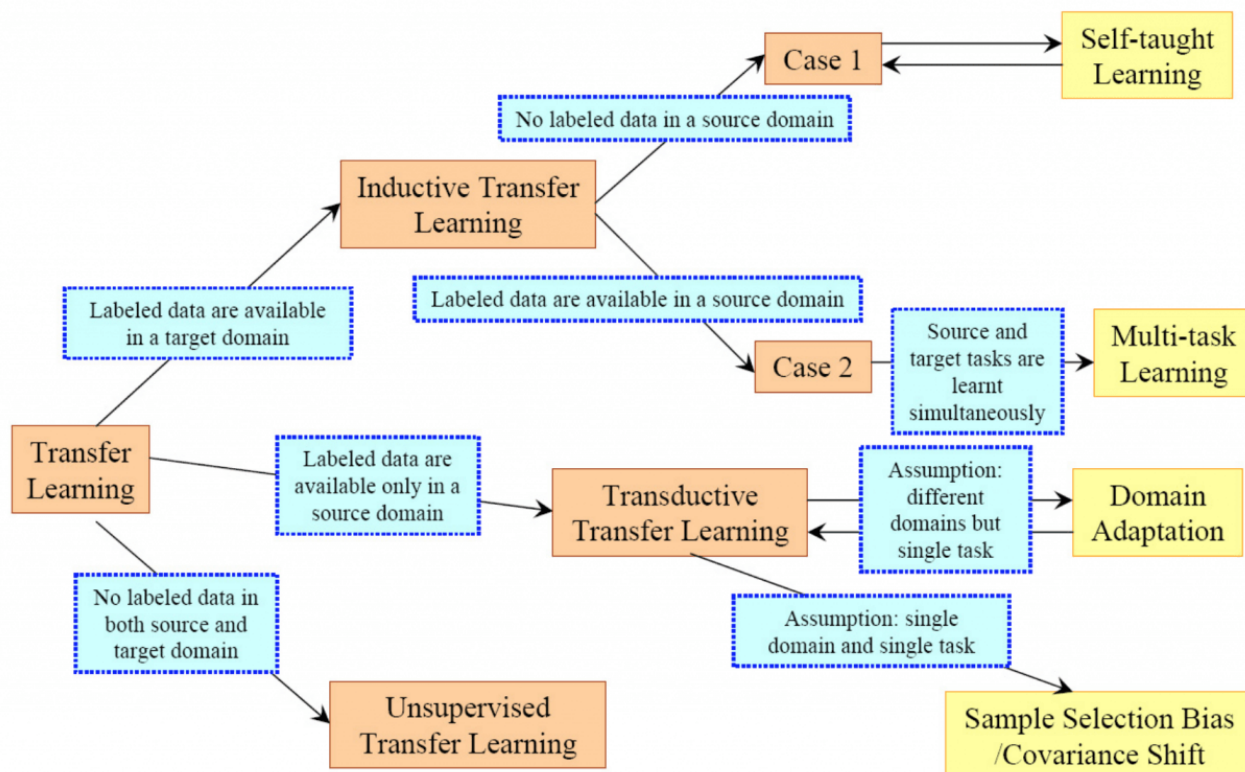
也有论文将域（domain）和 任务（task）合二为一称之为一个 dataset，cross-dataset 指的就是 domain 或者 task 不同。

## 2.3. 迁移学习的分类

### 按特征空间分

- 同构迁移学习（Homogeneous TL）：源域和目标域的特征空间相同（ $X_S = X_T$ ）
- 异构迁移学习（Heterogeneous TL）：源域和目标域的特征空间不同（ $X_S \neq X_T$ ）

### 按迁移情景分



(<https://lumingdong.cn/wp-content/uploads/2019/12/20191111213720.png>)

参考上图，根据所使用的传统机器学习算法，迁移学习方法可分类为：

- **归纳式迁移学习（Inductive Transfer learning）**：在该场景中，源域和目标域相同，但源任务和目标任务彼此不同。算法尝试利用来自源域的归纳偏差帮助改进目标任务。根据源域中是否包含标记数据，归纳式迁移学习可以进一步分为类似于**多任务学习（Multi-task Learning）**和**自学习（Self-taught Learning）**这两类方法。

- **直推式迁移学习 (Transductive Transfer Learning) :** 源域和目标域不同，学习任务相同。在该场景中，源任务和目标任务之间存在一些相似之处，但相应的域不同。通常源域具有大量标记数据，而目标域没有。根据特征空间或边缘概率的设置不同，直推式迁移学习可进一步分类为多个子类。
- **无监督迁移学习 (Unsupervised Transfer Learning) :** 源域和目标域均没有标签。该场景类似于归纳式迁移学习，重点关注目标域中的无监督任务。

下表总结了上述迁移学习策略在不同场景和领域下的对比：

Learning Strategy	Related Areas	Source & Target Domains	Source Domain Labels	Target Domain Labels	Source & Target Tasks	Tasks
Inductive Transfer Learning	Multi-task Learning	The Same	Available	Available	Different but Related	Regression Classification
	Self-taught Learning	The Same	Unavailable	Available	Different but Related	Regression Classification
Unsupervised Transfer Learning		Different but Related	Unavailable	Unavailable	Different but Related	Clustering Dimensionality Reduction
Transductive Transfer Learning	Domain Adaptation, Sample Selection Bias & Co-variate Shift	Different but Related	Available	Unavailable	The Same	Regression Classification

(<https://lumingdong.cn/wp-content/uploads/2019/11/20191111214248.png>)

按迁移内容来分：

- **基于样本的迁移学习 (Instance transfer) :** 通常，理想场景是源域中的知识可重用到目标任务。但是在大多数情况下，源域数据是不能直接重用的。然而，源域中的某些实例是可以与目标数据一起重用，达到改善结果的目的。对于归纳式迁移，已有一些研究利用来自源域的训练实例来改进目标任务，例如 Dai 及其合作研究者对 AdaBoost 的改进工作。
- **基于特征表示的迁移学习 (Feature-representation transfer) :** 该类方法旨在通过识别可以从源域应用于目标域的良好特征表示，实现域差异最小化，并降低错误率。根据标记数据的可用性情况，基于特征表示的迁移可采用有监督学习或无监督学习。
- **基于参数的迁移学习 (Parameter transfer) :** 该类方法基于如下假设：针对相关任务的模型间共享部分参数，或超参数的先验分布。不同于同时学习源和目标任务的多任务学习，在迁移学习中我们可以对目标域的应用额外的权重以提高整体性能。
- **基于关系知识的迁移学习 (Relational-knowledge transfer) :** 与前面三类方法不同，基于关系知识的迁移意在处理非独立同分布 (i.i.d) 数据即每个数据点均与其他数据点存在关联。例如，社交网络数据就需要采用基于关系知识的迁移学习技术。

下表清晰地总结了不同迁移内容分类和不同迁移学习策略间关系 <sup>2</sup>：

	Inductive Transfer Learning	Transductive Transfer Learning	Unsupervised Transfer Learning
Instance-transfer	✓	✓	
Feature-representation-transfer	✓	✓	✓
Parameter-transfer	✓		
Relational-knowledge-transfer	✓		

(<https://lumingdong.cn/wp-content/uploads/2019/11/20191111214554.png>)

