软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2013,24(11):2584−2596 [doi: 10.3724/SP.J.1001.2013.04485]

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

具有 Fisher 一致性的代价敏感 Boosting 算法*

曹 莹, 苗启广, 刘家辰, 高 琳

(西安电子科技大学 计算机学院,陕西 西安 710071)

通讯作者: 苗启广, E-mail: qgmiao@mail.xidian.edu.cn

摘 要: AdaBoost 是一种重要的集成学习元算法,算法最核心的特性“Boosting”也是解决代价敏感学习问题的有效方法.然而,各种代价敏感 Boosting 算法,如 AdaCost、AdaC 系列算法、CSB 系列算法等采用启发式策略,向 AdaBoost 算法的加权投票因子计算公式或权值调整策略中加入代价参数,迫使算法聚焦于高代价样本.然而,这些启发式策略没有经过理论分析的验证,对原算法的调整破坏了 AdaBoost 算法最重要的 Boosting 特性。AdaBoost 算法收敛于贝叶斯决策,与之相比,这些代价敏感 Boosting 并不能收敛到代价敏感的贝叶斯决策.针对这一问题,研究严格遵循 Boosting 理论框架的代价敏感 Boosting 算法.首先,对分类间隔的指数损失函数以及 Logit 损失函数进行代价敏感改造,可以证明新的损失函数具有代价意义下的 Fisher 一致性,在理想情况下,优化这些损失函数最终收敛到代价敏感贝叶斯决策;其次,在 Boosting 框架下使用函数空间梯度下降方法优化新的损失函数得到算法 AsyB 以及 AsyBL.二维高斯人工数据上的实验结果表明,与现有代价敏感 Boosting 算法相比,AsyB 和 AsyBL 算法能够有效逼近代价敏感贝叶斯决策;UCI 数据集上的测试结果也进一步验证了 AsyB 以及 AsyBL 算法能够生成有更低错分类代价的代价敏感分类器,并且错分类代价随迭代呈指数下降.

关键词: 代价敏感学习;贝叶斯决策;Fisher 一致性;AdaBoost;二分类

中图法分类号: TP181 **文献标识码**: A

中文引用格式: 曹莹,苗启广,刘家辰,高琳.具有 Fisher 一致性的代价敏感 Boosting 算法.软件学报,2013,24(11):2584−2596. http://www.jos.org.cn/1000-9825/4485.htm

英文引用格式: Cao Y, Miao QG, Liu JC, Gao L. Fisher consistent cost sensitive Boosting algorithm. Ruan Jian Xue Bao/ Journal of Software, 2013,24(11):2584−2596 (in Chinese). http://www.jos.org.cn/1000-9825/4485.htm

Fisher Consistent Cost Sensitive Boosting Algorithm

CAO Ying, MIAO Qi-Guang, LIU Jia-Chen, GAO Lin

(School of Computer Science and Technology, Xidian University, Xi'an 710071, China)

Corresponding author: MIAO Qi-Guang, E-mail: qgmiao@mail.xidian.edu.cn

Abstract: AdaBoost is a meta ensemble learning algorithm. The most important theoretical property behind it is “Boosting”, which also plays an important role in cost sensitive learning. However, available cost sensitive Boosting algorithms, such as AdaCost, AdaC1, AdaC2, AdaC3, CSB0, CSB1 and CSB2, are just heuristic. They add cost parameters into voting weight calculation formula or sample weights updating strategy of AdaBoost, so that the algorithms are forced to focus on samples with higher misclassification costs. However, these heuristic modifications have no theoretical foundations. The worst thing is that they break the most important theoretical property of AdaBoost, namely “Boosting”. Compared to AdaBoost which converges to optimal Bayes decision rule, those cost sensitive algorithms do not converge to cost sensitive decision rule. This paper studies the problem of designing cost sensitive Boosting algorithms strictly under Boosting theory. First, two new loss functions are constructed by making exponential loss and logit loss cost sensitive. It can be proved

* 基金项目: 国家自然科学基金(61072109, 61272280, 41271447, 61272195); 新世纪优秀人才支持计划(NCET-12-0919); 西安市科技局项目(CXY1341(6)); 中央高校基本科研业务费专项资金(K5051203020, K5051203001, K5051303016, K5051303018, K505131 00006)

收稿时间: 2013-05-31; 修改时间: 2013-07-17; 定稿时间: 2013-08-27

that the new loss functions are Fisher consistent in cost sensitive setting, therefore optimizing them finally leads to cost sensitive Bayes decision rule. Performing gradient decent in functional space to optimize these two loss functions then results in new cost sensitive Boosting algorithms: AsyB and AsyBL. Experimental results on synthetic Gaussian data prove that in comparison with other cost sensitive Boosting algorithms, AsyB and AsyBL always better approximate cost sensitive Bayes decision rule. Experimental results on UCI datasets further prove that AsyB and AsyBL generate better cost sensitive classifiers with lower misclassification costs and the misclassification costs decrease exponentially with iterations.

Key words: cost sensitive learning; Bayes decision; Fisher consistent; AdaBoost; binary classification

代价敏感学习是机器学习研究的一类重要问题.经典学习问题假设样本错分类到各个类别的代价相同,学习算法以最小化期望错误率为目标.而在代价敏感学习中,样本被错分到不同类别的代价不相同,算法以最小化期望代价为目标.代价敏感学习能够更好地满足医疗诊断、金融预测、恶意程序检测等问题的实际需求.

首先给出代价敏感学习问题的形式化描述.输入空间 $X\subseteq R^N$ 是 N 维向量空间,$Y=\{-1,+1\}$是输出空间,D 是定义在 $X\times Y$ 上一个固定但未知的分布.训练样本集 $S\{\langle \boldsymbol{x}_1,y_1\rangle,\langle \boldsymbol{x}_2,y_2\rangle,\ldots,\langle \boldsymbol{x}_m,y_m\rangle\}$独立同分布地从 D 中抽取.代价矩阵 $cost(i,j)$定义了将第 i 类样本错分为第 j 类样本的代价.代价敏感学习算法的目标是从假设空间 H 中找到一个假设 h,使期望代价 $h=\arg\min_{H} E_{X,Y\sim D}[\![cost(y,yh(\boldsymbol{x}))]\!]$ 最小.在二分类问题中,简记 $cost(i,i)=0(i=1,2)$,$cost(1,2)=C_1$,$cost(2,1)=C_2$,且假定 $C_1>C_2$.

代价敏感学习通常分为基于特定算法的代价敏感学习和代价敏感学习元方法两类[1].第 1 种方法从算法原理入手,改变算法的优化目标,从假设空间中直接选择一个让期望代价最小的假设,例如在决策树算法中使用代价敏感的叶结点分裂准则以及代价敏感的剪枝策略[2].第 2 种方法在标准学习算法的基础上附加一个过程,将使期望错误率最小的标准学习算法转化使期望代价最小的代价敏感学习算法,常用方法包括移动决策阈值[3]、重采样[4,5]以及集成学习方法[6].这类方法的通用性好,在解决实际问题时经常使用,也是本文的研究内容.

以 AdaBoost 为代表的 Boosting 算法是一类重要的集成学习元算法[7],并且代价敏感 Boosting 算法也是最重要的代价敏感学习元方法之一.AdaBoost 算法能够将分类准确率仅比随机猜测略好的弱分类器提升为分类准确率任意高的强分类器.算法维护一个定义在全体训练样本上的分布,每一轮迭代依此分布生成新的子分类器,并根据子分类器对训练样本分类的准确率计算投票权值,以加权投票方式对子分类器进行集成.在一轮迭代的最后,当前错误分类样本的权值加重,正确分类样本权值降低,迫使下一个生成的子分类器聚焦于当前难以被正确分类样本.理想情况下,AdaBoost 可以完美拟合任意训练样本集,当训练样本足够多时,算法的训练错误率可以任意接近 0;同时,分类错误率随迭代以指数速率下降,这是 Boosting 算法最重要也最吸引人的理论特性.许多研究也将 Boosting 思想用于代价敏感学习,对错分类代价进行提升,试图得到错分类代价很低的代价敏感分类器.研究提出了 AdaCost[8],AdaC1,AdaC2,AdaC3[9,10],CSB0,CSB1,CSB2[11],Asymmetric Boosting[12] 以及 Asymmetric LogitBoost[13]等代价敏感 Boosting 算法.然而,这些算法大多采用启发策略向 AdaBoost 算法的加权投票因子,或是样本权更新策略中加入代价因子,许多修改未经理论分析的验证,反而破坏了 AdaBoost 算法最重要的 Boosting 特性.

