计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2015.20148280
52(11):2545-2554, 2015

面向大数据流的多任务加速在线学习算法

李志杰 李元香 王 峰 匡 立
(软件工程国家重点实验室(武汉大学) 武汉 430072)
(lzj0019@163.com)

Accelerated Multi-Task Online Learning Algorithm for Big Data Stream

Li Zhijie, Li Yuanxiang, Wang Feng, and Kuang Li
(*State Key Laboratory of Software Engineering* (*Wuhan University*), *Wuhan* 430072)

Abstract Conventional machine learning and data mining techniques with batch computing mode suffer from many limitations when being applied to big data stream analytics tasks. Multi-task online learning framework with stream computing mode is a promising tool for big data stream analysis. However, current multi-task online learning algorithm has low convergence rate, such as $O(1/\sqrt{T})$ up to the T-th iteration, and its low convergence rate has become a bottleneck of online algorithm performance. In this paper, we propose a novel multi-task accelerated online learning algorithm, called ADA-MTL(accelerated dual averaging method for multi-task learning), which simultaneously obtains low computational time complexity and optimal convergence rate $O(1/T^2)$. The proof of a closed-form solution theorem which efficiently updates the weight matrix $\boldsymbol{W}_t$ at each iteration is provided, and detailed theoretical analysis for the algorithm convergence rate is conducted. The experimental results on real-world datasets demonstrate the merits of the proposed multi-task accelerated online learning algorithm for large-scale dynamic data stream problems. Since this multi-task accelerated online learning algorithm can obviously improve the real-time performance and the scalability for big data stream analysis, it is a realistic method for big data stream analytics tasks.

Key words big data stream; multi-task; accelerated; online learning; convergence analysis

摘 要 多任务在线学习框架采用直接数据处理的流式计算模式,是大数据流分析很有前途的一种工具.然而目前的多任务在线学习算法收敛率低,仅为 $O(1/\sqrt{T})$,T 为算法迭代次数.提出一种新颖的多任务加速在线学习算法 ADA-MTL(accelerated dual averaging method for multi-task learning),在保持多任务在线学习快捷计算优势的基础上,达到最优收敛率 $O(1/T^2)$.对多任务权重学习矩阵 $\boldsymbol{W}_t$ 的迭代闭式解表达式进行了推导,对提出算法的收敛性进行了详细的理论分析.实验表明,提出的多任务加速在线学习算法能够更好地保障大数据流处理的实时性和可伸缩性,有较广泛的实际应用价值.

关键词 大数据流;多任务;加速;在线学习;收敛分析

中图法分类号 TP181

云计算、物联网、社交网络等新兴信息技术和应用模式的快速发展,越来越多的软件系统部署在以互联网为代表的开放、动态、难控和不确定的变化环境,产生了急剧增加的大数据流.大数据流以其实时性、

收稿日期:2014-11-24;**修回日期**:2015-03-11
基金项目:国家自然科学基金项目(61070009,61103125);国家"八六三"高技术研究发展计划基金项目(2007AA01Z290)
通信作者:李元香(yxli@whu.edu.cn)

无序性、无限性、易失性、突发性等显著特征,使得其与传统批量大数据在数据计算的要求、方式等方面有着明显的不同[1].大数据的批量计算首先进行数据的存储,然后再对存储的静态数据进行集中计算.Hadoop是典型的大数据批量计算框架,由Hadoop分布式文件系统(Hadoop distributed file system, HDFS)负责静态数据的存储,并通过MapReduce将计算逻辑分配到各数据节点进行数据计算和价值发现.而在流式计算中,无法确定数据的到来时刻和到来顺序,也无法将全部数据存储起来.因此,不再进行流式数据的存储,而是当流动的数据到来后在内存中直接进行数据的实时计算.如Twitter的Storm、Yahoo的S4就是典型的流式数据计算框架.因此,如何通过创新性的数据分析方法实现对大数据流的快速、高效、及时地分析与计算是大数据分析技术领域面临的新挑战[1-2].

在围绕实时大数据流处理这一需求展开的研究中,在线机器学习算法采用数据流直接处理的模式,每次迭代处理一个随机流数据,学习变量的迭代更新只经过简单的计算,从而在实时性和准确率之间取得一个平衡,是解决该难题很有前途的方案,特别适合于训练流式大数据[3].

研究表明[3-14],许多大数据实际应用领域,如金融时间序列预测、自然语言处理、生物信息学多任务数据集分析等,同时学习多个相关的任务获取共享信息模式要优于单个任务分别学习的方式.还有一些应用,如协同过滤的用户特征矩阵优化与物品特征矩阵优化、矩阵分类等,优化对象本身是矩阵变量,直接类同于多任务学习算法的权重矩阵变量的求解.因此,近年来许多相关研究侧重于多任务学习框架,如文献[3-14].不过这些工作大多是批训练模式,在线多任务学习框架的研究工作很少.

文献[15]提出的正则化对偶平均单任务学习方法(regularized dual averaging method for single task learning, DA-STL),以凸函数优化为基础,通过迭代计算过去所有损失函数梯度值的平均值,降低优化目标表达式的求解复杂度,达到高效更新权重向量的目的,是单任务在线学习领域的标志性研究工作.基于正则化对偶平均思想,Yang等人[3,14]提出了一种多任务的在线学习框架并应用于多任务特征选择DA-MTFS(dual averaging based multi-task feature selection).DA-MTFS算法收敛率为$O(1/\sqrt{T})$,每次迭代时间与空间复杂度均为$O(dQ)$.其中,T为算法迭代次数,d为数据特征维度,Q是任务个数.显然,该算法依然存在在线算法面临的普遍问题:低计算复杂性常常与算法的低收敛率联系在一起.

加速在线学习方面,文献[15]提出了正则化对偶平均在线学习框架并讨论了加速分支的情况.文献[16]提出了随机优化与在线学习的加速梯度下降方法.不过,这些算法的加速效果取决于参数的满足条件,极端情况下才能获得最优收敛率$O(1/T^2)$,并且讨论的都是单任务在线算法的加速效果.

文献[17]提出了一种迹范数最小化加速梯度下降方法,在理论上证明了多任务学习算法的最优收敛率可以达到$O(1/T^2)$.不过,该算法是基于批处理模式,并且由于应用了矩阵的奇异值分解(singular value decomposition, SVD),每次迭代的昂贵计算代价使得该方法并不适用于大规模数据场景.

本文结合使用加速技术和正则化对偶平均方法,提出一种新颖的多任务加速在线学习算法,算法计算高效且加速收敛.迭代更新权重学习矩阵的闭式解计算表达式,以及算法的收敛率分析,均有完整的数学推导过程.多任务加速在线学习算法可以很方便地应用于金融时间序列预测、在线推荐系统协同过滤等动态多变的大数据流场景中,实验结果表明它能显著提升大规模数据流处理的实时性和可伸缩性,是有效学习大数据流的一种很有前途的工具.

1 多任务在线学习框架

1.1 问题构建

定义1. 多任务学习(multi-task learning, MTL).假设有Q个任务,它们的数据来自于同一空间$X\times Y$,其中,$X\subset\mathbb{R}^d$,$Y\subset\mathbb{R}$.每个任务有N_q个数据点$D_q=\{\boldsymbol{z}_i^q=(\boldsymbol{x}_i^q,y_i^q)\}_{i=1}^{N_q}$,它们组成多任务数据集$D=\bigcup_{q=1}^{Q}D_q$.$D_q$取样于$X\times Y$空间的分布$P_q$,每个任务的$P_q$各不相同,但这些不同任务的$P_q$是相关的.多任务学习的目标是学习$Q$个函数$f_q:\mathbb{R}^d\to\mathbb{R}$,$q=1,2,\cdots,Q$,使得$f_q(\boldsymbol{x}_i^q)$估计$y_i^q$.当$Q=1$,它是标准的单任务学习问题.

