系 统 科 学 与 数 学
J. Sys. Sci. & Math. Scis.
x(x) (20xx, x), 1–18

基于多任务深度学习的自适应软参数共享方法*

汪红霞[1] 金 骁[2] 杜玉坤[3] 张 楠[4]

(南京审计大学统计与数据科学学院, 南京; 211815)

摘要 在软参数共享模型的基础上, 通过任务之间的相似度与参数之间的关系, 设置自适应正则项系数λ^*, 自适应参数衰减比例θ, 本文提出了基于多任务深度学习的自适应软参数共享方法. 在基于均值约束的L_2范数基础上, 通过自适应地去除损失函数中正则项中的项数, 去除任务间相似度不高的信息. 本文的方法将软参数多任务学习动态地转化为软参数多任务与单任务联合学习, 相对于软参数多任务学习方法, 该方法减少了负迁移现象带来的影响. 相对于单任务学习方法, 该方法可以极大地降低局部最小解的风险. 模拟研究和案例分析都验证了该方法的有效性, 该方法的预测精度优于传统的多任务学习和单任务学习.

关键词 多任务学习, 软参数共享, 自适应参数衰减比例, 自适应正则项系数.

MR(2000)主题分类号 68Q10, 65Y20

DOI 10.12341/jssms

Adaptive Soft Parameter Sharing Method Based on Multi-task Deep Learning

WANG Hongxia[1] JIN Xiao[2] DU Yukun[3] ZHANG Nan[4]

(*School of Statistics and Data Science, Nanjing Audit University, Nanjing* 211815)

Abstract On the basis of the soft parameter sharing model, we set the adaptive regular term coefficient λ^* and adaptive parameter decay ratio θ by the similarity between tasks and the relationships between parameters. In this paper, we propose an adaptive soft parameter sharing method based on multi-task deep learning. On the basis of L_2 norm based on the mean constraint, the effect of removing information with low similaritities between tasks can achieve by adaptively removing the number of terms in the regular term of the loss function. The approach in this paper

*国家社会科学基金资助项目(22BTJ021); 江苏省研究生科研创新计划(KYCX22_2154).
收稿日期: 200x-xx-xx, 收到修改稿日期: 200x-xx-xx.
通信作者: 汪红霞, Email:hxwang@nau.edu.cn
编委:

dynamically transforms soft parameter multi-task learning into joint soft parameter multi-task and single-task learning. Compared with soft parameter multi-task learning methods, this method reduces the impact of negative migration phenomena. Compared with single-task learning method, this method can greatly reduce the risk of local minimum solution. Both simulation studies and case analyses have confirmed the effectiveness of this approach, demonstrating that its achieves superior predictive accuracy compared to traditional multi-task learning and single-task learning methods.

Keywords multi-task learning, soft parameter sharing, adaptive parameter decay ratio, adaptive regular term coefficients.

1 引 言

多任务学习在环境科学, 经济学, 空气污染等许多领域中有重要的应用. 多任务学习旨在利用任务之间的有效信息来提高多个相关任务的学习性能[1]. 张钰等[2]指出虽然任务数据采集的来源和分布是相似的, 但是由于学习的目的不完全相同, 不能简单地将它们合并为一个任务. 此时可以将它们看作是由多个相关的任务组成, 选择多个任务联合学习, 从而获得一些潜在信息以提高各自任务的学习效果. 对于给定的模型, 利用任务之间的相关性, 对多个任务进行联合训练, 更好地概括原始任务, 通过共享信息, 从而提高模型在多个任务上的学习能力.

基于深度学习的多任务学习最早可追溯到1993年, Caruana[3]提出了神经网络中硬参数共享的结构, 同时基于实证研究提出了多任务学习起作用的几个可能的机制. 与多任务学习相结合的深度学习框架主要有下面两种方法: 1) 基于硬参数共享的多任务深度学习方法. 硬参数共享允许多个任务共享一些模型参数, 并享受降低存储成本的好处. 硬参数共享在每一层共享所有参数的信息, 例如Collobert和Weston[4]将词性标注, 词块分割, 命名实体识别及词语相似度任务统一到一个语言模型中, 利用其他任务中自动学习的特征来提升语义角色标注任务的性能. Long等[5]提出多线性关系网络共享AlexNet的前五个卷积层, 并将特定任务的全连接层用于不同的任务. Ruder等[6]提出了多任务学习的硬参数共享结构. 尽管参数的硬共享机制在许多场景中有用, 但是若任务间的联系不那么紧密, 则硬参数共享技术效果不佳. 2) 基于软参数共享的多任务深度学习方法. 软参数共享机制, 需要考虑任务的相关性, 相对于硬参数共享机制而言, 软参数共享机制对于联系不那么紧密的任务也可以起到很好的优化效果[7]. 软参数共享可以充分利用多任务学习中的隐世数据增加机制, 窃听机制, 注意力集中机制, 正则化机制等优势[7]. 软参数共享网络可以对模型参数之间的距离进行正则化, 以鼓励参数相似. 例如Duong等[8]使用L_2距离进行正则化, Misra等[9]提出一个十字绣网络结构来学习不同任务相对位置特征的线性组合, Mrini等[10]提出一种渐进软参数共享的多任务学习方法, 使用加权损失函数同时优化多个任务, 提高了多个任务的性能.

目前, 在联合训练多个任务时, 提高模型在某一项任8务上的性能会损害具有不

同需求任务的性能, 称为负迁移现象[11]. 近年来最小化负迁移现象是多任务学习的一个关键目标. 王先兰等[12]提出多任务联合学习模型来减少负迁移现象, 提高了模型的鲁棒性以及预测精度. 郭辉和郭静纯[13]提出一种基于混合共享机制的多任务深度学习方法, 对组内和组间任务分别应用硬参数共享和软参数共享, 提高了多任务学习的性能. 徐薇等[14]为解决任务差异的内在冲突会损害部分任务预测精度的问题, 提出了一种基于相关性学习层的多任务学习模型.

在日常收集到的数据中并不是所有任务的联系都那么紧密, 软参数共享更合适. 软参数共享中正则化约束的选择决定了预测模型的性能. 基于此, 本文在Li等[15]提出的均值约束共享的基础上, 使用L_2范数约束, 通过任务之间参数的差异性, 设置软参数正则项系数λ, 自适应正则项系数λ^*, 自适应参数衰减比例θ, 得到累积自适应参数衰减比例θ^*. 将软参数多任务学习自适应地动态转化为多个单任务和多任务联合学习. 在此过程中, 寻找预测性能最好的自适应软参数共享模型.

本文的创新点: 本文在软参数共享机制的基础上, 提出了一种可以自适应调整正则项项数的方法. 1) 该方法可以随着模型迭代次数的增加, 动态调整模型训练机制, 是一种从多任务训练逐步调整为多个单任务训练的一种方法. 2) 该方法可以极大减小多任务学习中负迁移现象带来的影响, 并且不会提高模型的复杂度, 可以进一步提高模型的泛化能力. 3) 该方法涉及到的超参数较少, 通常只需要很少的调参.

2 模型与方法

2.1 软参数共享模型

软参数共享模型的每个任务都有自己特定的模型和参数, 不同任务之间不共用底层的参数, 模型的参数之间通过正则化约束, 保障了参数空间的相似性, 从而达到多任务联合训练的效果.

软参数共享与常用的硬参数共享相比, 它的优点在于可以不需要提前考虑任务间的相关性强弱, 效果上不再受到任务差异和数据分布差异带来的影响, 在任务差异未知的情况下效果更好. 它与单任务学习相比, 可以充分利用任务间的共享信息进行学习, 极大程度地避免多个任务陷入局部最小解.

