计算机研究与发展
Journal of Computer Research and Development
DOI:10.7544/issn1000-1239.2016.20150267
53(5):1053-1062,2016

一种基于最大值损失函数的快速偏标记学习算法

周 瑜[1] 贺建军[1,2] 顾 宏[1] 张俊星[2]
[1](大连理工大学电子信息与电气工程学部 辽宁大连 116024)
[2](大连民族大学信息与通信工程学院 辽宁大连 116600)
(yuzhou829@sina.com)

A Fast Partial Label Learning Algorithm Based on Max-loss Function

Zhou Yu[1], He Jianjun[1,2], Gu Hong[1], and Zhang Junxing[2]
[1](*Faculty of Electronic Information and Electrical Engineering, Dalian University of Technology, Dalian, Liaoning* 116024)
[2](*College of Information and Communication Engineering, Dalian Nationalities University, Dalian, Liaoning* 116600)

Abstract In the age of big data, learning with weak supervision has become one of the hot research topics in machine learning field. Partial label learning, which deals with the problem where each training example is associated with a set of candidate labels among which only one label corresponds to the ground-truth, is an important weakly-supervised machine learning frameworks proposed recently and can be widely used in many real world tasks. The max-loss function may be used to accurately capture the relationship between the partial labeled sample and its labels. However, since the max-loss function usually brings us a nondifferentiable objective function difficult to be solved, it is rarely adopted in the existing algorithms. Moreover, the existing partial label learning algorithms can only deal with the problem with small-scale data, and rarely can be used to deal with big data. To cure above two problems, this paper presents a fast partial label learning algorithm with the max-loss function. The basic idea is to transform the nondifferentiable objective to a differentiable concave function by introducing the aggregate function to approximate the max(·) function involved in the max-lass function, and then to solve the obtained concave objective function by using a stochastic quasi-Newton method. The experimental results show that the proposed algorithm can not only achieve higher accuracy but also use shorter computing time than the state-of-the-art algorithms with average-loss functions. Moreover, the proposed algorithm can deal with the problems with millions samples within several minutes.

Key words partial label learning; max-loss function; aggregate function; weakly-supervised learning; classification accuracy

摘 要 在弱监督信息条件下进行学习已成为大数据时代机器学习领域的研究热点,偏标记学习是最近提出的一种重要的弱监督学习框架,主要解决在只知道训练样本的真实标记属于某个候选标记集合的情况下如何进行学习的问题,在很多领域都具有广泛应用.最大值损失函数可以很好地描述偏标记学习

收稿日期:2015-03-26;**修回日期**:2015-07-13
基金项目:国家自然科学基金项目(61503058,61374170,61502074,U1560102);高等学校博士学科点专项科研基金项目(20120041110008);中央高校基本科研业务费专项资金项目(DC201501055,DC201501060201)
This work was supported by the National Natural Science Foundation of China (61503058,61374170,61502074,U1560102), the Research Fund for the Doctoral Program of Higher Education of China (20120041110008), and the Fundamental Research Funds for the Central Universities (DC201501055,DC201501060201).
通信作者:顾宏(guhong@dlut.edu.cn)

中的样本与候选标记间的关系,但是由于建立的模型通常是一个难以求解的非光滑函数,目前还没有建立基于该损失函数的偏标记学习算法.此外,已有的偏标记学习算法都只能处理样本规模比较小的问题,还没看到面向大数据的算法.针对以上2个问题,先利用凝聚函数逼近最大值损失函数中的max(•)将模型的目标函数转换为一个光滑的凹函数,然后利用随机拟牛顿法对其进行求解,最终实现了一种基于最大值损失函数的快速偏标记学习算法.仿真实验结果表明,此算法不仅要比基于均值损失函数的传统算法取得更好的分类精度,运行速度上也远远快于这些算法,处理样本规模达到百万级的问题只需要几分钟.

关键词 偏标记学习;最大值损失函数;凝聚函数;弱监督学习;分类精度

中图法分类号 TP391

在传统分类技术中,通常需要通过一个样本的真实标记信息准确给定的训练样本集来训练分类器,而在很多实际问题中,准确确定样本的真实标记信息是很困难或者需要付出很大代价的.例如在医疗诊断中,医生通过一个病人的症状(如关节疼、发烧、血沉高),往往可以判断他得了哪几种疾病(可能是风湿性关节炎或布氏杆菌病),而很难确定具体是哪种疾病,在利用传统分类技术构造疾病诊断的预测算法时,这类数据需要被排除在外,而其实这类数据对于缩小疾病的可能范围往往是非常重要的;再如,在利用众筹的方式标注训练数据时,由于标注人员个人素质的差异,标注的结果往往会不一致,在传统的分类算法中,这种不一致的数据往往会被舍弃,其实,虽然标注结果不一致,但是也已经缩小了类别的可能范围,所以这类数据也是有用的.偏标记集学习(partial label learning[1-3],也称 learning from candidate labeling sets[4] ambiguous label learning[5], superset label learning[6]),就是研究这类数据(即不能准确确定训练样本的真实标记而只知道它属于类别标记集合的某一子集)的分类问题的一种新机器学习框架.由于偏标记学习是对传统分类技术的一个扩展,放松了构造训练集的条件,因此它与传统分类技术一样具有广阔的应用空间,可以解决图像处理[7]、文本挖掘[8]、医疗诊断[9]等领域的各种实际问题.

设 X 为样本的特征空间,$Y=\{y_1,y_2,\cdots,y_Q\}$ 为类别标记集合.偏标记学习的目的就是利用一个训练集 $D=\{(\boldsymbol{x}_1,Y_1),(\boldsymbol{x}_2,Y_2),\cdots,(\boldsymbol{x}_n,Y_n)\}$(其中 $\boldsymbol{x}_i\in X$ 是样本的特征向量;$Y_i\equiv\{y_{i1},y_{i2},\cdots,y_{im_i}\}\subset Y$ 是样本 $\boldsymbol{x}_i$ 的真实标记的一个候选集合)确定一个函数 $f:X\to Y$,使得 f 可以正确输出新(待预测)样本 $\boldsymbol{x}^*\in X$ 的类别标记.可以看出,在偏标记学习框架下,所能利用的训练数据的标记信息不再具有单一性和明确性,真实标记湮没于候选标记集合中,这就使得学习算法的构建变得比传统分类算法更加困难.最早的偏标记学习算法可以追溯到2002年Grandvalet[10]对Logistic回归模型的拓展研究,随后Jin和Ghahramani[11]将偏标记学习归结为一种新的机器学习框架,提出了基于辨识模型的学习算法.在这2组科研人员早期工作的推动下,偏标记学习逐渐引起了人们的关注,尝试将传统的机器学习模型引入偏标记学习算法的构建问题中.文献[5]提出了一种基于 k 近邻方法的偏标记学习算法;文献[12]提出了基于最大似然估计和信度函数的学习算法;Luo和Orabona[4]提出了一种最大间隔偏标记学习算法;Cour等人[2]和Nguyen等人[13]分别提出了一种线性支持向量机偏标记学习算法;Liu和Dietterich[6]提出了一种条件混合多变量模型;Zhang[3]利用纠错输出编码技术提出了一种基于集成分类器的偏标记学习算法;文献[14]提出了一种针对结构化输入数据的偏标记学习算法.这些已有成果主要通过2种策略来构建偏标记学习算法,第1种策略是将偏标记学习问题的训练数据集变换为传统的分类问题的训练集,然后利用已有的传统分类算法求解问题,例如文献[3]首先利用ECOC技术将偏标记学习问题的训练数据集变换为具有确切标记信息的多个二分类问题的数据集,然后在每个二分类问题数据集上建立一个二分类器,最后将各个分类器的结果综合来输出预测结果,该类策略的一个主要问题是建立的二分类数据集中正负样本的比例可能会极度不平衡,最终影响算法的精度;第2种策略是通过定义新的损失函数来改进传统分类模型,使得其可以处理偏标记集学习问题,基于该策略人们提出了基于 k-NN(k-nearest neighbor)[5]、最大间隔[4]、Logistic[10]、线性支持向量机[2]等几种模型的学习算法.根据文献[2]的理论分析可知,最大值

损失 $L(f(\boldsymbol{x}_i),Y_i)=\max\limits_{y\in Y_i}L(f(\boldsymbol{x}_i),y)$可以很好地描述偏标记学习中的样本与候选标记间的关系,但是由于 max(•)是一个非光滑函数,基于最大值损失建立的模型很难求解,目前还没见到基于最大值损失函数的偏标记学习算法.另外,随着大数据时代的来临,数据产生速度的持续加快,这也使得在很多应用领域规模巨大的训练数据唾手可得,所以研究能够有效利用大数据的偏标记学习算法也具有非常重要的实际意义,而目前还没有看到相关的研究成果.针对以上 2 个问题,本文将尝试建立一种基于最大值损失函数的快速偏标记学习算法,基本思想是利用凝聚函数来逼近 max(•)将模型的目标函数转换为一个可以由常用的凸优化方法求解的可微凹函数,然后通过引入一种称作随机拟牛顿的方法来建立一种快速的模型求解方法,从而建立可以有效处理大数据问题的偏标记学习算法.