通常,在多任务学习模型中,第q个任务的决策函数f_q假定是以模型权重向量$\boldsymbol{w}_q$为参数的一个超平面,即$f_q(\boldsymbol{x})=\boldsymbol{w}_q^{\mathrm{T}}\boldsymbol{x}$,$q=1,2,\cdots,Q$.$Q$个权重向量$\boldsymbol{w}_q$组成大小$d\times Q$的矩阵$\boldsymbol{W}$,

$$\boldsymbol{W}=(\boldsymbol{w}_1,\boldsymbol{w}_2,\cdots,\boldsymbol{w}_Q)=$$
$$(\boldsymbol{W}_{\cdot 1},\boldsymbol{W}_{\cdot 2},\cdots,\boldsymbol{W}_{\cdot Q})=(\boldsymbol{W}_{1\cdot}^{\mathrm{T}},\boldsymbol{W}_{2\cdot}^{\mathrm{T}},\cdots,\boldsymbol{W}_{d\cdot}^{\mathrm{T}})^{\mathrm{T}},$$

这里,d 表示数据特征维度.

权重矩阵 $\boldsymbol{W}$ 是多任务学习的目标.多任务学习是通过最小化经验风险以及权重的正则化项来完成学习权重矩阵 $\boldsymbol{W}$ 的目标.批训练方式的多任务学习目标 $\boldsymbol{W}$ 优化表达式如下:

$$\min_{\boldsymbol{W}} \varphi(\boldsymbol{W}) = \sum_{q=1}^{Q} \frac{1}{N_q} \sum_{i=1}^{N_q} l^q(\boldsymbol{W}_{\cdot q}, \boldsymbol{z}_i^q) + \Omega_\lambda(\boldsymbol{W}), \quad (1)$$

其中,$\lambda \geqslant 0$ 是平衡损失与正则化项的一个常数,$l^q(\boldsymbol{W}_{\cdot q}, \boldsymbol{z}_i^q)$定义第 q 个任务的样本点 $\boldsymbol{z}_i^q = (\boldsymbol{x}_i^q, y_i^q)$的损失.本文损失函数设定为平滑可导的强凸函数,

$$l(\boldsymbol{w}, \boldsymbol{z}) = \frac{1}{2}[y - \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}]^2. \quad (2)$$

基于对偶平均的在线算法,采用直接数据处理的流式计算模式,多任务学习目标 $\boldsymbol{W}$ 优化表达式如下:

$$\min_{\boldsymbol{W}} \phi(\boldsymbol{W}) = \left\{ \langle \bar{\boldsymbol{G}}_t, \boldsymbol{W} \rangle + \Omega_\lambda(\boldsymbol{W}) + \frac{\beta_t}{t} h(\boldsymbol{W}) \right\}, \quad (3)$$

其中,对偶变量 $\boldsymbol{G}_t = \partial l_t$ 表示第 t 次迭代的损失函数 l_t 对原变量 $\boldsymbol{W}$ 的偏梯度;$\bar{\boldsymbol{G}}_t$ 则表示第 1 步到第 t 步迭代的偏梯度平均值;$h(\boldsymbol{W})$是附加的强凸函数,$\{\beta_t\}_{t \geqslant 1}$是非递减的非负输入序列.

因此,对偶平均在线算法的每次迭代过程主要由 3 步组成:1)计算损失函数关于权重的偏梯度;2)计算偏梯度的平均值;3)基于平均偏梯度计算更新权重变量.

对于式(2)的损失函数,单个任务偏梯度计算式如下:

$$\partial l_t(\boldsymbol{w}) = \partial l(\boldsymbol{w}, \boldsymbol{z}_t) = (\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_t - y_t)\boldsymbol{x}_t, \quad (4)$$

其中,$\boldsymbol{w}$ 表示这个任务的权重向量,$(\boldsymbol{x}_t, y_t)$代表该任务在第 t 次迭代时的特征向量和响应值.

1.2 基于对偶平均的多任务特征选择在线算法

Yang 等人在文献[3]提出的基于对偶平均的多任务特征选择在线算法 DA-MTFS 是典型的多任务在线学习框架,为便于比较与启发思路,我们在算法 1 中概述其基本思想.

算法 1 中,$\|\cdot\|_{\mathrm{F}}$ 表示矩阵的 Frobenius 范数,正则化项 $\Omega_\lambda(\boldsymbol{W})$取权重矩阵 $\boldsymbol{W}$ 混合范数 $L_{1/2,1}$,由 $\boldsymbol{W}$ 的 $L_{1,1}$范数$\|\boldsymbol{W}_{j\cdot}^{\mathrm{T}}\|_1$ 和 $L_{2,1}$ 范数$\|\boldsymbol{W}_{j\cdot}^{\mathrm{T}}\|_2$线性组合而成,

$$\Omega_\lambda(\boldsymbol{W}) = \lambda \sum_{j=1}^{d} (c\|\boldsymbol{W}_{j\cdot}^{\mathrm{T}}\|_1 + \|\boldsymbol{W}_{j\cdot}^{\mathrm{T}}\|_2), \quad (5)$$

其中,c 是控制任务选择稀疏性的一个常数.$\Omega_\lambda(\boldsymbol{W})$中的 $L_{1/2,1}$混合范数具有特征与任务选择的功能.算法 1 的权重矩阵 $\boldsymbol{W}$ 更新优化表达式(6),由多任务目标优化式(3)的输入序列取 $\beta_t = \gamma\sqrt{t}$ 得到.算法采用流式计算模式,数据在内存直接处理,这就要求优化式(6)的求解简单化,从而获得在线算法急需的计算效率.根据优化式(6),推导出了权重矩阵 $\boldsymbol{W}$ 更新的闭式解(closed-form solution)计算表达式,这样,优化式(6)每次更新计算 $\boldsymbol{W}$ 的时间变为常数.算法 1 的一次迭代时间复杂度为 $O(dQ)$,收敛率为 $O(1/\sqrt{T})$,其基本特征为计算高效但收敛率低[3].

算法 1. 多任务特征选择在线学习框架 DA-MTFS.

输入:

$$\boldsymbol{W}_0 = \arg\min_{\boldsymbol{W}} h(\boldsymbol{W}), h(\boldsymbol{W}) = \frac{1}{2}\|\boldsymbol{W}\|_{\mathrm{F}}^2,$$

给定常数 $\lambda > 0, \gamma > 0$;

输出:$\boldsymbol{W}_{t+1}$.

初始化:令 $\boldsymbol{W}_1 = \boldsymbol{W}_0, \bar{\boldsymbol{G}}_0 = \boldsymbol{0}$.

for $t = 1, 2, \cdots$ do

1) Q 个任务依次出现一个实例 $\boldsymbol{z}_t^q, q = 1, 2, \cdots, Q$,根据式(4)分别计算 $\boldsymbol{g}_t^q = \partial l(\boldsymbol{w}^q, \boldsymbol{z}_t^q)$值,构成一个 $d \times Q$ 矩阵,$\boldsymbol{G}_t = (\boldsymbol{g}_t^1, \boldsymbol{g}_t^2, \cdots, \boldsymbol{g}_t^Q)$;

2) 计算偏梯度的平均值:

$$\bar{\boldsymbol{G}}_t = \frac{t-1}{t}\bar{\boldsymbol{G}}_{t-1} + \frac{1}{t}\boldsymbol{G}_t;$$

3) 基于 $\bar{\boldsymbol{G}}_t$ 计算更新下次迭代的权重矩阵 $\boldsymbol{W}_{t+1}$:

$$\boldsymbol{W}_{t+1} = \arg\min_{\boldsymbol{W}} \phi(\boldsymbol{W}),$$

$$\phi(\boldsymbol{W}) = \langle \bar{\boldsymbol{G}}_t, \boldsymbol{W} \rangle + \Omega_\lambda(\boldsymbol{W}) + \frac{\gamma}{\sqrt{t}} h(\boldsymbol{W}); \quad (6)$$

end for

2 加速的多任务在线学习算法

2.1 加速的多任务在线学习框架

我们在正则化对偶平均方法框架内,结合使用加速技术的微批量方法(mini-batch approach),提出一种新颖的多任务加速在线学习框架 ADA-MTL(accelerated dual averaging method for multi-task learning).