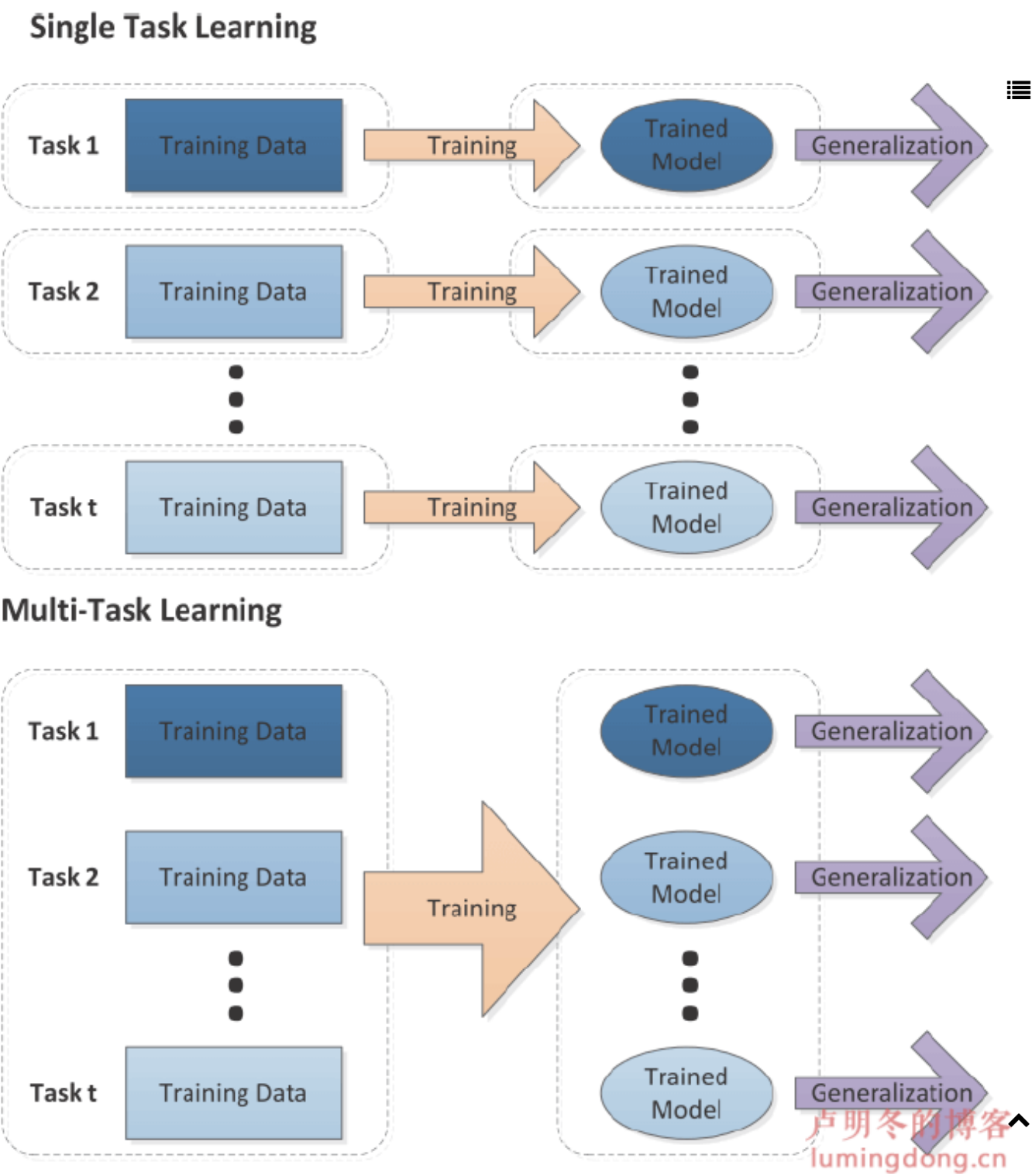
### 3. 多任务学习



从上一小节我们了解到，多任务学习其实是迁移学习的一个分支，或者可以说是迁移学习中一类特殊的解决方法。多任务学习同时学习若干任务，并不区分源和目标。与经典的迁移学习相比，多任务学习的学习器最初并不知道目标任务，它一次接收多个任务的相关信息。

### 3.1. 单任务学习和多任务学习

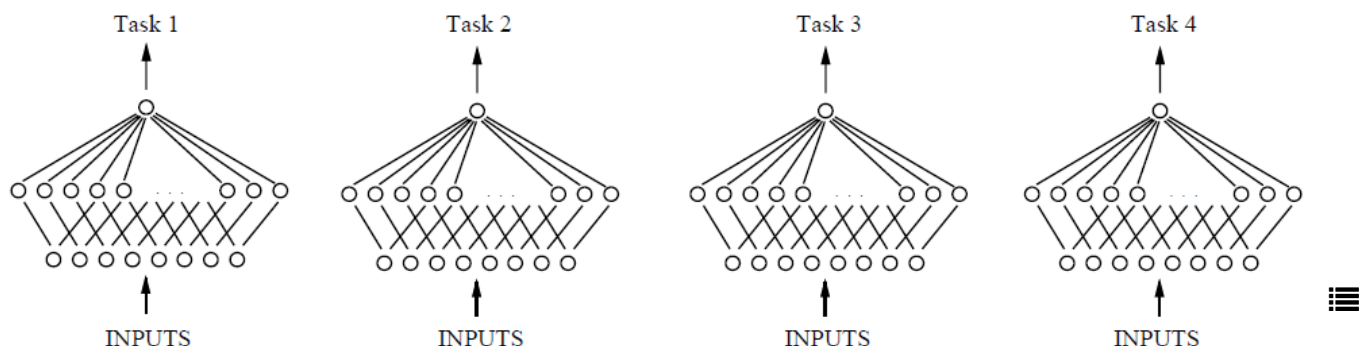
单任务学习时，各个任务之间的模型空间（Trained Model）是相互独立的，而多任务学习时，多个任务之间的模型空间（Trained Model）是共享的，对比如下图所示：



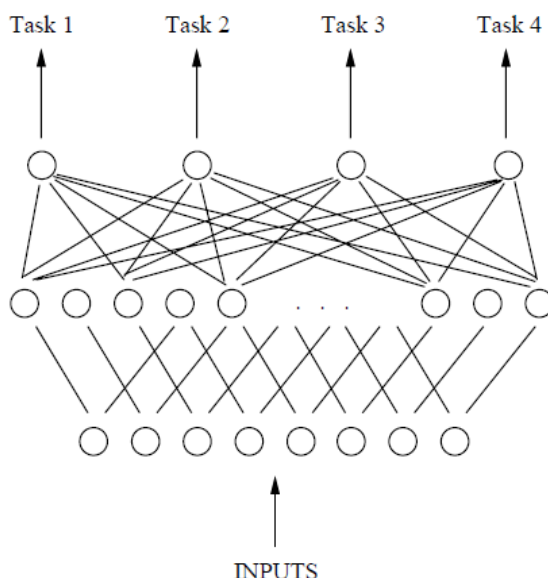


(<https://lumingdong.cn/wp-content/uploads/2019/11/20191115175726.png>)

另外一个明显对比是利用神经网络结构来学习，多任务学习时，多个任务之间的浅层表示共享（shared representation），而单任务学习时，各个任务的学习是相互独立的，如下图：



Single Task Backprop (STL) of four tasks with the same inputs



Multitask Backprop (MTL) of four tasks with the same inputs

(<https://lumingdong.cn/wp-content/uploads/2019/11/20191115181828.png>)

## 3.2. 多任务学习的定义

多任务学习（Multitask learning）是基于共享表示（shared representation），把多个相关的任务放在一起学习的一种机器学习方法。多任务学习涉及多个相关的任务同时并行学习，梯度同时反向传播，利用包含在相关任务训练信号中的特定领域的信息来改进泛化能力。

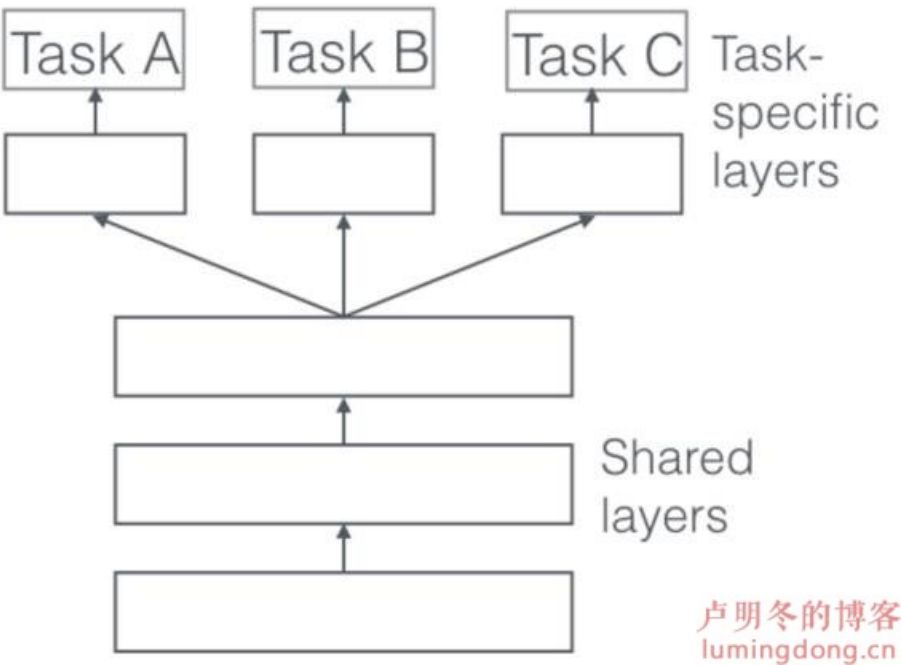
可以发现，多任务学习的定义中，有两个非常关键的限定，也是多任务得以实现的前提条件：多个任务之间必须具有相关性以及拥有可以共享的底层表示。

在多任务学习的定义中，共享表示是一个非常重要的限定，个人认为**共享表示对于最终任务的学习有两类作用**：一类是**促进作用**，通过浅层的共享表示互相分享、互相补充学习到的领域相关信息，从而互相促进学习，提升对信息的穿透和获取能力；另外一类是**约束作用**，在多个任务同时进行反向传播时，共享表示则会兼顾到多个任务的反馈，由于不同的任务具有不同的噪声模式，所以同时学习多个任务的模型就会通过平均噪声模式从而学习到更一般的表征，这个有点像正则化的意思，因此相对于单任务，过拟合风险会降低，泛化能力增强。

因此在深度神经网络中，执行多任务学习有两种最常用的方法：<sup>3</sup>

- 参数的硬共享机制（基于参数的共享，Parameter Based）

共享 Hard 参数是神经网络 MTL 最常用的方法。在实际应用中，通常通过在所有任务之间共享隐藏层，同时保留几个特定任务的输出层来实现，如下图所示：



(<https://lumingdong.cn/wp-content/uploads/2019/11/20191130194455.jpeg>)