软参数共享模型如图1所示, 假设空间足够大, 可以同时包含多个任务的解. 假设数据一共有N个样本, 输入M个特征, 输出K个任务, 神经网络中共有s 个隐藏层. 记输入矩阵为$\boldsymbol{X}$, 输出矩阵为$\boldsymbol{Y}$; 则第n个样本的输入向量为$\boldsymbol{X}_n = (x_n^{(1)}, x_n^{(2)}, \cdots, x_n^{(M)})$, 第$n$个样本的输出向量为$\boldsymbol{Y}_n = (y_n^{(1)}, y_n^{(2)}, \cdots, y_n^{(K)})$. 第$m$维特征所有样本的输入向量为$\boldsymbol{X}^{(m)} = (x_1^{(m)}, x_2^{(m)}, \cdots, x_N^{(m)})^{\mathsf{T}}$, 第$j$个任务所有样本的输出向量为$\boldsymbol{Y}^{(j)} = (y_1^{(j)}, y_2^{(j)}, \cdots, y_N^{(j)})^{\mathsf{T}}$. $(n = 1, 2, \cdots, N, m = 1, 2, \cdots, M, j = 1, 2, \cdots, K)$. 图中每个隐藏层$i (i = 1, 2, \cdots, s)$ 一共有$q^{(i)}$个节点. 记第i层第j个任务的节点矩阵为$\boldsymbol{H}^{(i)(j)}$, 其中$\boldsymbol{H}^{(i)(j)} = (h_1^{(i)(j)}, h_2^{(i)(j)}, \cdots, h_{q^{(i)}}^{(i)(j)})$ $(i = 1, 2, \cdots, s, j = 1, 2, \cdots, K)$. 记所有数据的权重矩阵为$\boldsymbol{W}$, 偏置矩阵为$\boldsymbol{b}$; 第$j$个任务的权重矩阵为$\boldsymbol{W}^{(j)}$, 偏置矩阵为$\boldsymbol{b}^{(j)}$. 对于第$j$个任务, 记隐藏层$i-1$到隐藏层$i$的权重矩阵为$\boldsymbol{W}^{(i-1)(j)}$, 偏置矩阵为$\boldsymbol{b}^{(i-1)(j)}$; 则第$j$个任务每一个隐藏层的权重矩

阵可以被分解为$\boldsymbol{W}^{(j)}=(\boldsymbol{W}^{(1)}(j),\boldsymbol{W}^{(2)(j)},\cdots,\boldsymbol{W}^{(s)(j)})$, 第$j$个任务每一个隐藏层的偏置矩阵可以被分解为$\boldsymbol{b}^{(j)}=(\boldsymbol{b}^{(1)}(j),\boldsymbol{b}^{(2)(j)},\cdots,\boldsymbol{b}^{(s)(j)})$.

对于第j个任务, 其每一个隐藏层$i-1$到每个隐藏层i的权重矩阵$\boldsymbol{W}^{(i-1)(j)}$, 它可以被分解为第$i$个隐藏层的每个节点分别与第$i-1$个隐藏层所有结点的权重向量, 记为$\boldsymbol{W}^{(i-1)(j)}=(\boldsymbol{W}_1^{(i-1)(j)},\boldsymbol{W}_2^{(i-1)(j)},\cdots,\boldsymbol{W}_{q^{(i)}}^{(i-1)(j)})$. 对于第$i$个隐藏层中任意一个节点$p(p=1,\cdots,q^{(i)})$ 与第$i-1$个隐藏层所有结点的权重向量$\boldsymbol{W}_p^{(i-1)(j)}$, 其可以被分解为任意一个节点$p(p=1,\cdots,q^{(i)})$与第$i-1$个隐藏层每一个结点的权重标量$\boldsymbol{W}_p^{(i-1)(j)}=(w_{p1}^{(i-1)(j)},w_{p2}^{(i-1)(j)},\cdots,w_{pq^{(i-1)}}^{(i-1)(j)})^{\mathsf{T}}$. 对于第$j$个任务, 其每一个隐藏层$i-1$到每个隐藏层$i$的偏置矩阵$\boldsymbol{b}^{(i-1)(j)}$, 它可以被分解为第$i$个隐藏层的每个节点分别与第$i-1$个隐藏层的所有结点的偏置标量, 记为$\boldsymbol{b}^{(i-1)(j)}=(b_1^{(i-1)(j)},b_2^{(i-1)(j)},\cdots,b_{q^{(i)}}^{(i-1)(j)})$.

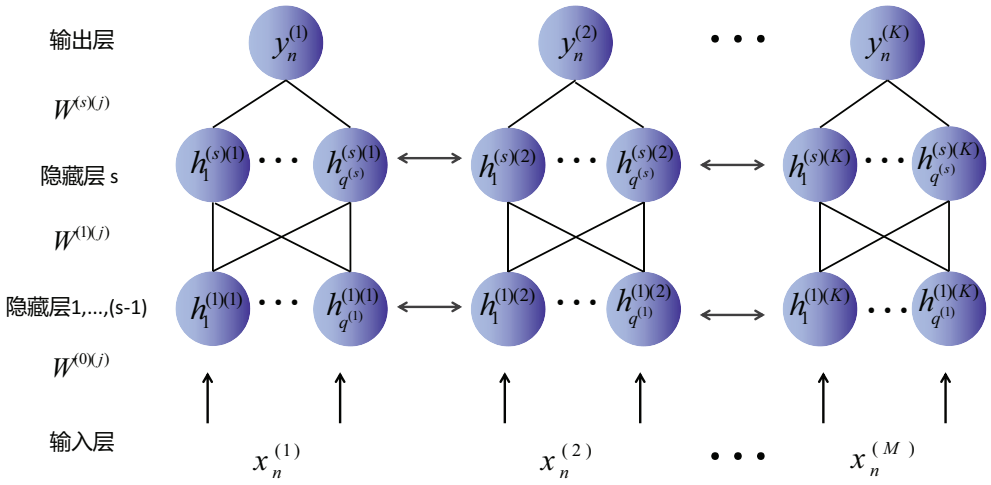

图1 软参数共享模型

(Figure 1 Soft parameter sharing model)

2.2 均值约束共享的软参数多任务学习方法

软参数共享通过参数与参数之间的关系, 使得多个任务可以有效地利用它们之间的信息, 从而享受多任务联合训练的优势. 具体考虑到 1) 使用L_2距离正则化在计算的时候相较于其它的惩罚项(例如L_1正则项)而言, 具有显著的耗时短, 预测精度高的优势. 2) 使用均值约束共享的方法在对于软参数多任务学习时, 可以充分利用任务之间的相似度与任务参数之间的关系进行建模. 均值约束共享的方法相较于其它的方法而言, 其对于大于2个任务的软参数多任务学习也能达到充分利用任务之间的具体信息进行建模的效果, 从而预测效果更好. 本文采用Duong等[8]提出的L_2距离正则化来建立多个任务之间的关系. 在软参数共享中, 参数不完全共用, 只是用

正则化项来建立参数之间的关系. 记$y_n^{(j)}$作为第n个样本任务j的真实值，那么第n个样本任务j的预测值可以表示为:

$$\widehat{y}_n^{(j)} = f(\boldsymbol{X}_n; \boldsymbol{W}^{(j)}), \tag{2.1}$$

其中, $\boldsymbol{W}^{(j)}$是模型的参数. f是一个具有网络结构$(\boldsymbol{X}_n, \boldsymbol{W}^{(j)})$的神经网络. 本文采用Li等[15]提出的基于均值约束共享的正则化方法来建立模型, 屈武和阎高伟[16]也指出均值共享约束方法优于传统的软参数共享方法, 本文在基于均值约束共享的软参数多任务学习方法上进行改进, 具体改进如下：本文中的均值约束共享正则化方法认为任务之间有一定的相关性, 但是不用考虑任务之间相关性的强弱. 认为不同模型任务之间的相关性是通过模型参数相互接近来量化的, 考虑了每一个任务的模型参数与所有任务的模型参数的均值, 来度量任务之间的相关性.

每个样本在训练集上的损失函数计算公式为:

$$L(y_n^{(j)}, f(\boldsymbol{X}_n; \boldsymbol{W}^{(j)})) = (y_n^{(j)} - f(\boldsymbol{X}_n; \boldsymbol{W}^{(j)}))^2, \tag{2.2}$$

$$Loss_{train} = \sum_{j=1}^{K} L(y_n^{(j)}, f(\boldsymbol{X}_n; \boldsymbol{W}^{(j)})) + \lambda \left\| \sum_{j=1}^{K} (\boldsymbol{W}^{(j)} - \frac{1}{K} \sum_{j=1}^{K} \boldsymbol{W}^{(j)}) \right\|_2^2, \tag{2.3}$$

$$\left\| \sum_{j=1}^{K} (\boldsymbol{W}^{(j)} - \frac{1}{K} \sum_{j=1}^{K} \boldsymbol{W}^{(j)}) \right\|_2^2 = \left\| \sum_{i=1}^{s} \sum_{p=1}^{q^{(i)}} \sum_{u=1}^{q^{(i-1)}} \sum_{j=1}^{K} (w_{pu}^{(i-1)(j)} - \frac{1}{K} \sum_{j=1}^{K} w_{pu}^{(i-1)(j)}) \right\|_2^2, \tag{2.4}$$

其中, 正则项系数λ代表了多个任务之间的相似度, λ 越大, 任务之间的参数就越接近, 任务之间的关系就越紧密.