算法 2. 加速的多任务在线学习框架 ADA-MTL.

输入:

$$\boldsymbol{W}_0 = \arg\min_{\boldsymbol{W}} h(\boldsymbol{W}), h(\boldsymbol{W}) = \frac{1}{2}\|\boldsymbol{W}\|_{\mathrm{F}}^2,$$

给定常数 $\lambda > 0$;

输出:$\boldsymbol{W}_t$.

初始化:令 $\boldsymbol{U}_1=\boldsymbol{W}_0$,$\bar{\boldsymbol{G}}_0=\boldsymbol{0}$,$\alpha_1=1$.

for $t=1,2,\cdots$ do

1) Q 个任务依次出现一个实例 $\boldsymbol{z}_t^q$,$q=1,2,\cdots,Q$,计算损失函数偏梯度在查询点 $\boldsymbol{U}_t$ 的值 $\boldsymbol{G}_t$;

2) 计算偏梯度的平均值:$\bar{\boldsymbol{G}}_t=\frac{t-1}{t}\bar{\boldsymbol{G}}_{t-1}+\frac{1}{t}\boldsymbol{G}_t$;

3) 输出:

$$\boldsymbol{W}_t=\arg\min_{\boldsymbol{W}}\phi(\boldsymbol{W},\boldsymbol{U}_t)=\left\{\langle\bar{\boldsymbol{G}}_t,\boldsymbol{W}\rangle+\Omega_\lambda(\boldsymbol{W})+\frac{1}{2}\|\boldsymbol{W}-\boldsymbol{U}_t\|_{\mathrm{F}}^2\right\};\quad(7)$$

4) 更新输入序列:

$$\alpha_{t+1}=\frac{1+\sqrt{1+4\alpha_t^2}}{2};\quad(8)$$

5) 计算查询点 $\boldsymbol{U}_{t+1}$:

$$\boldsymbol{U}_{t+1}=\boldsymbol{W}_t+\frac{\alpha_t-1}{\alpha_{t+1}}(\boldsymbol{W}_t-\boldsymbol{W}_{t-1});\quad(9)$$

end for

比较算法 1 和算法 2,不难看出,算法 2 增加了一个查询点 $\boldsymbol{U}_t$,每次迭代计算的 $\boldsymbol{G}_t$ 是损失函数偏梯度在查询点 $\boldsymbol{U}_t$ 的值.查询点由式(9)调整,其中 α_t 是由式(8)产生的输入序列,其作用是使查询点 $\boldsymbol{U}_t$ 尽量靠近权重矩阵 $\boldsymbol{W}_t$.必须指出,与算法 1、文献[15]的加速算法等其它算法不同的是,我们在算法 2 的优化式(7)中,附加的 $h(\boldsymbol{W})$ 强凸函数形式已由 $\frac{1}{2}\|\boldsymbol{W}\|_{\mathrm{F}}^2$ 改为 $\frac{1}{2}\|\boldsymbol{W}-\boldsymbol{U}_t\|_{\mathrm{F}}^2$.因为,这时不仅需要增加优化函数的凸性,更重要的是,通过把 $\frac{1}{2}\|\boldsymbol{W}-\boldsymbol{U}_t\|_{\mathrm{F}}^2$ 融入算法的优化目标,使得优化变量 $\boldsymbol{W}$ 与查询点 $\boldsymbol{U}_t$ 不断靠近,从而获得算法持续加速的必要条件.因此,我们在算法 2 中采用的加速技术是一种改进的微批量方法,加速效果更好.

2.2 求解 $\boldsymbol{W}_t$ 的闭式解

算法 2 的计算复杂度主要取决于优化式(7)权重矩阵 $\boldsymbol{W}_t$ 的求解.我们通过推导出优化式(7)的 $\boldsymbol{W}_t$ 闭式解,高效迭代更新权重矩阵 $\boldsymbol{W}_t$.

定理 1. 已知迭代 t 对偶平均梯度 $\bar{\boldsymbol{G}}_t$,正则化项 $\Omega_\lambda(\boldsymbol{W})=\lambda\sum_{j=1}^{d}(c\|\boldsymbol{W}_{j\cdot}^{\mathrm{T}}\|_1+\|\boldsymbol{W}_{j\cdot}^{\mathrm{T}}\|_2)$,如式(5)所示.令 $(\bar{\boldsymbol{R}}_{j,q})_t=[|(\bar{G}_{j,q})_t-(u_q)_t|-\lambda c]_+\cdot\mathrm{sgn}((\bar{G}_{j,q})_t-(u_q)_t)$,$j=1,2,\cdots,d$,$q=1,2,\cdots,Q$.则算法 2 可用式(10)计算的闭式解来高效更新优化式(7)的目标 $\boldsymbol{W}_t$,其中,运算 $[r]_+=r$,$r>0$;$[r]_+=0$,$r\leqslant 0$.

$$(\boldsymbol{W}_{j\cdot})_t=-\left[1-\frac{\lambda}{\|(\bar{\boldsymbol{R}}_{j\cdot})_t\|_2}\right]_+\cdot(\bar{\boldsymbol{R}}_{j\cdot})_t.\quad(10)$$

定理 1 的证明见附录 A.

由于定理 1 能够为算法 2 的优化式(7)提供 $\boldsymbol{W}_t$ 每次迭代更新的闭式解,这样,优化式(7)每次更新计算 $\boldsymbol{W}_t$ 的时间变为常数.同时,对偶偏梯度平均值 $\bar{\boldsymbol{G}}_t$ 充分利用上次迭代计算与存储过的平均值 $\bar{\boldsymbol{G}}_{t-1}$,计算简单.算法 2 每次迭代只需 $O(dQ)$ 空间来存储要求的信息:平均偏梯度和权重矩阵.算法 2 每次迭代的时间复杂度也是 $O(dQ)$,d 为数据特征维度,Q 是任务个数.因此,算法 2 与算法 1 一样,计算高效.与算法 1 不同的是,算法 2 能够加速收敛,通过 2.3 节的算法收敛理论分析,我们证明算法 2 的收敛率可以达到最优值 $O(1/T^2)$.

2.3 算法收敛理论分析

正则化随机多任务学习模型是对随机多任务样本 $\boldsymbol{Z}$ 的期望损失最小化加上正则化项 Ω_λ,

$$\min_{\boldsymbol{W}}\varphi(\boldsymbol{W})=E_{\boldsymbol{Z}}l(\boldsymbol{W},\boldsymbol{Z})+\Omega_\lambda(\boldsymbol{W}),\quad(11)$$

其中,$\varphi(\boldsymbol{W})$ 称为多任务学习的目标值.现实中,由于 $\boldsymbol{Z}$ 的分布一般难以确定,期望损失 $E_{\boldsymbol{Z}}l(\boldsymbol{W},\boldsymbol{Z})$ 常由经验平均值 $\sum_{q=1}^{Q}\frac{1}{N_q}\sum_{i=1}^{N_q}l^q(\boldsymbol{W}_{\cdot q},\boldsymbol{z}_i^q)$ 代替,如式(1)所示.而对于多任务对偶平均方法,期望损失则用 $\langle\bar{\boldsymbol{G}}_t,\boldsymbol{W}\rangle$ 代替如下:

$$\varphi(\boldsymbol{W})=\langle\bar{\boldsymbol{G}}_t,\boldsymbol{W}\rangle+\Omega_\lambda(\boldsymbol{W}).\quad(12)$$

定义 2. 多任务在线学习收敛率(convergence rate for multi-task online learning, CR-MTOL).多任务在线学习目标值 $\varphi(\boldsymbol{W})$ 定义如式(12),如果算法优化问题 $\min_{\boldsymbol{W}}\varphi(\boldsymbol{W})$ 存在最优解 $\boldsymbol{W}^*$,那么,该算法运行到第 T 步的收敛率为 υ_T,

$$\upsilon_T=\varphi(\boldsymbol{W}_T)-\varphi(\boldsymbol{W}^*).\quad(13)$$

显然,如果算法收敛快,则算法效率高,迭代次数少.