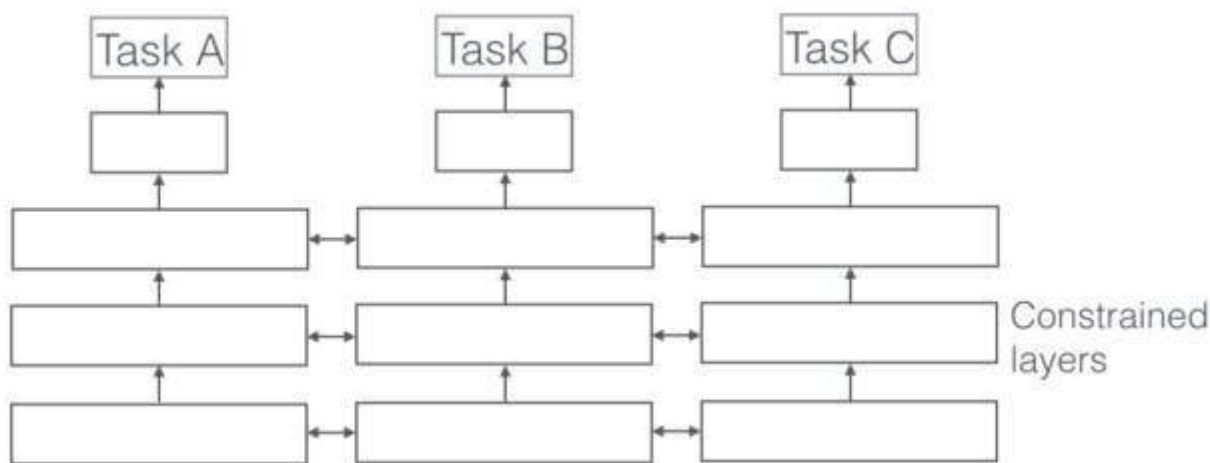
共享 Hard 参数大大降低了过拟合的风险。这点我们可以这样简单直观的理解：如果同时学习的任务越多，模型能够找到一个含有所有任务的表征就越困难，因而过拟合原始任务的可能性就越小。

- 参数的软共享机制（基于约束的共享，Regularization Based）

另外一种方式是共享 Soft 参数，每个任务都有自己的参数和模型。模型参数之间的距离是正则化的，以便鼓励参数相似化，例如使用 L2 距离进行正则化。







(<https://lumingdong.cn/wp-content/uploads/2019/11/20191130194609.jpeg>)



约束深度神经网络 Soft 参数共享的思想受到了 MTL 正则化技术的极大启发，这种思想已经用于其它模型开发。

MTL 是一种非常灵活的方法，可以使用不同形式的共享表示，从而形成得到不同的效果，这在深度神经网络中表现尤其明显。比如可以共享底层特征、共享中间层的隐藏单元、甚至共享模型某一层的结果，而共享表示之外各自独立的部分，也可根据各自任务的特点来灵活设计，可以用类似的特征组合和模型方法，也可以是多个任务使用完全不同的特征组合和模型方法。具体如何设计 MTL 的结构，首先在基于 MTL 基本原则情况下（存在共享表示和相关任务），应从任务特点、资源消耗、性能效果等方面去综合考虑。

因为 MTL 的结构设计比较灵活，所以 MTL 有很多形式：联合学习（joint learning）、自主学习（learning to learn）和带有辅助任务的学习（learning with auxiliary task）等。一般来说，优化多个损失函数就等同于进行多任务学习（与单任务学习相反）。即使只优化一个损失函数（如在典型情况下），也有可能借助辅助任务来改善原任务模型。

### 3.3. 为什么多任务学习是有效的

多任务学习之所以有效，是因为多任务学习的方式引入了归纳偏置（inductive bias），归纳偏置有两个效果，一个是互相促进，可以把多任务模型之间的关系看作是互相先验知识，也称归纳迁移（inductive transfer），有了对模型的先验假设，可以更好的提升模型的效果；另外一个效果是约束作用，借助多任务间的噪声平衡以及表征偏置来实现更好的泛化性能。我们下面详细解释：

1. 利用 MTL 减少深度学习对大数据量的依赖



解决数据稀疏性其实本身也是迁移学习的一个特性，在多任务学习中也同样会体现。以图像物体识别为例，假如 A 任务是识别汽车，而 B 任务是识别出越野车，可以知道，当样本数量有限时，B 任务的样本数量要远远小于 A 任务的样本数量。那么这个时候，多目标学习可以利用样本数量比较多的 A 任务，从 A 任务学习到一些处理后的高维抽象的 feature map 或者部分参数应用到 B 任务的学习，可以一定程度上缓解 B 任务数据稀疏问题。

## 2. 多任务间相互促进，对特征学习和表征更加充分，体现在以下三个方面：

### ◦ 多个模型特性互相弥补

不同的任务训练出的模型可以相互看做是一种 inductive transfer (**先验知识**)，通过提供 inductive bias (**某种对模型的先验假设**) 来提升模型效果\*\*。

点击率预估模型善于学习表面特征，其中包括标题党，美图带来的点击诱惑；而转化模型学习更深层的特征，比如体验、服务质量。MTL 可以很好地将这些特征充分学习到。

### ◦ 注意力机制

如果一个任务非常嘈杂或数据量有限并且高维，模型可能难以区分相关与不相关的特征。MTL 可以帮助模型将注意力集中在重要的特征上，因为其它任务将为这些特征的相关性或不相关性提供额外的证据。

### ◦ 窃听 (eavesdropping) 或者提示 (hint)

某特征 G 很容易被任务 B 学习，但是难以被另一个任务 A 学习。这可能是因为 A 以更复杂的方式与特征进行交互，或者因为其它特征阻碍了模型学习 G 的能力。通过 MTL，我们可以允许模型“窃听”，即通过任务 B 学习 G。最简单的方法是通过“提示”，即直接训练模型来预测最重要的特征。

## 3. 多任务间相互约束，提高泛化性，体现在以下两个方面：

### ◦ 隐式数据增加带来的正则化约束

由于不同的任务具有不同的噪声模式，所以同时学习多个任务的模型能够学习更一般的表征（类似 bagging 的思想）。只学习任务 A 有可能过拟合任务 A，而联合地学习 A 和 B 使模型能够通过平均噪声模式获得更好的表征。这也是一种特殊的正则化方式。

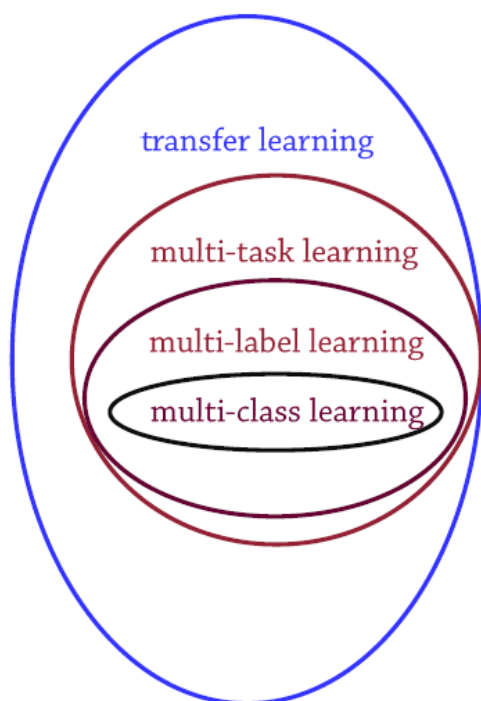
### ◦ 表征偏置



MTL 任务偏好其它任务也偏好的表征，这会造成模型偏差。但这将有助于模型在将来泛化到新任务，因为可以通过学习足够大的假设空间，在未来某些新任务中得到更好的泛化表现（解决冷启动），前提是这些任务都是同源的。

### 3.4. 多任务学习与其他学习方法的对比

多任务学习是迁移学习的一种，再往下，多标签学习属于多任务学习，而多类别学习又属于多标签学习，通过下图可以很好的区分这些概念：



#### ○ Transfer Learning

- Define source & target domains
- Learn on the source domain
- Generalize on the target domain



#### ○ Multi-task Learning

- Model the task relatedness
- Learn all tasks simultaneously
- Tasks may have different data/features

#### ○ Multi-label Learning

- Model the label relatedness
- Learn all labels simultaneously
- Labels share the same data/features

#### ○ Multi-class Learning

- Learn the classes independently
- All classes are exclusive

(<https://lumingdong.cn/wp-content/uploads/2019/11/20191115182010.png>)

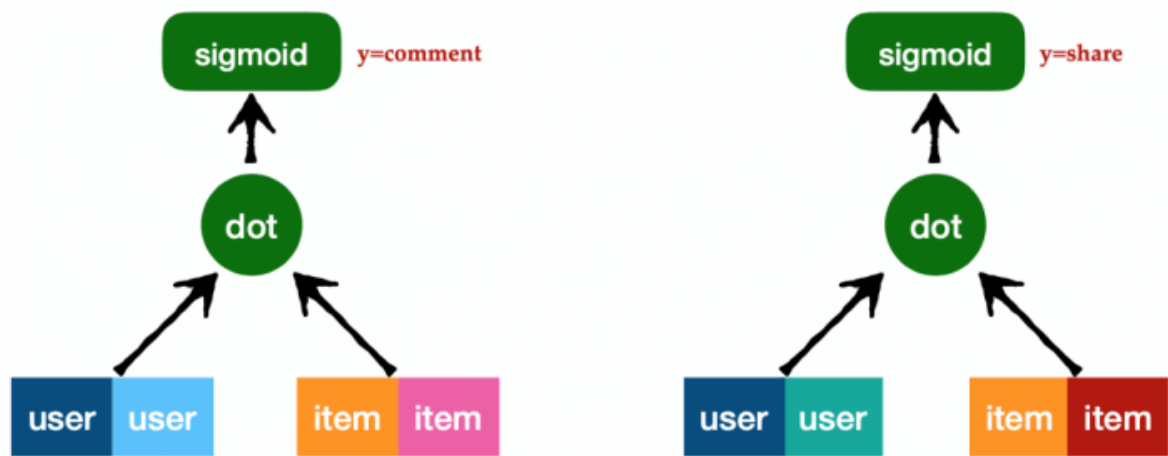
- **迁移学习 (transfer learning)** 定义一个源域一个目标域，从源域学习，然后把学习的知识信息迁移到目标域中，从而提升目标域的泛化效果。迁移学习一个非常经典的案例就是图像处理中的风格迁移。
- **多任务学习 (Multitask learning)** 是迁移学习的一种，对多个任务之间的相关性建模，同时学习多个任务，这些任务**可能拥有不同的数据或特征**。比如人脸识别中，同时识别人脸的性别和预估人脸主人的年龄。
- **多标签学习 (Multilabel learning)** 是多任务学习中的一种，对多个标签之间的相关性建模，同时学习多个标签，多个标签之间**共享相同的数据或特征**。多标签学习就是多标签分类学习，指的是一条数据**可能有一个或者多个标签**（类别标签数量不统一），比如一个病人的体检报告，它可能被标记上，高血压，高血糖等多个标签。