1 代价敏感 Boosting 算法分析

目前研究提出了 AdaCost、AdaC 系列算法、CSB 系列算法、Asymmetric Boosting、Asymmetric LogitBoost 等代价敏感 Boosting 算法.除了 Asymmetric Boosting 和 Asymmetric LogitBoost 以外,其余算法都是启发式的,它们分别以不同的策略向 AdaBoost 算法的加权投票因子计算公式和样本权调整策略中加入代价因子.

AdaCost 要求对代价矩阵进行归一化处理,使得 $c_i\in[0,1]$.算法实现代价敏感学习的核心是定义调整方程 $\beta(i)$[8].对正确分类样本$\beta_+(i)=-0.5C_i+0.5$,错误分类样本$\beta_-(i)=0.5C_i+0.5$,每一轮迭代计算:

$$\gamma_t=\sum_{i=1}^{m}D^t(i)y_ih_t(\boldsymbol{x}_i)\beta(i),\ \alpha_t=\frac{1}{2}\ln\frac{1+\gamma_t}{1-\gamma_t},\ D^{t+1}(i)=D^t(i)\frac{\exp(-\alpha_t y_i h_t(\boldsymbol{x}_i)\beta(i))}{Z_t}.$$

$\beta_+(i)$是代价因子的单调递减函数,$\beta_-(i)$则相反,以使高代价样本$\beta_+(i)$低,$\beta_-(i)$高.然而 AdaCost 算法中,α的计算非

常不合理,为保证加权投票因子有意义(α>0),每一轮迭代生成的子分类器必须满足:

$$\sum_{i=1}^{m} I[h(\boldsymbol{x}_i)=y_i]D(i)\beta_+(i) > \sum_{i=1}^{m} I[h(\boldsymbol{x}_i)\neq y_i]D(i)\beta_-(i).$$

考虑调整因子$\beta(i)$,高代价样本错误分类时权值增加,正确分类时权值降低,除非子分类器准确率非常高,否则这一条件难以满足.特别是当两类错误代价的比率很高时,常常难以生成可用的子分类器进行集成.同时,初始化时使用的代价敏感样本权,并没有为降低错分类代价带来实际效果,反而加剧了α计算的不合理.α的不合理,同样也间接地影响了样本权值更新策略的合理性,在实际使用中,AdaCost 效果并不好.

CSB 系列算法要求 C_2=1,C_1>1,不对代价矩阵进行归一化处理[11].CSB 系列算法使用了与 AdaBoost 算法相同的方法计算加权投票因子,仅对样本权更新策略进行修改.在 AdaBoost 算法中,加权投票因子与样本权调整是一个统一的整体,共同保证了 Boosting 特性.CSB 系列算法对 AdaBoost 的修改仅仅是启发式地考虑加重高代价样本的权值,以迫使算法聚焦于高代价错误,但这些修改并不合理,完全破坏了 AdaBoost 算法的 Boosting 的特性,CSB 系列算法的效果并不理想.

AdaC1,AdaC2 与 AdaC3 分别将代价因子加入到 AdaBoost 算法样本权更新公式的 3 个不同位置,并在此基础上推导加权投票因子α的计算公式[9,10].尽管 AdaC 系列算法看上去经过理论分析的验证,但是 AdaC2 和 AdaC3 算法推导过程有不合理之处.AdaC2 算法使用样本权更新策略 $D^{t+1}(i)=D^t(i)C_i\exp(-\alpha_t y_i h(\boldsymbol{x}_i))/Z_t$,推导的核心步骤是将 AdaBoost 算法训练错误率上界 $\frac{1}{m}|i:H(\boldsymbol{x}_i)\neq y_i| \leqslant \frac{1}{m}\sum_i \exp(-y_i H(\boldsymbol{x}_i)) = \prod_t Z_t$ 推广到代价敏感设置下.由于 AdaBoost 使用的指数损失函数是 0-1 损失的凸上界,因此,以上不等式总成立.然而在代价敏感设置下,$\frac{1}{m}|i:H(\boldsymbol{x}_i)\neq y_i| \leqslant \frac{1}{m}\sum_i \exp(-y_i H(\boldsymbol{x}_i)) = \prod_t Z_t$ 并不成立.对 AdaC3 算法,尽管作者提出算法时并不是优化特定损失函数,但从损失函数理论的角度分析,优化 AdaC3 使用的损失函数不是对代价敏感贝叶斯决策的合理近似,最终不会收敛到代价敏感贝叶斯决策.

Asymmetric Boosting 算法对代价敏感损失函数 $L(\boldsymbol{x},y,f(\boldsymbol{x}))=E_{X,Y\sim D}\left[\!\left[I(y=1)\mathrm{e}^{-C_1 y f(\boldsymbol{x})}+I(y=-1)\mathrm{e}^{-C_2 y f(\boldsymbol{x})}\right]\!\right]$ 使用函数空间梯度下降优化方法,但加权投票因子α必须通过二分法或是牛顿法等数值方法求解[12],与原 AdaBoost 相比,算法的复杂度提高了.Asymmetric LogitBoost 算法首先定义了代价敏感的 logit 变换:

$$P_c(y=1|\boldsymbol{x})=\frac{\mathrm{e}^{\frac{1}{2}(C_1+C_2)f(\boldsymbol{x})+\frac{1}{2}\log\frac{C_2}{C_1}}}{\mathrm{e}^{\frac{1}{2}(C_1+C_2)f(\boldsymbol{x})+\frac{1}{2}\log\frac{C_2}{C_1}}+\mathrm{e}^{-\frac{1}{2}(C_1+C_2)f(\boldsymbol{x})-\frac{1}{2}\log\frac{C_2}{C_1}}}.$$

然后,优化 Logit 损失函数 $y^*\log P_c(\boldsymbol{x})+(1-y^*)\log(1-P_c(\boldsymbol{x}))$,$y^*=(y+1)/2\in\{0,1\}$.可以证明,在 Asymmetric Boosting 和 Asymmetric LogitBoost 算法中,让损失函数最小的决策是代价敏感贝叶斯决策,因此在理想情况下,训练样本充足无噪声,则这两种算法会最终求得最优的代价敏感分类器.由于有 Boosting 理论的支持,Asymmetric Boosting 和 Asymmetric LogitBoost 算法在各类代价敏感 Boosting 算法中效果较好.

在文献[14]中,Landesa-Vázquez 等人使用类条件错误分解的方法解释 AdaBoost 中训练错误率的改变,指出:AdaBoost 算法本身就具备非对称学习能力,不用对算法进行任何修改,只要在样本权初始化时赋予代价敏感初始化权值 $D^1(i)=c_i\big/\sum_{i=1}^{m}c_i$,就可以直接进行代价敏感学习,AdaBoost 算法的一切理论特性都在代价敏感学习中继续保持.在本文的研究中,我们也将所有算法与具有代价敏感初始化样本权的 AdaBoost 算法进行比较,在下文中,简记作 AdaBoostC 算法.

为了更直观地说明以上代价敏感 Boosting 算法的实际运行结果,我们在高斯人工数据集上对以上算法得到的代价敏感分类器的决策边界进行可视化分析.进行人工数据集测试的原因是样本分布已知,因此能够确定最优决策边界的精确位置,能够将各算法求得的分类器的决策边界与理论上最优的贝叶斯决策边界进行直观的对比.首先,依高斯分布 $\mathrm{N}\left([-1,1],\begin{bmatrix}5.2 & 2\\ 2 & 3.7\end{bmatrix}\right)$ 和 $\mathrm{N}\left([1,-1],\begin{bmatrix}3.3 & 1.5\\ 1.5 & 4.6\end{bmatrix}\right)$ 生成两类测试数据,每一类各 1 000 个样本,取其中 70%数据用于训练,30%用于测试.指定代价因子 C_1/C_2=5,子分类器使用线性感知器,算法迭代 100 轮.

从图 1 中可以看到,在这些代价敏感 Boosting 算法中,AdaBoostC,Asymmetric Boosting 算法对代价敏感贝叶斯决策边界近似较好.除此之外的算法求得的分类器的决策边界与最优贝叶斯决策都相差较大,并不是对代价敏感贝叶斯决策的有效近似.这些算法都因为不合理的样本权调整,或是不合理的加权投票因子计算方式,过度地聚焦于高代价样本.不断加大代价因子,当两类错误代价相差越来越大时,这些算法得到的决策偏离最优代价敏感决策也越来越严重.

