软件学报 ISSN 1000-9825, CODEN RUXUEW
*Journal of Software*,2014,25(10):2282−2292 [doi: 10.13328/j.cnki.jos.004504]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

# 求解 AUC 优化问题的对偶坐标下降方法*

姜纪远, 陶 卿, 高乾坤, 储德军

(中国人民解放军 陆军军官学院 十一系,安徽 合肥 230031)

通讯作者: 姜纪远, E-mail: jyjianggle@gmail.com

**摘 要**: AUC 被广泛作为衡量不平衡数据分类性能的评价标准.与二分类问题不同,AUC 问题的损失函数由来自两个不同类别的样本对组成.如何提高其实际收敛速度,是一个值得研究的问题.目前的研究结果表明:使用 reservoir sampling 技术的在线方法(OAM)表现出很好的 AUC 性能,但 OAM 仍存在诸如收敛速度慢、参数选择复杂等缺点.针对 AUC 优化问题的对偶坐标下降(AUC-DCD)方法进行了系统的研究,给出 3 种算法,即 AUC-SDCD,AUC-SDCDperm 和 AUC-MSGD,其中,AUC-SDCD 和 AUC-SDCDperm 与样本数目有关,AUC-MSGD 与样本数目无关.理论分析指出,OAM 是 AUC-DCD 的一种特殊情形.实验结果表明,AUC-DCD 在 AUC 性能和收敛速度两方面均优于 OAM.研究结果表明,AUC-DCD 是求解 AUC 优化问题的首选方法.

**关键词**: 机器学习;优化方法;AUC;对偶坐标下降;支持向量机

**中图法分类号**: TP18

中文引用格式: 姜纪远,陶卿,高乾坤,储德军.求解 AUC 优化问题的对偶坐标下降方法.软件学报,2014,25(10):2282−2292. http://www.jos.org.cn/1000-9825/4504.htm

英文引用格式: Jiang JY, Tao Q, Gao QK, Chu DJ. Dual coordinate descent method for solving AUC optimization problem. Ruan Jian Xue Bao/Journal of Software, 2014,25(10):2282−2292 (in Chinese). http://www.jos.org.cn/1000-9825/4504.htm

## Dual Coordinate Descent Method for Solving AUC Optimization Problem

JIANG Ji-Yuan, TAO Qing, GAO Qian-Kun, CHU De-Jun

(11th Department, Army Officer Academy of PLA, Hefei 230031,China)

Corresponding author: JIANG Ji-Yuan, E-mail: jyjianggle@gmail.com

**Abstract**: AUC is widely used as a measure for the imbalanced classification problems. The AUC loss problem is a pairwise function between two instances from different classes, which is obviously different from that in standard binary classifications. How to improve its real convergence speed is an interesting problem. Recent study shows that the online method (OAM) using the reservoir sampling technique has better performance. However, there exist some shortcomings such as slow convergence rate and difficult parameter selection. This paper conducts a systematic investigation for solving AUC optimization problem by using the dual coordinate descent methods (AUC-DCD). It presents three kinds of algorithms: AUC-SDCD, AUC-SDCDperm and AUC-MSGD, where the first two algorithms depend on the size of training set while the last does not. Theoretical analysis shows that OAM is a special case of the AUC-DCD. Experimental results show that AUC-DCD is better than OAM on the AUC performance as well as the convergence rate. Therefore AUC-DCD is among the first optimization schemes suggested for efficiently solving AUC problems.

**Key words**: machine learning; optimization method; AUC; dual coordinate descent; support vector machine

在机器学习中,预测精度(或错误率)被普遍作为分类算法的性能评价指标[1].但是,如果遇到错分代价不相等或处理分类不平衡数据库等情况时,单一使用预测精度作为评价指标就不能完整地反映算法的真实性能[2,3].而基于接收者操作特性(receiver operation characteristic,简称 ROC)分析的 ROC 曲线下面积(area under the ROC

* 基金项目: 国家自然科学基金(61273296, 60975040); 安徽省自然科学基金(1308085QF121)
收稿时间: 2013-01-30; 定稿时间: 2013-09-30

curve,简称 AUC)[4]则能很好地度量算法的整体性能,因此在机器学习中备受关注.

目前,国内外许多著名学者对 AUC 优化问题的求解进行了深入的研究,并提出了多种求解算法[5],主要有基于 Hinge 损失的 SVM(support vector machine,简称 SVM)型算法、基于平方损失的最小二乘型算法、基于指数损失的 AdaBoost 型算法等.这些算法的主要区别在于使用了不同的代理损失函数,其中,基于 Hinge 损失的 SVM 型算法受到了广泛的关注[6,7].由于求解 AUC 问题需要优化来自两个不同类别的样本对构成的损失,损失数与样本数呈平方增长,导致对目标函数的优化时间长,收敛速度慢.在 2011 年的 ICML 会议上,Zhao 等人[8]首次提出了求解 AUC 优化问题的在线算法(online AUC maximization,简称 OAM),通过引进两个缓冲(buffer),极大地缩减了需要优化的损失函数的个数,从而减少了训练次数,并且在分类不均衡数据库上获得了较好的实验结果.

众所周知,在线方法考虑样本是随时间序列逐个产生的,每次只用一个样本更新解向量,即,来一个样本训练 1 次,训练之后的样本则被丢弃,不需要存储所有训练样本,有别于传统的批处理算法,因此可用于处理大规模数据库.2003 年,Zinkevich 在投影次梯度方法的基础上,针对一般凸问题提出了第一个在线优化算法[9],并证明了 $O(\sqrt{T})$ 的 regret 界.随后,Hazan 针对强凸问题得到了 $O(\log T)$的 regret 界[10],保证了算法的性能.2007 年,Shalev-Shwartz 等人提出的投影次梯度算法 Pegasos[11],首次在大规模数据上进行实验并得到了很好的结果,在线方法也重新引起了人们的广泛关注.考虑到 AUC 优化问题的优化目标是来自不同类别的样本对构成的损失之和,需要存储所有样本,因此不能直接使用在线方法求解.文献[8]巧妙地使用 reservoir sampling 技术引进两个缓冲,并设定缓冲大小,分别用于存储正负样本,当处理一个样本时,首先判断该样本的类别并存储于相应缓冲,然后与另一缓冲里的所有样本组合构成损失对,当缓冲满时,用新样本随机替换缓冲里的样本,以达到更新缓冲的目的.由于仅对缓冲内的样本进行优化,因此大大减少了损失函数的个数.文献[8]给出两种算法——OAMseq 和 OAMgra,并给出 OAM 在缓冲为无限大时的算法 OAMinf,其中,OAMinf 算法的 AUC 性能最好.