1　模型建立

本文拟以 Logistic 回归模型为建模工具来构建偏标记学习算法,虽然文献[10]已经提出了一种基于 Logistic 回归模型的偏标记学习算法,但是该算法是基于均值损失函数建立的,而本文拟建立的是一种基于最大值损失函数的算法,因此这二者是不同的.与传统的多分类 Logistic 回归模型一样,基本的思想是在特征空间 X 中为每个类别标记 $y_j(j=1,2,\cdots,Q)$定义一个潜变量函数 $f_{y_j}(\boldsymbol{x})=\overline{\boldsymbol{w}}_{y_j}^{\mathrm{T}}\boldsymbol{x}+b_{y_j}$,然后样本 $\boldsymbol{x}\in X$ 的真实类别标记是 $y_k(k=1,2,\cdots,Q)$的概率可以由这些潜变量函数$\{f_{y_j}\mid j=1,2,\cdots,Q\}$在 $\boldsymbol{x}$ 上的值来确定,即 $p(y_k\mid\boldsymbol{x},W)=\mathrm{e}^{f_{y_k}(\boldsymbol{x})}\Big/\Big(\sum\limits_{j=1}^{Q}\mathrm{e}^{f_{y_j}(\boldsymbol{x})}\Big)$,而模型参数 $W\equiv\{\overline{\boldsymbol{w}}_{y_j},b_{y_j}\mid j=1,2,\cdots,Q\}$可以通过最大化对数联合似然函数 $\ln L(W\mid D)=\sum\limits_{i=1}^{n}\ln p(Y_i\mid\boldsymbol{x}_i,W)$来得到.由于在偏标记学习中,我们只知道样本 $\boldsymbol{x}_i$ 的真实标记属于一个候选标记集合 $Y_i\equiv\{y_{i1},y_{i2},\cdots,y_{im_i}\}\subset Y$,而并不知道具体是候选标记集合中的哪个标记,因此构建偏标记学习算法的主要难点在于如何定义似然函数 $p(Y_i\mid\boldsymbol{x}_i,W)$.已有的偏标记学习算法(如文献[2,10])主要采用 2 种形式的似然函数:

$$p(Y_i\mid\boldsymbol{x}_i,W)=\frac{1}{\mid Y_i\mid}\sum_{y\in Y_i}p(y\mid\boldsymbol{x}_i,W)=\frac{1}{\mid Y_i\mid}\sum_{y\in Y_i}\mathrm{e}^{f_y(\boldsymbol{x}_i)}\Big/\Big(\sum_j\mathrm{e}^{f_{y_j}(\boldsymbol{x}_i)}\Big),\tag{1}$$

$$p(Y_i\mid\boldsymbol{x}_i,W)=\exp\Big(\frac{1}{\mid Y_i\mid}\sum_{y\in Y_i}f_y(\boldsymbol{x}_i)\Big)\Big/\Big(\sum_j\mathrm{e}^{f_{y_j}(\boldsymbol{x}_i)}\Big).\tag{2}$$

可以看出,以上 2 个似然函数都假设样本 $\boldsymbol{x}_i$ 的候选标记集 Y_i 中的每个标记都对似然函数 $p(Y_i\mid\boldsymbol{x}_i,W)$具有一样的贡献,这将导致最大化 $p(Y_i\mid\boldsymbol{x}_i,W)$时,与 Y_i 中的各个元素对应的潜变量同时取得最大值,因此这 2 个似然函数本质是假定了 Y_i 的所有元素都是 $\boldsymbol{x}_i$ 的真实标记,这与偏标记学习框架描述的"样本的候选标记中有且只有一个标记是样本的真实标记"的概念是不一致的.本文将定义一种新的似然函数:

$$p(Y_i\mid\boldsymbol{x}_i,W)=\max_{y\in Y_i}p(y\mid\boldsymbol{x}_i,W)=\max_{y\in Y_i}\Big\{\mathrm{e}^{f_y(\boldsymbol{x}_i)}\Big/\Big(\sum_{j=1}^{Q}\mathrm{e}^{f_{y_j}(\boldsymbol{x}_i)}\Big)\Big\}=\mathrm{e}^{\max\limits_{y\in Y_i}f_y(\boldsymbol{x}_i)}\Big/\Big(\sum_{j=1}^{Q}\mathrm{e}^{f_{y_j}(\boldsymbol{x}_i)}\Big),\tag{3}$$

该似然函数本质上把是真实标记的概率最大的候选标记作为样本的真实标记.由于概率模型中似然函数与参数模型中的损失函数本质上是一样的,而文献[2]的理论分析表明,最大值损失较均值损失更符合偏标记学习的实际情况,因此由式(3)定义的似然函数也要优于由式(1)(2)定义的似然函数.

为了避免过拟合,可以在目标函数中增加一个正则化项 $\sum\limits_{j=1}^{Q}\|\overline{\boldsymbol{w}}_{y_j}\|^2/2$.因此可以通过最大化如下目标函数来求解模型参数 $W\equiv\{\overline{\boldsymbol{w}}_{y_j},b_{y_j}\mid j=1,2,\cdots,Q\}$,

$$W=\arg\max_W\Big\{\ln L(W\mid D)-\rho\sum_{j=1}^{Q}\|\overline{\boldsymbol{w}}_{y_j}\|^2/2\Big\}=\arg\max_W\Big\{\sum_{i=1}^{n}\ln\max_{y\in Y_i}p(y\mid\boldsymbol{x}_i,W)-\rho\sum_{j=1}^{Q}\|\overline{\boldsymbol{w}}_{y_j}\|^2/2\Big\}=\arg\max_W\Big\{\sum_{i=1}^{n}\max_{y\in Y_i}f_y(\boldsymbol{x}_i)-\sum_{i=1}^{n}\ln\Big(\sum_{j=1}^{Q}\mathrm{e}^{f_{y_j}(\boldsymbol{x}_i)}\Big)-\rho\sum_{j=1}^{Q}\|\overline{\boldsymbol{w}}_{y_j}\|^2/2\Big\},\tag{4}$$

其中,$\rho>0$ 是正则化项的权重系数.由于$\max\limits_{y\in Y_i}f_y(\boldsymbol{x}_i)$是一个非光滑函数,式(4)并不能通过基于梯度的方法直接求解,这也是已有的偏标记学习算法不采用式(3)的主要原因.下面我们将通过凝聚函数将式(4)近似地表示为一个光滑的凹函数,从而采用常用的凸优化方法就可以求解.

凝聚函数(aggregate function)

$$G_l(\boldsymbol{x})=\frac{1}{l}\ln\Big(\sum_{j=1}^{m}\mathrm{e}^{lg_j(\boldsymbol{x})}\Big),\tag{5}$$

又名指数惩罚函数(exponential penalty function),是由 Li[15]在求解非线性规划问题时利用代理约束概念和最大熵原理推导出来的,这里 l 是一个正的控制参数.该函数和最大值函数 $G(\boldsymbol{x})=\max\limits_{j\in\{1,2,\cdots,m\}} g_j(\boldsymbol{x})$ 有关系式:

$$G(\boldsymbol{x})\leqslant G_l(\boldsymbol{x})\leqslant G(\boldsymbol{x})+\frac{1}{l}\ln m. \tag{6}$$

因此,对于有限正整数 m,当 $l\to+\infty$ 时,$G_l(\boldsymbol{x})$ 能单调一致地逼近 $G(\boldsymbol{x})$.根据式(6)描述的关系,我们可以得到 2 个关系式:

$$\max_{y\in Y_i} f_y(\boldsymbol{x}_i)\leqslant\frac{1}{l}\ln\Big(\sum_{y\in Y_i}\mathrm{e}^{lf_y(\boldsymbol{x}_i)}\Big)\leqslant \max_{y\in Y_i} f_y(\boldsymbol{x}_i)+\frac{\ln Q}{l},\ i=1,2,\cdots,n, \tag{7}$$

$$\sum_{i=1}^{n}\max_{y\in Y_i} f_y(\boldsymbol{x}_i)\leqslant\frac{1}{l}\sum_{i=1}^{n}\ln\Big(\sum_{y\in Y_i}\mathrm{e}^{lf_y(\boldsymbol{x}_i)}\Big)\leqslant \sum_{i=1}^{n}\max_{y\in Y_i} f_y(\boldsymbol{x}_i)+\frac{n\ln Q}{l}, \tag{8}$$