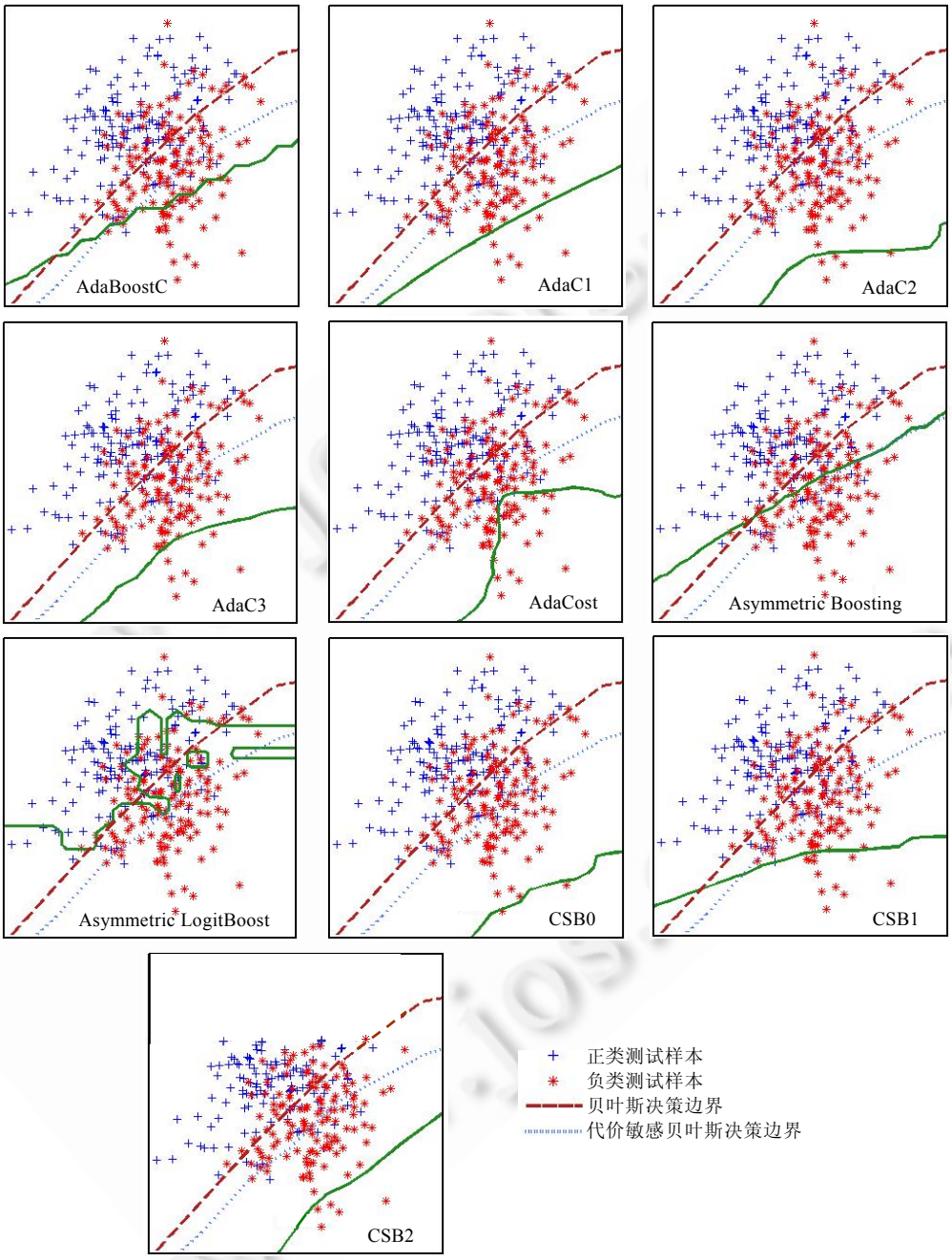

Fig.1 Decision boundaries of different cost sensitive Boosting algorithms on synthetic Gaussian data

图 1 各代价敏感 Boosting 算法在高斯人工数据集上的决策边界

AdaBoost 算法起源于 PAC 学习模型,算法的每一步都经过理论的分析和实验的验证,扎实的理论保证了算法的 Boosting 特性.然而,除了 Asymmetric Boosting 和 Asymmetric LogitBoost 以外,以上提出的代价敏感的 Boosting 算法都是启发式的,没有经过严格的理论分析,实验也并不充分.这些启发式的策略改变 AdaBoost 算法的加权投票因子计算公式,或是修改样本权更新策略,向其中加入代价参数,使得高代价样本若错误分类,则会更大地加重其权值,反之则会较小地降低其权值.这些启发式策略从定性分析出发,并非完全合理,对 AdaBoost 算法核心步骤的修改甚至破坏了算法最重要的 Boosting 特性.AdaBoost 算法收敛于贝叶斯决策,相比之下,除了 Asymmetric Boosting 以外,这些代价敏感 Boosting 算法并不收敛于代价敏感的贝叶斯决策.

2 具有 Fisher 一致性的代价敏感 Boosting 算法

为了严格遵循 Boosting 理论框架设计对错分类代价进行提升的代价敏感 Boosting 算法,本文研究首先对 AdaBoost 算法使用的分类间隔的指数损失函数以及 LogitBoost 使用的 Logit 损失函数进行代价敏感改造,保证优化新的代价敏感损失函数最终收敛到代价敏感的贝叶斯决策.

2.1 代价敏感损失函数

损失函数在机器学习算法中起着重要作用.$S_l\{(x_1,y_1),\ldots,(x_l,y_l)\}$是依分布 D 得到的独立同分布样本集,学习算法基于训练样本集 S_l 和损失函数 $L(\boldsymbol{x},y,f(\boldsymbol{x}))$从假设空间 H 中选择一个使经验风险最小的假设 $\hat{h}(\boldsymbol{x})$. 分类问题研究的一个基本问题是:随着 $l\to\infty$,能否构建起一个假设序列$\{h_n\}$,对这个序列,经验风险与期望风险以很高的概率任意接近最低可能风险 R^*,也就是一致性要求.特别地,当最小化问题:

$$\arg\min_{f(x)} E_{X,Y\sim D}[\![L(\boldsymbol{x},y,f(\boldsymbol{x}))]\!]=\arg\min_{f(x)} L(\boldsymbol{x},y,f(\boldsymbol{x}))P(y=1\mid\boldsymbol{x})+L(\boldsymbol{x},y,f(\boldsymbol{x}))P(y=-1\mid\boldsymbol{x})$$

有唯一解 $\hat{f}(\boldsymbol{x})$,并且 $\operatorname{sgn}(\hat{f}(\boldsymbol{x}))=\operatorname{sgn}\left(\log\dfrac{P(y=1\mid\boldsymbol{x})}{P(y=-1\mid\boldsymbol{x})}\right)$时,称 $L(\boldsymbol{x},y,f(\boldsymbol{x}))$具有 Fisher 一致性[15,16].Fisher 一致性要求对所有可测函数,损失函数 $L(\boldsymbol{x},y,f(\boldsymbol{x}))$的总体最小化最终都能收敛到贝叶斯决策.Fisher 一致性是分类问题中损失函数应该满足的最小条件.可以证明,AdaBoost 算法使用的分类间隔的指数损失函数、LogitBoost 使用的 Logit 损失函数都具有 Fisher 一致性.同理,在 Boosting 框架下设计代价敏感损失函数首先应该对算法使用的损失函数进行代价敏感改造,使得损失函数的最小化最终收敛到代价敏感贝叶斯决策,也就是代价敏感意义下的 Fisher 一致性.

引理. 公式(1)是分类间隔的代价敏感指数损失函数,公式(2)是代价敏感的 Logit 损失函数,令它们最小化的决策都是代价敏感贝叶斯决策 $f^*(\boldsymbol{x})=\dfrac{1}{2}\dfrac{P_{Y|X}(y=1\mid\boldsymbol{x})C_1}{P_{Y|X}(y=-1\mid\boldsymbol{x})C_2}$:

$$L_e=E_{Y|X}[I(y=+1)C_1\mathrm{e}^{-yf(x)}+I(y=-1)C_2\mathrm{e}^{-yf(x)}|\boldsymbol{x}] \tag{1}$$

$$\begin{cases} L_l=E_{Y|X}[y^*\log p_c(\boldsymbol{x})+(1-y^*)\log(1-p_c(\boldsymbol{x}))\mid\boldsymbol{x}] \\ p_c=\dfrac{C_2\mathrm{e}^{f(x)}}{C_1\mathrm{e}^{-f(x)}+C_2\mathrm{e}^{f(x)}} \\ y^*=(y+1)/2\in\{0,1\} \end{cases} \tag{2}$$

证明:

$E_{Y|X}[I(y=+1)C_1\mathrm{e}^{-yf(x)}+I(y=-1)C_2\mathrm{e}^{-yf(x)}|\boldsymbol{x}]=P(y=-1|\boldsymbol{x})C_2\mathrm{e}^{f(x)}$,对损失函数求导数并令导数为 0,于是有:

$$\frac{\partial L(\boldsymbol{x},y,f(\boldsymbol{x}))}{\partial f(\boldsymbol{x})}=-P(y=1\mid\boldsymbol{x})C_1\mathrm{e}^{-f(x)}+P(y=-1\mid\boldsymbol{x})C_2\mathrm{e}^{f(x)}\triangleq 0.$$

解得:

$$f^*(\boldsymbol{x})=\frac{1}{2}\frac{P_{Y|X}(y=1\mid\boldsymbol{x})C_1}{P_{Y|X}(y=-1\mid\boldsymbol{x})C_2},\ \text{且}\ \frac{\partial^2 L_e}{\partial f^2}\geqslant 0.$$

因此,L_e 有唯一最小化解 $f^*(\boldsymbol{x})$.