本文通过对文献[8]的分析发现:OAM 算法在每次更新解向量时,没有使用上一步更新的解信息,使算法的收敛速度很慢,并且算法运行时所使用的平衡参数随着缓冲的改变而改变,使得参数选择过于复杂.为了更好地解决这些问题,本文提出一种求解 AUC 优化问题的对偶坐标下降方法(AUC-DCD).坐标下降(coordinate descend,简称 CD)方法的思路比较简单,在计算时固定解的其他维坐标,一次仅对选中的一维坐标进行优化求解.根据所优化目标函数的不同,CD 可分为对偶坐标下降(DCD)[12]和原始坐标下降(PCD)[13].DCD 是 2008 年台湾大学林智仁教授领导的研究小组针对 SVM 的对偶问题提出来的,并且在大量的数据库上取得了比 PCD 和当前一些流行算法更快的收敛效果,被认为是处理大规模数据特别是文本数据的首选算法[14].DCD 以简洁的操作流程、低廉的计算代价和快速的实际收敛效果,成为处理优化问题最有效的方法之一,能够克服 AUC 优化问题的缺点.此外,2012 年,Shalev-Shwartz[15]在对 DCD 方法进行分析时,曾使用 Modified-SGD 算法为对偶变量赋初值, Modified-SGD 算法与随机梯度(stochastic gradient descent,简称 SGD)算法类似,但使用了目标函数的对偶信息求取步长.由于 Modified-SGD 算法无需额外的内存开销,且优化的目标函数与样本数无关,因此可用于 AUC 问题处理大规模数据库的情况.

在线方法与对偶坐标下降方法在求解子问题的过程中均是在每一步迭代时求解单个样本关于所有维数的子优化问题,两者有着密切的联系.本文采用基于 Hinge 损失的最小化 SVM 模型,提出一种求解 AUC 优化问题的对偶坐标下降方法(AUC-DCD),相应地,给出 3 种算法,并发现其中 OAM 算法是 AUC-DCD 的一种特殊情形.最后,通过在大量数据库上进行比较实验,进一步验证了 AUC-DCD 的有效性.

## 1 AUC 优化问题

求解 AUC 优化问题面临的一个关键问题就是如何计算 AUC 值,在机器学习领域,较常用的方法是非参数假定的 AUC 估计[16],它在数值上等价于排序的 WMW(Wilcoxon-Mann-Whitney)统计[17,18].对于给定的独立同分布的训练样本集 $S=\{(\boldsymbol{x}_i,y_i)\in R^d\times\{-1,+1\}|i\in[n]\}$,把集合 $S$ 分为正负两个样本集 $S^+=\{\boldsymbol{x}_i^+\mid i\in[n^+]\}$和$S^-=\{\boldsymbol{x}_j^-\mid j\in[n^-]\}$, 其中,$n^+$和 $n^-$分别表示正负样本数,且 $n=n^++n^-$,$0<n^+<n$.非参数假定的 AUC 估计反映的是决策函数 $g=\boldsymbol{w}^T\boldsymbol{t}+b$ 代入一个随机抽取的正类样本的函数值大于一个随机抽取的负类样本的函数值的概率,AUC 值越大,则所求解

的分类器性能就越优,其表达式为

$$AUC=\frac{\sum_{i=1}^{n^+}\sum_{j=1}^{n^-}I(g(\boldsymbol{x}_i^+)>g(\boldsymbol{x}_j^-))}{n^+n^-} \tag{1}$$

其中,$I(x)$为指标函数(indicator function),即: $I(x)=\begin{cases}1, & x\text{为真}\\ 0, & x\text{为假}\end{cases}$.

在决策函数 $g=\boldsymbol{w}^T\boldsymbol{t}+b$ 中,$b$ 为偏置,可通过对数据进行扩维.将偏置 $b$ 放入权重 $\boldsymbol{w}$ 中统一进行处理,即:$[\boldsymbol{x}^T,1]\rightarrow\boldsymbol{x}^T$,$[\boldsymbol{w}^T,b]\rightarrow\boldsymbol{w}^T$,此时,决策函数可写为 $g=\boldsymbol{w}^T\boldsymbol{t}$,代入公式(1)可得:

$$AUC=\frac{\sum_{i=1}^{n^+}\sum_{j=1}^{n^-}I(\boldsymbol{w}^T\boldsymbol{x}_i^+>\boldsymbol{w}^T\boldsymbol{x}_j^-)}{n^+n^-}=1-\frac{\sum_{i=1}^{n^+}\sum_{j=1}^{n^-}I(\boldsymbol{w}^T\boldsymbol{x}_i^+\leqslant\boldsymbol{w}^T\boldsymbol{x}_j^-)}{n^+n^-}.$$

对特定的数据库,使 AUC 为极大值的点等价于使 $\sum_{i=1}^{n^+}\sum_{j=1}^{n^-}I(\boldsymbol{w}^T\boldsymbol{x}_i^+\leqslant\boldsymbol{w}^T\boldsymbol{x}_j^-)\Big/n^+n^-$ 为极小值的点,因为 $I(x)$ 可视为 0-1 损失函数,非凸且不可微,在此选取其代理函数 Hinge 损失函数进行优化.在机器学习领域,一般在“损失函数+正则化项”的框架下对机器学习问题进行研究[19],常用的正则化项有 L1 和 L2 正则化,其中,L2 正则化具有二阶可导、对偶、强凸等特性,常用的 SVM 模型即采用 L2 正则化.在此,采用 SVM 模型讨论 AUC 优化问题,即,正则化项使用 L2 正则化,用于调节分类器的泛化性.因此,最终优化目标函数为

$$\min_{\boldsymbol{w}}\frac{\lambda}{2}\|\boldsymbol{w}\|^2+\frac{1}{n^+n^-}\sum_{i=1}^{n^+}\sum_{j=1}^{n^-}l(\boldsymbol{w}^T(\boldsymbol{x}_i^+-\boldsymbol{x}_j^-)) \tag{2}$$

其中,参数$\lambda$>0.Hinge 损失函数为 $l(\boldsymbol{w}^T(\boldsymbol{x}_i^+-\boldsymbol{x}_j^-))=\max\{0,1-\boldsymbol{w}^T(\boldsymbol{x}_i^+-\boldsymbol{x}_j^-)\}$.

## 2 求解 AUC 优化问题的对偶坐标下降方法

对偶坐标下降方法有多种形式,根据选取优化坐标方式的不同,可以分为循环对偶坐标下降(cyclic-DCD)算法、随机对偶坐标下降(stochastic-DCD)算法和使用 Permutation 策略的置换对偶坐标下降(stochastic-DCDperm)算法等.同理,本文给出 3 种求解 AUC 优化问题的算法,即,AUC-SDCD,AUC-SDCDperm 和 AUC-MSGD,其中,AUC-MSGD 也是随机选取一维坐标进行优化.为便于说明,本文使用 SDCD,SDCDperm 和 MSGD 作为相应算法的简记形式.若无特别说明,文中的 AUC 在线方法均指 OAM 算法.