定理 2. 设算法 2 优化问题 $\min_{\boldsymbol{W}}\varphi(\boldsymbol{W})$ 的最优解是 $\boldsymbol{W}^*$,则算法 2 运行到第 T 次迭代的收敛率为 $\upsilon_T(T\geqslant 1)$,

$$\upsilon_T=\varphi(\boldsymbol{W}_T)-\varphi(\boldsymbol{W}^*)\leqslant\frac{2\|\boldsymbol{W}_0-\boldsymbol{W}^*\|_{\mathrm{F}}^2}{(T+1)^2},$$

即 $\upsilon_T=O(1/T^2)$.

定理 2 的证明见附录 B.

3 评价与应用

本节我们分析比较的方法包括本文提出的多任务加速在线学习算法 ADA-MTL、文献[3]的多任务在线特征选择算法 DA-MTFS、文献[15]的正则化对偶平均单任务在线学习方法 DA-STL 和文献[9]的多任务批训练学习算法 BT-MTL(batch-training method for multi-task learning). 对提出的多任务加速在线学习算法 ADA-MTL,研究大数据流背景下,在金融时间序列预测(financial time series prediction, FTSP)场景的应用算法 ADA-MTL 和在协同过滤的概率矩阵分解(probabilistic matrix factorization, PMF)在线学习场景的应用算法 ADA-MTL,分析评价相应的大数据流挖掘系统的实时性、可伸缩性、数据吞吐量等性能指标. 数据流抽取(data stream sampling)算法采用环形循环滑动窗口方法[18]. 实验环境为 2.80 GHz CPU,3.93 GB 内存 PC,所有算法在 Matlab 上运行.

3.1 时空代价分析

大数据流的数据规模大,到达速率非常快,要求数据流挖掘算法在有限的内存空间中实时处理,这就要求面向大数据流分析算法的时间、空间复杂度低.

假设有 Q 个任务,每个任务有 k 个 d 维的样本,文献[19]的多任务批处理算法 BT-MTL 的浮点操作数(flops)的界为 $O(dkQ+dQ)=O(dkQ)$. 由于我们这里考虑的训练样本是流数据,因此,当一个新样本出现时,批处理算法必须重新训练. 当每个任务的样本数达到 N 时,则总浮点操作数的界是:

$$O\left(\sum_{k=2}^{N} dkQ\right)=O(dQN^2). \tag{14}$$

该多任务批训练算法的存储代价界是:

$$O(dQN). \tag{15}$$

不同的是,在线算法 DA-MTFS 和 ADA-MTL 每次迭代的最坏时间复杂度为 $O(dQ)$,并且当样本一个个顺次出现时它能相应地更新,不必重新训练. 这样,当每个任务的样本数是 N,总浮点操作数是:

$$O(dQN). \tag{16}$$

多任务在线学习算法的存储代价界是:

$$O(dQ). \tag{17}$$

比较式(14)(15)和式(16)(17),多任务的在线学习算法相对于多任务批训练算法耗费的时空代价分别下降了一个数量等级. 图 1 画出了 4 个算法的运行时间-训练样本数的 log-log 对数关系图.

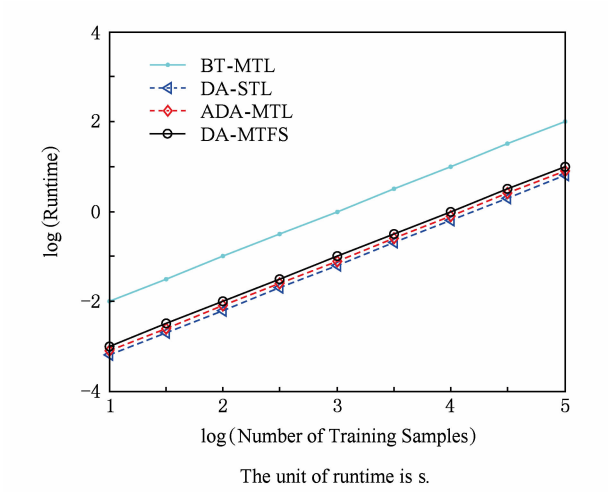

Fig. 1 The log-log plot of runtime on training samples.

图 1 运行时间-训练样本数的 log-log 对数关系图

3.2 收敛性

为了验证算法 2(ADA-MTL)相对于算法 1(DA-MTFS)的加速收敛效果,我们选择 4 个多任务数据集:yeast, letters, digits 和 DMOZ. yeast[20] 数据集由一个酵母基因分类问题得出,有 14 个任务;letters, digits[20] 分别是手写体单词(8 个任务)和数字数据集(10 个任务);DMOZ① 是文本分类数据集(10 个任务). 我们随机抽取 10%的各个任务的数据做训练样本,抽取数据流采用环形循环滑动窗口方法.

为了评估算法的效率,所有算法当目标值的相对变化小于 10^{-8} 时终止运行,所耗费的时间我们称之为收敛时间(convergence time). 图 2 比较了 2 个多任务在线算法 DA-MTFS 和 ADA-MTL 在 yeast, letters, digits 和 DMOZ 这 4 个数据集上分别进行 10 次随机试验的平均收敛时间. 我们观察到加速算法

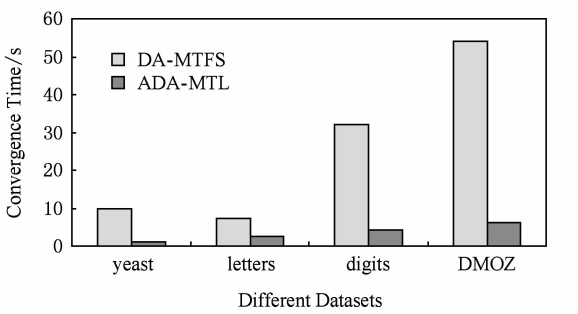

Fig. 2 Comparison of the convergence time.

图 2 算法 2 与算法 1 收敛时间的比较

① http://www.dmoz.org/

ADA-MTL 在 4 个数据集所需的收敛时间均大大小于算法 DA-MTFS,这是由于算法 ADA-MTL 收敛快,趋于稳定值的时间大大节省所致.

为了进一步观察比较算法 1 和算法 2 的收敛特性,图 3 画出了这 2 个多任务在线学习算法在 yeast 数据集上运行时的目标值迭代曲线.我们观察到算法 ADA-MTL 仅需 30 次左右迭代趋于稳定值,而算法 DA-MTFS 需 120 次左右的迭代才能收敛至稳定值.

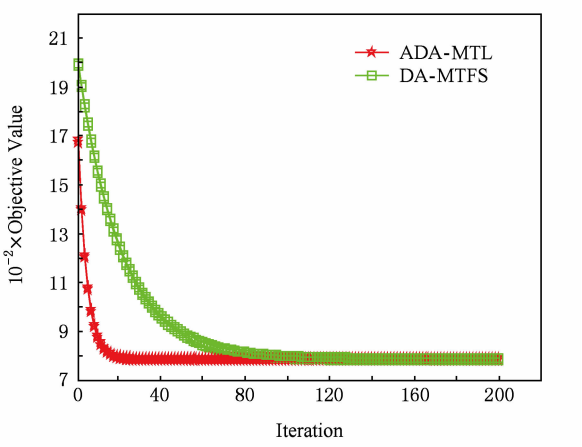

Fig. 3 Convergence of algorithms on yeast data set.