$E_{Y|X}[y^*\log p_c(\boldsymbol{x})+(1-y^*)\log(1-p_c(\boldsymbol{x}))|\boldsymbol{x}]=P(y^*=1|\boldsymbol{x})\log p_c(\boldsymbol{x})+P(y^*=0|\boldsymbol{x})\log(1-p_c(\boldsymbol{x}))$,对损失函数求导数,并令导数

为 0,于是有:

$$P(y^*=1\mid \boldsymbol{x})\frac{1}{p_c(\boldsymbol{x})}-P(y^*=0\mid \boldsymbol{x})\frac{1}{1-p_c(\boldsymbol{x})}\triangleq 0.$$

代入
$$p_c=\frac{C_2\mathrm{e}^{f(x)}}{C_1\mathrm{e}^{-f(x)}+C_2\mathrm{e}^{f(x)}},$$

解得:
$$f^*(\boldsymbol{x})=\frac{1}{2}\frac{P_{Y|X}(y=1\mid \boldsymbol{x})C_1}{P_{Y|X}(y=-1\mid \boldsymbol{x})C_2},\ \text{且}\ \frac{\partial^2 L_e}{\partial p_c^2}\geqslant 0.$$

$p_c(\boldsymbol{x})$是 $f(\boldsymbol{x})$的单调增函数,因此,L_l 有唯一最小化解 $f^*(\boldsymbol{x})$. □

在训练样本无限无噪声的理想情况下,对损失函数(1)和损失函数(2)使用经验风险最小化准则,会得到最优的代价敏感贝叶斯决策.即使是在训练样本有限、有噪声的情况下,优化这两个损失函数也是对代价敏感贝叶斯决策的一个合理近似.

2.2 AsyB算法

对分类间隔的代价敏感指数损失(1)使用函数空间梯度下降优化方法,首先从初始分布生成第 1 个子分类器,以后的每一轮迭代在现有模型 $F(\boldsymbol{x})$的基础上加入$\alpha f_t(\boldsymbol{x})$,得到的新模型 $F(\boldsymbol{x})+\alpha f_t(\boldsymbol{x})$应使损失函数下降最快,于是有:

$$\begin{aligned}(f_t(\boldsymbol{x}),\alpha_t)&=\arg\min_{f(x),\alpha}L_e(\boldsymbol{x},y,F(\boldsymbol{x})+\alpha f(\boldsymbol{x}))\\&=E_{Y|X}[I(y=+1)C_1\mathrm{e}^{-y(F(x)+\alpha f(x))}+I(y=-1)C_2\mathrm{e}^{-y(F(x)+\alpha f(x))}\mid \boldsymbol{x}]\\&=E_{Y|X}[I(y=+1)C_1\mathrm{e}^{-F(x)}\mathrm{e}^{-\alpha f(x)}+I(y=-1)C_2\mathrm{e}^{F(x)}\mathrm{e}^{\alpha f(x)}\mid \boldsymbol{x}]\\&=E_{Y|X}[I(y=+1)C_1\mathrm{e}^{-F(x)}I(f(\boldsymbol{x})=1)\mathrm{e}^{-\alpha}+\\&\quad I(y=+1)C_1\mathrm{e}^{-F(x)}I(f(\boldsymbol{x})=-1)\mathrm{e}^{\alpha}+\\&\quad I(y=-1)C_2\mathrm{e}^{-F(x)}I(f(\boldsymbol{x})=+1)\mathrm{e}^{\alpha}+\\&\quad I(y=-1)C_2\mathrm{e}^{-F(x)}I(f(\boldsymbol{x})=-1)\mathrm{e}^{-\alpha}]\\&=P_{Y|X}(y=+1\mid \boldsymbol{x})C_1\mathrm{e}^{-F(x)}I(f(\boldsymbol{x})=+1)\mathrm{e}^{-\alpha}+\\&\quad P_{Y|X}(y=+1\mid \boldsymbol{x})C_1\mathrm{e}^{-F(x)}I(f(\boldsymbol{x})=-1)\mathrm{e}^{\alpha}+\\&\quad P_{Y|X}(y=-1\mid \boldsymbol{x})C_2\mathrm{e}^{-F(x)}I(f(\boldsymbol{x})=+1)\mathrm{e}^{\alpha}+\\&\quad P_{Y|X}(y=-1\mid \boldsymbol{x})C_2\mathrm{e}^{-F(x)}I(f(\boldsymbol{x})=-1)\mathrm{e}^{-\alpha}.\end{aligned}$$

除以常数 $\mathrm{e}^{-F(x)}+\mathrm{e}^{F(x)}$,对分布进行归一化:

$$\begin{aligned}(f_t(\boldsymbol{x}),\alpha_t)=\arg\min_{f(x),\alpha}\Bigg\{&P_{Y|X}(y=+1\mid \boldsymbol{x})\frac{\mathrm{e}^{-F(x)}}{\mathrm{e}^{-F(x)}+\mathrm{e}^{F(x)}}C_1I(f(\boldsymbol{x})=+1)\mathrm{e}^{-\alpha}+\\&P_{Y|X}(y=+1\mid \boldsymbol{x})\frac{\mathrm{e}^{-F(x)}}{\mathrm{e}^{-F(x)}+\mathrm{e}^{F(x)}}C_1I(f(\boldsymbol{x})=-1)\mathrm{e}^{\alpha}+\\&P_{Y|X}(y=-1\mid \boldsymbol{x})\frac{\mathrm{e}^{F(x)}}{\mathrm{e}^{-F(x)}+\mathrm{e}^{F(x)}}C_2I(f(\boldsymbol{x})=+1)\mathrm{e}^{\alpha}+\\&P_{Y|X}(y=-1\mid \boldsymbol{x})\frac{\mathrm{e}^{-F(x)}}{\mathrm{e}^{-F(x)}+\mathrm{e}^{F(x)}}C_2I(f(\boldsymbol{x})=-1)\mathrm{e}^{-\alpha}\Bigg\}.\end{aligned}$$

令

$$P^{\varpi}_{Y|X}(y\mid \boldsymbol{x})=\frac{P_{Y|X}(y\mid x)\mathrm{e}^{-yF(x)}}{Z}\tag{3}$$

$$\begin{aligned}(f_t(\boldsymbol{x}),\alpha_t)=\arg\min_{f(x),\alpha}E_X\{&P^{\varpi}_{Y|X}(y=+1\mid \boldsymbol{x})I(f(\boldsymbol{x})=+1)C_1\mathrm{e}^{-\alpha}+P^{\varpi}_{Y|X}(y=+1\mid \boldsymbol{x})I(f(\boldsymbol{x})=-1)C_1\mathrm{e}^{\alpha}+\\&P^{\varpi}_{Y|X}(y=-1\mid \boldsymbol{x})I(f(\boldsymbol{x})=+1)C_2\mathrm{e}^{\alpha}+P^{\varpi}_{Y|X}(y=-1\mid \boldsymbol{x})I(f(\boldsymbol{x})=-1)C_2\mathrm{e}^{-\alpha}\}.\end{aligned}$$

用样本估计替代上式中的期望,得到:

$$(f_t(\boldsymbol{x}),\alpha_t)=\arg\min[\gamma^+C_1\mathrm{e}^{-\alpha}+\varepsilon^+C_1\mathrm{e}^{\alpha}+\gamma^-C_2\mathrm{e}^{\alpha}+\varepsilon^-C_2\mathrm{e}^{-\alpha}] \tag{4}$$

$$\begin{cases}\varepsilon^+=\sum_{i=1}^{m}D^t(i)I[y=1]I[h(\boldsymbol{x}_i)\neq y_i]\\ \varepsilon^-=\sum_{i=1}^{m}D^t(i)I[y=1]I[h(\boldsymbol{x}_i)\neq y_i]\end{cases} \tag{5}$$

$$\begin{cases}\gamma^+=\sum_{i=1}^{m}D^t(i)I[y=1]I[h(\boldsymbol{x}_i)= y_i]\\ \gamma^-=\sum_{i=1}^{m}D^t(i)I[y=1]I[h(\boldsymbol{x}_i)= y_i]\end{cases} \tag{6}$$