在验证集上计算不同任务的损失函数$L(y_n^{(j)}, f(\boldsymbol{X}_n^*; \boldsymbol{W}^{(j)}))$, 选择使损失函数最小的参数方案. 以测试集上的损失函数来作为软参数多任务学习预测性能的评价指标, 每个样本在验证集和测试集上损失函数的计算公式为:

$$Loss_{valid} = \sum_{j=1}^{K} L(y_n^{(j)}, f(\boldsymbol{X}_n^*; \boldsymbol{W}^{(j)})), \tag{2.5}$$

其中, $\boldsymbol{X}_n^*$代表验证集上的样本.

$$Loss_{test} = \sum_{j=1}^{K} L(y_n^{(j)}, f(\boldsymbol{X}_n^{**}; \boldsymbol{W}^{(j)})), \tag{2.6}$$

其中, $\boldsymbol{X}_n^{**}$代表测试集上的样本.

在多任务深度学习中常采用均方误差(mean square error, MSE)和平均绝对误差(mean absolute error, MAE)作为整体模型预测性能的评价指标，计算公式为:

$$MSE = \frac{1}{N} \sum_{n=1}^{N} \left(\sum_{j=1}^{K} L(y_n^{(j)}, f(\boldsymbol{X}_n^{**}; \boldsymbol{W}^{(j)})) \right)^2, \tag{2.7}$$

$$MAE = \frac{1}{N} \sum_{n=1}^{N} \left| \sum_{j=1}^{K} L(y_n^{(j)}, f(\boldsymbol{X}_n^{**}; \boldsymbol{W}^{(j)})) \right|. \tag{2.8}$$

2.3 自适应软参数共享的多任务学习算法

软参数共享可以让多个任务联合训练, 使得任务之间可以互相利用有效的信息进行学习, 但负迁移现象往往难以得到有效的处理. 软参数共享的多任务学习原理机制为: 任务之间的相似度越高, 则参数最优解的相似性也就越高; 正则项系数λ代表了多个任务之间的相似度, λ越大, 任务之间的参数就越接近, 任务之间的关系就越紧密[2]. 在多任务学习中常常存在多个损失函数的局部极小值点, 本文的目标是让多个任务更精确地达到它们损失函数的全局最小点. 基于此, 本文提出了自适应软参数共享的多任务学习算法. 为简单起见, 本节以两个任务为例.

在单任务学习时, 任务之间不能相互利用它们之间的有效信息进行共享帮助, 从而非常容易陷入损失函数的局部最小点[17]. 如图2所示, 两个任务在学习过程中都陷入了它们的局部极小值点且不会继续进行迭代学习, 最后它们分别在损失函数的某一个局部极小值点(图2中绿色的点)停止学习.

在进行软参数多任务学习时, 多个任务的局部最小点不同, 不同的任务共享底层特征, 交换信息, 相互学习; 任务之间可以利用共享信息, 帮助各自的任务逃离局部最小点. 但任务之间有不相关的信息作为噪声, 会产生负迁移现象, 从而很难达到它们单独的全局最小点[17]. 如图2所示, 两个任务相互帮助, 利用它们的相关信息, 逃离了它们损失函数的第一个局部最小点, 相比较于单任务学习而言, 增加了模型的预测性能. 但由于两个任务之间不相关信息产生的噪音引起的负迁移现象, 两个任务之间只能在它们损失函数的全局最小解的范围内(图2中蓝色虚线和红色的虚线)终止学习. 图2中黑色的点代表在进行软参数多任务学习时, 两个任务能够到达的损失函数极小值.

本文提出的自适应软参数多任务学习方法, 通过自适应地去除训练集上损失函数的正则项中数值较大的项, 使得任务之间可以利用共享信息, 帮助各自的任务逃离局部最小点后, 进一步去除任务之间不相关信息, 减少了任务之间的噪音, 极大减少负迁移现象. 该方法使得每个任务可以在全局最小解的范围内继续进行迭代学习, 从而更精确地找到每个任务的全局最小解. 如图2所示, 两个任务在利用它们的相关信息, 逃离了它们损失函数的第一个局部最小点后, 继续删除任务之间不相关信息(图2中红色的箭头). 该过程大大减少了任务之间的噪音, 使得两个任务可以继续进行有效的学习, 最终在它们损失函数的全局最小点(图2中红色的点)终止学习.

自适应软参数多任务学习通过去除训练集上损失函数中正则项里权重差最大的那部分参数矩阵, 使得相似度最低的部分任务独立地进行训练, 最小化负迁移现象. 自适应软参数多任务学习相较于多个任务单独学习而言, 它可以利用多任务学习中的注意力集中机制, 正则化机制等优势, 充分利用相关性较强的任务之间的有效信息进行学习. 它相较于软参数多任务学习而言, 可以在享受了相关性较强的任务之间的信息进行学习的同时, 去除相关性较弱的任务之间产生的干扰信息, 使得相关性弱的任务独立地进行学习, 进一步减少了负迁移现象, 可以达到“取其精华, 去其糟粕”的效果.

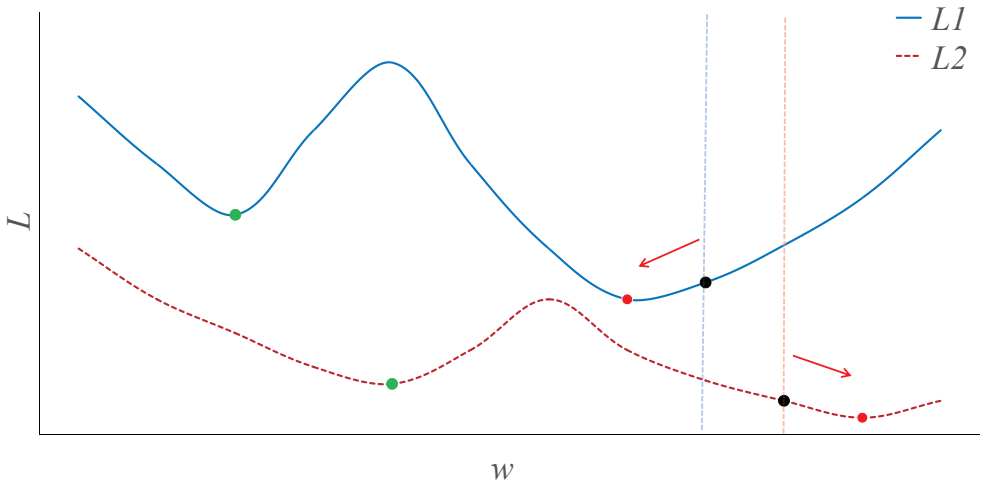

图2 单任务、软参数多任务以及自适应软参数多任务学习的对比图

(Figure 2 Comparison chart of single-task, soft parameter multi-task and adaptive soft parameter multi-task learning)

本文中的参数衰减指的是在$Loss_{train}$的正则项中对差值较大的参数项进行去除, 接下来将介绍本文中超参数的含义以及自适应软参数共享的多任务学习算法.

自适应参数衰减比例θ: 该超参数是指在进行每轮软参数自适应多任务学习迭代时, 在训练集正则项中删除数值较大的前θ比例的项. 例如在正则项$\sum_{i=1}^{s}\sum_{p=1}^{q^{(i)}}\sum_{u=1}^{q^{(i-1)}}\sum_{j=1}^{k}(w_{pu}^{(i-1)(j)}-\frac{1}{k}\sum_{j=1}^{k}w_{pu}^{(i-1)(j)})$ 中, 一共有$s\cdot q^{(i)}\cdot q^{(i-1)}\cdot k$ 个项, 在进行下一轮迭代前, 按数值从高到低排序, 删除前$s\cdot q^{(i)}\cdot q^{(i-1)}\cdot k\cdot\ \theta$ 个项.

累计自适应参数衰减比例θ^*: 该超参数是指在$Loss_{valid}^*$的参数方案下, 在第T^*轮迭代前, 正则项中累计删除的数值较大的$T^*\cdot\theta$ 比例的项, 它是θ的倍数, 其中, $Loss_{valid}^*$表示在$\frac{1}{\theta}$ 轮迭代中, 在验证集上自适应选出的损失函数最低点.

自适应正则项系数λ^*: 该超参数与在进行软参数多任务学习的正则化系数意义相同, 代表多个任务之间的相似度, 随着参数的衰减, 任务的相似度将会改变.