### 2.1 3种求解算法

讨论对偶坐标下降方法时,首先需要求解出原优化问题的对偶问题.为方便说明,本文记:

$$\sum_{ij}=\sum_{i=1}^{n^+}\sum_{j=1}^{n^-},\ k=n^+\times n^-,\boldsymbol{z}_{ij}=\boldsymbol{x}_i-\boldsymbol{x}_j,l_{ij}(\boldsymbol{z}_{ij}^T\boldsymbol{w})=\max\{0,1-\boldsymbol{z}_{ij}^T\boldsymbol{w}\},\text{其中},i\in[n^+],j\in[n^-],$$

则公式(2)可写为

$$\min_{\boldsymbol{w}}P(\boldsymbol{w}),\text{其中},P(\boldsymbol{w})=\frac{\lambda}{2}\|\boldsymbol{w}\|^2+\frac{1}{k}\sum_{ij}l_{ij}(\boldsymbol{z}_{ij}^T\boldsymbol{w}) \tag{3}$$

容易求得公式(3)的对偶问题(求解过程见附录 1)为

$$\max_{\boldsymbol{\alpha}}D(\boldsymbol{\alpha}),\text{其中},D(\boldsymbol{\alpha})=\frac{1}{k}\sum_{ij}\alpha_{ij}-\frac{\lambda}{2}\left\|\frac{1}{\lambda k}\sum_{ij}\alpha_{ij}\boldsymbol{z}_{ij}\right\|^2,\text{并且}\alpha_{ij}\in[0,1] \tag{4}$$

其中,$\boldsymbol{\alpha}$为对偶变量,且 $\boldsymbol{\alpha}=[\alpha_{11},...,\alpha_{ij},...,\alpha_{n^+n^-}]^T$. 通过求解过程易知,有关系式 $\boldsymbol{w}=\frac{1}{\lambda k}\sum_{ij}\alpha_{ij}\boldsymbol{z}_{ij}$ 成立.为方便起见,本文使用上述问题的等价问题进行求解,即:

$$\min_{\boldsymbol{\alpha}}\frac{\lambda}{2}\left\|\frac{1}{\lambda k}\sum_{ij}\alpha_{ij}\boldsymbol{z}_{ij}\right\|^2-\frac{1}{k}\sum_{ij}\alpha_{ij},\alpha_{ij}\in[0,1] \tag{5}$$

由于每次仅更新对偶变量$\boldsymbol{\alpha}$的一维,设第 $t$ 次迭代时使向量$\boldsymbol{\alpha}$的 $a_{ij}$ 维完成从 $a_{ij}^{t-1}$ 到 $a_{ij}^t$ 的更新,并设

$\Delta\alpha_{ij}=a_{ij}^{t}-a_{ij}^{t-1}$,所以公式(5)的单维变量子问题为

$$\begin{cases}\min\limits_{\Delta\alpha_{ij}} \dfrac{\lambda k}{2}\left\|\boldsymbol{w}^{t-1}+\dfrac{1}{\lambda k}\Delta\alpha_{ij}\boldsymbol{z}_{ij}\right\|^2-(a_{ij}^{t-1}+\Delta\alpha_{ij}) \\ \text{s.t. } 0\leqslant a_{ij}^{t}\leqslant 1\end{cases} \tag{6}$$

式(6)为关于$\Delta\alpha_{ij}$的二次函数,易求得其解析解为 $\Delta\alpha_{ij}=\lambda k(1-\boldsymbol{z}_{ij}^{T}\boldsymbol{w}^{t-1})\big/\boldsymbol{z}_{ij}^{T}\boldsymbol{z}_{ij}$.

算法 1 为 AUC-SDCD 的更新过程.

**算法 1**. AUC-SDCD.

•Given $\boldsymbol{\alpha}^0$ and the corresponding $\boldsymbol{w}^0=\dfrac{1}{\lambda k}\sum\limits_{ij}\alpha_{ij}^0\boldsymbol{z}_{ij}$

•Repeat $t$=1,2,...,$T$

1. Randomly choose $i\in[n^+], j\in[n^-]$
2. Let $\alpha_{ij}^{t}=\min(\max(\alpha_{ij}^{t-1}+\lambda k(1-\boldsymbol{z}_{ij}^{T}\boldsymbol{w}^{t-1})\big/\boldsymbol{z}_{ij}^{T}\boldsymbol{z}_{ij},0),1)$
3. $\boldsymbol{w}^{t}=\boldsymbol{w}^{t-1}+(\alpha_{ij}^{t}-\alpha_{ij}^{t-1})\boldsymbol{z}_{ij}\big/\lambda k$

•End

算法 1 从初始化$\boldsymbol{\alpha}$开始(一般初始化为零向量),每次迭代随机挑选对偶变量$\boldsymbol{\alpha}$的一维,通过解决单变量子问题(6)进行更新,最后,利用原问题解与对偶问题解之间的关系达到更新原问题解 $\boldsymbol{w}$ 的目的.

**算法 2**. AUC-SDCDperm.

•Given $\boldsymbol{\alpha}^0$ and the corresponding $\boldsymbol{w}^0=\dfrac{1}{\lambda k}\sum\limits_{ij}\alpha_{ij}^0\boldsymbol{z}_{ij}$

•Repeat $t$=1,2,...,$T$

1. Let $\{p_1,p_2,...,p_{n^+n^-}\}$ be a random permutation of $\{1,2,...,n^+n^-\}$
2. Repeat $s=1,2,...,n^+n^-$

(1) $a=p_s, i=(a-1)\big/n^-+1, j=(a-1)\bmod n^-+1$

(2) Do Step 2 ~ Step 3 of AUC-SDCD

End

•End

算法 2 为 AUC-SDCDperm 的执行过程.SDCDperm 算法与 SDCD 算法的处理过程类似,SDCDperm 在遍历 1 次样本时,仅对向量$\boldsymbol{\alpha}$的每一维更新 1 次,但 SDCD 算法则可能对$\boldsymbol{\alpha}$的某一维重复更新.

算法 1、算法 2 每次更新解向量时均使用了上一步更新的解信息,因此需要额外的存储空间,当处理大规模数据库时,所花费的内存代价相当惊人.为解决存储问题,我们给出 AUC-MSGD 算法,具体流程见算法 3.

**算法 3**. AUC-MSGD.

•Initialize $\boldsymbol{w}^0=\boldsymbol{0}$

•Repeat $t$=1,2,...,$T$

1. Randomly choose $i\in[n^+], j\in[n^-]$
2. Let $\alpha_{ij}=\min(\max(\lambda t(1-\boldsymbol{z}_{ij}^{T}\boldsymbol{w}^{t-1})\big/\boldsymbol{z}_{ij}^{T}\boldsymbol{z}_{ij},0),1)$
3. $\boldsymbol{w}^{t}=(1-1/t)\boldsymbol{w}^{t-1}+\alpha_{ij}\boldsymbol{z}_{ij}\big/\lambda t$

•End

MSGD 算法对对偶变量的更新与 SDCD 算法稍有不同,主要在于两种算法所解决的单变量子问题不同,MSGD 算法在第 $t$ 次迭代时通过解决如下子问题对解向量进行更新:

$$\begin{cases}\min\limits_{\alpha_{ij}} \dfrac{\lambda t}{2}\left\|\boldsymbol{w}^{t-1}+\dfrac{1}{\lambda t}\alpha_{ij}\boldsymbol{z}_{ij}\right\|^2-\alpha_{ij} \\ \text{s.t. } 0\leqslant a_{ij}\leqslant 1\end{cases} \tag{7}$$

与公式(6)比较易发现:SDCD 算法子问题的样本数是固定的,即,对某一数据库 $k=n^+\times n^-$是定值;而 MSGD 算法求解的子问题(7)则是随迭代次数 $t$ 的增加产生变化.