给定 $f_t(\boldsymbol{x})$,公式(4)对 α 求导,并令导数为 0:

$$\frac{\partial}{\partial\alpha}\gamma^+C_1\mathrm{e}^{-\alpha}+\varepsilon^+C_1\mathrm{e}^{\alpha}+\gamma^-C_2\mathrm{e}^{\alpha}+\varepsilon^-C_2\mathrm{e}^{-\alpha}\triangleq 0.$$

解得:

$$\alpha_t=\frac{1}{2}\ln\frac{\gamma^+C_1+\varepsilon^-C_2}{\varepsilon^+C_1+\gamma^+C_2}.$$

由公式(3)得到第 T+1 轮样本权值更新公式为

$$D^{t+1}(i)=D^t(i)\frac{\exp(-\alpha_t h_t(\boldsymbol{x}_i)y_i)}{Z_t}.$$

整理以上分析过程,得到代价敏感 Boosting 算法 AsyB(算法 1).

算法 1. AsyB 算法.

输入:$\langle\boldsymbol{x}_1,y_1\rangle,\langle\boldsymbol{x}_2,y_2\rangle,\ldots,\langle\boldsymbol{x}_m,y_m\rangle$;代价参数 C_1,C_2.

初始化: $D^1(i)=c_i\Big/\sum_{i=1}^{m}c_i$.

For i=1,…,T

1. 基于分布 D^t 训练子分类器 h_t.
2. 由公式(5)和公式(6)计算$\varepsilon^+,\varepsilon^-,\gamma^+$以及$\gamma^-$.
3. 计算加权投票因子 $\alpha_t=\frac{1}{2}\ln\frac{\gamma^+C_1+\varepsilon^-C_2}{\varepsilon^+C_1+\gamma^+C_2}$.
4. 更新样本分布: $D^{t+1}(i)=D^t(i)\frac{\exp(-\alpha_t h_t(\boldsymbol{x}_i)y_i)}{Z_t}$, Z_t 是归一化因子.

输出: $H(\boldsymbol{x})=\mathrm{sign}\left(\sum_{t=1}^{T}\alpha_t h_t(\boldsymbol{x})\right)$.

2.3 AsyBL算法

Friedman 在其研究中首次提出使用统计学观点解释 AdaBoost 算法,将算法看作是使用牛顿法在函数空间优化分类间隔的指数损失函数 $\mathrm{e}^{yf(x)}$.实际上,$\mathrm{e}^{yf(x)}$的二阶泰勒级展开与 Logistic 回归中使用的(负)Logit 损失函数等价,于是,Friedman 提出了直接优化 Logit 损失函数的 LogitBoost 算法.在实际应用中,LogitBoost 具有与 AdaBoost 算法相当的性能以及更强的抗噪声能力,是一种重要的 Boosting 算法[17].

对代价敏感 Logit 损失(2),使用函数空间牛顿法解可加逻谛斯克回归,每一轮迭代在现有模型 $F(\boldsymbol{x})$的基础上加入 $f_t(\boldsymbol{x})$,新模型 $F(\boldsymbol{x})+f_t(\boldsymbol{x})$应使损失函数 L_l 下降最快,于是有:

$$\begin{aligned}f_t(\boldsymbol{x})&=\arg\min_{f(x)}L_l(\boldsymbol{x},y,F(\boldsymbol{x})+f(\boldsymbol{x}))\\&=\arg\min_{f(x)}E_{Y|X}[2y^*(F(\boldsymbol{x})+f(\boldsymbol{x}))y^*\log C_2-(1-y^*)\log C_1-\log(C_1+C_2\mathrm{e}^{2(F(x)+f(x))})\,|\,\boldsymbol{x}].\end{aligned}$$

通常,使用牛顿法解逻谛斯克回归问题需要计算 L_l 在 $f(\boldsymbol{x})$=0 处的一阶和二阶导数:

$$s(\boldsymbol{x})=\left.\frac{\partial L_l(F+f)}{\partial f}\right|_{f=0}=E_{Y|X}\left[2y^*-\frac{2C_2\mathrm{e}^{2F(\boldsymbol{x})}}{C_1+C_2\mathrm{e}^{2F(\boldsymbol{x})}}\,|\,\boldsymbol{x}\right]=2E_{Y|X}[y^*-p_c(\boldsymbol{x})\,|\,\boldsymbol{x}],$$

$$H(\boldsymbol{x})=\left.\frac{\partial^2 L_l(F+f)}{\partial f^2}\right|_{f=0}=E_{Y|X}\left[2y^*-\frac{2C_2\mathrm{e}^{2F(\boldsymbol{x})}}{C_1+C_2\mathrm{e}^{2F(\boldsymbol{x})}}\,|\,\boldsymbol{x}\right]=-4E_{Y|X}[p_c(\boldsymbol{x})(1-p_c(\boldsymbol{x}))\,|\,\boldsymbol{x}],$$

则牛顿法每一步的更新为

$$\begin{aligned}F(\boldsymbol{x})\leftarrow F(\boldsymbol{x})-H^{-1}(\boldsymbol{x})s(\boldsymbol{x})&=F(\boldsymbol{x})+\frac{1}{2}\frac{E[y^*-p_c(\boldsymbol{x})\,|\,\boldsymbol{x}]}{E[p_c(\boldsymbol{x})(1-p_c(\boldsymbol{x}))\,|\,\boldsymbol{x}]}\\&=F(\boldsymbol{x})+\frac{1}{2}E_{\varpi}\left[\left.\frac{y^*-p_c(\boldsymbol{x})}{p_c(\boldsymbol{x})(1-p_c(\boldsymbol{x}))}\right|x\right],\end{aligned}$$

其中,$\varpi(\boldsymbol{x})=p_c(\boldsymbol{x})(1-p_c(\boldsymbol{x}))$.

Boosting 算法中每一轮迭代求 $f(\boldsymbol{x})$,等价于使用最小平方误差准则解回归问题(7),可以使用像回归树 CART (classification and regression tree)这样的方法:

$$\min_{f(x)}E_{\varpi}\left[F(\boldsymbol{x})+\frac{1}{2}\frac{y^*-p_c(\boldsymbol{x})}{p_c(\boldsymbol{x})(1-p_c(\boldsymbol{x}))}-(F(\boldsymbol{x})+f(\boldsymbol{x}))\right]^2 \tag{7}$$

整理以上分析过程,得到代价敏感 LogitBoost 算法 AsyBL(算法 2).

算法 2. AsyBL 算法.

输入:$\langle\boldsymbol{x}_1,y_1\rangle,\langle\boldsymbol{x}_2,y_2\rangle,\ldots,\langle\boldsymbol{x}_m,y_m\rangle$;$y^*=(y+1)/2\in\{0,1\}$;代价参数 C_1,C_2.

初始化: $\varpi_i^1=1/m, z_i^1=\dfrac{y_i^*-p_c(\boldsymbol{x}_i)}{p_c(\boldsymbol{x}_i)(1-p_c(\boldsymbol{x}_i))}, p_c(\boldsymbol{x})=C_2/(C_1+C_2), f(\boldsymbol{x})=0.$

For i=1,…,T

1. 以 $\boldsymbol{z}^t$ 为拟合值,以 $\boldsymbol{\varpi}^t$ 为权值,使用加权最小二乘准则求回归方程 $f_t(\boldsymbol{x})$.
2. 更新回归问题的拟合值 $z_i^{t+1}=\dfrac{y_i^*-p_c(\boldsymbol{x}_i)}{p_c(\boldsymbol{x}_i)(1-p_c(\boldsymbol{x}_i))}$, 权值 $\varpi_i^{t+1}=p_c(\boldsymbol{x}_i)(1-p_c(\boldsymbol{x}_i))$.
3. 更新 $F(\boldsymbol{x})\leftarrow F(\boldsymbol{x})+\dfrac{1}{2}f_t(\boldsymbol{x})$.

输出: $H(\boldsymbol{x})=\mathrm{sign}\left(\sum_{t=1}^{T}f_t(\boldsymbol{x})\right)$.