从式(7)(8)可以看出,当 $l\to+\infty$ 时,$\frac{1}{l}\sum\limits_{i=1}^{n}\ln\Big(\sum\limits_{y\in Y_i}\mathrm{e}^{lf_y(\boldsymbol{x}_i)}\Big)$ 可以一致逼近 $\sum\limits_{i=1}^{n}\max\limits_{y\in Y_i} f_y(\boldsymbol{x}_i)$,而且 $\frac{1}{l}\sum\limits_{i=1}^{n}\ln\Big(\sum\limits_{y\in Y_i}\mathrm{e}^{lf_y(\boldsymbol{x}_i)}\Big)$ 是光滑函数.因此,当选取适当的 l 后,就可以用 $\frac{1}{l}\sum\limits_{i=1}^{n}\ln\Big(\sum\limits_{y\in Y_i}\mathrm{e}^{lf_y(\boldsymbol{x}_i)}\Big)$ 来代替式(4)中的 $\sum\limits_{i=1}^{n}\max\limits_{y\in Y_i} f_y(\boldsymbol{x}_i)$,即:

$$\begin{aligned}\boldsymbol{W}&=\arg\max_{\boldsymbol{W}}\Big\{\sum_{i=1}^{n}\max_{y\in Y_i} f_y(\boldsymbol{x}_i)-\sum_{i=1}^{n}\ln\Big(\sum_{j=1}^{Q}\mathrm{e}^{f_{y_j}(\boldsymbol{x}_i)}\Big)-\rho\sum_{j=1}^{Q}\|\bar{\boldsymbol{w}}_{y_j}\|^2/2\Big\}\approx\\&\arg\max_{\boldsymbol{W}}\Big\{\frac{1}{l}\sum_{i=1}^{n}\ln\Big(\sum_{y\in Y_i}\mathrm{e}^{lf_y(\boldsymbol{x}_i)}\Big)-\sum_{i=1}^{n}\ln\Big(\sum_{j=1}^{Q}\mathrm{e}^{f_{y_j}(\boldsymbol{x}_i)}\Big)-\rho\sum_{j=1}^{Q}\|\bar{\boldsymbol{w}}_{y_j}\|^2/2\Big\}=\\&\arg\max_{\boldsymbol{W}}\Big\{\frac{1}{l}\sum_{i=1}^{n}\ln\Big(\sum_{y\in Y_i}\mathrm{e}^{l(\bar{\boldsymbol{w}}_y^{\mathrm{T}}\boldsymbol{x}_i+b_y)}\Big)-\sum_{i=1}^{n}\ln\Big(\sum_{j=1}^{Q}\mathrm{e}^{\bar{\boldsymbol{w}}_{y_j}^{\mathrm{T}}\boldsymbol{x}_i+b_{y_j}}\Big)-\rho\sum_{j=1}^{Q}\|\bar{\boldsymbol{w}}_{y_j}\|^2/2\Big\}\triangleq\\&\arg\max_{\boldsymbol{W}} Z(\boldsymbol{W}).\end{aligned} \tag{9}$$

显然,新目标函数 $Z(\boldsymbol{W})$ 不仅可以一致地逼近式(4),而且还是一个光滑函数.

下面证明 $Z(\boldsymbol{W})$ 还是一个凹函数.为了便于描述,我们重新定义 $\boldsymbol{W},\boldsymbol{x}_i,y_j$ 的形式:$\boldsymbol{W}$ 写成由所有模型参数 $\{\bar{\boldsymbol{w}}_{y_j},b_{y_j}\mid j=1,2,\cdots,Q\}$ 构成的列向量,即 $\boldsymbol{W}=(\bar{\boldsymbol{w}}_{y_1}^{\mathrm{T}},b_{y_1},\bar{\boldsymbol{w}}_{y_2}^{\mathrm{T}},b_{y_2},\cdots,\bar{\boldsymbol{w}}_{y_Q}^{\mathrm{T}},b_{y_Q})^{\mathrm{T}}$;类别标记 y_j 由一个长度为 Q 的列向量表示,第 j 个元素为 1,其他元素为 0,即 $\boldsymbol{y}_j=(\underbrace{0,\cdots,0}_{j},1,0,\cdots,0)^{\mathrm{T}}$;$\boldsymbol{x}_i$ 写成增广矩阵的形式,即 $\boldsymbol{x}_i=\begin{pmatrix}\boldsymbol{x}_i\\1\end{pmatrix}$.这样书写的好处是可以将潜变量函数 $f_{y_j}(\boldsymbol{x})$ 写成统一的向量乘积的形式 $f_{y_j}(\boldsymbol{x})=\bar{\boldsymbol{w}}_{y_j}^{\mathrm{T}}\boldsymbol{x}+b_{y_j}=\boldsymbol{W}^{\mathrm{T}}(\boldsymbol{y}_j\otimes\boldsymbol{x}_i)$,而且也便于用向量(矩阵)的形式表示目标函数 $Z(\boldsymbol{W})$ 关于 $\boldsymbol{W}$ 的一阶和二阶导数,其中 $\otimes$ 表示克罗克内积(Kroneckerproduct)[16].因此,目标函数 $Z(\boldsymbol{W})$ 可以重新写成如下的形式:

$$\boldsymbol{Z}(\boldsymbol{W})=\frac{1}{l}\sum_{i=1}^{n}\ln\Big(\sum_{\boldsymbol{y}_k\in Y_i}\mathrm{e}^{l\boldsymbol{W}^{\mathrm{T}}(\boldsymbol{y}_k\otimes\boldsymbol{x}_i)}\Big)-\sum_{i=1}^{n}\ln\Big(\sum_{j=1}^{Q}\mathrm{e}^{\boldsymbol{W}^{\mathrm{T}}(\boldsymbol{y}_j\otimes\boldsymbol{x}_i)}\Big)-\frac{\rho}{2}\boldsymbol{W}^{\mathrm{T}}\bar{\boldsymbol{E}}\boldsymbol{W}, \tag{10}$$

其中,$\bar{\boldsymbol{E}}$ 为将与 $\boldsymbol{W}$ 同价的单位矩阵的与 $\{b_{y_j}\mid j=1,2,\cdots,Q\}$ 对应的对角元素置为 0 后所得的矩阵.对 $\boldsymbol{Z}(\boldsymbol{W})$ 分别求一阶和二阶导数,可得:

$$\begin{aligned}\nabla\boldsymbol{Z}(\boldsymbol{W})=&\sum_{i=1}^{n}\sum_{\boldsymbol{y}_k\in Y_i}\mathrm{e}^{l\boldsymbol{W}^{\mathrm{T}}(\boldsymbol{y}_k\otimes\boldsymbol{x}_i)}(\boldsymbol{y}_k\otimes\boldsymbol{x}_i)\Big/\sum_{\boldsymbol{y}_k\in Y_i}\mathrm{e}^{l\boldsymbol{W}^{\mathrm{T}}(\boldsymbol{y}_k\otimes\boldsymbol{x}_i)}-\sum_{i=1}^{n}\sum_{j=1}^{Q}\mathrm{e}^{\boldsymbol{W}^{\mathrm{T}}(\boldsymbol{y}_j\otimes\boldsymbol{x}_i)}(\boldsymbol{y}_j\otimes\boldsymbol{x}_i)\Big/\sum_{j=1}^{Q}\mathrm{e}^{\boldsymbol{W}^{\mathrm{T}}(\boldsymbol{y}_j\otimes\boldsymbol{x}_i)}-\rho\bar{\boldsymbol{E}}\boldsymbol{W},\end{aligned} \tag{11}$$