- **多类别学习 (Multiclass learning)** 是多标签学习任务中的一种，也称多分类学习，对多个相互独立的类别 (classes) 进行建模。多类别学习指的是一条数据**只有一个标签，但是标签有多种类别**，经典的鸢尾花分类就是标准的多类别学习任务。<sup>4</sup>

### 3.5. 多任务学习在推荐系统中的简单应用示例

这里以简单的矩阵分解 (MF) 为例，同时学习预估评论和分享两个任务的模型，两个任务有可共享的特征，也有各自独有的特征，如下图，通过颜色来区分共享的特征和独有的特征：



(<https://lumingdong.cn/wp-content/uploads/2019/11/20191119160745.png>)


前面我们已经提到过通过多个任务之间的相互辅助和相互约束，可以得到泛化性能较好的效果，我们以上图为例，来看一下资源消耗情况。

假设有 100w 的用户、100w 的商品，每个用户每个商品都学习到 128 维的 Embedding 向量，其中有一半是共享的，一半是各自独有的，线上有 10 个目标，也就对应 10 个模型，如果我们来看一下单任务和多任务的在资源消耗上区别：

	存储量	计算量
10 single tasks	$128 \times 10 \times 200w$	$128 \times 10 \times K$
1 multi-task	$(64 + 64 \times 10) \times 200w$	$(64 + 64 \times 10) \times K$

其中，200w 为 100w 用户和 100w 商品，K 是对 K 个候选进行预估推荐，共享部分的 64 维不用增加存储和计算，因此能够节省存储和计算资源，非常适合工业界应用。

## 4. ESMM

 Entire Space Multi-Task Model: An Effective Approach for Estimating Post-Click Conversion Rate (<https://lumingdong.cn/go/2fkp0b>)

**点击率 (Click-Through Rate, CTR)** 和 **转化率 (Conversion Rate, CVR)** 在信息检索、推荐系统、在线广告等应用场景下都是非常重要的两个指标，因为它们直接关系到产品的盈利。关于 CVR，其实大部分情况下指的是点击后转化率，即 **pCVR (post-click Conversion Rate)**，这就会与 CTR 形成一个顺序和传递关系，因此在很多场景下，这个指标则更为重要，如：

- 在电商推荐中，最大化 GVM（商品交易总额）是平台的重要目标之一，而 GMV 可以拆解为  $\text{流量} \times \text{点击率} \times \text{转化率} \times \text{客单价}$ ，可见点击率和转化率是优化目标非常重要的两个因子，而这两个指标的共同优化，其实就是一个多目标排序问题；
- 在以 oCPC (<https://lumingdong.cn/go/l2z7lf>) (Optimized Cost Per Click，以目标转化为优化方式的点击出价) 为主的智能广告投放平台中，用 pCVR 作为目标来调整每次点击出价，从而使 CPM (Cost Per Mille, 千人成本) 最大化。



## 4.1. 传统CVR预估存在的问题

传统 pCVR 预估采用的是类似 CTR 预估的技术，即通过点击的样本子集进行训练，推理的时候对整个展现样本空间进行推断。但这种方法存在很多问题：


- **样本选择偏差 (Sample Selection Bias, SSB)**

后一阶段的模型基于上一阶段的采样后的样本子集进行训练，但是最终是在全样本空间进行推理，这带来了严重的模型的泛化性问题

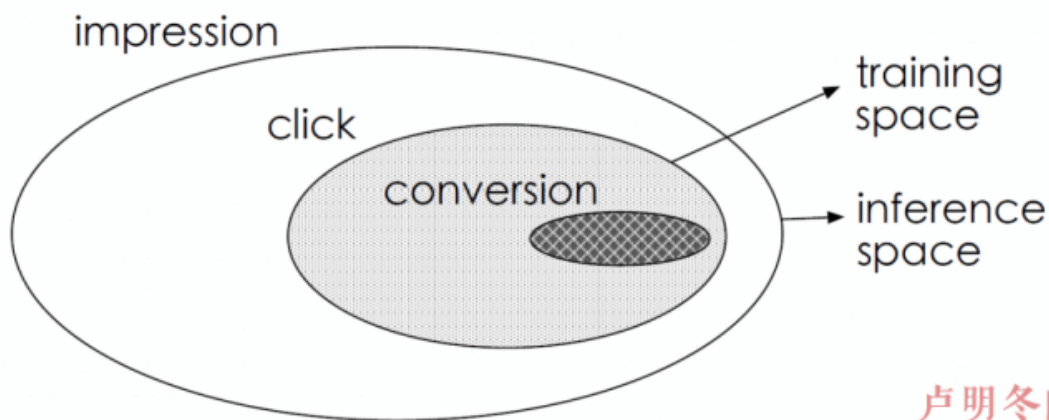
- **样本数据稀疏 (Data Sparsity, DS)**

通常后一阶段模型的训练样本规模通常远低于前一阶段任务，而且会出现正负样本极度不平衡。加大了模型训练的难度，同样带来了泛化性问题。

- **延迟反馈 (Delayed Feedback)**

点击后的转化 (conversion) 很可能延时发生，比如看过一个商品，但并没有马上去买，过了几天才去购买，给建模带来困难。感兴趣的同学可以阅读这篇 Paper:  **Modeling Delayed Feedback in Display Advertising** (<https://lumingdong.cn/go/iom1w3>)





(<https://lumingdong.cn/wp-content/uploads/2019/11/20191119231102.png>)



## 4.2. 学术界当前的解决方案

对于前两个问题，学术界也有很多解决方法，如：

**缓解 DS 问题：**

- **分层 CVR 模型 (Hierarchical Estimator)**

针对不同的特征构建不同的分层分类器，并分别估算分布参数，然后使用逻辑回归来组合这些单独的分类器。这种方法可以粗力度预估以解决 DS 问题，但依赖先验知识来构建分层结构，难以在具有数千万 user 和 item 的推荐系统中应用。

- **过采样方法 (Oversampling Method)**

通过复制稀少类的样本缓解不平衡，这有助于缓解数据的稀疏性，但是对采样率很敏感。

**缓解 SSB 问题：**

- **AMAN (All Missing As Negative, 缺失值作为负样本)**

应用随机抽样策略来选择未点击的展现作为负样本。它可以在一定程度上通过引入缺失观察的样本来消除 SSB 问题，但通常会导致预测值偏低。

- **无偏差采样 (Unbias Sampling)**

使用蒙特卡洛拒绝采样法 (<https://lumingdong.cn/go/qi9ft2>) (Rejection Sampling) 来拟合样本实际分布，制造无偏差采样，但会遇到数值不稳定问题。



上面的几种方法多多少少都存在一些问题，因此 SSB 和 DS 问题都没有被很好地解决，而且没有一个方法应用序列行为的信息。

2018 年阿里妈妈算法团队发表了一篇论文：《Entire Space Multi-Task Model: An Effective Approach for Estimating Post-Click Conversion Rate》，提出了 **ESMM (Entire Space Multi-Task Model, 完整空间多任务模型)**，ESMM 创新地利用用户行为序列数据，在完整的样本数据空间利用 MTL 同时学习点击率和转化率 (post-view clickthrough&conversion rate, CTCVR)，解决了传统 CVR 预估模型难以克服的样本选择偏差 (SSB) 和训练数据过于稀疏 (DS) 的问题。

### 4.3. ESMM 算法原理



之前我们提到过，从点击率到转化率，其实有个顺序和传递过程，更完整的业务流应该是：曝光 (impression) → 点击 (click) → 转化 (conversion)。用户的行为遵循一定的顺序决策模式，以电商和广告为例，用户看到曝光的推荐，对感兴趣的内容会点击进入详情页进一步了解，之后才考虑是否购买发生转化。因此，对于这种顺序递进关系，整个任务可进行数字化描述：

假设训练数据集为  $\mathcal{S} = \{(x_i, y_i \rightarrow z_i)\}_{i=1}^N$ ，其中的样本  $(x, y \rightarrow z)$  是从域  $\mathcal{X} \times \mathcal{Y} \times \mathcal{Z}$  中按照某种分布采样得到的， $\mathcal{X}$  是特征空间， $\mathcal{Y}$  和  $\mathcal{Z}$  是标签空间， $N$  为数据集中的样本总数量。在 CVR 预估任务中， $x$  是高维稀疏多域的特征向量， $y$  和  $z$  的取值为 0 或 1，分别表示是否点击和是否购买。 $y \rightarrow z$  揭示了用户行为的顺序性，即点击事件一般发生在购买事件之前。CVR 模型的目标是预估条件概率 pCVR，与其相关的两个概率为点击率 pCTR 和点击且转化率 pCTCVR，它们之间的关系如下：

$$\underbrace{p(y = 1, z = 1|x)}_{pCTCVR} = \underbrace{p(y = 1|x)}_{pCTR} \times \underbrace{p(z = 1|y = 1, x)}_{pCVR}.$$