3 实验评估

3.1 人工数据集实验

从第 1 节对人工数据集测试的结果可以看到,除了 AdaBoostC,Asymmetric Boosting 算法以外,其余基于启发式的代价敏感 Boosting 算法实际上并不是对代价敏感贝叶斯决策的有效近似,许多算法最终得到的代价敏感分类器极大地偏离了最优代价敏感决策.

为了验证 AsyB 以及 AsyBL 算法是否能够有效逼近代价敏感贝叶斯决策,在实验中使用与第 1 节图 1 相同的二维高斯人工数据集对 AsyB 以及 AsyBL 算法的决策边界进行可视化分析.指定代价因子 $f=C_1/C_2$ 分别为 2,5,8,10.AsyB 和 AsyBL 算法迭代 100 轮,分别使用线性感知器和回归树 CART 作为子分类器算法.

图 2 绘制出了对检测样本分类的决策边界,从图中可以直观地看到,调整代价因子,AsyB 和 AsyBL 算法始终能够较好地逼近代价敏感的贝叶斯决策.

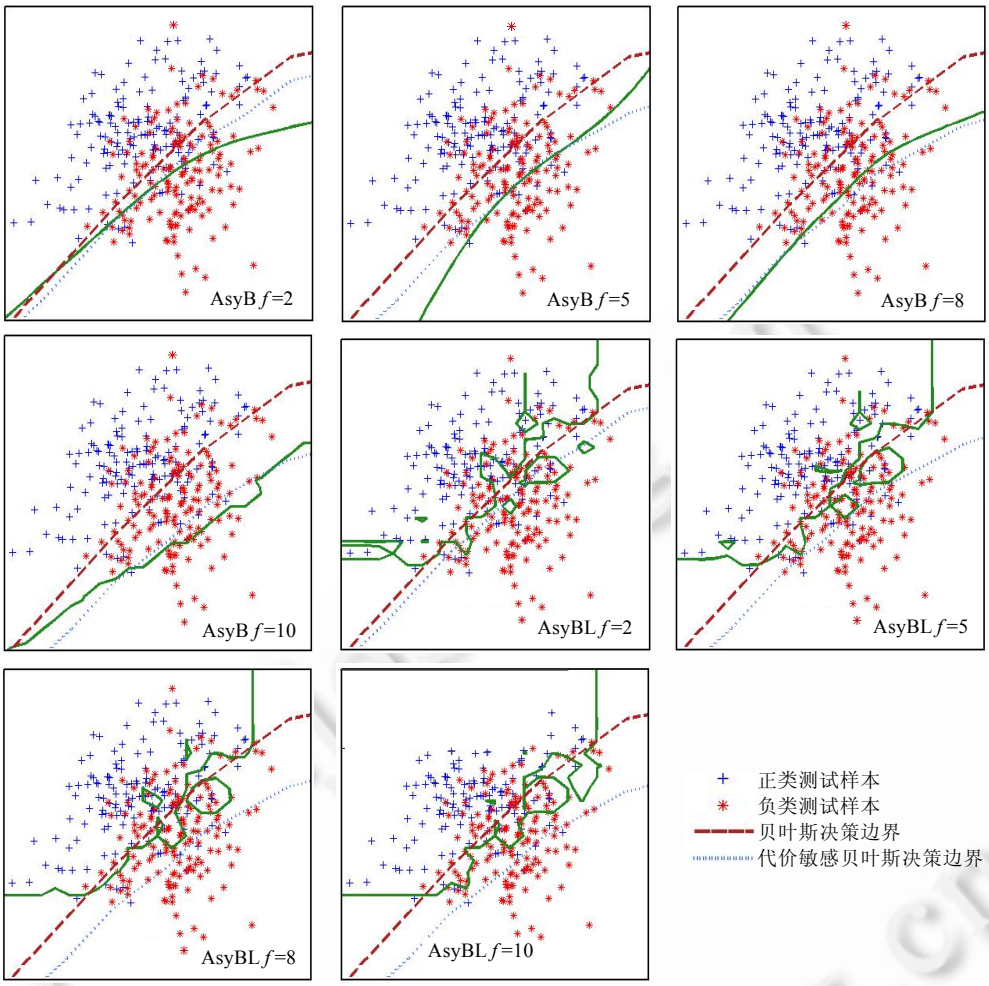

Fig.2 Decision boundaries of AsyB and AsyBL on synthetic Gaussian data

图 2 AsyB 以及 AsyBL 算法对高斯人工数据分类的决策边界

3.2 UCI数据集实验

在对人工数据集测试的基础上,接着选择 crx,page-block,phoneme,pima 以及 vowel 这 5 个 UCI 数据集对本文研究所涉及的各代价敏感 Boosting 算法进行评估.指定代价因子 C_1/C_2=5.所有算法均赋以代价敏感初始化样本权,迭代 50 次.除了 AsyBL 算法必须固定使用回归方法生成子分类器(实验中选择回归树 CART)以外,其余算法均使用决策树 C4.5 算法生成子分类器.同时,所有代价敏感 Boosting 算法与赋以代价敏感初始化样本权的 AdaBoost 算法进行比较.在五迭交叉验证样本集上计算各算法的错分类代价、Precision、Recall、*G*-mean、*F*-measure 以及 AUC 指标的均值与方差.表 1~表 5 是测试结果,粗体标注出了各评价指标下最好算法的结果.

表 1~表 5 的测试结果首先验证了文献[14]的结论:为 AdaBoost 算法赋以代价敏感的初始化样本权值可以直接用于代价敏感学习,结果甚至好于 AdaC 与 CSB 这样特别设计的代价敏感 Boosting 算法,但不如 AsyB 和 AsyBL 算法.因此,尽管 AdaBoost 算法本身也具有非对称学习能力,但是设计专门的代价敏感的 Boosting 算法依然是有价值的.与其他代价敏感 Boosting 算法相比,AsyB 和 AsyBL 稳定地具有更低的错分类代价,同时,无论是 Precision,Recall,*G*-mean,*F*-measure 这样的聚焦于单类分类性能的评价指标,还是 AUC 这样的评价性能指标,AsyB 以及 AsyBL 算法的性能都比较稳定.

从表 1~表 5 中的测试结果可以看出,在 AdaC 系列算法中,AdaC1 效果比 AdaC2 和 AdaC3 效果更好.因为 AdaC1 算法尽管在提出时采用启发式策略,但本质上恰巧优化与 Asymmetric Boosting 算法相同的损失函数,不同之处在于 Asymmetric Boosting 算法通过数值方法精确计算出加权投票因子α,而 AdaC1 求α的近似解.Precision 指标表示正类样本在所有预测为正类的样本中所占的比例,Recall 指标表示正类样本正确分类的比例.AdaC2,AdaC2 以及 CSB 系列算法的 Recall 指标常常为 1,但 Precision 指标却很低,这说明这些算法过度地聚焦于高代价样本,生成的分类器将所有样本不加选择地分为正类,这样的分类器是不合理的.

Table 1 Results of different cost sensitive Boosting algorithms on UCI dataset: crx

表 1 各代价敏感 Boosting 算法在 UCI 数据集上 crx 的检测结果

算法名称	错分类代价	Precision	Recall	*G*-mean	*F*-Measure	AUC
AdaBoostC	57.40 (9.04)	0.81 (0.06)	0.85 (0.03)	0.42 (0.02)	0.83 (0.03)	0.90 (0.05)
AdaCost	49.40 (16.98)	0.75 (0.11)	**0.90 (0.04)**	0.40 (0.04)	0.82 (0.07)	0.89 (0.07)
AdaC1	48.40 (19.32)	0.82 (0.12)	0.88 (0.04)	0.42 (0.04)	0.84 (0.08)	0.91 (0.07)
AdaC2	71.40 (0.55)	0.45 (0.00)	1.00 (0.00)	0.00 (0.00)	0.62 (0.00)	0.86 (0.08)
AdaC3	71.40 (0.55)	0.45 (0.00)	1.00 (0.00)	0.00 (0.00)	0.62 (0.00)	0.87 (0.07)
CSB0	71.40 (0.55)	0.45 (0.00)	1.00 (0.00)	0.00 (0.00)	0.62 (0.00)	0.89 (0.08)
CSB1	71.40 (0.55)	0.45 (0.00)	1.00 (0.00)	0.00 (0.00)	0.62 (0.00)	0.76 (0.08)
CSB2	70.20 (2.95)	0.46 (0.01)	1.00 (0.00)	0.03 (0.06)	0.63 (0.01)	0.82 (0.06)
AsyBoosting	54.80 (20.08)	0.88 (0.09)	0.84 (0.06)	0.43 (0.03)	**0.86 (0.07)**	0.91 (0.06)
AsyLogitBoost	73.00 (19.82)	0.75 (0.14)	0.81 (0.05)	0.39 (0.04)	0.78 (0.08)	0.80 (0.06)
AsyB	**45.00 (24.55)**	0.82 (0.11)	0.89 (0.06)	0.43 (0.04)	0.85 (0.08)	0.89 (0.07)
AsyBL	58.80 (20.24)	**0.88 (0.08)**	0.82 (0.06)	**0.43 (0.03)**	0.85 (0.06)	**0.92 (0.06)**