图 3 算法 1 与算法 2 在 yeast 数据集的收敛性比较

算法 ADA-MTL 的加速收敛与低时间复杂度特性,带来优异的实时性能.假设数据流挖掘系统要求新产生的数据流在 T 时间内处理完毕并反馈处理结果,我们称之为实时度 T.显然,实时度 T 越小,对算法的实时处理能力要求越高.如果数据流处理能力不能满足实时度 T 的要求,那么某些流数据会因为不能得到及时处理而丢失.我们观察到当实时度 T 低于某个阈值,例如 $T=0.5$ s,算法 DA-MTFS,尤其是算法 DA-STL,流数据实际平均处理率 P 开始显著下降;而这时算法 ADA-MTL 基本不变,维持近 100%的流数据处理率.这说明算法 ADA-MTL 的实时性能明显好于算法 DA-MTFS,尤其是算法 DA-STL.算法 ADA-MTL 优良的实时性能确保了它适应随机动态的大数据流场景的能力.

3.3 金融数据流回归分析

金融数据分析是一种典型大数据流分析.股票价格波动预测,属于大数据回归分析的范畴,是金融领域研究的焦点问题之一.我们将多任务在线回归分析算法应用于股市预测当中,并对 ADA-MTL,DA-MTFS,DA-STL,BT-MTL 等算法在金融时间序列预测中的性能进行比较,以验证我们提出的模型与在线算法 ADA-MTL 在金融数据流回归分析中的特性与优势.

中国股市自 1990-12-19 正式营业至今已有 20 多年的历史,交易数据量巨大.我们选取上证综合指数和四川长虹、民生银行等个股进行实例研究,每个实例视为一个任务,共 100 个任务.股票历史交易数据和实时交易数据从新浪财经网下载.训练数据集选取 2009-04-13—2010-03-30,一共 240 个交易日的数据,测试集选取 2010 年第 2,3 季度的交易日的数据,一共 120 个交易日的数据.使用证券交易信息作为任务的数据特征,具体信息如表 1 所示:

Table 1 Data Feature

表 1 数据特征信息表

Index	Feature Name	Index	Feature Name
1	Transdate	16	EMA12
2	Openprice	17	EMA26
3	Lastprice	18	Avg5
4	Topprice	19	Avg10
5	Lowprice	20	Avg20
6	DIFF	21	Avg30
7	DEA	22	BBI
8	MACD	23	UPR
9	K	24	DWN
10	D	25	No
11	J	26	EMPMA5
12	WR1	27	EMPMA10
13	WR2	28	EMPMA20
14	WR3	29	EMPMA30
15	Volume	30	EMPMA60

下载的原始数据集各特征值先进行预处理,即归一化处理,将其缩放到[-1,1]之间,再抽取数据流.算法参数可以利用 LIBSVM 工具的 gridregression.py 函数进行参数寻优过程.实验的方案是多任务流数据在线学习,训练样本顺次到达,算法迭代时每个任务都有一个待处理的样本.表 2 列出各比较算法在选定数据集上回归分析的性能平均值,这里使用预测误差 $RMSE$(root mean square error)和权重矩阵的 NNZs(number of non-zeros)值以及数据吞吐量(throughout)来衡量模型与算法性能.其中,预测误差 $RMSE$ 是真实值与预测值误差的平均平方根.2 个无数据的单元格表示这些项的数据意义不明显.

Table 2 *RMSE* and NNZs of the Regression Analysis

表 2 选定数据集上回归分析的 *RMSE* 和 NNZs 值

Algorithm	*RMSE*	NNZs	Throughout/number per second	Parameters	
				λ	c
BT-MTL	2.15	2981		4.2	0.01
DA-STL	2.16		151	0.8	0.01
DA-MTFS	**1.81**	**2059**	185	6.9	0.01
ADA-MTL	**1.81**	2129	**204**	6.9	0.01

图 4 给出了算法 ADA-MTL 的预测性能 *RMSE* 值与任务个数的关系.

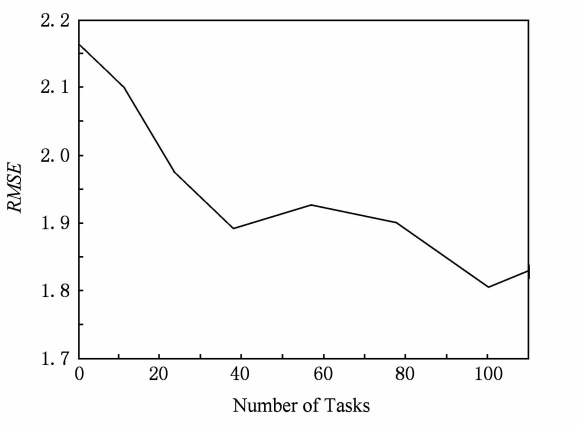

Fig. 4 *RMSE* of ADA-MTL vs number of tasks.

图 4 算法 ADA-MTL 的预测性能 *RMSE* 值与任务个数的关系

从表 2 和图 4 我们观察到:

1) 多任务学习方式($Q=100$)比各个任务单独学习方式的回归预测性能要好 20%左右.

2) 在线学习算法和相应批处理算法性能相近.

3) 算法 ADA-MTL 回归预测性能取得最好值,并且 NNZs 值仅高于算法 DA-MTFS,权重矩阵解的稀疏性较强.

4) 表 2 第 4 列是在数据流速 210 个/s、实时度 $T=0.3$ s 条件下,3 个在线算法的流数据吞吐量.算法 ADA-MTL 吞吐量最高,接近于数据流速;算法 DA-MTFS 次之;算法 DA-STL 最低.

3.4 协同过滤在线学习

协同过滤(collaborative filtering, CF)将用户评价物品作为一种偏好指示,根据已知的某些用户的偏好预测用户的未知偏好,是构建推荐系统的一种主要方式.随着 MovieLens, Yahoo!Music, Amazon 等大规模在线用户贡献网站和在线购物网站的出现,用户和物品库急剧扩大,系统始终处于随机获取新的三元组(u,i,r),即新的用户、新的物品和新的评价的动态场景中,呈现大数据流式计算的特征.这里,我们使用的协同过滤技术是 PMF[21-22].PMF 的目标是学习 2 个低秩矩阵,用户特征矩阵 $\boldsymbol{U}$ 和物品特征矩阵 $\boldsymbol{V}$,使用 $\boldsymbol{U}^{\mathrm{T}}\boldsymbol{V}$ 拟合评价矩阵 $\boldsymbol{R}$.因此,PMF 的优化矩阵是用户特征矩阵 $\boldsymbol{U}$ 和物品特征矩阵 $\boldsymbol{V}$,对应于算法 2 的权重矩阵 $\boldsymbol{W}$,可以分别使用算法 2 学习优化.我们选择数据集 Yahoo! Music① 和 MovieLens② 研究所比较算法的实验性能.Yahoo! Music 是一个很大的数据集,包含超过 25 000 万的评分值;但这个数据集非常稀疏,只有 0.4%的单元是已知的.表 3 给出了这 2 个数据集的基本统计.

Table 3 Statistics of Datasets

表 3 数据集的基本统计

Statistics	MovieLens	Yahoo!Music
No. Ratings	1 000 209	252 800 275
No. Users	6 040	1 000 990
No. Items	3 952	624 961
Rating Range	[1,5]	[0,100]

我们采用 $RMSE=\sqrt{\sum_{(u,i,r)\in Z}(\hat{r}_{u,i}-r)^2/|Z|}$ 来评估算法的性能,*RMSE* 是真实评分值与预测评分值误差的平均平方根.

为了评估本文的流数据在线挖掘算法的可伸缩性,我们的实验有下列 3 种设置:1) T_1:随机抽取所有三元组(u,i,r)的 10%训练,使用剩余的 90%作评估使用;2) T_5:随机抽取所有三元组(u,i,r)的 50%训练,使用剩余的 50%作评估使用;3) T_9:随机抽取所有三元组(u,i,r)的 90%训练,使用剩余的 10%作评估使用.

表 4 记录了各种不同设置下在线与批训练 PMF 算法的 *RMSE* 性能值.

从实验中,我们观察到下列现象:

1) 总的来说,在线 PMF 算法在不同数据集上的性能与批训练算法相近.

2) 在 T_1 设置下,在线算法的性能甚至要超出批训练算法一点点,这可能是由于下面事实引起:在很少训练样本的场景下,相对于批训练算法,在线学习算法陷入局部最优解的可能性要小一些.