$$\begin{aligned}\nabla\nabla\boldsymbol{Z}(\boldsymbol{W})=&\sum_{i=1}^{n}\Big(\sum_{\boldsymbol{y}_k\in Y_i}l\mathrm{e}^{l\boldsymbol{W}^{\mathrm{T}}(\boldsymbol{y}_k\otimes\boldsymbol{x}_i)}\cdot((\boldsymbol{y}_k\boldsymbol{y}_k^{\mathrm{T}})\otimes(\boldsymbol{x}_i\boldsymbol{x}_i^{\mathrm{T}}))\Big)\Big/\Big(\sum_{\boldsymbol{y}_k\in Y_i}\mathrm{e}^{l\boldsymbol{W}^{\mathrm{T}}(\boldsymbol{y}_k\otimes\boldsymbol{x}_i)}\Big)-\\&\sum_{i=1}^{n}\Big(\Big(\sum_{\boldsymbol{y}_k\in Y_i}\mathrm{e}^{l\boldsymbol{W}^{\mathrm{T}}(\boldsymbol{y}_k\otimes\boldsymbol{x}_i)}(\boldsymbol{y}_k\otimes\boldsymbol{x}_i)\Big)\cdot\Big(\sum_{\boldsymbol{y}_k\in Y_i}l\mathrm{e}^{l\boldsymbol{W}^{\mathrm{T}}(\boldsymbol{y}_k\otimes\boldsymbol{x}_i)}(\boldsymbol{y}_k\otimes\boldsymbol{x}_i)\Big)^{\mathrm{T}}\Big)\Big/\Big(\Big(\sum_{\boldsymbol{y}_k\in Y_i}\mathrm{e}^{l\boldsymbol{W}^{\mathrm{T}}(\boldsymbol{y}_k\otimes\boldsymbol{x}_i)}\Big)^2\Big)-\\&\sum_{i=1}^{n}\frac{\sum\limits_{j=1}^{Q}\mathrm{e}^{\boldsymbol{W}^{\mathrm{T}}(\boldsymbol{y}_j\otimes\boldsymbol{x}_i)}((\boldsymbol{y}_j\boldsymbol{y}_j^{\mathrm{T}})\otimes(\boldsymbol{x}_i\boldsymbol{x}_i^{\mathrm{T}}))}{\sum\limits_{j=1}^{Q}\mathrm{e}^{\boldsymbol{W}^{\mathrm{T}}(\boldsymbol{y}_j\otimes\boldsymbol{x}_i)}}+\\&\sum_{i=1}^{n}\Big(\Big(\sum_{j=1}^{Q}\mathrm{e}^{\boldsymbol{W}^{\mathrm{T}}(\boldsymbol{y}_j\otimes\boldsymbol{x}_i)}(\boldsymbol{y}_j\otimes\boldsymbol{x}_i)\Big)\cdot\Big(\sum_{j=1}^{Q}\mathrm{e}^{\boldsymbol{W}^{\mathrm{T}}(\boldsymbol{y}_j\otimes\boldsymbol{x}_i)}(\boldsymbol{y}_j\otimes\boldsymbol{x}_i)\Big)^{\mathrm{T}}\Big)\Big/\Big(\Big(\sum_{j=1}^{Q}\mathrm{e}^{\boldsymbol{W}^{\mathrm{T}}(\boldsymbol{y}_j\otimes\boldsymbol{x}_i)}\Big)^2\Big)-\rho\bar{\boldsymbol{E}},\end{aligned} \tag{12}$$

假设对每个 i 都只有一个 $\boldsymbol{y}_s\in Y_i$,使得:$\boldsymbol{f}_{y_s}(\boldsymbol{x}_i)\equiv\max\limits_{\boldsymbol{y}_k\in Y_i}\boldsymbol{f}_{\boldsymbol{y}_k}(\boldsymbol{x}_i)=\max\limits_{\boldsymbol{y}_k\in Y_i}\boldsymbol{W}^{\mathrm{T}}(\boldsymbol{y}_k\otimes\boldsymbol{x}_i)$,即在$\{\boldsymbol{W}^{\mathrm{T}}(\boldsymbol{y}_k\otimes\boldsymbol{x}_i)\mid\boldsymbol{y}_k\in Y_i\}$中只有一个元素能达到最大值,则当 $l\to+\infty$时,$\nabla\boldsymbol{Z}(\boldsymbol{W})$,$\nabla\nabla\boldsymbol{Z}(\boldsymbol{W})$可以近似表示为

$$\nabla\boldsymbol{Z}(\boldsymbol{W})\approx\sum_{i=1}^{n}\Big(\bar{\boldsymbol{Y}}_i\otimes\boldsymbol{x}_i-\sum_{j=1}^{Q}\mathrm{e}^{\boldsymbol{W}^{\mathrm{T}}(\boldsymbol{y}_j\otimes\boldsymbol{x}_i)}(\boldsymbol{y}_j\otimes\boldsymbol{x}_i)\Big/\sum_{j=1}^{Q}\mathrm{e}^{\boldsymbol{W}^{\mathrm{T}}(\boldsymbol{y}_j\otimes\boldsymbol{x}_i)}\Big)-\rho\bar{\boldsymbol{E}}\boldsymbol{W}=\sum_{i=1}^{n}(\bar{\boldsymbol{Y}}_i\otimes\boldsymbol{x}_i-\boldsymbol{d}_i\otimes\boldsymbol{x}_i)-\rho\bar{\boldsymbol{E}}\boldsymbol{W},\tag{13}$$

$$\nabla\nabla\boldsymbol{Z}(\boldsymbol{W})\approx-\sum_{i=1}^{n}\Big(\sum_{j=1}^{Q}\mathrm{e}^{\boldsymbol{W}^{\mathrm{T}}(\boldsymbol{y}_j\otimes\boldsymbol{x}_i)}((\boldsymbol{y}_j\boldsymbol{y}_j^{\mathrm{T}})\otimes(\boldsymbol{x}_i\boldsymbol{x}_i^{\mathrm{T}}))\Big)\Big/\Big(\sum_{j=1}^{Q}\mathrm{e}^{\boldsymbol{W}^{\mathrm{T}}(\boldsymbol{y}_j\otimes\boldsymbol{x}_i)}\Big)+\sum_{i=1}^{n}\Big(\Big(\sum_{j=1}^{Q}\mathrm{e}^{\boldsymbol{W}^{\mathrm{T}}(\boldsymbol{y}_j\otimes\boldsymbol{x}_i)}(\boldsymbol{y}_j\otimes\boldsymbol{x}_i)\Big)\cdot\Big(\sum_{j=1}^{Q}\mathrm{e}^{\boldsymbol{W}^{\mathrm{T}}(\boldsymbol{y}_j\otimes\boldsymbol{x}_i)}(\boldsymbol{y}_j\otimes\boldsymbol{x}_i)\Big)^{\mathrm{T}}\Big)\Big/\Big(\Big(\sum_{j=1}^{Q}\mathrm{e}^{\boldsymbol{W}^{\mathrm{T}}(\boldsymbol{y}_j\otimes\boldsymbol{x}_i)}\Big)^2\Big)-\rho\bar{\boldsymbol{E}}=-\sum_{i=1}^{n}(\mathrm{diag}(\boldsymbol{d}_i)-\boldsymbol{d}_i\boldsymbol{d}_i^{\mathrm{T}})\otimes(\boldsymbol{x}_i\boldsymbol{x}_i^{\mathrm{T}})-\rho\bar{\boldsymbol{E}},\tag{14}$$

其中,$\bar{\boldsymbol{Y}}_i=\arg\max\limits_{\boldsymbol{y}_k\in Y_i}\{\boldsymbol{W}^{\mathrm{T}}(\boldsymbol{y}_k\otimes\boldsymbol{x}_i)\}$,$\boldsymbol{d}_i=(d_{i1},d_{i2},\cdots,d_{iQ})^{\mathrm{T}}$,$d_{is}=\dfrac{\mathrm{e}^{\boldsymbol{W}^{\mathrm{T}}(\boldsymbol{y}_s\otimes\boldsymbol{x}_i)}}{\sum\limits_{j=1}^{Q}\mathrm{e}^{\boldsymbol{W}^{\mathrm{T}}(\boldsymbol{y}_j\otimes\boldsymbol{x}_i)}}$ $(s=1,2,\cdots,Q)$,$\mathrm{diag}(\boldsymbol{d}_i)$表示以 $\boldsymbol{d}_i$ 的元素为对角元的对角矩阵.

下面证明 $\mathrm{diag}(\boldsymbol{d}_i)-\boldsymbol{d}_i\boldsymbol{d}_i^{\mathrm{T}}$ 是一个半正定矩阵.由于 $d_{ij}\geqslant0$ 且 $\sum\limits_{j=1}^{Q}d_{ij}=1$,所以对于任意向量 $\boldsymbol{a}=(a_1,a_2,\cdots,a_Q)^{\mathrm{T}}$,根据柯西不等式可以得到 $\boldsymbol{a}^{\mathrm{T}}\boldsymbol{d}_i\boldsymbol{d}_i^{\mathrm{T}}\boldsymbol{a}=\Big(\sum\limits_{j=1}^{Q}a_jd_{ij}\Big)^2=\Big(\sum\limits_{j=1}^{Q}(a_j\sqrt{d_{ij}})\sqrt{d_{ij}}\Big)^2\leqslant\Big(\sum\limits_{j=1}^{Q}a_j^2d_{ij}\Big)\times\Big(\sum\limits_{j=1}^{Q}d_{ij}\Big)=\Big(\sum\limits_{j=1}^{Q}a_j^2d_{ij}\Big)=\boldsymbol{a}^{\mathrm{T}}\mathrm{diag}(\boldsymbol{d}_i)\boldsymbol{a}$,从而有 $\boldsymbol{a}^{\mathrm{T}}(\mathrm{diag}(\boldsymbol{d}_i)-\boldsymbol{d}_i\boldsymbol{d}_i^{\mathrm{T}})\boldsymbol{a}\geqslant0$,因此 $\mathrm{diag}(\boldsymbol{d}_i)-\boldsymbol{d}_i\boldsymbol{d}_i^{\mathrm{T}}$ 是一个半正定矩阵.