自适应软参数共享的多任务学习算法详细描述了损失函数的变化过程, 在进行该算法前, 首先设置了自适应参数衰减比例θ和自适应正则项系数λ^*. 设置软参数正则项系数λ和自适应正则项系数λ^*, 是为了通过任务与任务之间的相关性进行建模. 设置了自适应参数衰减比例θ是为了自适应更精确地找到验证集上损失函数的最小值.

本文方法中超参数的λ^*是根据任务之间的相关性来确定的, 一般来说任务越相似, 则λ 越大. 参数衰减比例θ 并没有固定的调参的方法, 一般初始值在$0.01-0.05$ 中选择, 而λ^*随着惩罚项项数越小, λ^*也会越来越小.

记$\boldsymbol{W}^{[1]}$和$\boldsymbol{b}^{[1]}$为软参数多任务学习迭代收敛时的参数矩阵与偏置矩阵. $\boldsymbol{W}^{[T]}$和$\boldsymbol{b}^{[T]}$为软参数多任务学习收敛之后, 自适应软参数多任务学习中的第T轮迭代收敛时的权重矩阵和偏置矩阵.

算法 1 自适应软参数共享的多任务学习算法

输入: 训练集$Z = \{(\boldsymbol{X}_1, \boldsymbol{Y}_1), (\boldsymbol{X}_2, \boldsymbol{Y}_2), \cdots, (\boldsymbol{X}_N, \boldsymbol{Y}_N)\}$, 其中$\boldsymbol{X}_n = (x_n^{(1)}, x_n^{(2)}, \cdots, x_n^{(M)}) \in \mathcal{X} \subseteq \boldsymbol{R}^M$, $\boldsymbol{Y}_n = (y_n^{(1)}, y_n^{(2)}, \cdots, y_n^{(K)}) \in \mathcal{Y} \subseteq \boldsymbol{R}^k, n = 1, 2, \cdots, N$. 软参数正则项系数$\lambda$，自适应正则项系数$\lambda^*$，自适应参数衰减比例$\theta$.

输出: 第T轮的自适应参数矩阵$\boldsymbol{W}^{[T]}$, 偏置矩阵$\boldsymbol{b}^{[T]}$, 训练集上的损失函数$Loss_{train}$, 验证集上的损失函数$Loss_{valid}$, 测试集上的损失函数$Loss_{test}$, 累积自适应参数衰减比例θ^*.

1. 设置软参数正则项系数λ, 根据*Adam*算法, 通过误差反向传播更新参数矩阵$\boldsymbol{W}$, 偏置矩阵$\boldsymbol{b}$, 直至收敛后停止迭代. 得到$\boldsymbol{W}^{[1]}$和$\boldsymbol{b}^{[1]}$.

2. 设置自适应正则项系数λ^*, 自适应参数衰减比例θ, 将(2.4)式的$(w_{pu}^{(i-1)(j)} - \frac{1}{K}\sum_{j=1}^{K} w_{pu}^{(i-1)(j)})$中数值较大的前$s \cdot q^{(i)} \cdot q^{(i-1)} \cdot k \cdot \theta$ 个项删除.

3. 进行第2轮迭代, 直到收敛, 得到$\boldsymbol{W}^{[2]}$和$\boldsymbol{b}^{[2]}$. 继续将(2.4)式的$(w_{pu}^{(i-1)(j)} - \frac{1}{K}\sum_{j=1}^{K} w_{pu}^{(i-1)(j)})$中数值较大的前$s \cdot q^{(i)} \cdot q^{(i-1)} \cdot k \cdot \theta$ 个项删除.

4. 重复步骤3, 则在第T轮迭代结束后, (2.4)式的$(w_{pu}^{(i-1)(j)} - \frac{1}{K}\sum_{j=1}^{K} w_{pu}^{(i-1)(j)})$中共删除了数值较大的前$s \cdot q^{(i)} \cdot q^{(i-1)} \cdot k \cdot \theta \cdot T$ 个项, 一共进行$\frac{1}{\theta}(\frac{1}{\theta} \geq T)$ 轮迭代, 记每轮迭代后验证集上收敛时的损失函数为$Loss_{valid}^{T}, (T = 1, \cdots, \frac{1}{\theta})$.

5. 在$\frac{1}{\theta}$ 轮迭代中寻找在验证集上收敛时的损失函数最低点, 记其为第T^*轮迭代, 该轮迭代收敛时的参数矩阵为$\boldsymbol{W}^{[*]}$, 偏置矩阵为$\boldsymbol{b}^{[*]}$, 损失函数为$Loss_{valid}^{*}$, 且输出θ^*.

6. 在测试集上使用参数矩阵$\boldsymbol{W}^{[*]}$, 偏置矩阵$\boldsymbol{b}^{[*]}$, 计算$Loss_{test}$.

3 模拟研究

为了简单起见，本节$M = 4, K = 2, s = 2$，模型如下：

$$\boldsymbol{Y} = (\boldsymbol{Y}^{(1)}, \boldsymbol{Y}^{(2)}), \tag{3.1}$$

$$\boldsymbol{Y}^{(j)} = (\boldsymbol{X}\boldsymbol{W}^{(1)(j)} + \boldsymbol{b}^{(1)(j)})\boldsymbol{W}^{(2)(j)} + \boldsymbol{b}^{(2)(j)} + \boldsymbol{\varepsilon}^{(j)}, \ \ j = 1, 2. \tag{3.2}$$

情形1: $\boldsymbol{X}$是一个$N * 4$维矩阵, 其中$\boldsymbol{X}_{ik} \sim U(0, 8)$, $i = 1, 2, \cdots, N$, $k = 1, 2, 3, 4$. $\boldsymbol{W}^{(1)(j)}$是一个4*3维矩阵, 其中$\boldsymbol{W}_{ik}^{(1)(j)} \sim U(0, 8)$, $i = 1, 2, 3, 4$, $k = 1, 2, 3$. $\boldsymbol{W}^{(2)(j)}$是一个3*1维向量, 其中$\boldsymbol{W}_{ik}^{(2)(j)} \sim U(0, 8)$, $i = 1, 2, 3$, $k = 1$. $\boldsymbol{b}^{(1)(j)}$是一个$N * 3$维矩阵, 其中$\boldsymbol{b}_{ik}^{(1)(j)} \sim N(0, 1)$, $i = 1, 2, \cdots, N$, $k = 1, 2, 3$. $\boldsymbol{b}^{(2)(j)}$是一个$N * 1$维向量, 其中$\boldsymbol{b}_{ik}^{(2)(j)} \sim N(0, 1)$, $i = 1, 2, \cdots, N$, $k = 1$. $\boldsymbol{\varepsilon}^{(j)}$ 是一个$N * 1$维向量, 其中$\boldsymbol{\varepsilon}_{ik}^{(j)} \sim N(0, 1)$, $i = 1, 2, \cdots, N$, $k = 1$. $\boldsymbol{Y}$是一个$N * 2$维矩阵.

情形2: $\boldsymbol{X}$是一个$N * 4$维矩阵, 其中$\boldsymbol{X}_{ik} \sim N(0, 1)$, $i = 1, 2, \cdots, N$, $k = 1, 2, 3, 4$. $\boldsymbol{W}^{(1)(j)}$是一个4*3维矩阵, 其中$\boldsymbol{W}_{ik}^{(1)(j)} \sim N(0, 1)$, $i = 1, 2, 3, 4$, $k = 1, 2, 3$. $\boldsymbol{W}^{(2)(j)}$是一个3*1维向量, 其中$\boldsymbol{W}_{ik}^{(2)(j)} \sim N(0, 1)$, $i = 1, 2, 3$, $k = 1$. $\boldsymbol{b}^{(1)(j)}$是一个$N * 3$维矩阵, 其中$\boldsymbol{b}_{ik}^{(1)(j)} \sim N(0, 1)$, $i = 1, 2, \cdots, N$, $k = 1, 2, 3$. $\boldsymbol{b}^{(2)(j)}$是一个$N * 1$维向量, 其中$\boldsymbol{b}_{ik}^{(2)(j)} \sim N(0, 1)$, $i = 1, 2, \cdots, N$, $k = 1$. $\boldsymbol{\varepsilon}^{(j)}$ 是一个$N * 1$维向量, 其中$\boldsymbol{\varepsilon}_{ik}^{(j)} \sim N(0, 1)$, $i = 1, 2, \cdots, N$,

$k=1$. $\boldsymbol{Y}$是一个$N*2$维矩阵.