Table 2 Results of different cost sensitive Boosting algorithms on UCI dataset: page-block

表 2 各代价敏感 Boosting 算法在 UCI 数据集上 page-block 的检测结果

算法名称	错分类代价	Precision	Recall	*G*-mean	*F*-Measure	AUC
AdaBoostC	92.80 (35.87)	0.76 (0.10)	0.89 (0.04)	**0.28 (0.01)**	0.82 (0.07)	0.98 (0.02)
AdaCost	98.60 (34.06)	**0.84 (0.09)**	0.86 (0.05)	**0.28 (0.01)**	0.85 (0.06)	0.98 (0.01)
AdaC1	101.00 (33.56)	0.83 (0.08)	0.85 (0.05)	**0.28 (0.01)**	0.84 (0.06)	0.98 (0.01)
AdaC2	655.00 (204.16)	0.15 (0.04)	0.99 (0.01)	0.16 (0.07)	0.26 (0.05)	0.98 (0.02)
AdaC3	758.20 (314.85)	0.15 (0.07)	1.00 (0.00)	0.10 (0.12)	0.25 (0.10)	0.98 (0.01)
CSB0	908.40 (113.73)	0.11 (0.01)	0.99 (0.02)	0.05 (0.07)	0.20 (0.02)	0.98 (0.02)
CSB1	552.20 (314.36)	0.23 (0.17)	0.99 (0.02)	0.18 (0.11)	0.35 (0.20)	0.96 (0.02)
CSB2	598.00 (338.62)	0.20 (0.10)	0.99 (0.01)	0.17 (0.10)	0.32 (0.14)	0.96 (0.04)
AsyBoosting	97.60 (41.41)	0.83 (0.09)	0.86 (0.06)	**0.28 (0.01)**	0.84 (0.07)	0.98 (0.01)
AsyLogitBoost	165.20 (37.73)	0.70 (0.10)	0.77 (0.05)	0.26 (0.01)	0.73 (0.07)	0.86 (0.04)
AsyB	**77.40 (35.83)**	0.81 (0.09)	**0.91 (0.05)**	**0.28 (0.01)**	0.85 (0.06)	**0.99 (0.01)**
AsyBL	81.60 (37.41)	0.83 (0.09)	0.89 (0.05)	**0.28 (0.01)**	**0.86 (0.06)**	0.97 (0.02)

Table 3 Results of different cost sensitive Boosting algorithms on UCI dataset: phoneme

表 3 各代价敏感 Boosting 算法在 UCI 数据集上 phoneme 的检测结果

算法名称	错分类代价	Precision	Recall	*G*-mean	*F*-Measure	AUC
AdaBoostC	296.00 (45.14)	0.69 (0.02)	0.90 (0.02)	0.39 (0.01)	0.78 (0.02)	0.93 (0.01)
AdaCost	417.60 (36.51)	0.44 (0.02)	**0.99 (0.01)**	0.31 (0.02)	0.61 (0.02)	0.92 (0.00)
AdaC1	313.20 (13.63)	0.76 (0.02)	0.86 (0.01)	0.40 (0.00)	0.81 (0.01)	0.95 (0.01)
AdaC2	708.20 (53.89)	0.31 (0.02)	1.00 (0.00)	0.09 (0.09)	0.47 (0.02)	0.86 (0.01)
AdaC3	738.40 (34.65)	0.30 (0.01)	1.00 (0.00)	0.05 (0.07)	0.46 (0.01)	0.88 (0.02)
CSB0	737.00 (36.76)	0.30 (0.01)	1.00 (0.00)	0.05 (0.07)	0.46 (0.01)	0.93 (0.01)
CSB1	744.60 (42.26)	0.30 (0.01)	1.00 (0.00)	0.03 (0.07)	0.46 (0.01)	0.77 (0.10)
CSB2	719.00 (48.51)	0.31 (0.01)	1.00 (0.00)	0.09 (0.07)	0.47 (0.02)	0.78 (0.08)
AsyBoosting	331.00 (20.95)	0.80 (0.01)	0.83 (0.01)	0.40 (0.00)	0.81 (0.01)	**0.95 (0.00)**
AsyLogitBoost	552.60 (43.16)	0.61 (0.01)	0.75 (0.03)	0.35 (0.01)	0.67 (0.02)	0.77 (0.04)
AsyB	**254.80 (22.15)**	0.71 (0.02)	0.91 (0.01)	0.40 (0.00)	0.80 (0.01)	0.94 (0.01)
AsyBL	265.20 (20.39)	**0.81 (0.01)**	0.88 (0.01)	**0.41 (0.00)**	**0.84 (0.01)**	**0.95 (0.00)**

Table 4 Results of different cost sensitive Boosting algorithms on UCI dataset: pima

表 4 各代价敏感 Boosting 算法在 UCI 数据集上 pima 的检测结果

算法名称	错分类代价	Precision	Recall	*G*-mean	*F*-Measure	AUC
AdaBoostC	92.00 (8.89)	0.60 (0.02)	0.76 (0.03)	0.35 (0.01)	0.67 (0.02)	0.80 (0.03)
AdaCost	210.00 (114.05)	0.28 (0.24)	0.41 (0.38)	0.20 (0.13)	0.33 (0.29)	0.43 (0.35)
AdaC1	99.60 (9.40)	0.60 (0.04)	0.72 (0.04)	0.35 (0.01)	0.66 (0.03)	**0.82 (0.02)**
AdaC2	100.00 (0.00)	0.35 (0.00)	1.00 (0.00)	0.00 (0.00)	0.52 (0.00)	0.74 (0.05)
AdaC3	100.00 (0.00)	0.35 (0.00)	1.00 (0.00)	0.00 (0.00)	0.52 (0.00)	0.76 (0.02)
CSB0	100.00 (0.00)	0.35 (0.00)	1.00 (0.00)	0.00 (0.00)	0.52 (0.00)	0.77 (0.04)
CSB1	99.40 (1.34)	0.35 (0.00)	1.00 (0.00)	0.02 (0.04)	0.52 (0.01)	0.72 (0.03)
CSB2	99.20 (1.30)	0.35 (0.00)	1.00 (0.00)	0.03 (0.04)	0.52 (0.00)	0.71 (0.07)
AsyBoosting	95.20 (20.56)	0.58 (0.05)	0.75 (0.07)	0.35 (0.02)	0.66 (0.06)	0.81 (0.04)
AsyLogitBoost	154.40 (10.62)	0.47 (0.02)	0.55 (0.06)	0.29 (0.01)	0.50 (0.02)	0.63 (0.04)
AsyB	**83.00 (10.84)**	0.59 (0.01)	**0.80 (0.04)**	**0.36 (0.01)**	**0.68 (0.02)**	0.81 (0.01)
AsyBL	105.40 (8.56)	**0.61 (0.01)**	0.69 (0.03)	0.35 (0.01)	0.65 (0.02)	0.80 (0.01)

Table 5 Results of different cost sensitive Boosting algorithms on UCI dataset: vowel

表 5 各代价敏感 Boosting 算法在 UCI 数据集上 vowel 的检测结果

算法名称	错分类代价	Precision	Recall	*G*-mean	*F*-Measure	AUC
AdaBoostC	12.20 (8.07)	0.76 (0.23)	0.94 (0.07)	0.27 (0.01)	0.82 (0.13)	0.97 (0.05)
AdaCost	45.60 (66.65)	0.59 (0.40)	0.98 (0.03)	0.24 (0.07)	0.67 (0.37)	0.95 (0.10)
AdaC1	12.40 (7.83)	**0.84 (0.20)**	0.91 (0.07)	0.27 (0.01)	0.86 (0.12)	0.99 (0.01)
AdaC2	72.40 (33.29)	0.23 (0.09)	1.00 (0.00)	0.22 (0.03)	0.36 (0.12)	0.99 (0.01)
AdaC3	47.80 (39.71)	0.41 (0.28)	0.97 (0.05)	0.24 (0.04)	0.53 (0.25)	0.99 (0.01)
CSB0	93.20 (86.80)	0.45 (0.46)	0.98 (0.05)	0.15 (0.14)	0.50 (0.41)	0.80 (0.27)
CSB1	46.60 (40.04)	0.37 (0.18)	0.99 (0.02)	0.24 (0.04)	0.51 (0.20)	0.99 (0.01)
CSB2	64.00 (28.31)	0.25 (0.11)	1.00 (0.00)	0.23 (0.03)	0.39 (0.13)	0.99 (0.01)
AsyBoosting	17.80 (10.92)	0.75 (0.26)	0.89 (0.09)	0.26 (0.01)	0.79 (0.16)	0.98 (0.02)
AsyLogitBoost	14.20 (6.18)	0.71 (0.18)	0.93 (0.05)	0.27 (0.01)	0.79 (0.11)	0.95 (0.06)
AsyB	**8.60 (5.03)**	0.80 (0.19)	**0.97 (0.05)**	**0.28 (0.01)**	0.86 (0.11)	**0.99 (0.00)**
AsyBL	9.20 (7.46)	0.81 (0.20)	0.96 (0.05)	0.28 (0.01)	**0.87 (0.12)**	0.99 (0.01)