(<https://lumingdong.cn/wp-content/uploads/2019/11/20191120115158.png>)

CVR 模型是预估条件概率 pCVR，上式可以转化为：

$$p(z = 1|y = 1, x) = \frac{p(y = 1, z = 1|x)}{p(y = 1|x)}$$

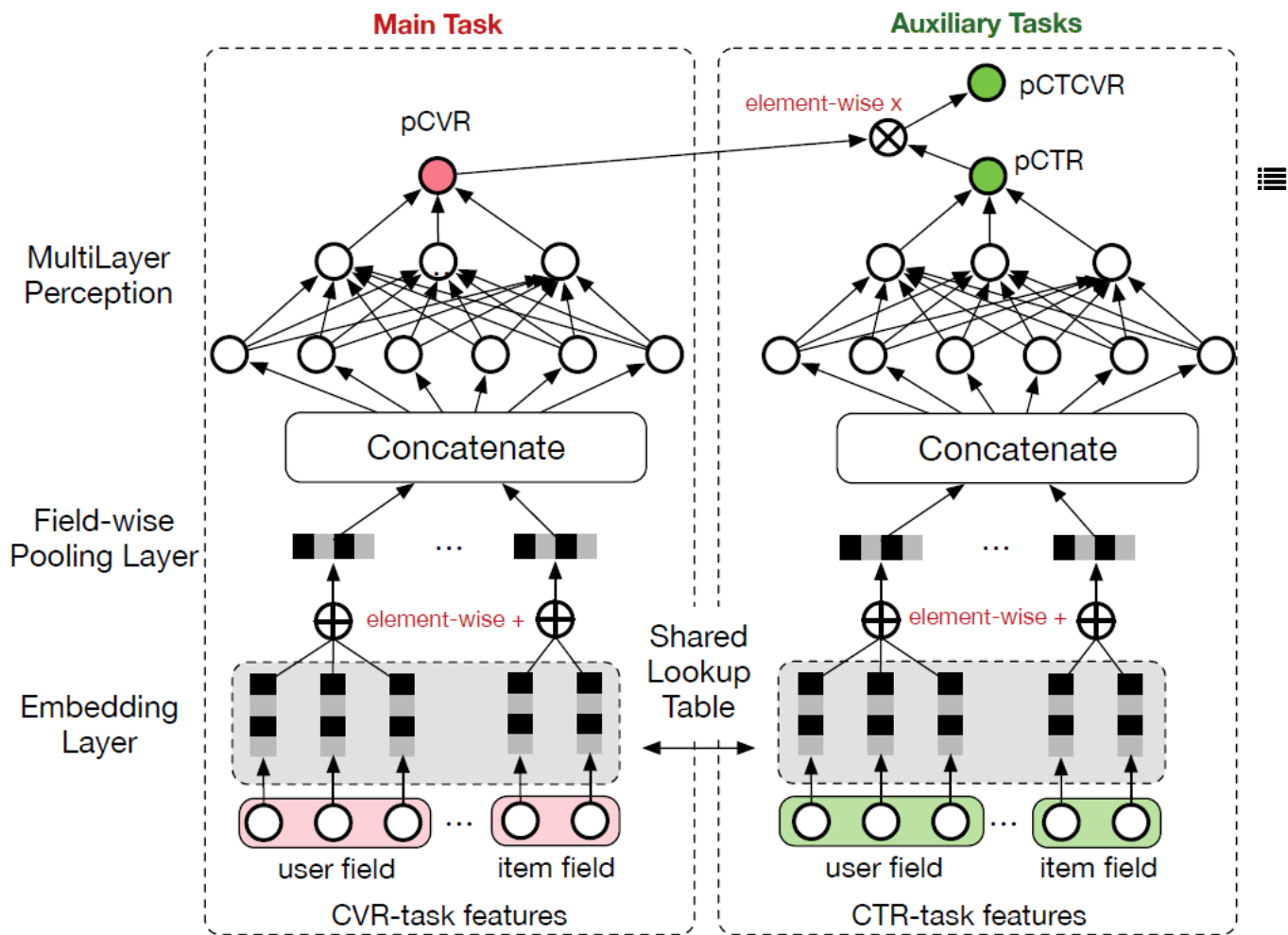
可以看到 pCVR 可以在先估计出 pCTR 和 pCTCVR 之后就可推导出来。从原理上来说，相当于分别单独训练两个模型拟合出 pCTR 和 pCTCVR，再通过 pCTCVR 除以 pCTR 得到最终的拟合目标 pCVR。实际操作中，由于 pCTR 通常很小，pCTCVR 除这个很小的数，容易溢出。故





ESMM 采用了乘法的形式，避免了除法，且能够使 pCVR 的值在  $[0,1]$  区间。由此可见，这三个预估模型都可以在整个样本空间上建模得到。

根据上面的分析，可知 pCTR 和 pCTCVR 是模型需要估计的**两个主要因子**，ESMM 使用了 MTL 方法，引入两个辅助任务，分别拟合 pCTR 和 pCTCVR，把 pCVR 当做一个中间变量，模型结构如下：



(<https://lumingdong.cn/wp-content/uploads/2019/11/20191120115548.png>)

整体来看，对于一个给定的展现，ESMM 能够同时输出预估的 pCTR、pCVR 和 pCTCVR。它主要由两个子神经网络组成，左边的子网络用来拟合 pCVR，右边的子网络用来拟合 pCTR。两个子网络的结构是完全相同的，并且共享 Embedding 参数。这里把子网络命名为 BASE 模型（这个 Base 模型结构，也是近年来基于 DL 的 CVR 预估模型比较经典的网络结构，因此论文中也简单的使用了这种结构，但真正用于生产环境下的结构可能要比这个复杂的多），两个子网络的输出结果相乘之后即得到 pCTCVR，并作为整个任务的输出。

ESMM 的损失函数由两部分组成，对应于 pCTR 和 pCTCVR 两个子任务，其形式如下：





$$L(\theta_{cvr}, \theta_{ctr}) = \sum_{i=1}^N l(y_i, f(x_i; \theta_{ctr})) + \sum_{i=1}^N l(y_i \& z_i, f(x_i; \theta_{ctr}) \times f(x_i; \theta_{cvr}))$$

其中， $\theta_{ctr}$  和  $\theta_{cvr}$  分别是 CTR 网络和 CVR 网络的参数， $l(\cdot)$  是交叉熵损失函数。在 CTR 任务中，有点击行为的展现事件构成的样本标记为正样本，没有点击行为发生的展现事件标记为负样本；在 CTCVR 任务中，同时有点击和购买行为的展现事件标记为正样本，否则标记为负样本。

ESMM 可以有效解决传统 CVR 预估模型的一些问题，具体表现如下：

### 解决样本选择（BBS）问题

- 全空间建模：和 CTR 一样，在全部展现样本上建模。

pCTCVR 和 pCTR，pCVR 都定义在全样本空间。通过分别估算单独训练的模型 pCTR 和 pCTCVR 并通过关系式可以获得 pCVR，三个关联的任务共同训练分类器，能够利用数据的序列模式并相互传递信息，保障物理意义

### 解决样本稀疏（DS）问题

- 迁移学习：在 ESMM 中，CVR 网络的 Embedding 参数与 CTR 任务共享，遵循特征表示迁移学习范式。Embedding Layer 将大规模稀疏输入映射到低维稠密向量中，主导深度网络参数。CTR 任务所有展现样本规模比 CVR 任务要丰富多个量级，该参数共享机制使 ESMM 中的 CVR 网络可以在未点击展现样本中进行学习。