情形3: $\boldsymbol{X}$是一个$N*4$维矩阵, 其中$\boldsymbol{X}_{ik} \sim Exp(5)$, $i=1,2,\cdots,N$, $k=1,2,3,4$. $\boldsymbol{W}^{(1)(j)}$是一个4*3维矩阵, 其中$\boldsymbol{W}_{ik}^{(1)(j)} \sim N(0,1)$, $i=1,2,3,4$, $k=1,2,3$. $\boldsymbol{W}^{(2)(j)}$是一个3*1维向量, 其中$\boldsymbol{W}_{ik}^{(2)(j)} \sim N(0,1)$, $i=1,2,3$, $k=1$. $\boldsymbol{b}^{(1)(j)}$是一个$N*3$维矩阵, 其中$\boldsymbol{b}_{ik}^{(1)(j)} \sim N(0,1)$, $i=1,2,\cdots,N$, $k=1,2,3$. $\boldsymbol{b}^{(2)(j)}$是一个$N*1$维向量, 其中$\boldsymbol{b}_{ik}^{(2)(j)} \sim N(0,1)$, $i=1,2,\cdots,N$, $k=1$. $\boldsymbol{\varepsilon}^{(j)}$是一个$N*1$维向量, 其中$\boldsymbol{\varepsilon}_{ik}^{(j)} \sim N(0,1)$, $i=1,2,\cdots,N$, $k=1$. $\boldsymbol{Y}$是一个$N*2$维矩阵.

本文采用*Adam*算法更新参数[18], *Adam*的公式如下:

$$\boldsymbol{W}=\boldsymbol{W}-\frac{\eta\widehat{m}_t}{\sqrt{\widehat{v}_t}+c}, \tag{3.3}$$

$$\widehat{m}_t=\frac{m_t}{1-\alpha_1^t}, \tag{3.4}$$

$$\widehat{v}_t=\frac{v_t}{1-\alpha_2^t}, \tag{3.5}$$

$$m_t=\alpha_1 m_{t-1}+(1-\alpha_1)g_t, \tag{3.6}$$

$$v_t=\alpha_2 v_{t-1}+(1-\alpha_2)g_t^2, \tag{3.7}$$

其中, η表示学习率, t表示更新的步长, g_t表示梯度, m_t表示对g_t的指数加权移动平均, v_t表示对g_t平方的指数加权移动平均, $\widehat{m}_t$表示m_t的偏置矫正, $\widehat{v}_t$表示v_t的偏置矫正, c是为了维持数值稳定性而添加的常数, 一般为10^{-8}. α_1和α_2代表的是指数衰减系数.

本文构建的算法程序包括: *CPU*为$r7-4800H$, *GPU*为$RTX2060$, 操作系统为64位windows, 开发语言为python. 所有方法的实现均基于pytorch框架. 设置输入层的神经元数为m, 共享层1的神经元数为4, 共享层2的神经元数为3, 输出层的神经元数为2. *Adam*优化算法学习率η=0.001, 指数衰减率α_1=0.99, α_2=0.9999, 常数$c=10^{-8}$.

随着迭代的进行, λ^*的值逐渐下降, 本文表格中的λ^*是迭代到损失函数最小时的最终值. 对于情形1, 情形2和情形3, 经过调参可以得到最优的超参数b如表1和表2.

表1 评价指标为MSE的最优超参数表

(Table 1 The optimal hyperparameter table with evaluation metric as MSE)

	λ	λ^*	θ	θ^*
情形1	0.001	0.00043	0.07	0.63
情形2	0.04	0.0176	0.02	0.38
情形3	0.0006	0.00034	0.10	0.80

表2 评价指标为MAE的最优超参数表

(Table 2 The optimal hyperparameter table with evaluation metric as MAE)

	λ	λ^*	θ	θ^*
情形1	0.0009	0.00037	0.062	0.558
情形2	0.055	0.0142	0.016	0.304
情形3	0.00075	0.00059	0.08	0.72

λ表示在进行软参数多任务学习时, 使得软参数多任务学习的损失函数达到最小值的最优参数. λ^*和θ表示在进行自适应软参数多任务学习时, 使得自适应软参数多任务学习的损失函数达到最小值的最优参数, θ^*是与之相对应的最优参数.

对于上述情形分别设置与软参数多任务学习和两个单任务学习的对比实验, 为保证对比的有效性, 我们分别将三种学习损失函数的最低值进行对比. 对于上面三种模拟情形, 分别取样本量N为500, 1000, 2000, 3000和4000. 整体数据集按7:3的比例分别随机划分为训练集和测试集, 将训练集中30%的数据作为验证集, 用于选择出最优的模型. 测试集用于评估模型性能和稳健性, 并采用100轮结果的均值作为实验结果.

表3 三种情形100次模拟$MSE(10^{-4})$的均值

(Table 3 The mean of 100 simulations of $MSE(10^{-4})$ in three scenarios)

	情形1			情形2			情形3		
	前馈网络	软参数	自适应	前馈网络	软参数	自适应	前馈网络	软参数	自适应
$N=500$	3.957	3.159	3.015	11.862	8.983	8.638	1.438	1.186	1.069
$N=1000$	3.632	2.853	2.749	10.973	8.164	7.964	1.260	0.974	0.813
$N=2000$	3.115	2.588	2.417	9.620	7.272	6.593	0.874	0.682	0.645
$N=3000$	2.770	2.167	1.974	7.631	6.086	5.770	0.679	0.457	0.392
$N=4000$	1.474	1.239	1.116	6.258	4.628	4.183	0.285	0.213	0.190

表4 三种情形100次模拟$MAE(10^{-3})$的均值

(Table 4 The mean of 100 simulations of $MAE(10^{-3})$ in three scenarios)

	情形1			情形2			情形3		
	前馈网络	软参数	自适应	前馈网络	软参数	自适应	前馈网络	软参数	自适应
$N=500$	5.816	5.013	4.972	10.237	6.780	6.692	1.134	0.891	0.847
$N=1000$	5.437	4.762	4.536	9.106	6.251	5.963	0.940	0.792	0.761
$N=2000$	4.819	4.395	4.143	7.851	5.478	5.006	0.771	0.594	0.569
$N=3000$	4.219	3.860	3.627	6.372	4.739	4.317	0.582	0.355	0.329
$N=4000$	3.071	2.475	2.253	4.893	3.214	2.865	0.247	0.196	0.179

表3和表4分别为情形1, 情形2和情形3在前馈网络, 软参数多任务学习和自适应软参数多任务学习的对比实验结果. 从表3和表4可以看出, 自适应软参数多任务学习优于软参数多任务学习, 软参数多任务学习优于前馈网络, 且随着样本量的增大, 三种情形的MSE和MAE的值越来越小. 性能提升值p的计算公式如下:

$$p=\frac{s-a}{s}\times 100\%, \tag{3.8}$$

其中, s代表软参数多任务学习的MSE和MAE, a代表自适应软参数多任务学习的MSE和MAE.

表5 三种情形下的平均耗时(s)以及性能提升值(%)(MSE)

(Table 5 The average time(s) and performance improvement values (%) in three scenarios(MSE))

	软参数耗时	自适应耗时	性能提升值
情形1	91	106	6.6
情形2	82	98	10.1
情形3	86	97	5.4

表6 三种情形下的平均耗时(s)以及性能提升值(%)(MAE)

(Table 6 The average time(s) and performance improvement values (%) in three scenarios(MAE))

	软参数耗时	自适应耗时	性能提升值
情形1	125	143	5.7
情形2	117	136	8.6
情形3	112	131	4.2

将自适应软参数多任务学习和软参数多任务学习做对比. 预测性能的衡量仍采用MSE和MAE作为评价指标. 表5和表6展示了当N为2000时的性能提升值和耗时. 从提升值来看, 无论是在MSE还是MAE这两个评价指标下, 自适应软参数多任务学习相比于软参数多任务学习的性能提升值都超过了4.2%. 在耗时方面, 自适应软参数耗时与软参数相比差值在19S 以内. 从$p > 0$可以看出, 自适应软参数多任务学习的整体性能优于软参数多任务学习, 说明了自适应软参数多任务学习相较于软参数多任务学习具有良好的预测和估计能力, 代价仅仅是少量的时间.