① http://kddcup.yahoo.com

② http://www.cs.umn.edu/Research/Group-Lens

Table 4 *RMSE* of PMF Algorithm on Different Settings

表 4 各种设置下算法的 *RMSE* 比较

Algorithm	MovieLens			Yahoo!Music	
	T_1	T_5	T_9	T_1	T_5
BT-MTL	1.002	0.902	**0.868**	28.88	23.54
DA-STL	**0.896**	0.895	0.869	29.24	23.57
DA-MTFS	0.992	0.901	0.902	28.41	22.84
ADA-MTL	0.898	**0.894**	0.901	**28.35**	**22.82**

3) 由于 Yahoo!Music 数据集的巨大数据,我们不能使用 T_9 设置执行在协同过滤的 PMF 在线学习场景中应用批训练算法 BT-MTL. 在 T_5 设置下,用于概率矩阵分解的算法 BT-MTL 完成 120 次迭代收敛所用时间多于 6 h,而使用我们提出的用于概率矩阵分解的多任务加速在线算法 ADA-MTL 只用大约 1 min 完成所有 18 000 万个评分的处理并达到相似的性能指标. 时间的节省十分惊人. 因此,算法 ADA-MTL 十分轻松地适应了 T_1, T_5, T_9 等不同的环境设置,具有很好的可伸缩性,可以满足大规模数据流的需求.

4 结束语

本文首次提出一种面向大数据流的、新颖的多任务加速在线算法,每次迭代时空复杂度为 $O(dQ)$,最优收敛率为 $O(1/T^2)$. 与当前的在线算法和批训练算法相比,提出的算法在各种应用实验中均取得相当或更好的分析预测性能. 更重要的是,该算法显著提升了大规模数据流处理的实时性和可伸缩性,在大数据流领域有较广泛的实际应用价值.

本研究工作尚存需要进一步完善之处:1) p-范数 RDA 方法能够更好地适应学习问题的几何结构,如何与加速机制结合在特定情况下获得更好的稀疏解与收敛率值得研究;2)大数据环境下的多核学习问题是大数据机器学习的基础性问题之一,如何设计出高效的多任务加速在线多核学习算法,更好地满足大数据流应用的各种需求是我们下一步研究要做的工作.

参 考 文 献

[1] Sun Dawei, Zhang Guangyan, Zheng Weimin. Big data stream computing: Technologies and instances [J]. Journal of Software, 2014, 25(4): 839-862 (in Chinese)
(孙大为, 张广艳, 郑纬民. 大数据流式计算: 关键技术及系统实例[J]. 软件学报, 2014, 25(4): 839-862)

[2] Meng Xiaofeng, Ci Xiang. Big data management: Concepts, techniques and challenges [J]. Journal of Computer Research and Development, 2013, 50(1): 146-169 (in Chinese)
(孟小峰, 慈祥. 大数据管理:概念,技术与挑战[J]. 计算机研究与发展, 2013, 50(1): 146-169)

[3] Yang Haiqin, Lyu M R, King I. Efficient online learning for multi-task feature selection [J]. ACM Trans on Knowledge Discovery from Data, 2013, 1(1): 1-28

[4] Ando R K, Zhang Tong. A framework for learning predictive structures from multiple tasks and unlabeled data [J]. Journal of Machine Learning Research, 2005, 6(11): 1817-1853

[5] Argyriou A, Evgeniou T, Pontil M. Multi-task feature learning [C] //Proc of Neural Information Processing Systems. Cambridge, MA: MIT Press, 2006: 41-48

[6] Ben D S, Borbely R S. A notion of task relatedness yielding provable multiple-task learning guarantees [J]. Machine Learning, 2008, 73(3): 273-287

[7] Caruana R. Multitask learning [J]. Maching Learning, 1997, 28(1): 41-75

[8] Evgeniou T, Pontil M. Learning multiple tasks with kernel methods [J]. Journal of Machine Learning Research, 2005, 6(4): 615-637

[9] Liu Jun, Ji Shuiwang, Ye Jieping. Multi-task feature learning via efficient $l_{2,1}$ norm minimization [C] //Proc of Uncertainty in Artificial Intelligence. Arlington, VA: AUAI, 2009: 339-348

[10] Pong Tingkei, Tseng Paul, Pi Shuiwang, et al. Trace norm regularization: Reformulations, algorithms, and multi-task learning [J]. SIAM Journal on Optimization, 2010, 20(6): 3465-3489

[11] Yang Haiqin, King I, Lyu M R. Multi-task learning for one-class classification [C] //Proc of Int Joint Conf on Neural Network. Piscataway, NJ: IEEE, 2010: 1-8

[12] Zhang Yi. Multi-task active learning with output constraints [C] //Proc of American Association for Artificial Intelligence. Menlo Park, CA: AAAI, 2010: 667-672

[13] Yang Haiqin, Xu Z L, King I, et al. Online learning for group lasso [C]//Proc of the 38th Int Conf on Machine Learning. New York: ACM, 2010: 1191-1198

[14] Yang Haiqin, King I, Lyu M R. Online learning for multi-task feature selection [C] //Proc of the 19th ACM Conf on Information and Knowledge Management. New York: ACM, 2010: 1693-1696

[15] Xiao L. Dual averaging method for regularized stochastic learning and online optimization [J]. Journal of Machine Learning Research, 2010, 11(10): 2543-2596.

[16] Hu C H, Kwok J T, Pan W K. Accelerated gradient methods for stochastic optimization and online learning [C] // Proc of Advances in Neural Information Processing Systems. Cambridge, MA: MIT Press, 2012: 781-789

[17] Ji Shuiwang, Ye Jieping. An accelerated gradient method for trace norm minimization [C] //Proc of the 26th Int Conf on Machine Learning. New York: ACM, 2009

[18] Zhan Ying, Wu Chunming, Wang Baojun. An algorithm for data stream sampling based on ring circular sliding window tightly coupled with buffer [J]. Acta Electronica Sinica, 2011, 39(4): 894-898 (in Chinese)
(詹英, 吴春明, 王宝军. 一种与缓冲区紧耦合的环形循环滑动窗口的数据流抽取算法[J]. 电子学报, 2011, 39(4): 894-898)

[19] Argyriou A, Evgeniou T, Pontil M. Convex multi-task feature learning [J]. Machine Learning, 2008, 73(3): 243-272

[20] Mairal J, Bach F, Ponce J, et al. Online learning for matrix factorization and sparse coding [J]. Machine Learning Research, 2010(11): 19-60

[21] Liu Nathann, Yang Qiang. Eigenrank: A ranking-oriented approach to collaborative filtering [C] //Proc of Special Interest Group on Information Retrieval. New York: ACM, 2008: 83-90

[22] Liu Nathann, Zhao Min, Yang Qiang. Probabilistic latent preference analysis for collaborative filtering [C] //Proc of Conf on Information and Knowledge Management. New York: ACM, 2009: 759-766

Li Zhijie, born in 1964. PhD candidate from Wuhan University. Associate professor. Member of China Computer Federation. His current research interests include machine learning and data mining.

Li Yuanxiang, born in 1962. Professor and PhD supervisor in Wuhan University. His current research interests include intelligent computation and parallel computing etc.

Wang Feng, born in 1981. Associate professor in Wuhan University. Her main research interests include data mining and machine learning.

Kuang Li, born in 1986. PhD candidate from Wuhan University. His current research interests include complex network and evolutionary algorithm.

附录 A. 定理 1 的证明.