## 4.4. ESMM的性能提升

在阿里公开的数据集 (<https://lumingdong.cn/go/cya1qk>) 上，各个方法对比如下表所示：

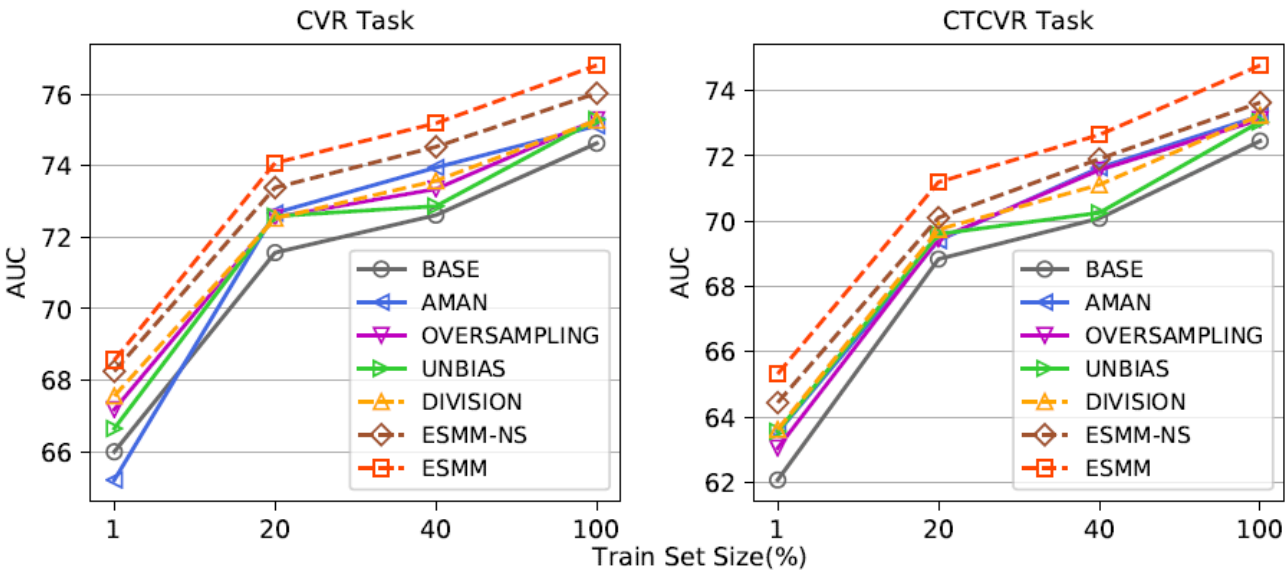
Model	AUC(mean ± std) on CVR task	AUC(mean ± std) on CTCVR task
BASE	66.00 ± 0.37	62.07 ± 0.45
AMAN	65.21 ± 0.59	63.53 ± 0.57
OVERSAMPLING	67.18 ± 0.32	63.05 ± 0.48
UNBIAS	66.65 ± 0.28	63.56 ± 0.70
DIVISION	67.56 ± 0.48	63.62 ± 0.09
ESMM-NS	68.25 ± 0.44	64.44 ± 0.62
ESMM	68.56 ± 0.37	65.32 ± 0.49

(<https://lumingdong.cn/wp-content/uploads/2019/11/20191120162533.png>)

其中，几个模型对应的方法如下：

- BASE：经典的 CVR 预估模型结构，也就是 ESMM 左半边部分；
- AMAN、OVERSAMPLING、UNBIAS：分别对应上面提到的几种学术界缓解 SSB 和 DB 问题的方法；
- DIVISION：先分别训练出拟合 CTR 和 CTCVR 的模型，再拿 CTCVR 模型的预测结果除以 CTR 模型的预测结果得到对 CVR 模型的预估；
- ESMM-NS：是 ESMM 模型的一个变种，其在 ESMM 模型的基础上去掉了特征表示共享的机制。

在**淘宝生产环境数据集**上几种不同算法的性能测试对比：



(<https://lumingdong.cn/wp-content/uploads/2019/11/20191120171327.png>)

ESMM 在线 AB 测试效果的提升：

Metric	CTR	CVR	GMV
%Improved	0.7%	6.9%	4.9%


(<https://lumingdong.cn/wp-content/uploads/2019/11/20191120171843.png>)

## 4.5. ESMM代码实现

参考 GitHub：ESMM Code (<https://lumingdong.cn/go/2wj7dk>)



## 5. MMoE

 Modeling Task Relationships in Multi-task Learning with Multi-gate Mixture-of-Experts (<https://lumingdong.cn/go/4kiqt2>)

阿里团队提出 ESMM 模型利用 MTL 的方法极大地提升了 CVR 预估的性能，同时解决了传统 CVR 模型预估的一些弊病。我们从模型的网络结构可以了解到，ESMM 是典型的 share-bottom 结构，即底层特征共享方式。这种 MTL 共享结构的一大特点是在任务之间都比较相似或者相关性比较大的场景下能带来很好的效果，归纳偏置的作用也能够很好的发挥出来，而对于任务间差异比较大的场景，这种 MTL 共享结构就有点捉襟见肘了。

### 5.1. MMoE要解决的问题





说到底，多任务学习的本质就是共享表示以及相关任务的相互影响。通常，相似的子任务也拥有比较接近的底层特征，那么多任务学习中，他们就可以很好地进行底层特征共享；而对于不相似的子任务，他们的底层表示差异很大，在进行参数共享时很有可能会互相冲突或噪声太多，导致多任务学习的模型效果不佳。

实际的应用场景中，我们可能不止有像 CTR、CVR 这样的非常相关的子任务，还会遇到子任务间关系没那么紧密的多任务学习场景，而且很多情况下，你很难判断任务在数据层面是否是相似的。所以多任务学习如何在相关性不高的任务上获得好效果是一件很有挑战性也很有实际意义的事，这也是本小节所提到的模型 “MMoE” 主要解决的问题。

其实在 MMoE 之前，已经有一些其他结构来解决这个问题了，比如两个任务的参数不共用，而是对不同任务的参数增加 L2 范数的限制；或者对每个任务分别学习一套隐层然后学习所有隐层的组合。这些结构和 Shared-Bottom 结构相比，其构成的模型会针对每个任务添加更多参数以适应任务间差异，虽然能够带来一定的效果提升，但是增加了更多的参数也就意味着需要更大的数据样本来训练模型，而且这些方法会使模型变得更复杂，也不利于在真实生产环境中部署使用。<sup>5</sup>

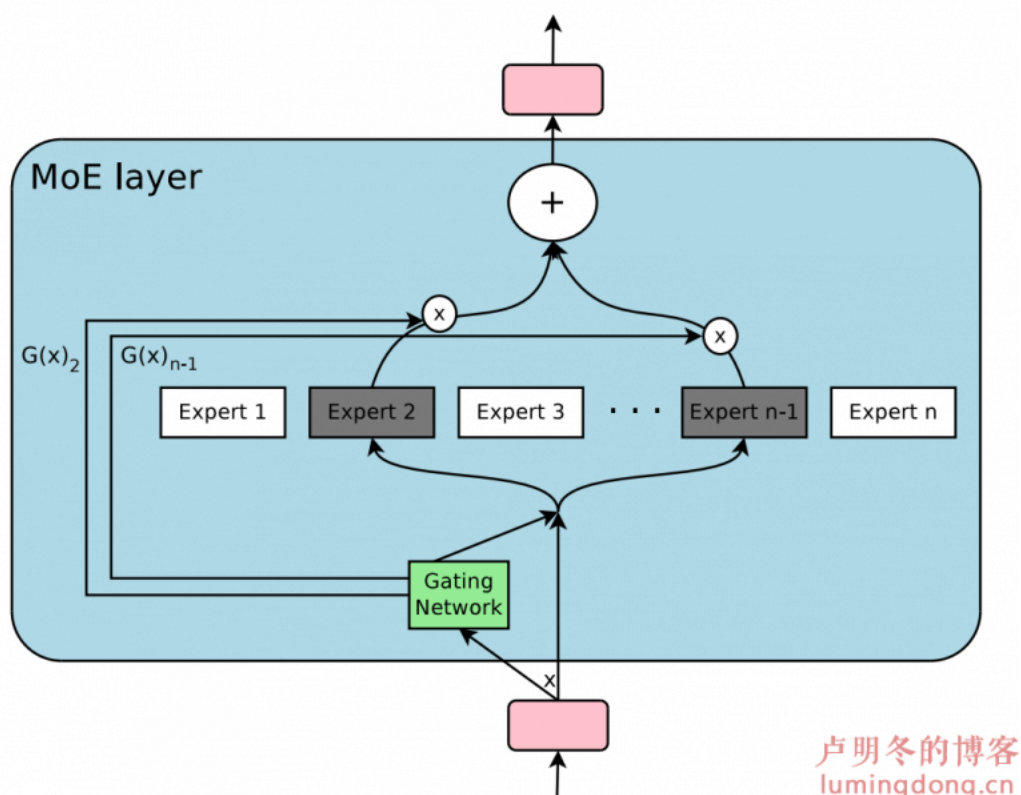
**MMoE (Multi-gate Mixture-of-Experts)** 是 Google 在 2018 年 KDD 上发表的论文《Modeling Task Relationships in Multi-task Learning with Multi-gate Mixture-of-Experts》里提出的，它是一种新颖的多任务学习结构。MMoE 模型刻画了任务相关性，基于共享表示来学习特定任务的函数，避免了明显增加参数的缺点。

### 5.2. MoE神经网络结构

MMoE 很重要的一步是把 MoE 引入了多任务学习中。早在 2017 年，谷歌大脑团队的两位科学家：大名鼎鼎的深度学习之父 *Geoffrey Hinton* 和 谷歌首席架构师 *Jeff Dean* 发表论文  **Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer** 

(<https://lumingdong.cn/go/tyrcm0>)》并提出了“稀疏门控制的混合专家层”（Sparsely-Gated Mixture-of-Experts layer, MoE），这里的 MoE 是一种特殊的神经网络结构层，结合了专家系统和集成思想在里面。

MoE 由许多“专家”组成，每个“专家”都有一个简单的前馈神经网络和一个可训练的网关网络（gating network），该网关网络选择“专家”的一个稀疏组合来处理每个输入，它可以实现自动分配参数以捕获多个任务可共享的信息或是特定于某个任务的信息，而无需为每个任务添加很多新参数，而且网络的所有部分都可以通过反向传播一起训练。MoE 结构图如下所示：



(<https://lumingdong.cn/wp-content/uploads/2019/11/20191123223402.png>)

MoE 可以作为一个基本的组成单元，也可以是多个 MoE 结构堆叠在一个大网络中。比如一个 MoE 层可以接受上一层 MoE 层的输出作为输入，其输出作为下一层的输入使用。在谷歌大脑的论文中，MoE 就是作为循环神经网络中的一个循环单元。