表7 三种软参数多任务学习方法的$MSE(10^{-4})$均值

(Table 7 The mean MSE values(10^{-4}) of three soft parameter multi-task learning methods)

	MAE			MSE		
	L_2正则	均值约束共享	两者结合	L_2正则	均值约束共享	两者结合
情形1	4.1890	4.1765	4.1434	0.2480	0.2465	0.2417
情形2	5.0378	5.0451	5.0062	0.6563	0.6581	0.6539
情形3	0.5713	0.5721	0.5691	0.0651	0.0652	0.0645

将基于L_2正则和均值约束共享两者结合的方法与单独使用L_2距离正则化和均值约束共享做对比. 表7分别表示三种方法分别在三种情形, 2000个样本, 100次模拟的情况下MAE和MSE的效果对比. 从表7中可以看出, 在进行模型预测性能的测试时, 基于L_2正则和均值约束共享的软参数学习方法分别比单独基于L_2 正则和单独基于均值约束共享的软参数学习方法具有更小的MAE和MSE. 说明基于L_2正则和均值约束共享两者结合的软参数共享方法具有更精确的预测和估计能力.

为了具体展示自适应软参数多任务学习的细节, 图3-5分别展示了样本量为2000, 分布为均匀分布、标准正态分布和指数分布在500次迭代之后, 继续迭代的软参数

多任务学习与自适应软参数多任务学习的MSE变化对比过程. 图3-5中黑点代表软参数多任务学习损失函数的最低点, 黑点之后代表软参数多任务学习与自适应软参数多任务学习的损失函数变化对比. 图3-5 中红点代表累计自适应参数衰减比例θ^*取得时的自适应软参数多任务学习损失函数的最低点. 从图3-5中可以看出在软参数多任务学习迭代到最低点的时候, 三种模拟情形可以通过自适应软参数多任务学习算法进一步优化, 使得损失函数进一步减小, 说明了自适应软参数多任务学习良好的预测性能.

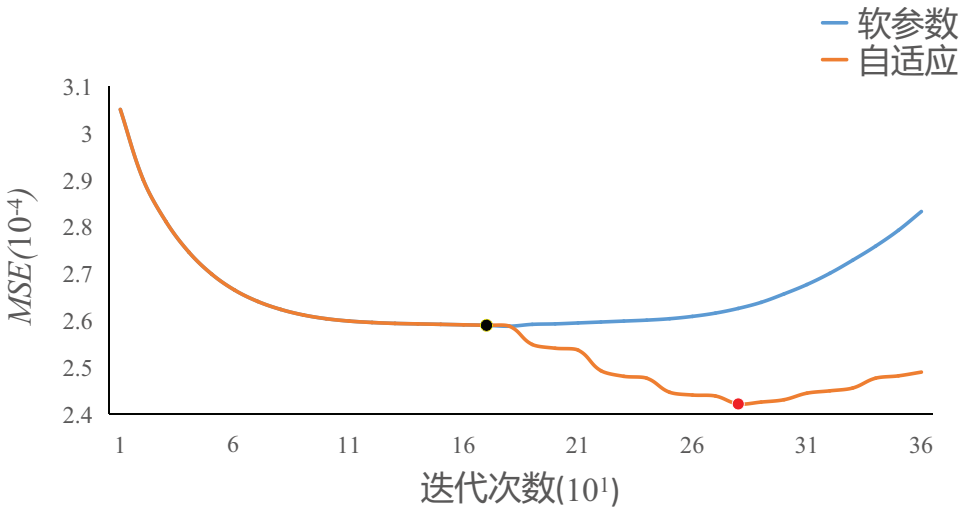

图3 均分分布的软参数多任务与自适应软参数多任务学习的MSE变化对比图

(Figure 3 Comparison chart of the MSE variation between equally-distributed soft parameter multi-task and adaptive soft parameter multi-task learning)

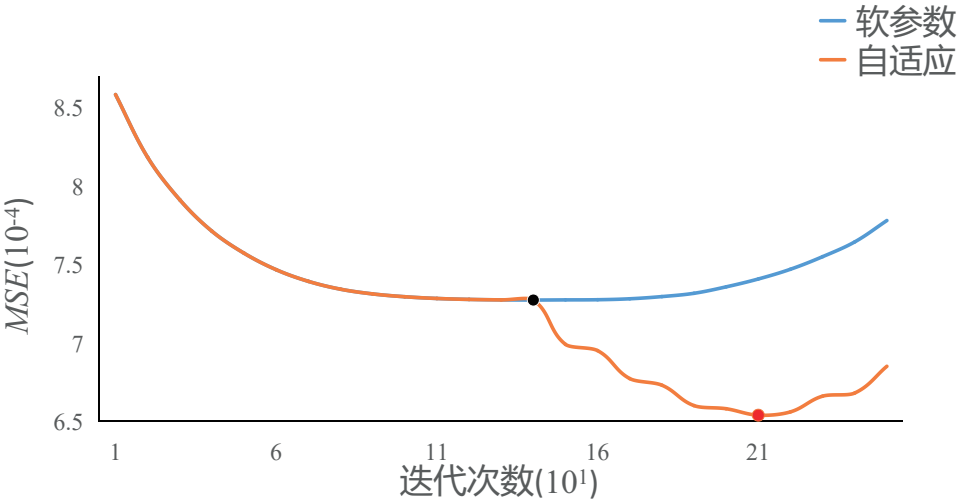

图4 标准正态分布的软参数多任务与自适应软参数多任务学习的MSE变化对比图

(Figure 4 Comparison chart of the MSE variation between standard normal distribution soft parameter multi-task and adaptive soft parameter multi-task learning)

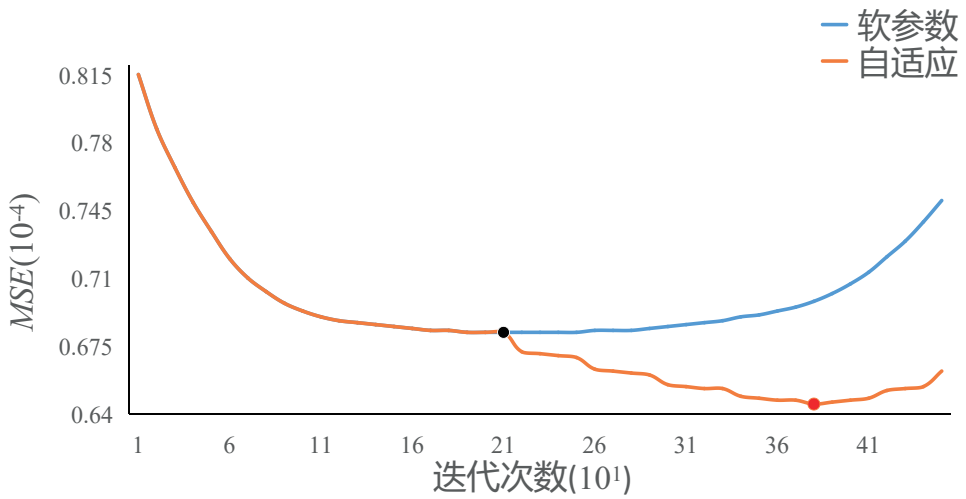

图5 指数分布的软参数多任务与自适应软参数多任务学习的MSE变化对比图

(Figure 5 Comparison chart of the MSE variation between exponential distribution soft parameter multi-task and adaptive soft parameter multi-task learning)

4 案例分析

4.1 两个任务的实例

李云祯等[19][20]采用统计学方法, 说明了天气影响因素之间的相关性. 本节考虑南京市空气污染数据, 数据来源于中国气象数据网和真气环境大数据中心(https://www.aqistudy.cn/historydata/). 数据包含了南京市2013年12月2日至2022年11月23日的$PM_{2.5}$浓度($\mu g/m^3$), PM_{10}浓度($\mu g/m^3$), CO浓度(mg/m^3), SO_2浓度($\mu g/m^3$), NO_2浓度($\mu g/m^3$) 和O_3_8h($\mu g/m^3$)这6个变量. $PM_{2.5}$浓度和O_3_8h($\mu g/m^3$)分别为输出变量$\boldsymbol{Y}^{(1)}$和$\boldsymbol{Y}^{(2)}$. PM_{10}浓度, CO浓度, SO_2 浓度和NO_2 浓度分别为输入变量$\boldsymbol{X}^{(1)}$, $\boldsymbol{X}^{(2)}$, $\boldsymbol{X}^{(3)}$和$\boldsymbol{X}^{(4)}$. 样本量N为3274个.