证明. 对于在式(5)定义的 $L_{1/2,1}$-范数正则化,式(7)是基于向量 $\boldsymbol{W}_{j\cdot}^{\mathrm{T}}$ 和 $\bar{\boldsymbol{G}}_{j\cdot}^{\mathrm{T}}$ 的. 我们使用 $\boldsymbol{w}$ 表示 $\boldsymbol{W}_{j\cdot}^{\mathrm{T}}$,则式(7)变为

$$\phi(\boldsymbol{w}_t)=(\bar{\boldsymbol{G}}_{j\cdot})_t\boldsymbol{w}_t+\lambda(c\|\boldsymbol{w}_t\|_1+\|\boldsymbol{w}_t\|_2)+\frac{1}{2}\|\boldsymbol{w}_t-\boldsymbol{u}_t\|_2^2, \tag{A1}$$

式(A1)中的目标向量是基于任务属性的,因此我们可以考虑一个任务 $q\in[1,Q]$,式(A1)变成:

$$\phi((w_q)_t)=(\bar{G}_{j,q})_t\times(w_q)_t+\lambda c\mid(w_q)_t\mid+\lambda\|(w_q)_t\|_2+\frac{1}{2}((w_q)_t-(u_q)_t)^2. \tag{A2}$$

显然:1) 当 $\bar{G}_{j,q}=(u_q)_t$,$(w_q)_t=0$;

2) 当 $\bar{G}_{j,q}>(u_q)_t$,$(w_q)_t<0$;

3) 当 $\bar{G}_{j,q}<(u_q)_t$,$(w_q)_t>0$.

对于 2),式(A2)变成:

$$\phi((w_q)_t)=((\bar{G}_{j,q})_t-\lambda c)\times(w_q)_t+\lambda\|(w_q)_t\|_2+\frac{1}{2}((w_q)_t-(u_q)_t)^2.$$

显然,当 $\bar{G}_{j,q}-(u_q)_t\leqslant\lambda c$,$(w_q)_t=0$;

类似地,对于 3),当 $-\lambda c\leqslant\bar{G}_{j,q}-(u_q)_t$,$(w_q)_t=0$.

综上,当 $|(\bar{G}_{j,q})_t-(u_q)_t|-\lambda c\leqslant 0$ 时,$(w_q)_t=0$. 因此,令:

$$(\bar{\boldsymbol{R}}_{j,q})_t=[|(\bar{G}_{j,q})_t-(u_q)_t|-\lambda c]_+\times\operatorname{sgn}((\bar{G}_{j,q})_t-(u_q)_t).$$

对所有的 $q=1,2,\cdots,Q$,式(A1)就变成:

$$\varphi(\boldsymbol{w}_t)=(\bar{\boldsymbol{R}}_{j\cdot})^{\mathrm{T}}\boldsymbol{w}_t+\lambda\|\boldsymbol{w}_t\|_2+\frac{1}{2}(\|\boldsymbol{w}_t\|_2^2+\|\boldsymbol{u}_t\|_2^2). \tag{A3}$$

不难验证,式(A3)的最优解应该是 $\boldsymbol{w}_t=k(\bar{\boldsymbol{R}}_{j\cdot})_t$,$k\leqslant 0$. 因此,式(A3)的目标变为

$$\min_{k\leqslant 0}k\|(\bar{\boldsymbol{R}}_{j\cdot})_t\|_2^2-\lambda k\|(\bar{\boldsymbol{R}}_{j\cdot})_t\|_2+\frac{1}{2}k^2\|(\bar{\boldsymbol{R}}_{j\cdot})_t\|_2^2. \tag{A4}$$

构建上述最优问题的拉格朗日式,并应用 KKT 条件,最优解必须满足:

$$\frac{\partial\varphi}{\partial k}=\|(\bar{\boldsymbol{R}}_{j\cdot})_t\|_2^2-\lambda\|(\bar{\boldsymbol{R}}_{j\cdot})_t\|_2+k\|(\bar{\boldsymbol{R}}_{j\cdot})_t\|_2^2+v=0,\ vk=0,v>0.$$

由此解得:

$$k=-\left(1-\frac{\lambda}{\|(\bar{\boldsymbol{R}}_{j\cdot})_t\|_2}+\frac{v}{\|(\bar{\boldsymbol{R}}_{j\cdot})_t\|_2^2}\right). \tag{A5}$$

根据 KKT 条件,$k<0$ 当且仅当:$v=0$,$\lambda<\|(\bar{\boldsymbol{R}}_{j\cdot})_t\|_2$.因此得到:

$$(\boldsymbol{W}_{j\cdot})_t = k(\bar{\boldsymbol{R}}_{j\cdot})_t = -\left[1-\frac{\lambda}{\|(\bar{\boldsymbol{R}}_{j\cdot})_t\|_2}\right]_+ \times (\bar{\boldsymbol{R}}_{j\cdot})_t.$$

证毕.

附录 B. 定理 2 的证明.

证明. 令算法 2 的学习权重矩阵 $\boldsymbol{W}$ 最优值:

$$\boldsymbol{W}^* = \arg\min_{\boldsymbol{W}}\{\varphi(\boldsymbol{W}) = \langle \bar{\boldsymbol{G}}_t, \boldsymbol{W}\rangle + \Omega_\lambda(\boldsymbol{W})\}, \tag{B1}$$

那么,$v_T=\varphi(\boldsymbol{W}_T)-\varphi(\boldsymbol{W}^*)$ 代表算法迭代 T 次收敛率.

根据算法 2 的式(7),令 $p(\boldsymbol{Y})=\arg\min_{\boldsymbol{X}}\phi(\boldsymbol{X},\boldsymbol{Y})$,则 $\boldsymbol{W}_t=\arg\min_{\boldsymbol{W}}\phi(\boldsymbol{W},\boldsymbol{U}_t)=p(\boldsymbol{U}_t)$.

由于算法 2 损失函数 l 强凸,Ω_λ 是泛凸函数,有:

$$l(\boldsymbol{X}) \geqslant l(\boldsymbol{Y}) + \langle \boldsymbol{X}-\boldsymbol{Y}, \nabla l(\boldsymbol{Y})\rangle,$$
$$\Omega_\lambda(\boldsymbol{X}) \geqslant \Omega_\lambda(p(\boldsymbol{Y})) + \langle \boldsymbol{X}-p(\boldsymbol{Y}), g(p(\boldsymbol{Y}))\rangle.$$

这里,$\boldsymbol{g}(p(\boldsymbol{Y}))\in\partial\Omega_\lambda(p(\boldsymbol{Y}))$.上面 2 个不等式相加,得:

$$\varphi(\boldsymbol{X}) \geqslant l(\boldsymbol{Y}) + \langle \boldsymbol{X}-\boldsymbol{Y}, \nabla l(\boldsymbol{Y})\rangle + \Omega_\lambda(p(\boldsymbol{Y})) + \langle \boldsymbol{X}-p(\boldsymbol{Y}), \boldsymbol{g}(p(\boldsymbol{Y}))\rangle. \tag{B2}$$

因为:

$$\phi(p(\boldsymbol{Y}),\boldsymbol{Y}) = \varphi(p(\boldsymbol{Y})) + \frac{1}{2}\|p(\boldsymbol{Y})-\boldsymbol{Y}\|_{\mathrm{F}}^2,$$
$$\phi(p(\boldsymbol{Y}),\boldsymbol{Y}) = l(\boldsymbol{Y}) + \langle p(\boldsymbol{Y})-\boldsymbol{Y}, \nabla l(\boldsymbol{Y})\rangle + \Omega_\lambda(p(\boldsymbol{Y})) + \frac{1}{2}\|p(\boldsymbol{Y})-\boldsymbol{Y}\|_{\mathrm{F}}^2,$$
$$\partial\phi = \nabla l(\boldsymbol{Y}) + (p(\boldsymbol{Y})-\boldsymbol{Y}) + \boldsymbol{g}(p(\boldsymbol{Y})) = 0. \tag{B3}$$

由式(B2)和式(B3),我们得到:

$$\varphi(\boldsymbol{X})-\varphi(p(\boldsymbol{Y})) \geqslant \varphi(\boldsymbol{X})-\phi(p(\boldsymbol{Y}),\boldsymbol{Y}) \geqslant \langle \boldsymbol{X}-p(\boldsymbol{Y}), \nabla l(\boldsymbol{Y})+\boldsymbol{g}(p(\boldsymbol{Y}))\rangle - \frac{1}{2}\|p(\boldsymbol{Y})-\boldsymbol{Y}\|_{\mathrm{F}}^2 = \langle \boldsymbol{Y}-\boldsymbol{X}, p(\boldsymbol{Y})-\boldsymbol{Y}\rangle + \frac{1}{2}\|p(\boldsymbol{Y})-\boldsymbol{Y}\|_{\mathrm{F}}^2.$$