由于 $\mathrm{diag}(\boldsymbol{d}_i)-\boldsymbol{d}_i\boldsymbol{d}_i^{\mathrm{T}}$,$\boldsymbol{x}_i\boldsymbol{x}_i^{\mathrm{T}}$,$\rho\bar{\boldsymbol{E}}$ 都是半正定矩阵,因此$\nabla\nabla\boldsymbol{Z}(\boldsymbol{W})$是一个半负定矩阵,这表明 $\boldsymbol{Z}(\boldsymbol{W})$是一个凹函数,因此可以利用通常的凸优化方法求其最大值点.需要强调的一点是"$\boldsymbol{Z}(\boldsymbol{W})$是一个凹函数"这个结论是在"对每个 i 都只有一个 $\boldsymbol{y}_s\in Y_i$ 使得 $\boldsymbol{f}_{y_s}(\boldsymbol{x}_i)\equiv\max\limits_{\boldsymbol{y}_k\in Y_i}\boldsymbol{f}_{\boldsymbol{y}_k}(\boldsymbol{x}_i)$"这个假设条件得到的,否则未必成立.由于这个假设条件的对立假设是一个小概率事件,因此在实际问题求解的过程中可以抛开这个假设不管,直接把 $\boldsymbol{Z}(\boldsymbol{W})$看作凹函数来求解.

2　模型求解

由于目标函数 $\boldsymbol{Z}(\boldsymbol{W})$是一个凹函数,因此很多常用的优化方法都可以用来求其最大值点.本文将建立 2 种模型求解方法,对于样本规模比较小的问题采用阻尼牛顿法来求解,对于样本规模比较大的问题将利用随机拟牛顿法来建立一种快速求解方法.

2.1　阻尼牛顿法

阻尼牛顿法的迭代公式如下:

$$\boldsymbol{W}^{k+1}=\boldsymbol{W}^k-\lambda_k(\nabla\nabla\boldsymbol{Z}(\boldsymbol{W}^k))^{-1}\nabla\boldsymbol{Z}(\boldsymbol{W}^k),\tag{15}$$

其中,$\nabla\boldsymbol{Z}(\boldsymbol{W})$,$\nabla\nabla\boldsymbol{Z}(\boldsymbol{W})$的详细表达式分别见式(13)(14),步长 λ_k 可以通过如下一维优化问题求得:$\lambda_k=\arg\max\limits_{\lambda}\boldsymbol{Z}(\boldsymbol{W}^k-\lambda(\nabla\nabla\boldsymbol{Z}(\boldsymbol{W}^k))^{-1}\nabla\boldsymbol{Z}(\boldsymbol{W}^k))\triangleq\arg\max\limits_{\lambda}\psi(\lambda)$.由于 $\psi(\lambda)$也是一个凹函数,所以我们通过利用二分法求解 $\psi(\lambda)$的导数 $\psi'(\lambda)$的零点方式来求解 λ_k,二分法的结束条件是二分区间的长度小于 0.000 01.

2.2　随机拟牛顿法

为了处理大数据问题,本文拟采用文献[17]提出的随机拟牛顿法来建立一种快速模型求解方法.为了便于描述,我们将式(9)(10)描述的最大值问题变换为最小值问题来求解:

$$\boldsymbol{W}=\arg\max_{W}\boldsymbol{Z}(\boldsymbol{W})=\arg\max_{W}\Big\{\frac{1}{l}\sum_{i=1}^{n}\ln\Big(\sum_{\boldsymbol{y}_k\in Y_i}\mathrm{e}^{l\boldsymbol{W}^{\mathrm{T}}(\boldsymbol{y}_k\otimes\boldsymbol{x}_i)}\Big)-\sum_{i=1}^{n}\ln\Big(\sum_{j=1}^{Q}\mathrm{e}^{\boldsymbol{W}^{\mathrm{T}}(\boldsymbol{y}_j\otimes\boldsymbol{x}_i)}\Big)-\frac{\rho}{2}\boldsymbol{W}^{\mathrm{T}}\bar{\boldsymbol{E}}\boldsymbol{W}\Big\}=\arg\min_{W}\Big\{\frac{1}{n}\Big(\sum_{i=1}^{n}\Big(-\frac{1}{l}\ln\Big(\sum_{\boldsymbol{y}_k\in Y_i}\mathrm{e}^{l\boldsymbol{W}^{\mathrm{T}}(\boldsymbol{y}_k\otimes\boldsymbol{x}_i)}\Big)+\ln\Big(\sum_{j=1}^{Q}\mathrm{e}^{\boldsymbol{W}^{\mathrm{T}}(\boldsymbol{y}_j\otimes\boldsymbol{x}_i)}\Big)+\frac{\bar{\rho}}{2}\boldsymbol{W}^{\mathrm{T}}\bar{\boldsymbol{E}}\boldsymbol{W}\Big)\Big)\Big\}\triangleq\arg\min_{W}\bar{\boldsymbol{Z}}(\boldsymbol{W})\triangleq\arg\min_{W}\frac{1}{n}\sum_{i=1}^{n}\boldsymbol{z}(\boldsymbol{W},\boldsymbol{x}_i,Y_i),\tag{16}$$

其中,$\bar{\rho}=\rho/n$,$\boldsymbol{z}(\boldsymbol{W},\boldsymbol{x}_i,Y_i)=-\dfrac{1}{l}\ln\Big(\sum\limits_{\boldsymbol{y}_k\in Y_i}\mathrm{e}^{l\boldsymbol{W}^{\mathrm{T}}(\boldsymbol{y}_k\otimes\boldsymbol{x}_i)}\Big)+\ln\Big(\sum\limits_{j=1}^{Q}\mathrm{e}^{\boldsymbol{W}^{\mathrm{T}}(\boldsymbol{y}_j\otimes\boldsymbol{x}_i)}\Big)+\dfrac{\bar{\rho}}{2}\boldsymbol{W}^{\mathrm{T}}\bar{\boldsymbol{E}}\boldsymbol{W}$.根据式(13)(14),可以得到 $\boldsymbol{z}(\boldsymbol{W},\boldsymbol{x}_i,Y_i)$的一阶、二阶导数为

$$\nabla\boldsymbol{z}(\boldsymbol{W},\boldsymbol{x}_i,Y_i)=(\boldsymbol{d}_i-\bar{\boldsymbol{Y}}_i)\otimes\boldsymbol{x}_i+\rho\bar{\boldsymbol{E}}\boldsymbol{W},\tag{17}$$

$$\nabla\nabla\bar{\boldsymbol{z}}(\boldsymbol{W},\boldsymbol{x}_i,Y_i)=(\mathrm{diag}(\boldsymbol{d}_i)-\boldsymbol{d}_i\boldsymbol{d}_i^{\mathrm{T}})\otimes(\boldsymbol{x}_i\boldsymbol{x}_i^{\mathrm{T}})+\bar{\rho}\bar{\boldsymbol{E}}.\tag{18}$$

对于训练样本规模比较大的问题,计算 $\bar{\boldsymbol{Z}}(\boldsymbol{W})$ 的一阶、二阶导数通常比较费时,因此在随机拟牛顿法中,每一步迭代都随机选取训练数据集的一个子集来近似计算 $\bar{\boldsymbol{Z}}(\boldsymbol{W})$ 的一、二阶导数,即:

$$\nabla\bar{\boldsymbol{Z}}(\boldsymbol{W})\approx\frac{1}{|S|}\sum_{(\boldsymbol{x}_i,Y_i)\in S}\nabla\bar{z}(\boldsymbol{W},\boldsymbol{x}_i,Y_i),\quad(19)$$

$$\nabla\nabla\bar{\boldsymbol{Z}}(\boldsymbol{W})\approx\frac{1}{|S_{\boldsymbol{H}}|}\sum_{(\boldsymbol{x}_i,Y_i)\in S_{\boldsymbol{H}}}\nabla\nabla\bar{z}(\boldsymbol{W},\boldsymbol{x}_i,Y_i),\quad(20)$$

其中,$S,S_{\boldsymbol{H}}$ 为随机选取的训练数据集 D 的子集.