MoE 神经网络结构有两个非常明显的好处：

### 1. 实现一种多专家集成的效果

MoE 的思想是训练多个神经网络（也就是多个专家），每个神经网络（专家）通过网关网络（Gating NetWork）被指定应用于数据集的不同部分，最后再通过网关网络将多个专家的结果进行组合。单个模型往往善于处理一部分数据，不擅长处理另外一部分数据 ^

（在这部分数据上犯错多），而多专家系统则很好的解决了这个问题：系统中的每一个神经网络，也就是每一个专家都会有一个擅长的数据区域，在这组区域上该专家就是“权威”，要比其他专家表现得好。因此多专家系统是单一全局模型或者多个局部模型的一个很好的折中，这样的网络结构能够处理更加复杂的数据分布，在相应的任务中，性能也会有很大的提升。

2. 只需增加很小的计算力，便能高效地提升模型的性能。

神经网络吸收信息的能力受其参数数量的限制。有人在理论上提出了条件计算（conditional computation）的概念，作为大幅提升模型容量而不会大幅增加计算力需求的一种方法。MoE 就是条件计算的一种实现，并在论文中证实，这种网络结构可实现

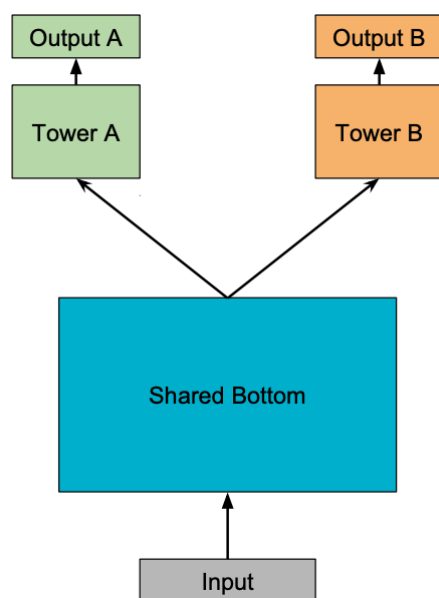


### 5.3. MoE与MTL的结合

有了以上信息，我们就很容易理解为什么要将 MoE 引入多任务学习中了，因为多专家集成的机制刚好可以用来解决多任务间的差异问题，并且还不会带来很大计算损耗。

接下来我们来看 MoE 如何与 MTL 结合。

MTL 最经典的 Shared-Bottom DNN 网络结构，如下图所示：

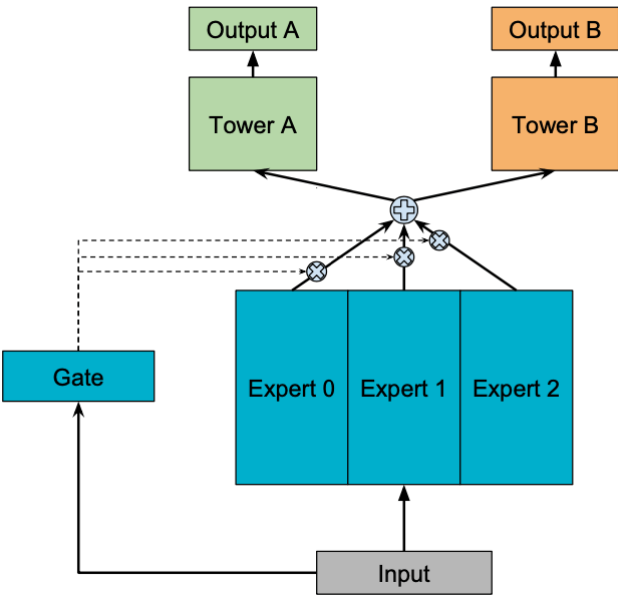


(<https://lumingdong.cn/wp-content/uploads/2019/12/20191201125500.png>)

Shared-Bottom 网络通常位于底部，表示为函数  $f$ ，多个任务共用这一层。往上， $K$  个子任务分别对应一个 tower network，表示为  $h^k$ ，每个子任务的输出  $y_k = h^k(f(x))$ 。



然后用一组由专家网络（expert network）组成的神经网络结构来替换掉 Shared-Bottom 部分（函数  $f$ ），这里的每个“专家”都是一个前馈神经网络，再加上一个门控网络，就构成了 MoE 结构的 MTL 模型。因为只有一个门网络，所以在论文中，为了与 MMoE 对应，也称这种结构为 OMoE（One-gate Mixture-of-Experts），其结构如下图所示：



(<https://lumingdong.cn/wp-content/uploads/2019/12/20191201125713.png>)

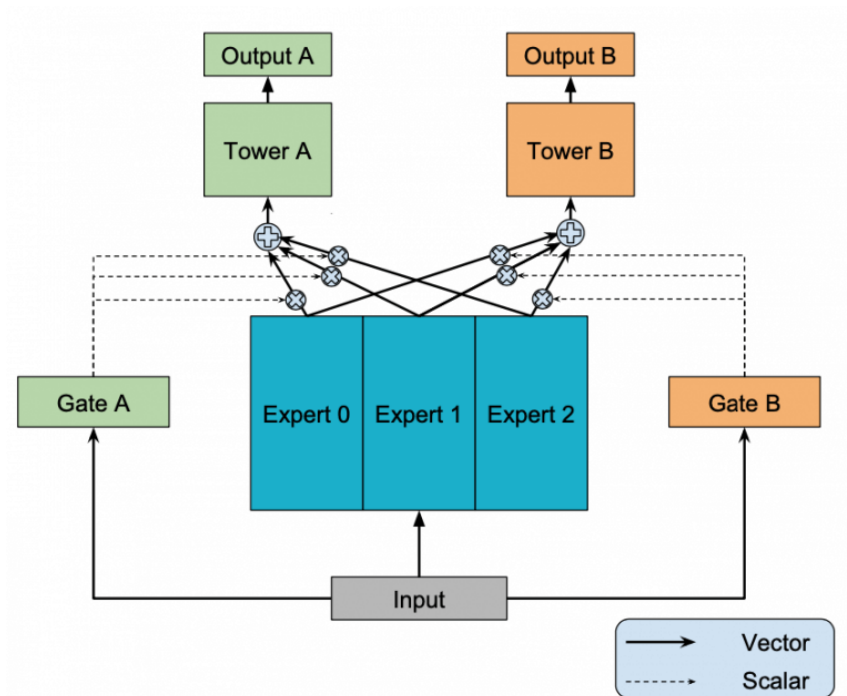
MoE 模型可以形式化表示为：

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

其中， $f_i(i = 1, \dots, n)$  是  $n$  个专家（expert）网络， $g$  是组合 experts 结果的门控网络（gating network）， $\sum_{i=1}^n g(x)_i = 1$ ，具体来说  $g$  产生  $n$  个 experts 上的概率分布，最终的输出是**所有 experts 的带权加和**。显然，MoE 可看做基于多个独立模型的集成方法。

知道了 OMoE 结构，那么 MMoE（Multi-gate Mixture-of-Experts）的结构就很容易猜出来了，通过名字里的 Multi-gate 很容易想到，MMOE 就是在 OMoE 的基础上，用了多个门控网络，结构如下图所示：





(<https://lumingdong.cn/wp-content/uploads/2019/12/20191201125739.png>)

MMoE 可以形式化表达为：

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

其中,  $g^k(x) = \text{softmax}(W_{gk}x)$ , 它的输入是 input feature, 输出就是所有 Experts 上的权重。

可见 MMoE 其实是 MoE 针对 MTL 的变种和优化, 相对于 OMoe 的结构中所有任务共享一个门控网络, MMoE 的结构优化为每个任务都单独使用一个门控网络。这样的改进可以针对不同任务得到不同的 Experts 权重, 从而实现对 Experts 的选择性利用, 不同任务对应的门控网络可以学习到不同的 Experts 组合模式, 因此模型更容易捕捉到子任务间的相关性和差异性。

## 5.4. MMoE的性能提升

论文分别在以下三个数据集上进行了实验, 来从不同的角度验证 MMoE 的性能。

### 1. 人工合成数据集 (Synthetic Dataset)

因为在真实数据集中我们无法准确度量和控制任务之间的相关性, 不太方便研究任务相关性对多任务模型的影响, 因此在论文中, 人工构建了两个回归任务的数据集, 并通过正弦函数来引入非线性机制。如此可以利用两个任务标签的皮尔逊相关系数作为任务相



关性的度量，来观察在不同相关性的任务下，Shared-Bottom、OMoE、MMoE 三种结构在训练过程中对 loss 的影响，最终结果如下图：

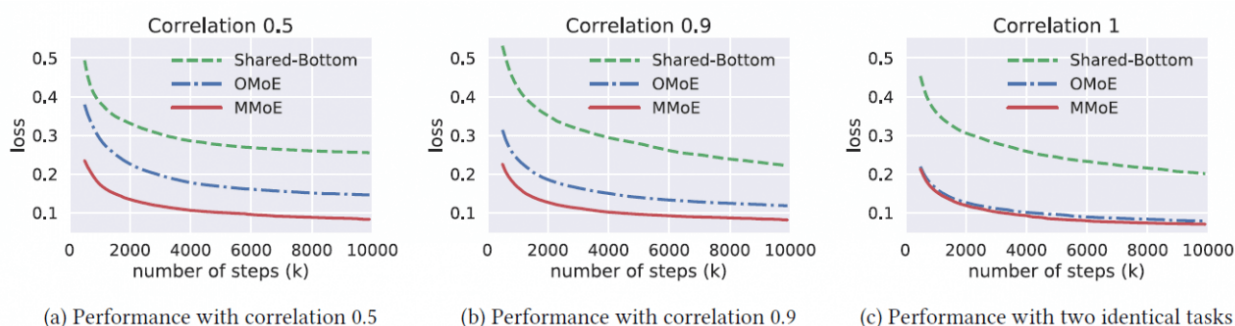


Figure 4: Average performance of MMoE, OMoE, and Shared-Bottom on synthetic data with different correlations.