### 2.2 AUC-SDCD与OAM的区别

文献[8]提出了求解 AUC 优化问题的在线算法(OAM),并给出缓冲取无限大时的算法 OAMinf,OAMinf 通过解决如下子问题来更新解向量 $\boldsymbol{w}$:

$$\min_{\boldsymbol{w}} \frac{\lambda k}{2}\|\boldsymbol{w}-\boldsymbol{w}^t\|^2+l_{ij}(\boldsymbol{z}_{ij}^T\boldsymbol{w}) \tag{8}$$

Crammer[20]在 2006 年通过问题(8)的对偶问题求得解析解 $\boldsymbol{w}=\boldsymbol{w}^t+\alpha_{ij}\boldsymbol{z}_{ij}/\lambda k$ (求解过程见附录 2),其中,

$$\alpha_{ij}=\min(\max(\lambda k(1-\boldsymbol{z}_{ij}^T\boldsymbol{w}^t)/\boldsymbol{z}_{ij}^T\boldsymbol{z}_{ij},0),1).$$

该子问题的求解方法本质上与公式(6)给出的求解方法相同.通过与算法 1 比较容易发现:若初始化向量 $\boldsymbol{\alpha}=\boldsymbol{0}$,OAMinf 算法是 SDCD 算法按顺序遍历 1 次所有样本时的特殊情况,但 SDCD 算法却能够得到比 OAMinf 算法更快的收敛速度,本文实验部分对这一结论进行了详细验证.

### 2.3 收敛性分析

对于一般形式坐标下降算法的收敛性问题,Luo 和 Tseng 进行了深入而系统的研究[21].针对求解 SVM 对偶优化问题的 DCD 方法,文献[12]指出:当遍历 $k$ 次样本集时,能够得到$(1-u)^k$,$0<u<1$ 的线性收敛速度.但 Shalev-Shwartz 等人[15]对此给出两点质疑:一是参数 $u$ 可能无限逼近 0,且迭代次数没有限制,可以无限大;二是所作的收敛性分析只是针对对偶目标函数,而优化的最终目的是原始目标函数,即,对偶问题的收敛速度并不能说明原问题具有同样的收敛速度.在文献[15]中,Shalev-Shwartz 和 Zhang 两位著名学者使用 Fenchel-Young 不等式[22]通过对偶间隙使得原始目标函数与对偶目标函数之间建立关系,从而给出原目标函数的收敛速度.

设算法进行 $T_0$ 次迭代后,令 $\bar{\boldsymbol{w}}=1/(T-T_0)\sum_{t=T_0+1}^{T}\boldsymbol{w}^t$,$\bar{\boldsymbol{\alpha}}=1/(T-T_0)\sum_{t=T_0+1}^{T}\boldsymbol{\alpha}^t$,$\boldsymbol{w}^*=\arg\min\limits_{\boldsymbol{w}} P(\boldsymbol{w})$, 由于原目标函数 $P(\boldsymbol{w})$为凸函数,根据凸函数性质可知 $P(\bar{\boldsymbol{w}})\leqslant 1/(T-T_0)\sum_{t=T_0+1}^{T}P(\boldsymbol{w}^t)$,又因为原目标函数与对偶目标函数满足关系式 $P(\boldsymbol{w}^*)\geqslant D(\bar{\boldsymbol{\alpha}})$, 所以有 $E[P(\bar{\boldsymbol{w}})]-P(\boldsymbol{w}^*)\leqslant E[P(\bar{\boldsymbol{w}})-D(\bar{\boldsymbol{\alpha}})]$. 这表明,可以用对偶间隙的收敛速度衡量原目标函数的收敛速度.因此,关于 AUC-SDCD 算法的收敛性,本文给出如下引理.

**引理 1**. 假设 AUC-SDCD 算法初始化$\boldsymbol{\alpha}=\boldsymbol{0}$,$\varepsilon>0$ 是迭代若干步后目标函数值与理论最优目标函数值之间的误差,当迭代次数 $T>T_0$ 时,若使对偶间隙满足 $E[P(\bar{\boldsymbol{w}})-D(\bar{\boldsymbol{\alpha}})]\leqslant\varepsilon$, 则迭代次数 $T$ 需满足:

$$T\geqslant T_0+k+\frac{4}{\lambda\varepsilon}\geqslant\max\left\{0,\left\lceil k\log(\lambda k/2)\right\rceil\right\}+k+\frac{20}{\lambda\varepsilon},$$

其中,$k=n^+\times n^-$,$E$ 表示对随机抽取的样本对序列取期望.

类似地,AUC-SDCDperm 算法的收敛性与 AUC-SDCD 算法相同.

针对 Modified-SGD 算法,当算法遍历 1 次 $n$ 个样本的数据库时,文献[15]给出对偶目标函数 $O((\log n)/n)$的收敛速度.同样,对于 AUC-MSGD 算法本文给出引理 2.

**引理 2**. 假设样本满足独立同分布,令 $\boldsymbol{\alpha}^*=\arg\max\limits_{\boldsymbol{\alpha}} D(\boldsymbol{\alpha})$, 当 AUC-MSGD 算法遍历 1 次样本时有:

$$E[D(\boldsymbol{\alpha}^*)-D(\boldsymbol{\alpha})]\leqslant\frac{2\log(ek)}{\lambda k},$$

其中,$k=n^+\times n^-$,$E$ 表示对随机抽取的样本对序列取期望.

由于引理 1、引理 2 的证明与文献[15]中的定理 1、定理 3 类似,这里不再赘述.

## 3 数值实验

本节通过实验对文中所提算法的正确性和有效性进行验证.实验中,算法在 LIBLINEAR[23]平台上实现,该平台设计了科学、合理的数据结构,可以高效地处理数据和安排内存开销,并用多种编程语言实现,便于开发新算法,受到了众多国家教育科研机构的青睐.

### 3.1 算法及数据库描述

实验在 10 个数据库上实现,这些数据库可从 LIBSVM 网站中获得,具体下载地址为 http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/,表 1 给出数据库的相关描述.

**Table 1** Details of the datasets

**表 1** 数据库描述

| 数据库 | 样本数 | 样本维数 | 失衡率 |
|---|---|---|---|
| sonar | 208 | 60 | 1.144 3 |
| fourclass | 862 | 2 | 1.807 8 |
| german | 1 000 | 24 | 2.333 3 |
| svmguide3 | 1 243 | 22 | 3.199 3 |
| dna | 2 000 | 180 | 3.310 3 |
| svmguide4 | 300 | 10 | 5.818 2 |
| pendigits | 7 494 | 16 | 8.620 0 |
| vowel | 528 | 10 | 10.000 0 |
| letter | 15 000 | 16 | 24.252 5 |
| sector | 6 412 | 55 197 | 148.116 3 |

表 1 中,sonar,fourclass,german 和 svmguide3 为二分类数据库,其余为多分类数据库.实验未对数据进行任何预处理,假设正样本为少数样本,为了构造不平衡数据库,对多分类数据库,采取把其中一类作为正样本,其他类作为负样本的处理策略.表中失衡率定义为负样本数与正样本数之比.