随机拟牛顿法[17]的迭代为

$$\boldsymbol{W}^{k+1}=\boldsymbol{W}^k-\alpha_k\boldsymbol{H}_t\,\nabla\bar{\boldsymbol{Z}}(\boldsymbol{W}^k),\quad(21)$$

其中,α_k 为步长,α_k 的选取会对算法的收敛速度有一定的影响,本文采用与文献[17]一样的定义方式,即 $\alpha_k=\beta/k$(β 为事先给定的常数);$\boldsymbol{H}_t$ 为海森矩阵的近似矩阵,通过与有限储存拟牛顿方法(limited-memory Broyden-Fletcher-Goldfarb-Shanno Broyden, L-BFGS)[18]中一样的方法来求得. 算法 1、算法 2 给出了随机拟牛顿法的算法流程图,关于该方法的详细描述见文献[17].

算法 1. 随机拟牛顿法的算法流程.

输入:$D,\bar{\rho},\beta,b,b_{\boldsymbol{H}},M,L$,$D$ 训练数据集,$\bar{\rho}$ 正则化项的权重系数,β 步长 α_k 的常系数,$b,b_{\boldsymbol{H}}$ 计算目标函数的一、二阶导数时选取的训练子集 $S,S_{\boldsymbol{H}}$ 的样本个数,M 设定的计算 $\boldsymbol{H}_t$ 的存储长度,L 控制主循环迭代 L 次更新 $\boldsymbol{H}_t$ 一次;

输出:$\boldsymbol{W}$.

① 初始化:$\boldsymbol{W}^1=\boldsymbol{1}$ 为全 1 向量,$\bar{\boldsymbol{W}}_t=\boldsymbol{0}$ 为零向量,$t=-1$;
② for $k=1,2,\cdots,K-1$
③ 随机选取训练子集 $S\subset D$,根据式(19)计算目标函数在 $\boldsymbol{W}^k$ 处的梯度值 $\nabla\bar{\boldsymbol{Z}}(\boldsymbol{W}^k)$;
④ 更新 $\bar{\boldsymbol{W}}_t=\bar{\boldsymbol{W}}_t+\boldsymbol{W}^k$;
⑤ if $k\leqslant 2L$
⑥ 更新 $\boldsymbol{W}^{k+1}=\boldsymbol{W}^k-\alpha_k\,\nabla\bar{\boldsymbol{Z}}(\boldsymbol{W}^k)$;
⑦ else
⑧ 根据算法 2 计算 $\boldsymbol{H}_t$;
⑨ 更新 $\boldsymbol{W}^{k+1}=\boldsymbol{W}^k-\alpha_k\boldsymbol{H}_t\,\nabla\bar{\boldsymbol{Z}}(\boldsymbol{W}^k)$;
⑩ end if
⑪ if $\mathrm{mod}(k,L)=0$
⑫ 更新 $t=t+1,\bar{\boldsymbol{W}}_t=\bar{\boldsymbol{W}}_t/L$;
⑬ if $t>0$
⑭ 随机选取训练子集 $S_{\boldsymbol{H}}\subset D$,
⑮ 根据式(20)计算目标函数在 $\bar{\boldsymbol{W}}_t$ 处的二阶导数 $\nabla\nabla\bar{\boldsymbol{Z}}(\bar{\boldsymbol{W}}_t)$;
⑯ 计算 $\bar{\boldsymbol{s}}_t=\bar{\boldsymbol{W}}_t-\bar{\boldsymbol{W}}_{t-1},\bar{\boldsymbol{y}}_t=\nabla\nabla\bar{\boldsymbol{Z}}(\bar{\boldsymbol{W}}_t)(\bar{\boldsymbol{W}}_t-\bar{\boldsymbol{W}}_{t-1})$;
⑰ end if
⑱ 更新 $\bar{\boldsymbol{W}}_t=\boldsymbol{0}$;
⑲ end if
⑳ end for

算法 2. 海森矩阵 $\boldsymbol{H}_t$ 的计算流程.

输入:$\{\bar{\boldsymbol{s}}_j,\bar{\boldsymbol{y}}_j\,|\,j=t-\min\{t,M\}+1,\cdots,t\}$;

输出:$\boldsymbol{H}_t$.

① 初始化:$\boldsymbol{H}=(\bar{\boldsymbol{s}}_t^{\mathrm{T}}\bar{\boldsymbol{y}}_t)/(\bar{\boldsymbol{y}}_t^{\mathrm{T}}\bar{\boldsymbol{y}}_t)\boldsymbol{I}$,$\boldsymbol{I}$ 为单位矩阵;
② for $j=t-\min\{t,M\}+1,\cdots,t$
③ 计算 $\rho_j=1/(\bar{\boldsymbol{y}}_j^{\mathrm{T}}\bar{\boldsymbol{s}}_j)$;
④ 更新 $\boldsymbol{H}=(\boldsymbol{I}-\rho_j\bar{\boldsymbol{s}}_j\bar{\boldsymbol{y}}_j^{\mathrm{T}})\boldsymbol{H}(\boldsymbol{I}-\rho_j\bar{\boldsymbol{y}}_j\bar{\boldsymbol{s}}_j^{\mathrm{T}})+\rho_j\bar{\boldsymbol{s}}_j\bar{\boldsymbol{s}}_j^{\mathrm{T}}$;
⑤ end for;
⑥ $\boldsymbol{H}_t=\boldsymbol{H}$.

3 仿真实验

本文的主要创新在于使用了最大值似然函数和建立了面向大数据问题的快速偏标记学习算法,因此,下面将分别从这 2 个方面验证所建算法的性能. 为了便于区分,在下面的仿真实验中我们用 PLLOG-Max 表示本文建立的基于最大值似然函数的偏标记学习算法,PLLOG-Max-DN 表示基于阻尼牛顿法求解的算法,PLLOG-Max-SQN 表示基于随机拟牛顿法求解的算法.

3.1 验证最大值似然函数对算法性能的影响

为了验证最大值似然函数对算法性能的影响,我们先将 PLLOG-Max-DN 算法与 CLPL 算法[2]以及使用由式(1)(2)定义的 2 种均值似然函数所建立的算法(分别简记为 PLLOG-Naive,PLLOG-Average 算法)进行了比较. 对于 PLLOG-Average 算法,也是采用阻尼牛顿法对模型进行求解的;对于 PLLOG-Naive 算法,由于目标函数不是一个凸(或凹)函数,因此采用文献[19]提出的 CCCP 算法对其进行求解,这 2 个算法的其他推导过程都是与 PLLOG-Max-DN 算法一样的. PLLOG-Max-DN, PLLOG-Naive, PLLOG-Average 算法都只有一个需要给定的模型参数,即正则化项的系数 ρ,在本节的仿真实验中,ρ 是通过在训练集上利用 3 折交叉验证法选

取得到的,CLPL 算法的所有参数都是按照文献[2]的建议设置的.我们在 5 个 UCI 数据集[20]和 2 个真实偏标记学习数据集[3]上对这 4 个算法进行了比较,这 7 个数据集的详细信息如表 1 所示:

Table 1　The Data Sets for Validating the Performance of Max-likelihood Based Algorithm

表 1　验证基于最大值似然函数的算法所用数据集

Source of Data Sets	Data Sets	Number of Samples	Number of Characteristics	Number of Classes	Number of Partial Label Samples		
					Minimum	Maximum	Average
UCI	Abalone	4 177	7	29			
	CTG	2 126	21	10			
	Yeast	1 484	8	10			
	Image Segmentation(Segment)	2 310	19	7			
	Movement	360	90	15			
Real World	BirdSong	4 998	38	13	1	4	2.18
	MSRCv2	1 758	48	23	1	7	3.16

由于 UCI 数据集是标准的传统分类数据集,我们采用了 2 个控制参数 p,r 来把它们变换为了偏标记数据集,其中 p 表示偏标记样本(即 $|Y_i|>1$)在整个样本集中的比例,r 表示偏标记样本的除真实标记以外的候选标记个数,即 $r=|Y_i|-1$.变换过程如下:对于给定的每一对参数(p,r),先从原始的 UCI 数据集中随机选取 pn 个样本(n 为样本总数),然后对于每一个被选取的样本,随机地从它的真实标记以外的类别标记中选取 r 个与真实标记一起构成该样本的候选标记.表 2 给出了 4 个算法在由不