即

$$\varphi(\boldsymbol{X})-\varphi(p(\boldsymbol{Y})) \geqslant \langle \boldsymbol{Y}-\boldsymbol{X}, p(\boldsymbol{Y})-\boldsymbol{Y}\rangle + \frac{1}{2}\|p(\boldsymbol{Y})-\boldsymbol{Y}\|_{\mathrm{F}}^2. \tag{B4}$$

令:

$$v_t = \varphi(\boldsymbol{W}_t)-\varphi(\boldsymbol{W}^*),$$
$$\boldsymbol{V}_t = \alpha_t\boldsymbol{W}_t - (\alpha_t-1)\boldsymbol{W}_{t-1} - \boldsymbol{W}^*. \tag{B5}$$

应用式(B4),分别设 $\boldsymbol{X}=\boldsymbol{W}_t$,$\boldsymbol{Y}=\boldsymbol{U}_{t+1}$ 和 $\boldsymbol{X}=\boldsymbol{W}^*$,$\boldsymbol{Y}=\boldsymbol{U}_{t+1}$,得到:

$$2(v_t-v_{t+1}) \geqslant \|\boldsymbol{W}_{t+1}-\boldsymbol{U}_{t+1}\|_{\mathrm{F}}^2 + 2\langle \boldsymbol{W}_{t+1}-\boldsymbol{U}_{t+1}, \boldsymbol{U}_{t+1}-\boldsymbol{W}_t\rangle, \tag{B6}$$
$$-2v_{t+1} \geqslant \|\boldsymbol{W}_{t+1}-\boldsymbol{U}_{t+1}\|_{\mathrm{F}}^2 + 2\langle \boldsymbol{W}_{t+1}-\boldsymbol{U}_{t+1}, \boldsymbol{U}_{t+1}-\boldsymbol{W}^*\rangle. \tag{B7}$$

式(B6)两边乘以 $\alpha_{t+1}-1$ 并加式(B7):

$$2((\alpha_{t+1}-1)v_t - \alpha_{t+1}v_{t+1}) \geqslant \alpha_{t+1}\|\boldsymbol{W}_{t+1}-\boldsymbol{U}_{t+1}\|_{\mathrm{F}}^2 + 2\langle \boldsymbol{W}_{t+1}-\boldsymbol{U}_{t+1}, \alpha_{t+1}\boldsymbol{U}_{t+1} - (\alpha_{t+1}-1)\boldsymbol{W}_t - \boldsymbol{W}^*\rangle.$$

上面不等式两边乘以 α_{t+1} 并利用算法 2 的式(8)导出的关系式:$\alpha_k^2=\alpha_{k+1}^2-\alpha_{k+1}$,我们得到:

$$2(\alpha_t^2 v_t - \alpha_{t+1}^2 v_{t+1}) \geqslant \|\alpha_{t+1}(\boldsymbol{W}_{t+1}-\boldsymbol{U}_{t+1})\|_{\mathrm{F}}^2 + 2\alpha_{t+1}\langle \boldsymbol{W}_{t+1}-\boldsymbol{U}_{t+1}, \alpha_{t+1}\boldsymbol{U}_{t+1} - (\alpha_{t+1}-1)\boldsymbol{W}_t - \boldsymbol{W}^*\rangle. \tag{B8}$$

因为,对任何大小相同的矩阵 $\boldsymbol{A},\boldsymbol{B},\boldsymbol{C}$,均有:

$$\|\boldsymbol{B}-\boldsymbol{A}\|_{\mathrm{F}}^2 + 2\langle \boldsymbol{B}-\boldsymbol{A}, \boldsymbol{A}-\boldsymbol{C}\rangle = \|\boldsymbol{B}-\boldsymbol{C}\|_{\mathrm{F}}^2 - \|\boldsymbol{A}-\boldsymbol{C}\|_{\mathrm{F}}^2.$$

因此,式(B8)可变为

$$2(\alpha_t^2 v_t - \alpha_{t+1}^2 v_{t+1}) \geqslant \|\alpha_{t+1}\boldsymbol{W}_{t+1} - (\alpha_{t+1}-1)\boldsymbol{W}_t - \boldsymbol{W}^*\|_{\mathrm{F}}^2 - \|\alpha_{t+1}\boldsymbol{U}_{t+1} - (\alpha_{t+1}-1)\boldsymbol{W}_t - \boldsymbol{W}^*\|_{\mathrm{F}}^2.$$

利用算法 2 的式(9)以及式(B5)中 $\boldsymbol{V}_t$ 的定义,得:

$$2\alpha_t^2 v_t - 2\alpha_{t+1}^2 v_{t+1} \geqslant \|\boldsymbol{V}_{t+1}\|_{\mathrm{F}}^2 - \|\boldsymbol{V}_t\|_{\mathrm{F}}^2. \tag{B9}$$

应用式(B9),t 分别取 $1,2,\cdots,t-1$,并将各式相加,得:

$$2v_1 - 2\alpha_t^2 v_t \geqslant \|\boldsymbol{V}_t\|_{\mathrm{F}}^2 - \|\boldsymbol{V}_1\|_{\mathrm{F}}^2. \tag{B10}$$

应用式(B4),设 $\boldsymbol{X}=\boldsymbol{W}^*$,$\boldsymbol{Y}=\boldsymbol{U}_1$,得:

$$\varphi(\boldsymbol{W}^*)-\varphi(\boldsymbol{W}_1) = \varphi(\boldsymbol{W}^*)-\varphi(p(\boldsymbol{U}_1)) \geqslant \langle \boldsymbol{U}_1-\boldsymbol{W}^*, p(\boldsymbol{U}_1)-\boldsymbol{U}_1\rangle + \frac{1}{2}\|p(\boldsymbol{U}_1)-\boldsymbol{U}_1\|_{\mathrm{F}}^2 = \frac{1}{2}(\|\boldsymbol{W}_1-\boldsymbol{U}_1\|_{\mathrm{F}}^2 + 2\langle \boldsymbol{U}_1-\boldsymbol{W}^*, \boldsymbol{W}_1-\boldsymbol{U}_1\rangle) = \frac{1}{2}\|\boldsymbol{W}_1-\boldsymbol{W}^*\|_{\mathrm{F}}^2 - \frac{1}{2}\|\boldsymbol{U}_1-\boldsymbol{W}^*\|_{\mathrm{F}}^2,$$

即

$$2v_1 \leqslant \|\boldsymbol{U}_1-\boldsymbol{W}^*\|_{\mathrm{F}}^2 - \|\boldsymbol{W}_1-\boldsymbol{W}^*\|_{\mathrm{F}}^2. \tag{B11}$$

把式(B11)代入式(B10):

$$\|\boldsymbol{U}_1-\boldsymbol{W}^*\|_{\mathrm{F}}^2 - 2\alpha_t^2 v_t \geqslant \|\boldsymbol{V}_t\|_{\mathrm{F}}^2 \geqslant 0.$$

由于 $\alpha_t\geqslant(t+1)/2$,于是有:

$$v_t = \varphi(\boldsymbol{W}_t)-\varphi(\boldsymbol{W}^*) \leqslant \frac{2\|\boldsymbol{W}_0-\boldsymbol{W}^*\|_{\mathrm{F}}^2}{(t+1)^2}.$$

若算法 1 运行到第 T 步,则收敛率为

$$v_T = \varphi(\boldsymbol{W}_T)-\varphi(\boldsymbol{W}^*) \leqslant \frac{2\|\boldsymbol{W}_0-\boldsymbol{W}^*\|_{\mathrm{F}}^2}{(T+1)^2} = O\left(\frac{1}{T^2}\right).$$

证毕.