文献[8]给出两种求解 AUC 优化问题的在线算法:OAMseq 和 OAMgra,并给出 OAM 算法在缓冲无限大时的算法 OAMinf.其中,OAMinf 算法得到的 AUC 性能优于 OAMseq 和 OAMgra.因此,本节实验仅对算法 OAMinf,SDCD,SDCDperm 和 MSGD 进行比较验证.为公平比较,实验中,算法 SDCD 和 MSGD 的迭代次数均取 $T=n^+\times n^-$,且 SDCDperm 算法仅执行 1 次外循环,即,保证 4 种算法迭代次数相同.

### 3.2 实验结果及结论

实验把每个数据库随机平均分成 5 组,其中 4 组作为训练样本,剩下 1 组作为测试样本,每次分组求得 5 组 AUC 值,每个数据库进行 4 次随机分组,最终结果为 20 组 AUC 值的平均值和方差.

4 种算法的参数选择范围为 $10^{-9}$~$10^{1}$,在实验中选取最优参数.

图 1 给出 4 种算法在各数据库上的参数取值与 AUC 值的关系.

表 2 列出了 4 种算法在相应数据库上的实验结果,其中,记号"$A\pm B$"中 $A$ 为 AUC 平均值,$B$ 为相应的均方差(mean square error).均方差按照公式 $MSE=\sqrt{1/(n-1)\sum_{i=1}^{n}(\bar{x}-x_i)^2}$ ($n$=20,$x_i$ 表示 AUC 值,$\bar{x}$ 表示 AUC 的平均值)来计算.

**Table 2** Comparison of AUC performances on the datasets

**表 2** 在各数据库上的 AUC 性能比较

| 数据库 | OAMinf | SDCD | SDCDperm | MSGD |
|---|---|---|---|---|
| sonar | 0.8611±0.0632 | 0.8652±0.0583 | 0.8650±0.0590 | 0.8653±0.0609 |
| fourclass | 0.8298±0.0266 | 0.8310±0.0314 | 0.8309±0.0314 | 0.8337±0.0260 |
| german | 0.7134±0.0475 | 0.7797±0.0284 | 0.7912±0.0325 | 0.7805±0.0271 |
| svmguide3 | 0.7689±0.0364 | 0.7545±0.0408 | 0.7905±0.0345 | 0.7527±0.0375 |
| dna | 0.9890±0.0041 | 0.9569±0.0119 | 0.9894±0.0041 | 0.9584±0.0114 |
| svmguide4 | 0.8065±0.0837 | 0.8354±0.0775 | 0.8389±0.0753 | 0.8294±0.0742 |
| pendigits | 0.9825±0.0058 | 0.9603±0.0160 | 0.9846±0.0041 | 0.9617±0.0131 |
| vowel | 0.8965±0.0382 | 0.9072±0.0360 | 0.9081±0.0363 | 0.9081±0.0382 |
| letter | 0.9882±0.0026 | 0.9643±0.0129 | 0.9888±0.0025 | 0.9618±0.0129 |
| sector | 0.9964±0.0058 | 0.8436±0.1042 | 0.9989±0.0012 | 0.8681±0.0769 |

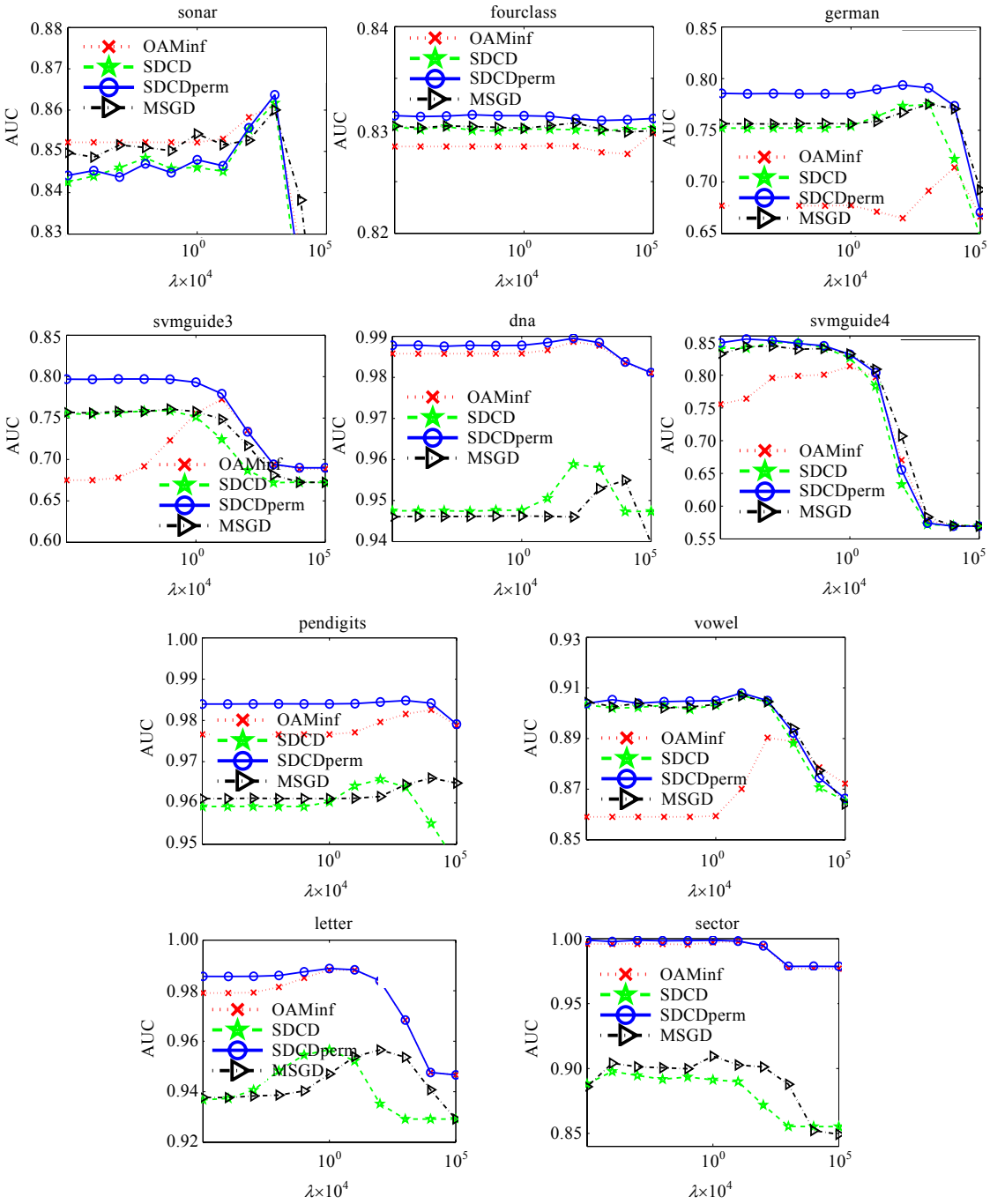


Fig.1 Comparison of AUC value with different parameters

图 1 取不同参数时的 AUC 值比较

从表中 2 可以看出:在 AUC 值的比较上,SDCDperm 算法在多数数据库上明显优于 OAMinf,SDCD 和 MSGD 算法;从均方差的比较中也可以看出,SDCDperm 算法的稳定性较好.SDCD 和 MSGD 算法的 AUC 性能仅在部分数据库上好于 OAMinf 算法,这与随机抽取的样本对的顺序有关.