(<https://lumingdong.cn/wp-content/uploads/2019/11/20191127153212.png>)

可以看出：

- OMoE 和 MMoE 的效果在不同相关度任务的数据中都好于 Shared-Bottom；
- 如果任务相关度非常高，则 OMoE 和 MMoE 的效果近似；
- 但是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。**

此外，在这组实验中，作者还发现 **MMoE 更容易训练，并且在多次训练运行时收敛到更好的损失**。这一发现也与最近的研究结果一致，门控机制在训练非凸深层神经网络时可以提高模型的可训练性。（模型的可训练性，就是指模型在超参数设置和模型初始化范围内的鲁棒性。）

论文针对数据和模型初始化中的随机性研究模型的鲁棒性，并在每种设置下重复进行多次实验，每次从相同的分布生成数据，但随机种子不同，并且模型也分别初始化，绘制了重复运行的最终损失值的直方图：



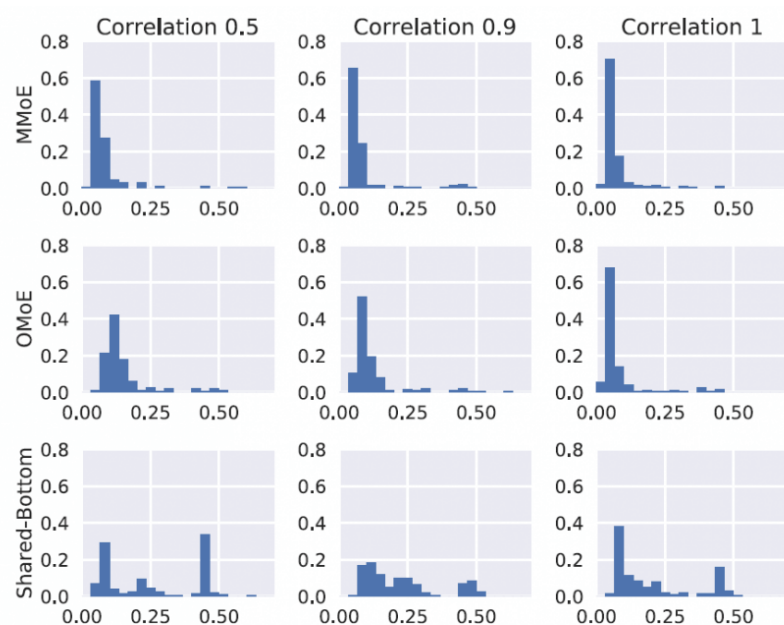


Figure 5: Histogram of performance of MMoE, OMoE, and Shared-Bottom multi-task model on synthetic data with different correlations.

(<https://lumingdong.cn/wp-content/uploads/2019/11/20191127153415.png>)

从直方图中可以发现三个有趣的结果：

- 首先，在所有任务相关性设置中，Shared-Bottom 模型的性能差异（方差）远大于基于 MoE 的模型的性能差异（方差）。这意味着，与基于 MoE 的模型相比，Shared-Bottom 模型通常具有较差的质量局部最小值。
- 其次，虽然任务相关性为 1 时 OMoE 模型的性能方差与 MMoE 模型相似，但当任务相关性降低到 0.5 时，OMoE 的鲁棒性明显下降。MMoE 和 OMoE 之间的唯一区别是是否存在多门结构。这验证了多门结构在解决由任务差异造成的不良局部最小值方面的有用性。
- 最后，值得一提的是，这三个模型中的最低损失是可比的。这并不奇怪，因为神经网络在理论上是通用近似器。具有足够的模型容量，应该存在一个“正确”的共享底部模型，该模型可以很好地学习这两个任务。但这只是 200 个独立实验的分布，对于更大，更复杂的模型（例如，当共享底层网络是递归神经网络时），获得任务关系“正确”模型的机会会更低，如果这样，仍然需要对任务关系进行显式建模。

## 2. UCI 人口普查收入数据集（UCI Census-income Dataset）

论文进一步评估了 MMoE 在基准数据集 UCI 人口普查收入数据集上的表现，并与几种最先进的多任务模型进行了比较，这些模型通过软参数共享对任务关系进行建模，两组结果数据如下：

**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	<b>0.9933</b>	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	<b>0.9361</b>	0.9917	0.9922
Tensor-Factorization	0.7460	0.6765	0.8175	0.8412
OMoE	0.9387	0.9319	0.9928	0.9923
MMoE	<b>0.9410</b>	0.9359	0.9926	<b>0.9927</b>

(<https://lumingdong.cn/wp-content/uploads/2019/11/20191127171007.png>)

**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	<b>0.9933</b>	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	<b>0.8860</b>	<b>0.8826</b>	0.9932	<b>0.9924</b>

(<https://lumingdong.cn/wp-content/uploads/2019/11/20191127171020.png>)

### 3. 大规模内容推荐数据集

最后，论文在真实的大型内容推荐系统上测试 MMoE，在向用户推荐项目时可同时学习两个分类任务。实验使用 1e 数十亿个训练样本来训练 MMoE 模型，并将其与基于 Shared-Bottom 的生产模型进行比较，见下图。从结果来看，AUC 等离线性能指标有着显著的提高。

Table 3: Engagement performance on the real large-scale recommendation system.

Metric	AUC@2M	AUC@4M	AUC@6M	R2@2M	R2@4M	R2@6M
Shared-Bottom	0.6879	0.6888	0.6900	0.08812	0.09159	0.09287
MMoE	0.6966	0.6981	0.6995	0.09668	0.09920	0.09912



(<https://lumingdong.cn/wp-content/uploads/2019/11/20191127172254.png>)

同时，实时实验中也发现 MMoE 的在线指标表现也不错。

Table 4: Live experiment results

Live experiment	Engagement Metric	Satisfaction Metric
Shared-Bottom Improvement over Single-Task	-0.22% *	19.72% **
MMoE Improvement over Shared-Bottom	0.25% **	2.65% **

\* indicates confidence interval level 90%

\*\* indicates confidence interval level 95%

(<https://lumingdong.cn/wp-content/uploads/2019/11/20191127172333.png>)

## 6. 工业界案例

多目标排序和多任务学习的应用，这两年在工业界有很多尝试，下面是一些其他案例。

- 美团 “猜你喜欢” 深度学习排序模型实践 (<https://lumingdong.cn/go/gx0i5t>)
- 多任务学习在美图推荐排序的近期实践 (<https://lumingdong.cn/go/8wp9qs>)
- 多目标学习系统：如何让知乎互动率提升 100%? (<https://lumingdong.cn/go/v3hee1>)



- MTL 在工业界如何更胜一筹 (<https://lumingdong.cn/go/5fxkx1>)
- YouTube 多目标排序系统：如何推荐接下来观看的视频 (<https://lumingdong.cn/go/6usa1u>)
- 信息流短视频时长多目标优化 (<https://lumingdong.cn/go/bkard1>)

## 7. 参考资料

[ 1 ] 罗老师. multi-objective ranking. 七月在线↵

[ 2 ] Dipanjan (DJ) Sarkar. A Comprehensive Hands-on Guide to Transfer Learning with Real-World Applications in Deep Learning. Towards Datascience (<https://lumingdong.cn/go/0vzfvt>)↵

[ 3 ] Sebastian Ruder. An Overview of Multi-Task Learning in Deep Neural Networks. arXiv (<https://lumingdong.cn/go/gc8h6b>)↵

[ 4 ] [Slides]Zhou&Chen&Ye. Multi-Task Learning: Theory, Algorithms, and Applications. SDM (<https://lumingdong.cn/go/qve79m>)↵

[ 5 ] 杨铭铭. 详解谷歌之多任务学习模型 MMoE(KDD 2018). 知乎 (<https://lumingdong.cn/go/od8fpu>)↵

© 除特别注明外，本站所有文章均为卢明冬的博客 (<https://lumingdong.cn>)原创，转载请注明作者和文章链接。

© 本文链接：<https://lumingdong.cn/multi-task-learning-in-recommendation-system.html>  
(<https://lumingdong.cn/multi-task-learning-in-recommendation-system.html>)

👁 7536 VIEWS

🔖 工程实践 (<https://lumingdong.cn/Tag/%E5%B7%A5%E7%A8%8b%E5%Ae%9e%E8%B7%B5>) , 排序算法 (<https://lumingdong.cn/Tag/%E6%8e%92%E5%Ba%8f%E7%Ae%97%E6%B3%95>) , 论文解读 (<https://lumingdong.cn/Tag/%E8%Ae%Ba%E6%96%87%E8%A7%A3%E8%Af%Bb>)

< 上一篇

百年孤独 (<https://lumingdong.cn/one-hundred-years-of-solitude.html>)

下一篇 >

推荐系统技术演进趋势：从召回到排序再到重排 (<https://lumingdong.cn/technology-evolution-trend-of-recommendation-system.html>)