在本节中, s=2, 则空气污染预测模型如下:

$$\boldsymbol{Y} = (\boldsymbol{Y}^{(1)}, \boldsymbol{Y}^{(2)}), \tag{4.1}$$

$$\boldsymbol{Y}^{(j)} = \sigma((\sigma(\boldsymbol{X}\boldsymbol{W}^{(1)(j)} + \boldsymbol{b}^{(1)(j)}))\boldsymbol{W}^{(2)(j)} + \boldsymbol{b}^{(2)(j)}) + \boldsymbol{\varepsilon}^{(j)}, \ \ j = 1, 2, \tag{4.2}$$

其中, $\boldsymbol{X}$是一个$N*4$维矩阵, $\boldsymbol{W}^{(1)(j)}$是一个4*3维矩阵, $\boldsymbol{W}^{(2)(j)}$是一个3*1维向量, $\boldsymbol{b}^{(1)(j)}$是一个N*3维矩阵, $\boldsymbol{b}^{(2)(j)}$是一个N*1维向量, σ为$sigmoid$激活函数. $\boldsymbol{\varepsilon}^{(j)}$ 是一个$N*1$维向量, 其中$\varepsilon_{ik}^{(j)} \sim N(0,1)$, $i = 1, 2, \cdots, N$, $k = 1$. 输出$\boldsymbol{Y}$ 是一个N*2维矩阵.

对此预测模型, 经过调参可以得到最优的超参数如表8.

表8 空气污染实例的最优超参数表

(Table 8 The optimal hyperparameter table for air pollution instance)

超参数 评价指标	MSE	MAE
λ	0.0115	0.0115
λ^*	0.0071	0.0071
θ	0.045	0.045
θ^*	0.54	0.54

将两个任务的自适应软参数多任务学习和软参数多任务学习做对比, 且估计效率的衡量仍采用MSE和MAE. 由表9可知, 当评价指标为MSE时, 自适应软参数多任务学习比软参数多任务学习多了13s, 自适应软参数多任务学习比软参数多任务学习的效果提升了3.4%. 当评价指标为MAE时, 自适应软参数多任务学习比软参数多任务学习多了16s, 自适应软参数多任务学习比软参数多任务学习的效果提升了3.0%.

表9 两种模型的损失函数值(10^{-2}), 耗时(s)及性能提升值(%)

(Table 9 The loss function values(10^{-2}), time consumption(s) and performance improvement values of two models(%))

	软参数	自适应	时间差	性能提升值
MSE	1.489	1.438	13	3.4
MAE	8.827	8.563	16	3.0

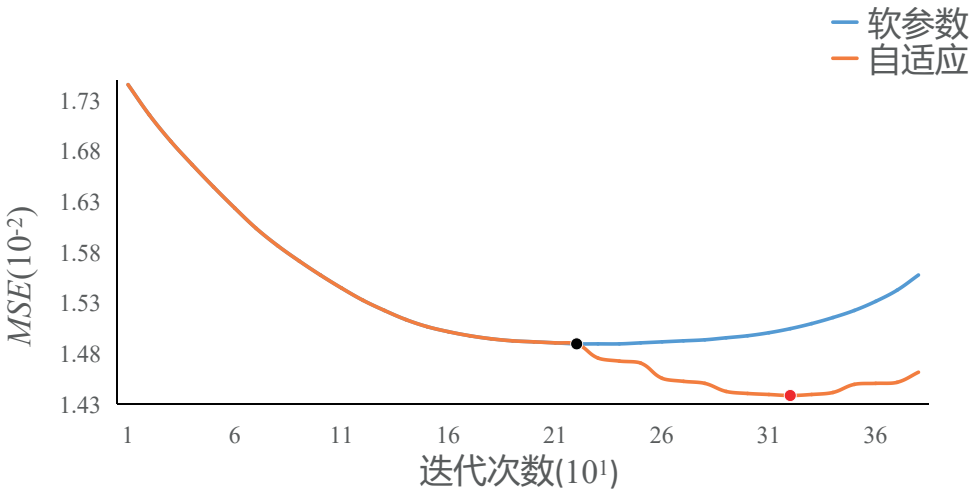

图6 南京市空气污染数据软参数多任务与自适应软参数多任务学习的MSE变化对比图

(Figure 6 Comparison chart of the MSE variation between soft parameter multi-task and adaptive soft parameter multi-task learning for air pollution data in Nanjing)

图6是南京市空气污染数据软参数多任务与自适应软参数多任务学习的MSE变化对比图, 图6中的标记点与模拟数据时的标记点含义一样. 从图6中可以看出对于

南京市空气污染数据, 在软参数多任务学习达到其损失函数的最低点后, 通过自适应软参数多任务学习仍可以进一步优化, 使得损失函数进一步下降, 说明了自适应软参数多任务学习良好的预测性能.

4.2 三个任务的实例

Zhu Y[21]采用改进的结构方程模型, 说明了学生的数学成绩, 阅读能力以及写作成绩呈现一定的相关关系. 本节考虑学生成绩数据, 该数据集来源于kaggle(https://www.kaggle.com/datasets/rkiattisak/student-performance-in-mathematics). 数据包含了高中生在数学方面的表现信息: 种族/民族, 父母教育程度, 午餐, 备考程度, 数学成绩, 阅读成绩和写作成绩这7个变量. 本文中数学成绩, 阅读分数和写作分数分别为输出变量$\boldsymbol{Y}^{(1)}$, $\boldsymbol{Y}^{(2)}$ 和$\boldsymbol{Y}^{(3)}$. 种族, 父母的教育水平, 午餐和考试备考程度分别为输入变量$\boldsymbol{X}^{(1)}$, $\boldsymbol{X}^{(2)}$, $\boldsymbol{X}^{(3)}$和$\boldsymbol{X}^{(4)}$, 样本量N为1000个. 在本节中, $s=2$, 则学生成绩预测模型如下:

$$\boldsymbol{Y}=(\boldsymbol{Y}^{(1)},\boldsymbol{Y}^{(2)},\boldsymbol{Y}^{(3)}), \tag{4.3}$$

$$\boldsymbol{Y}^{(j)}=\sigma((\sigma(\boldsymbol{X}\boldsymbol{W}^{(1)(j)}+\boldsymbol{b}^{(1)(j)}))\boldsymbol{W}^{(2)(j)}+\boldsymbol{b}^{(2)(j)})+\boldsymbol{\varepsilon}^{(j)},\ \ j=1,2,3, \tag{4.4}$$

其中, $\boldsymbol{X}$是一个$N*4$维矩阵, $\boldsymbol{W}^{(1)(j)}$是一个4*3维矩阵, $\boldsymbol{W}^{(2)(j)}$是一个3*1维向量, $\boldsymbol{b}^{(1)(j)}$是一个N*3维矩阵, $\boldsymbol{b}^{(2)(j)}$是一个N*1维向量, σ为$sigmoid$激活函数. $\boldsymbol{\varepsilon}^{(j)}$ 是一个$N*1$维向量, 其中$\boldsymbol{\varepsilon}_{ik}^{(j)}\sim N(0,1)$, $i=1,2,\cdots,N$, $k=1$. 输出$\boldsymbol{Y}$是一个N*3维矩阵.

对此预测模型, 经过调参可以得到最优的超参数如表10.

表10 成绩预测实例的超参数

(Table 10 The hyperparameters for grade prediction instances)

超参数 评价指标	MSE	MAE
λ	0.0418	0.0367
λ^*	0.00359	0.00321
θ	0.026	0.0245
θ^*	0.364	0.392

将三个任务的自适应软参数多任务学习和软参数多任务学习做对比, 且估计效率的衡量仍采用MSE和MAE. 由表11可知, 当评价指标为MSE时, 自适应软参数多任务学习比软参数多任务学习多了17s, 自适应软参数多任务学习比软参数多任务学习的效果提升了4.5%. 当评价指标为MAE时, 自适应软参数多任务学习比软参数多任务学习多了21s, 自适应软参数多任务学习比软参数多任务学习的效果提升了3.9%.

表11 两种模型的损失函数值(10^{-1}), 耗时(s)及性能提升值(%)

(Table 11 The loss function values(10^{-1}), time consumption(s) and performance improvement values of two models(%))

	软参数	自适应	时间差	性能提升值
MSE	0.715	0.683	17	4.5
MAE	4.926	4.735	21	3.9

图7是高中学生综合成绩数据软参数多任务与自适应软参数多任务学习的MSE变化对比图, 图7中的标记点与模拟数据时的标记点含义一样. 从图7中可以看出对于学生成绩数据, 在软参数多任务学习达到其损失函数的最低点后, 通过自适应软参数多任务学习仍可以进一步优化, 使得损失函数进一步下降, 说明了自适应软参数多任务学习模型良好的泛化能力.