为了说明 AUC-DCD 方法在收敛速度这一性能上优于在线算法,图 2 给出 4 种算法每次迭代求解的 AUC 值和目标函数值的下降速度比较实验(20 次结果的平均值).限于篇幅,这里仅给出在 sonar,svmguide4 和 vowel 数据库上的实验结果.

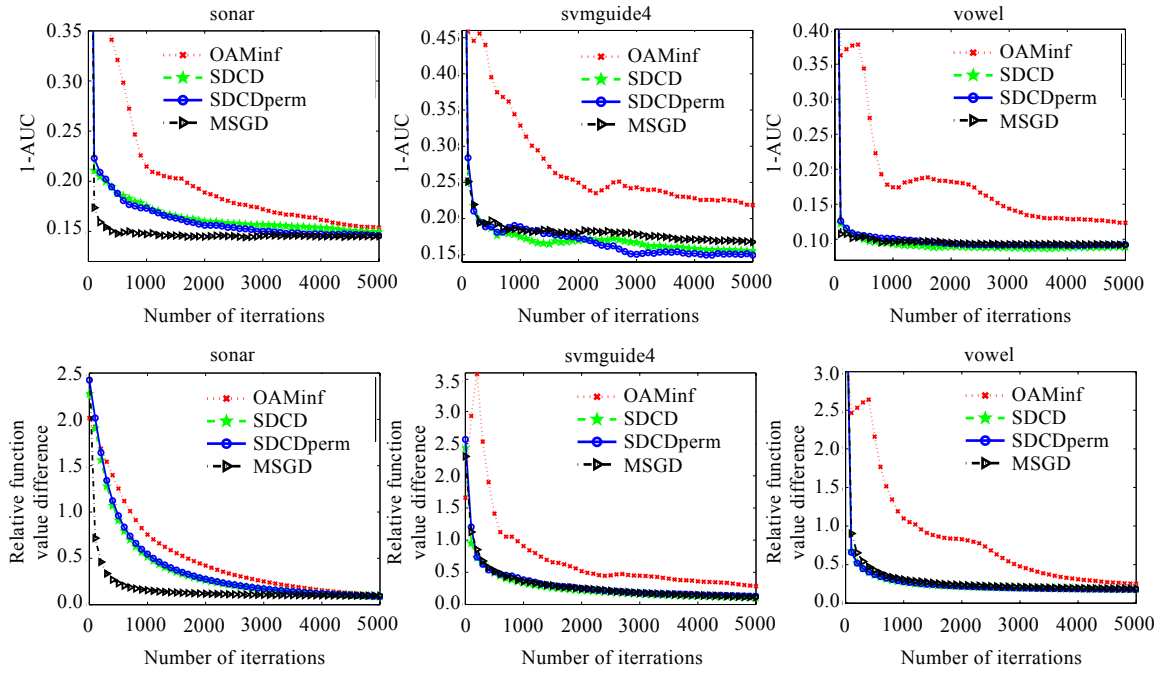


Fig.2 Comparison of convergence rates of AUC and relative function value difference

图 2 AUC 及相对目标函数值收敛速度比较

图 2 中,6 幅图的横坐标均为迭代次数,每迭代运行 100 步做一个记录点.由于在执行一定的迭代次数之后算法已接近收敛,因此,图中仅画出算法迭代 5 000 次的实验结果.为方便比较,图中用“1-AUC”表示 AUC 值的收敛速度,用相对目标函数值(relative function value difference)表示目标函数值的下降速度.其中,

相对目标函数值=(当前目标函数值−最优目标函数值)/最优目标函数值.

计算时,最优目标函数值用最小的目标函数值代替.

从上述图中可以得出如下结论:

(1) 算法 MSGD,SDCD 和 SDCDperm 无论是在 AUC 的收敛速度还是目标函数值的收敛速度上都明显快于 OAMinf 算法;特别地,在 sonar 数据库,MSGD 的收敛速度快于其他 3 种算法;

(2) SDCD 算法的目标函数值的下降速度稍快于 SDCDperm;

(3) 当迭代次数足够大时,SDCDperm 算法得到的 AUC 性能总是最优.

## 4 总 结

本文提出一种求解 AUC 优化问题的对偶坐标下降(AUC-DCD)方法,并给出相应的 3 种算法.理论分析表明,求解 AUC 优化问题的在线方法(OAM)是一种特殊的坐标下降方法.并且指出,AUC-DCD 方法可以得到比 OAM 方法更优的 AUC 性能及更快的收敛速度.最后,通过实验,对我们的结论进行了验证.

**References**:

[1] Duda RO, Hart P E, Stork DG. Pattern Classification. 2nd., New York: Wiley, 2001.

[2] Cortes C, Mohri M. AUC optimization vs. error rate minimization. Advances in Neural Information Processing Systems, 2004, 16(16):313−320.

[3] Maloof MA. Learning when data sets are imbalanced and when costs are unequal and unknown. In: Proc. of the Workshop on Learning from Imbalanced Data Sets II (ICML). Washington, 2003. 421−428. http://www.informatik.uni-trier.de/~ley/db/conf/icml/icml2003.html

[4] Hanely JA, McNeil BJ. The meaning and use of the area under a receiver operating characteristic (ROC) curve. Radiology, 1982, 143(1):29−36. [doi: 10.1148/radiology.143.1.7063747]

[5] Wang YN, Chen SC. A survey of evaluation and design for classifier based on AUC. Pattern Recognition and Artificial Intelligence, 2011,24(1):64−71 (in Chinese with English abstract).

[6] Rakotomamonjy A. Optimizing area under roc curve with SVMs. In: Proc. of the European Conf. on Artificial Intelligence Workshop on ROC Analysis and Artificial Intelligence. Valencia: IOS Press, 2004. 71−80.