Table 2　The Accuracy of Each Algorithm on the UCI Data Sets

表 2　各个算法在各个 UCI 数据集上预测精度　　%

Data Sets	Algorithm	$r=1$			$r=2$			$r=3$		
		$p=0.15$	$p=0.45$	$p=0.75$	$p=0.15$	$p=0.45$	$p=0.75$	$p=0.15$	$p=0.45$	$p=0.75$
Abalone	PLLOG-Average	17.74	17.34	16.74	17.96	17.01	16.24	17.86	17.17	16.42
	PLLOG-Max-DN	**18.42**	**19.68**	**19.96**	**19.21**	**19.64**	**20.70**	**19.60**	**20.97**	**23.08**
	PLLOG-Naive	18.39	18.53	18.37	18.41	18.16	20.00	18.35	18.24	19.73
	CLPL	23.63	23.63	23.51	23.73	23.50	22.51	23.69	23.46	22.57
CTG	PLLOG-Average	67.44	64.66	60.86	67.75	64.16	62.67	66.99	65.03	60.85
	PLLOG-Max-DN	**68.36**	**69.29**	**69.43**	**68.49**	**70.00**	**69.21**	68.02	**69.23**	**68.98**
	PLLOG-Naive	68.31	68.39	68.28	67.96	69.06	68.42	**68.16**	68.57	67.54
	CLPL	65.89	65.24	65.23	66.39	65.22	65.35	66.27	65.18	64.66
Yeast	PLLOG-Average	51.16	51.18	50.25	51.55	51.59	49.58	51.20	51.74	49.49
	PLLOG-Max-DN	**52.02**	**52.48**	**52.77**	**51.88**	**52.24**	51.09	**51.89**	**52.14**	49.57
	PLLOG-Naive	51.28	51.63	51.19	51.61	52.00	**51.12**	51.28	51.99	**51.34**
	CLPL	52.71	52.41	53.21	53.12	52.84	51.17	52.13	53.11	51.15
Segment	PLLOG-Average	**89.42**	87.09	85.40	88.19	86.80	85.83	**88.75**	87.58	85.59
	PLLOG-Max-DN	87.52	**87.75**	88.03	**87.55**	**87.87**	**88.53**	87.61	87.38	**88.66**
	PLLOG-Naive	87.48	87.74	**88.26**	87.53	87.85	88.28	87.76	**87.61**	88.41
	CLPL	82.94	82.68	82.60	82.81	82.64	82.30	82.87	82.42	82.67
Movement	PLLOG-Average	63.03	64.21	54.60	64.57	61.87	57.06	61.90	61.96	57.61
	PLLOG-Max-DN	**66.43**	**67.82**	66.11	65.76	**66.72**	64.21	**64.31**	64.03	62.83
	PLLOG-Naive	66.24	67.18	**66.75**	**65.92**	66.33	**65.00**	64.29	**65.71**	**65.25**
	CLPL	53.98	54.11	49.83	54.91	49.39	46.61	52.96	49.20	43.63

同的控制参数(p,r)生成的数据集上的预测精度,每个数据集上的结果都是5次10折交叉验证的平均结果.表2中黑体显示的是PLLOG-Max-DN,PLLOG-Naive,PLLOG-Average三个算法中最好的结果,下划线表示的是PLLOG-Max-DN算法和CLPL算法之间相比最好的结果.可以看出,就PLLOG-Max-DN,PLLOG-Naive,PLLOG-Average三个算法相比,PLLOG-Max-DN算法在生成的45个控制数据集中的33个上取得了最好的结果,PLLOG-Naive算法在10个数据集上取得了最好结果,而PLLOG-Average算法只在2个数据集上取得了最好结果,而且对于那些没有取得最好结果的数据集,PLLOG-Max-DN算法的结果也基本与最好结果相差很少,这就表明最大值似然函数要优于其他2种均值似然函数;而PLLOG-Max-DN和CLPL算法相比,PLLOG-Max-DN算法在45个控制数据集中的29个上取得了最好的结果,CLPL算法在16个上取得了最好结果,而且其中在6个上是以微弱的优势胜出,因此PLLOG-Max-DN算法要优于CLPL算法.

表3给出了4个算法在2个真实数据集上的预测精度,遗憾的是PLLOG-Max-DN算法只在一个数据集上取得了最好的结果,主要原因是BirdSong数据集是从一个多标记学习问题中采集的,可能BirdSong数据集中的一部分样本的候选标记本身都是样本的真实标记,从而影响了算法的性能.

Table 3 The Accuracy of Each Algorithm on the Real-world Data Sets

表3 各个算法在真实数据集上预测精度 %

Data Sets	PLLOG-Average	PLLOG-Max-DN	PLLOG-Naive	CLPL
Birdsong	57.0±0.2	59.3±0.2	65.3±0.1	64.0±0.1
MSRCv2	36.6±0.5	42.2±0.5	40.6±0.4	41.3±0.3

3.2 验证快速模型求解算法的性能

本节将在5个规模比较大的UCI数据集上对本文建立的快速偏标记学习算法PLLOG-Max-SQN的性能进行评估,这些UCI数据集的详细信息如表4所示.

所有的实验结果都是通过随机选取3/4的样本作为训练数据,1/4的样本作为测试数据,然后重复进行5次实验得到的平均结果.由于PLLOG-Max-SQN算法的参数比较多,所以我们首先在由Letter数据集生成的一个偏标记数据集(控制参数$p=0.6$,$r=3$)上观测了这些参数的不同取值对预测精度的影响.图1给出了在固定了其他参数后(参数固定后的值$\bar{\rho}=0.005$,$\beta=2$,$b=50$,$b_H=50$,$M=20$,$L=20$)各个参数的不同取值与算法的预测精度之间的关系,对于参数b,b_H,除取值为10时,预测精度会随着迭代次数的变化有较大变化外,取值为其他值时变化都不大;对于参数L,M,取值太大和太小效果都不好,$L=20$时效果最好,M的取值为5或者10时效果最好;对于参数β,没有明显的规律,$\beta=2$时效果较好.由于参数选择是一件比较费力的事情,因此根据在Letter数据集上的分析结果,我们将PLLOG-Max-SQN算法的β,b,b_H,M,L五个参数默认设置为$\beta=2$,$b=50$,$b_H=50$,$M=10$,$L=20$,对于正则化项系数$\bar{\rho}$,由于该参数对算法的精度影响比较大,我们是利用交叉验证法选取的.

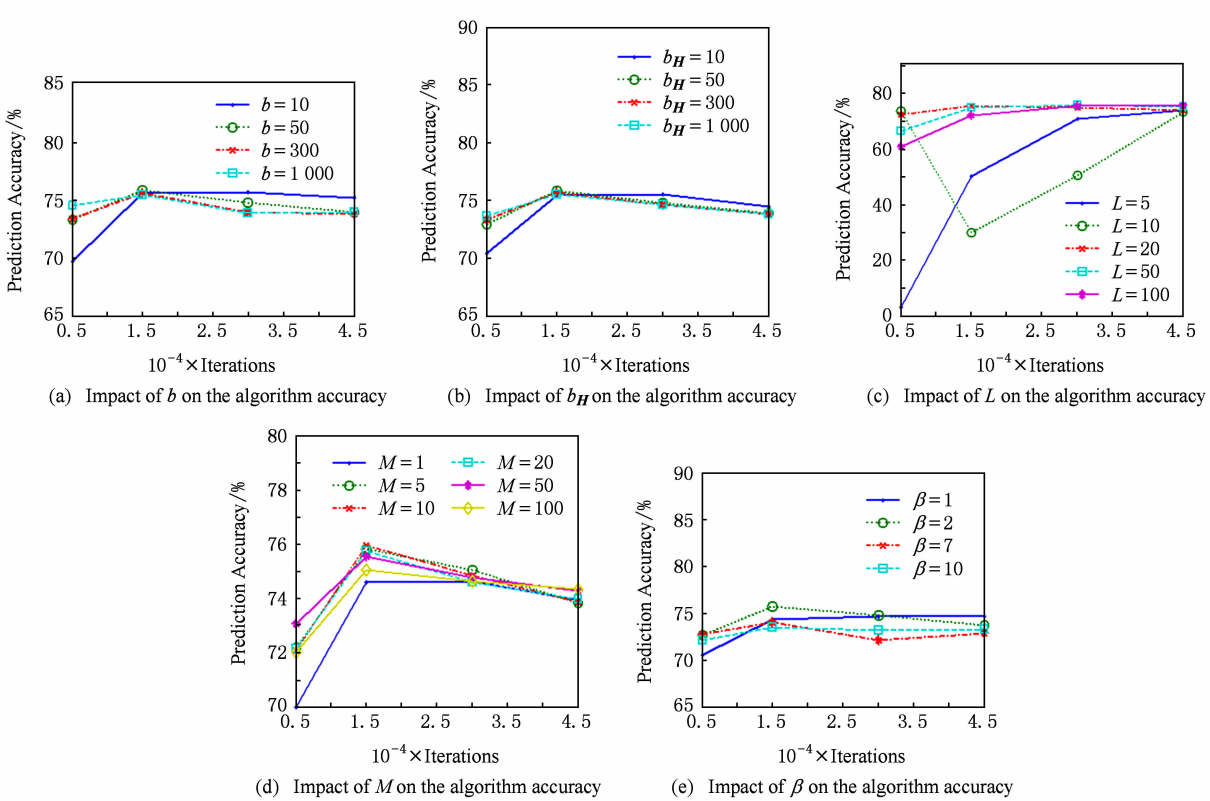
Table 4 The UCI Data Sets for Validating the Performance of the Fast Algorithm for Solving the Proposed Model

表4 验证快速模型求解算法所用UCI数据集

Data Sets	Number of Samples	Number of Characteristics	Number of Classes
Letter	20 000	16	26
Shuttle	58 000	9	7
Connect4	67 557	42	3
Covtype	581 012	54	7
Poker-hand	1 025 010	10	10

表5给出了PLLOG-Max-SQN算法按照以上的参数设定方法在各个UCI数据集上的平均预测精度和计算时间,由于目前还没有其他的快速偏标记学习算法,因此只将PLLOG-Max-SQN算法与我们建立的PLLOG-Max-DN算法进行了比较,这2个算法的结束条件是PLLOG-Max-SQN的迭代次数是$2n/3$(n训练样本个数),PLLOG-Max-DN的迭代次数是100次,所有结果都是在CPU主频为2.5 GHz、内存为8 GB的笔记本电脑上运行得到的.