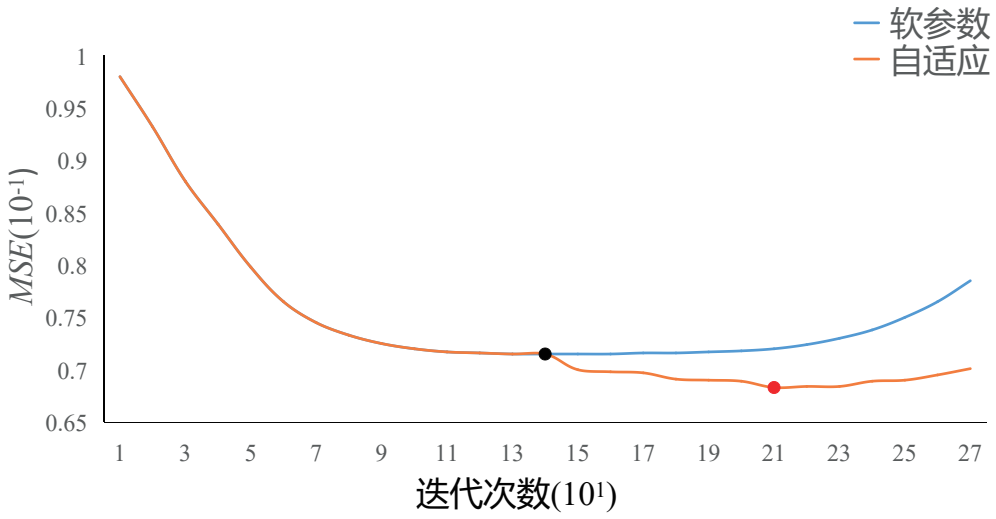

图7 高中学生综合成绩数据软参数多任务与自适应软参数多任务学习的MSE变化对比图

(Figure 7 Comparison chart of the MSE variation between soft parameter multi-task and adaptive soft parameter multi-task learning for comprehensive score data of high school students)

5 结论与展望

本文基于多任务深度学习方法, 结合软参数共享网络, 给出一种自适应软参数共享的多任务学习方法. 在正则项中采用L_2范数进行均值约束的基础上, 设置软参数正则项系数λ, 自适应正则项系数λ^*, 自适应参数衰减比例θ, 结合$Adam$优化算法, 得到自适应参数衰减比例θ^*, 将软参数多任务学习自适应地动态转化为多个单任务和软参数多任务联合学习. 在充分利用了多任务学习的优势后, 进一步减少了软参数多任务学习中的负迁移现象.

数值模拟的结果显示, 本文给出的自适应软参数多任务学习方法比单任务学习和软参数多任务学习方法拥有更好预测性能. 实际例子也验证了自适应软参数多任务学习方法的有效性和可行性, 代价仅仅是少量的时间. 模拟与实例的自适应软参数多任务学习方法与软参数多任务学习方法的MSE变化对比图详细描述了自适应软参数多任务学习方法在后期迭代学习的变化过程.

本文对于多个任务采用自适应软参数多任务学习方法提出了原理依据, 该方法能进一步降低损失函数, 达到减少负迁移现象的显著优势. 在未来的研究中可将自适应软参数多任务学习方法用到其它形式的软参数正则项约束当中, 对于自适应软参数多任务学习方法的理论证明, 是我们继续研究的方向.

参 考 文 献

[1] Zhang Y, Yang Q. An overview of multi-task learning. *National Science Review*, 2018, **5**(1): 30–43.

[2] 张钰, 刘建伟, 左信. 多任务学习. 计算机学报, 2020, **43**(07): 1340–1378.
(Zhang Yu, Liu Jianwei, Zuo Xin. Multi-task learning. *Chinese Journal of Computers*, 2020, **43**(07): 1340–1378.)

[3] Caruana R. Multitask learning: A knowledge-based source of inductive bias1. *Proceedings of the Tenth International Conference on Machine Learning*, 1993: 41–48.

[4] Collobert R, Weston J. A unified architecture for natural language processing: Deep neural networks with multitask learning. *Proceedings of the 25th International Conference on Machine Learning*, 2008: 160–167.

[5] Long M, Cao Z, Wang J, et al. Learning multiple tasks with multilinear relationship networks. *Advances in Neural Information Processing Systems*, 2017, **30**: 1593–1602.

[6] Ruder S, Bingel J, Augenstein I, et al. Latent multi-task architecture learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019, **33**(01): 4822–4829.

[7] 冯兴杰, 赵新阳, 冯小荣. 基于软参数共享的事件联合抽取方法. 计算机应用研究, 2023, **40**(01): 91–96.
(Feng Xingjie, Zhao Xinyang, Feng Xiaorong. Event joint extraction method based on soft parameter sharing. *Computer Applications Research*, 2023, **40**(01): 91–96.)

[8] Duong L, Cohn T, Bird S, et al. Low resource dependency parsing: Cross-lingual parameter sharing in a neural network parser. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, 2015, **2**: 845–850.

[9] Misra I, Shrivastava A, Gupta A, et al. Cross-stitch networks for multi-task learning. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016: 3994–4003.

[10] Mrini K, Dernoncourt F, Yoon S, et al. A gradually soft multi-task and data-augmented approach to medical question understanding. *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, 2021, **1**: 1505–1515.

[11] Lee H B, Yang E, Hwang S J. Deep asymmetric multi-task feature learning. *International Conference on Machine Learning*, 2018, **80**: 2956–2964.

[12] 王先兰, 周金坤, 穆楠等. 基于多任务联合学习的跨视角地理定位方法. 计算机应用, 2023, **43**(05): 1625–1635.
(Wang Xianlan, Zhou Jinkun, Mu Nan, et al. Cross-view geolocation method based on multi-task joint learning. *Computer Applications*, 2023, **43**(05): 1625–1635.)

[13] 郭辉, 郭静纯. 基于混合共享机制的多任务深度学习方法. 计算机工程与设计, 2023, **44**(02): 556–562.
(Guo Hui, Guo Jingchun. Multi-task deep learning method based on hybrid sharing mechanism. *Computer Engineering and Design*, 2023, **44**(02): 556–562.)

[14] 徐薇, 骆剑平, 李霞等. 基于相关性学习的多任务模型及其应用. 深圳大学学报(理工版) 2023, 1–10.
(Xu Wei, Luo Jianping, Li Xia, et al. Multi-task models and their applications based on correlation learning. *Journal of Shenzhen (University Science and Engineering)*, 2023: 1–10.)

[15] Li A, Wu Z, Lu H, et al. Collaborative self-regression method with nonlinear feature based on multi-task learning for image classification. *IEEE Access*, 2018, **6**: 43513–43525.

[16] 屈武, 阎高伟. 集成最大均值差异正则约束的迁移子空间软测量. 重庆理工大学学报(自然科学), 2020, **34**(04): 108–114
(Qu Wu, Yan Gaowei. Soft measurement with integrated maximum mean discrepancy regularization constraint in transfer subspace. *Journal of Chongqing University of Technology (Natural Science)*, 2020, **34**(04): 108–114.)

[17] Chen Z, Badrinarayanan V, Lee C Y, et al. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. *International Conference on Machine Learning*, 2018: 794–803.

[18] Jais I K M, Ismail A R, Nisa S Q. Adam optimization algorithm for wide and deep neural network. *Knowledge Engineering and Data Science*, 2019, **2**(01): 41–46.

[19] 李云祯, 周平, 陈军辉, 等. 成都市空气质量指数与雾霾的关系研究. 生态环境学报, 2016, **25**(11): 1760–1766
(Li Yunzhen, Zhou Ping, Chen Junhui, et al. Study on the relationship between air quality index and haze in chengdu city. *Journal of Ecology and Environment*, 2016, **25**(11): 1760–1766.)

[20] 程美英, 钱乾, 倪志伟, 等. 基于虚拟多任务二元粒子群算法和分形维数的雾霾天气预测方法. 系统科学与数学, 2018, **38**(05): 623–637.
(Cheng Meiying, Qian Qian, Ni Zhiwei, et al. Haze weather prediction method based on virtual multi-task binary particle swarm algorithm and fractal dimension. *Journal of Systems Science and Mathematical Sciences*, 2018, **38**(05): 623–637.)

[21] Zhu Y. Reading matters more than mathematics in science learning: An analysis of the relationship between student achievement in reading, mathematics, and science. *International Journal of Science Education*, 2022, **44**(01): 1–17.