[7] Brefeld U, Scheffer T. AUC maximizing support vector learning. In: Proc. of the ICML 2005 Workshop on ROC Analysis in Machine Learning. Bonn, 2005. 377−384. http://users.dsic.upv.es/~flip/ROCML2005/papers.html

[8] Zhao PL, Hoi SCH, Jin R, Yang TB. Online AUC maximization. In: Proc. of the 28th Int'l Conf. on Machine Learning. 2011. 233−240. http://www.informatik.uni-trier.de/~ley/db/conf/icml/icml2011.html

[9] Zinkevich M. Online convex programming and generalized infinitesimal gradient ascent. In: Proc. of the 20th Annual Int'l Conf. on Machine Learning. 2003. 856−863. http://www.informatik.uni-trier.de/~ley/db/conf/icml/icml2003.html

[10] Hazan E, Agarwal A, Kale S. Logarithmic regret algorithms for online convex optimization. Machine Learning, 2007,69(2-3): 169−192. [doi: 10.1007/s10994-007-5016-8]

[11] Shalev-Shwartz S, Singer Y, Srebro N. Pegasos: Primal estimated sub-gradient solver for SVM. In: Proc. of the Int'l Conf. on Machine Learning. 2007. 807−814. [doi: 10.1145/1273496.1273598]

[12] Hsieh CJ, Chang KW, Lin CJ, Keerthi SS, Sundararajan S. A dual coordinate descent method for large-scale linear SVM. In: Proc. of the 28th Int'l Conf. on Machine Learning. 2008. 408−415. [doi: 10.1145/1390156.1390208]

[13] Chang KW, Hsieh CJ, Lin CJ. Coordinate descent method for large-scale l2-loss linear support vector machines. Journal of Machine Learning Research, 2008,9:1369−1398.

[14] Yuan GX, Chang KW, Hsieh CJ, Lin CJ. A comparison of optimization methods and software for large-scale l1-regularized linear classification. The Journal of Machine Learning Research, 2010,9999:3183−3234.

[15] Shalev-Shwartz S, Zhang T. Stochastic dual coordinate ascent methods for regularized loss minimization. Journal of Machine Learning Research, 2013,14:567−599.

[16] Wu SM, Flach P. A scored AUC metric for classifier evaluation and selection. In: Proc. of the ICML 2005 Workshop on ROC Analysis in Machine Learning. Bonn, 2005. 378−389. http://users.dsic.upv.es/~flip/ROCML2005/papers.html

[17] Mann HB, Whitney DR. On a test of whether one of two random variables is stochastically larger than the other. The Annals of Mathematical Statistics, 1947,18(1):50−60. [doi: 10.1214/aoms/1177730491]

[18] Yan L, Dodier R, Mozer MC, Wolniewicz R. Optimizing classifier performance via an approximation to the Wilcoxon-Mann-Whitney statistic. In: Proc. of the Int'l Conf. on Machine Learning. Washington, 2003. 848−855. http://www.informatik.uni-trier.de/~ley/db/conf/icml/icml2003.html

[19] Sun ZY, Tao Q. Statistical machine learning: A review of the loss function and optimization. Communications of The CCF, 2009, 5(8):7−14 (in Chinese with English abstract).

[20] Crammer K, Dekel O, Keshet J, Shalev-Shwartz S, Singer Y. Online passive-aggressive algorithms. The Journal of Machine Learning Research, 2006,7:551−585.

[21] Luo ZQ, Tseng P. On the convergence of the coordinate descent method for convex differentiable minimization. Journal of Optimization Theory and Applications, 1992,72(1):7−35. [doi: 10.1007/BF00939948]

[22] Shalev-Shwartz S, Singer Y. Convex repeated games and fenchel duality. In: Advances in Neural Information Processing Systems. 2006. 1265−1272. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.87.1617

[23] Fan RE, Chang KW, Hsieh CJ, Wang XR, Lin CJ. LIBLINEAR: A library for large linear classification. The Journal of Machine Learning Research, 2008,9:1871−1874.

**附中文参考文献**:

[5] 汪云云,陈松灿.基于 AUC 的分类器评价和设计综述.模式识别与人工智能,2011,24(1):64−71.

[19] 孙正雅,陶卿.统计机器学习综述:损失函数与优化求解.中国计算机学会通讯,2009,5(8):7−14.

## 附录 1. 对偶问题的求解过程

为了便于求解,令 $c=\lambda/2$,$f(\boldsymbol{w})=\|\boldsymbol{w}\|^2$,$\boldsymbol{w}_0=\boldsymbol{w}_{ij}=\boldsymbol{w}$,$\forall i\in[n^+]$,$\forall j\in[n^-]$,并引入 $k$ 个等式约束,则公式(3)的等价优化问题为

$$\begin{cases}\min\limits_{\boldsymbol{w}_0,\boldsymbol{w}_{11},\ldots,\boldsymbol{w}_{n^+n^-}} cf(\boldsymbol{w}_0)+\dfrac{1}{k}\sum\limits_{ij}l_{ij}(\boldsymbol{z}_{ij}^T\boldsymbol{w}_{ij})\\ \text{s.t.}\ \ \dfrac{1}{k}\boldsymbol{z}_{ij}^T\boldsymbol{w}_{ij}=\dfrac{1}{k}\boldsymbol{z}_{ij}^T\boldsymbol{w}_0\end{cases} \tag{a}$$

为第 $ij$ 个等式约束引入拉格朗日乘子 $\alpha_{ij}$,则公式(a)的拉格朗日函数为

$$L(\boldsymbol{w}_0,\boldsymbol{w}_{11},\ldots,\boldsymbol{w}_{n^+n^-},\alpha_{11},\ldots,\alpha_{n^+n^-})=cf(\boldsymbol{w}_0)+\frac{1}{k}\sum_{ij}l_{ij}(\boldsymbol{z}_{ij}^T\boldsymbol{w}_{ij})+\sum_{ij}\alpha_{ij}\left(\frac{1}{k}\boldsymbol{z}_{ij}^T\boldsymbol{w}_{ij}-\frac{1}{k}\boldsymbol{z}_{ij}^T\boldsymbol{w}_0\right) \tag{b}$$

所以,公式(a)的对偶函数为

$$\begin{aligned}D(\boldsymbol{\alpha})&=\inf_{\boldsymbol{w}_0,\boldsymbol{w}_{11},\ldots,\boldsymbol{w}_{n^+n^-}}L(\boldsymbol{w}_0,\boldsymbol{w}_{11},\ldots,\boldsymbol{w}_{n^+n^-},\alpha_{11},\ldots,\alpha_{n^+n^-})\\&=\inf_{\boldsymbol{w}_0,\boldsymbol{w}_{11},\ldots,\boldsymbol{w}_{n^+n^-}}\left\{cf(\boldsymbol{w}_0)+\frac{1}{k}\sum_{ij}l_{ij}(\boldsymbol{z}_{ij}^T\boldsymbol{w}_{ij})+\sum_{ij}\alpha_{ij}\left(\frac{1}{k}\boldsymbol{z}_{ij}^T\boldsymbol{w}_{ij}-\frac{1}{k}\boldsymbol{z}_{ij}^T\boldsymbol{w}_0\right)\right\}\\&=\inf_{\boldsymbol{w}_0,\boldsymbol{w}_{11},\ldots,\boldsymbol{w}_{n^+n^-}}\left\{-\frac{1}{k}\sum_{ij}\alpha_{ij}\boldsymbol{z}_{ij}^T\boldsymbol{w}_0+cf(\boldsymbol{w}_0)+\frac{1}{k}\sum_{ij}[\alpha_{ij}\boldsymbol{z}_{ij}^T\boldsymbol{w}_{ij}+l_{ij}(\boldsymbol{z}_{ij}^T\boldsymbol{w}_{ij})]\right\}\\&=\inf_{\boldsymbol{w}_0,\boldsymbol{w}_{11},\ldots,\boldsymbol{w}_{n^+n^-}}\left\{-c\left[\frac{1}{ck}\sum_{ij}\alpha_{ij}\boldsymbol{z}_{ij}^T\boldsymbol{w}_0-f(\boldsymbol{w}_0)\right]-\frac{1}{k}\sum_{ij}[-\alpha_{ij}\boldsymbol{z}_{ij}^T\boldsymbol{w}_{ij}-l_{ij}(\boldsymbol{z}_{ij}^T\boldsymbol{w}_{ij})]\right\}\\&=-c\sup_{\boldsymbol{w}_0}\left[\left(\frac{1}{ck}\sum_{ij}\alpha_{ij}\boldsymbol{z}_{ij}\right)^T\boldsymbol{w}_0-f(\boldsymbol{w}_0)\right]-\frac{1}{k}\sum_{ij}\sup_{\boldsymbol{w}_{ij}}[-\alpha_{ij}\boldsymbol{z}_{ij}^T\boldsymbol{w}_{ij}-l_{ij}(\boldsymbol{z}_{ij}^T\boldsymbol{w}_{ij})]\\&=-cf^*\left(\frac{1}{ck}\sum_{ij}\alpha_{ij}\boldsymbol{z}_{ij}\right)-\frac{1}{k}\sum_{ij}l_{ij}^*(-\alpha_{ij})\end{aligned} \tag{c}$$