从表5可以看出,2个算法的预测精度基本都相当,但是计算时间上PLLOG-Max-SQN算法要远远快于PLLOG-Max-DN算法.为了进一步明确2个算法的预测精度之间究竟有无明显差距,我们利用5%显著性水平的t检验方法对2个算法在各个数据集上的结果进行了分析,结果表明除在2个控制数据集Shuttle($p=0.6$,$r=1$),Connect4($p=0.6$,$r=1$)上PLLOG-Max-DN算法比PLLOG-Max-SQN算法的精度略高以外,在其他的16个控制数据集上,这2个算法的结果在统计意义上是一样的.



(a) Impact of b on the algorithm accuracy　(b) Impact of b_H on the algorithm accuracy　(c) Impact of L on the algorithm accuracy

(d) Impact of M on the algorithm accuracy　(e) Impact of β on the algorithm accuracy

Fig. 1　The performance of PLLOG-Max-SQN algorithm on Letter data set when varying the value of each parameter.

图 1　每个参数取不同值时 PLLOG-Max-SQN 算法在 Letter 数据集上的性能

Table 5　The Performance of PLLOG-Max-DN and PLLOG-Max-SQN Algorithms on Large-scale UCI Data Sets

表 5　PLLOG-Max-DN 与 PLLOG-Max-SQN 算法在大规模 UCI 数据集上的性能比较

Data Sets	Algorithm	Classification Accuracy/%(Running Time/min)			
		$r=1$		$r=3$	
		$p=0.3$	$p=0.6$	$p=0.3$	$p=0.6$
Letter	PLLOG-Max-SQN	75.12(0.22)	75.08(0.20)	75.20(0.22)	75.30(0.21)
	PLLOG-Max-DN	74.84(4.14)	75.35(4.20)	74.91(4.13)	75.59(4.35)
Shuttle	PLLOG-Max-SQN	92.34(0.22)	92.06(0.22)	92.30(0.22)	92.18(0.22)
	PLLOG-Max-DN	92.54(1.17)	92.48(1.44)	92.51(1.12)	92.37(1.44)
Connect4	PLLOG-Max-SQN	64.49(0.23)	64.03(0.23)		
	PLLOG-Max-DN	64.99(0.78)	63.85(0.86)		
Covtype	PLLOG-Max-SQN	59.81(1.77)	60.16(1.72)	59.99(1.72)	59.37(1.73)
	PLLOG-Max-DN	60.17(32.64)	60.14(34.48)	60.18(32.79)	59.65(37.29)
Poker-hand	PLLOG-Max-SQN	50.11(4.84)	50.21(4.81)	50.08(4.78)	50.12(4.72)
	PLLOG-Max-DN	50.11(31.42)	50.21(32.05)	50.08(31.58)	50.12(31.74)

4　结束语

损失函数的设计体现了算法关于学习框架本质的刻画,是影响算法泛化性能的关键因素之一.最大值损失函数可以较均值损失函数更好地描述偏标记学习框架下的样本与标记之间的关系,但是由于最大值损失函数是一个非光滑函数,导致建立模型的目标函数通常也是一个难于求解的非光滑函数,针对这个问题,本文通过引入凝聚函数,将基于最大值损失函数的偏标记学习模型的目标函数转换成了一种易于求解的光滑凹函数.此外,针对已有算法不能

处理大数据的问题,本文通过引入随机拟牛顿法对模型进行求解,建立一种快速偏标记学习算法,可以有效处理大数据.考虑到本文建立的算法是一种线性算法,有时不能很好地处理非线性问题,下一步我们将尝试以支持向量机或者高斯过程模型为建模工具,建立一种基于核方法的快速偏标记学习算法.

参 考 文 献

[1] Zhang Minling. Research on partial label learning [J]. Journal of Data Acquisition & Processing, 2015, 30(1): 77-87 (in Chinese)
(张敏灵. 偏标记学习研究综述[J]. 数据采集与处理, 2015, 30(1): 77-87)

[2] Cour T, Sapp B, Taskar B. Learning from partial labels [J]. Journal of Machine Learning Research, 2011, 12: 1501-1536

[3] Zhang Minling. Disambiguation-free partial label learning [C] //Proc of the 14th SIAM Int Conf on Data Mining. Philadelphia, PA: SIAM, 2014: 37-45

[4] Luo J, Orabona F. Learning from candidate labeling sets [C] //Advances in Neural Information Processing Systems. Montreal: Neural Information Processing System Foundation, 2010: 1504-1512

[5] Hüllermeier E, Beringer J. Learning from ambiguously labeled examples [J]. Intelligent Data Analysis, 2006, 10(5): 419-439

[6] Liu L, Dietterich T. A conditional multinomial mixture model for superset label learning [C] //Advances in Neural Information Processing Systems. Montreal: Neural Information Processing System Foundation, 2012: 557-565

[7] Zeng Z, Xiao S, Jia K, et al. Learning by associating ambiguously labeled images [C] //Proc of the 26th IEEE Int Conf on Computer Vision and Pattern Recognition. Piscataway, NJ: IEEE, 2013: 708-715

[8] Grandvalet Y, Bengio Y. Learning from partial labels with minimum entropy, 2004s-28 [R]. Montreal: Center for Interuniversity Research and Analysis of Organizations, 2004

[9] Fung G, Dundar M, Krishnapuram B, et al. Multiple instance learning for computer aided diagnosis [C] // Advances in Neural Information Processing Systems. Cambridge, MA: MIT Press, 2007: 425-432

[10] Grandvalet Y. Logistic regression for partial labels [C] // Proc of the 9th Int Conf on Information Processing and Management of Uncertainty in Knowledge-Based Systems. Annecy, Haute-Savoie: Institute for the Physics and Mathematics of the Universe (IPMU), 2002: 1935-1941

[11] Jin R, Ghahramani Z. Learning with multiple labels [C] // Advances in Neural Information Processing Systems. Montreal: Neural Information Processing System Foundation, 2002: 897-904

[12] Côme E, Oukhellou L, Denoeux T, et al. Learning from partially supervised data using mixture models and belief functions [J]. Pattern Recognition, 2009, 42(3): 334-348

[13] Nguyen N, Caruana R. Classification with partial labels [C] //Proc of the 14th ACM SIGKDD Int Conf on Knowledge Discovery and Data Mining. New York: Association for Computing Machinery, 2008: 381-389

[14] Li C, Zhang J, Chen Z. Structured output learning with candidate labels for local parts [C] //Proc of the European Conf on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECMLPKDD 2013). Berlin: Springer, 2013

[15] Li X. An aggregate function method for nonlinear programming [J]. Science in China Series A-Mathematics, 1991, 34(12): 1467-1473

[16] Horn R, Johnson C. Topics in Matrix Analysis [M]. Cambridge, UK: Cambridge University Press, 1991: 239-297

[17] Byrd R, Hansen S, Nocedal J, et al. A stochastic quasi-Newton method for large-scale optimization [OL]. [2015-02-18]. http://arxiv.org/pdf/1401.7020v2.pdf

[18] Nocedal J, Wright S. Numerical Optimization [M]. 2nd ed. Berlin: Springer, 1999: 195-204

[19] Yuille A, Rangarajan A. The concave-convex procedure [J]. Neural Computation, 2003, 15(4): 915-936

[20] Bache K, Lichman M. UCI machine learning repository [OL]. 2013 [2015-03-10]. http://archive.ics.uci.edu/ml

Zhou Yu, born in 1982. PhD candidate. Her research interests include machine learning and data mining.

He Jianjun, born in 1983. PhD and lecturer. His research interests include machine learning and pattern recognition.

Gu Hong, born in 1961. Professor and PhD supervisor. His research interests include machine learning, big data and bioinformatics.

Zhang Junxing, born in 1969. PhD and professor. His research interests include data mining and speech signal processing.