其中,$f^*(\cdot)$和 $l_{ij}^*(\cdot)$ 分别为函数 $f(\cdot)$和 $l_{ij}(\cdot)$的凸共轭函数.由凸共轭函数的定义可知:

$$f^*(u)=\sup_a\{ua-f(a)\}=\sup_a\{ua-a^2\}=\frac{1}{4}u^2,l_{ij}^*(u)=\sup_a\{ua-l_{ij}(a)\}=\sup_a\{ua-\max\{0,1-a\}\}=u,u\in[-1,0].$$

把上式及 $c=\lambda/2$ 代入公式(c)中,即得到原问题的对偶问题:

$$D(\boldsymbol{\alpha})=\frac{1}{k}\sum_{ij}\alpha_{ij}-\frac{\lambda}{2}\left\|\frac{1}{\lambda k}\sum_{ij}\alpha_{ij}\boldsymbol{z}_{ij}\right\|^2,\text{其中},\alpha_{ij}\in[0,1].$$

对公式(b)求偏导,有:

$$\frac{\partial L}{\partial\boldsymbol{w}_0}=\lambda\boldsymbol{w}_0-\frac{1}{k}\sum_{ij}\alpha_{ij}\boldsymbol{z}_{ij}=0\Rightarrow\boldsymbol{w}_0=\frac{1}{\lambda k}\sum_{ij}\alpha_{ij}\boldsymbol{z}_{ij},$$

$$\frac{\partial L}{\partial\boldsymbol{w}_{ij}}=\frac{1}{k}\frac{\partial l_{ij}(\boldsymbol{z}_{ij}^T\boldsymbol{w}_{ij})}{\partial\boldsymbol{w}_{ij}}\boldsymbol{z}_{ij}+\frac{1}{k}\alpha_{ij}\boldsymbol{z}_{ij}=0\Rightarrow\frac{\partial l_{ij}(\boldsymbol{z}_{ij}^T\boldsymbol{w}_{ij})}{\partial\boldsymbol{w}_{ij}}=-\alpha_{ij}.$$

又因为 $\boldsymbol{w}_0=\boldsymbol{w}_{ij}=\boldsymbol{w}$,$\forall i\in[n^+]$,$\forall j\in[n^-]$,所以有如下关系式成立:

$$\boldsymbol{w}=\frac{1}{\lambda k}\sum_{ij}\alpha_{ij}\boldsymbol{z}_{ij},\frac{\partial l_{ij}(\boldsymbol{z}_{ij}^T\boldsymbol{w})}{\partial\boldsymbol{w}}=-\alpha_{ij}.$$

## 附录 2. 子问题(8)解析解求解过程

令 $\xi_{ij}=\max\{0,1-\boldsymbol{z}_{ij}^T\boldsymbol{w}\},\forall i\in[n^+],\forall j\in[n^-]$,易知公式(8)的等价问题为

$$\begin{cases} \min\limits_{w} \dfrac{\lambda k}{2} \| \boldsymbol{w} - \boldsymbol{w}^t \|^2 + \xi_{ij} \\ \text{s.t. } 1 - \boldsymbol{z}_{ij}^T \boldsymbol{w} \leqslant \xi_{ij} \\ \quad \xi_{ij} \geqslant 0 \end{cases} \tag{d}$$

构造公式(d)的拉格朗日函数:

$$L(\boldsymbol{w}, \xi_{ij}, \alpha_{ij}, \beta_{ij}) = \frac{\lambda k}{2} \| \boldsymbol{w} - \boldsymbol{w}^t \|^2 + \xi_{ij} + \alpha_{ij}(1 - \boldsymbol{z}_{ij}^T \boldsymbol{w} - \xi_{ij}) - \beta_{ij}\xi_{ij} \tag{e}$$

对公式(e)求偏导,得:

$$\frac{\partial L}{\partial \boldsymbol{w}} = \lambda k(\boldsymbol{w} - \boldsymbol{w}^t) - \alpha_{ij}\boldsymbol{z}_{ij} = 0 \Rightarrow \boldsymbol{w} = \boldsymbol{w}^t + \frac{1}{\lambda k}\alpha_{ij}\boldsymbol{z}_{ij},$$

$$\frac{\partial L}{\partial \xi_{ij}} = 1 - \alpha_{ij} - \beta_{ij} = 0.$$

将上述两式代入公式(e)中并整理,得:

$$L(\boldsymbol{\alpha}) = -\frac{1}{2\lambda k} \| \alpha_{ij}\boldsymbol{z}_{ij} \|^2 + (1 - \boldsymbol{z}_{ij}^T \boldsymbol{w}^t)\alpha_{ij}, \text{且} \alpha_{ij} \in [0,1].$$

上式对$\alpha_{ij}$求偏导并置 0,得子问题解析解:

$$\alpha_{ij} = \min\left( \max\left( \frac{\lambda k(1 - \boldsymbol{z}_{ij}^T \boldsymbol{w}^t)}{\boldsymbol{z}_{ij}^T \boldsymbol{z}_{ij}}, 0 \right), 1 \right).$$


**姜纪远**(1989－),男,安徽涡阳人,硕士,主要研究领域为机器学习,模式识别.
E-mail: jyjianggle@gmail.com

**高乾坤**(1989－),男,硕士,主要研究领域为机器学习,模式识别.
E-mail: gaospeed@gmail.com

**陶卿**(1965－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为机器学习,模式识别,应用数学.
E-mail: taoqing@gmail.com

**储德军**(1978－),男,讲师,主要研究领域为模式识别,凸优化及其在机器学习中的应用.
E-mail: djun.chu@gmail.com