3.3 错分类代价随迭代变化

AdaBoost 算法最重要的特性之一是分类错误率随着迭代的进行呈指数速率下降.将 Boosting 思想应用于代价敏感学习,也是希望能够对错分类代价进行提升,使得错分类代价随着迭代的进行不断下降.为了验证各代价敏感 Boosting 算法是否能够对错分类代价进行提升,使得错分类代价随着迭代过程不断下降,我们对第 3.2 节实验中,在 phoneme 的五迭交叉验证样本集上训练的各代价敏感 Boosting 算法绘制训练样本集与检测样本集上错分类代价随迭代进行的变化曲线,如图 3 所示.

从图 3 中可以看到:

- AsyB 和 AsyBL 算法能够对错分类代价进行提升,训练样本集与检测样本机上错分类代价随着迭代的进行呈指数速率下降.
- 此外,赋予代价敏感初始化样本权的 AdaBoost,AdaC1,Asymmetric Boosting,Asymmetric LogitBoost 算法也能对错分类代价进行提升,但是提升效果比 AsyB 算法要差.并且在更多数据集的实验中可以观察到,与 AsyB 和 AsyBL 算法相比,这些算法对错分类代价的提升作用并不稳定.
- AdaC2,AdaC3 以及 CSB 系列算法对 AdaBoost 算法的修改破坏了算法的 Boosting 特性,随着迭代的进行,错分类代价甚至出现不下降反而升高的现象.

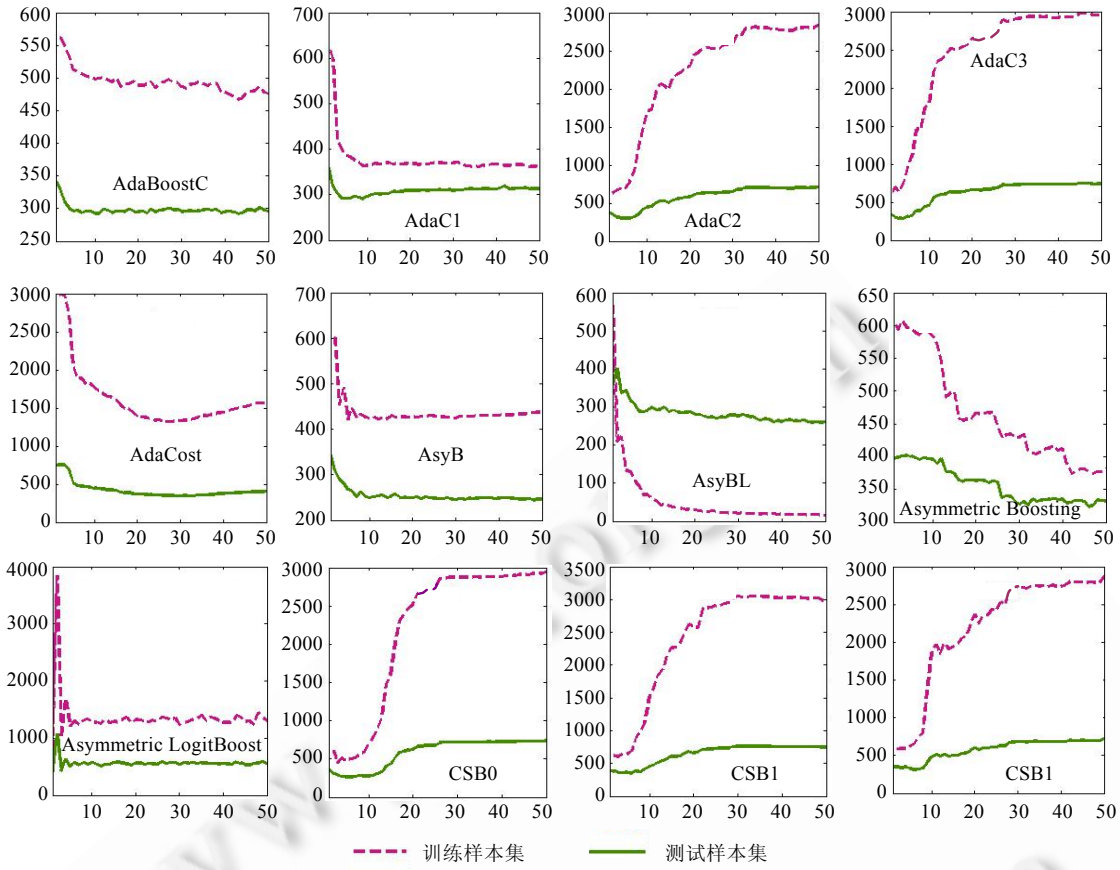

Fig.3 Cost curves of different cost sensitive Boosting algorithms on UCI dataset phoneme

图 3 UCI 数据集 phoneme 上各代价敏感 Boosting 算法代价曲线

4 结 论

Boosting 是一种重要的集成学习元方法,也是一种重要的优化思想,在代价敏感学习中起着重要的作用.代价敏感 Boosting 算法能够将以最小错误率为目标的标准学习算法改造为降低错分类代价的代价敏感学习算法.本文算法首先对 AdaBoost 算法使用的分类间隔的指数损失函数和 LogitBoost 使用的 Logit 损失函数进行代价敏感改造.可以证明,理想情况下,令新的损失函数最小的决策是代价敏感贝叶斯决策.然后使用函数空间梯度下降优化方法,通过理论分析确定 Boosting 算法中的加权投票因子以及权值更新策略,提出了代价敏感 Boosting 算法:AsyB 和 AsyBL.当两类分类错误代价相等时,AsyB 和 AsyBL 算法分别退化为标准 AdaBoost 和 LogitBoost 算法,并且与原 AdaBoost 算法相比,AsyB 与 AsyBL 算法的复杂度没有任何改变.对人工数据、UCI 数据的测试结果表明:AsyB 和 AsyBL 算法能够合理地逼近最优代价敏感贝叶斯决策,使得错分类代价随着迭代的进行呈指数下降,最终得到具有更低错分类代价的代价敏感分类器.

References:

[1] López V, Fernández A, Moreno-Torres JG, Herrera F. Analysis of preprocessing vs. cost-sensitive learning for imbalanced classification. Open problems on intrinsic data characteristics. Expert Systems with Applications, 2012,39(7):6585−6608. [doi: 10.1016/j.eswa.2011.12.043]

[2] Lomax S, Vadera S. A survey of cost-sensitive decision tree induction algorithms. ACM Computing Surveys (CSUR), 2013,45(2): 16−50. [doi: 10.1145/2431211.2431215]

[3] Domingos P. Metacost: A general method for making classifiers cost-sensitive. In: Proc. of the 5th ACM SIGKDD Int'l Conf. on Knowledge Discovery and Data Mining (KDD'99). New York: ACM Press, 1999. 155−164. [doi: 10.1145/312129.312220]

[4] Elkan C. The foundations of cost-sensitive learning. In: Proc. of the 17th Int'l Joint Conf. on Artificial Intelligence (IJCAI 2001). San Francisco: Morgan Kaufmann Publishers, 2001. 973−978.

[5] Zadrozny B, Langford J, Abe N. Cost-Sensitive learning by cost-proportionate example weighting. In: Proc. of the 3rd IEEE Int'l Conf. on Data Mining (ICDM 2003). New York: IEEE, 2003. 435−442. [doi: 10.1109/ICDM.2003.1250950]

[6] Krawczyk B, Woźniak M. Designing cost-sensitive ensemble-genetic approach. Advances in Intelligent and Soft Computing, 2011, 102:227−234. [doi: 10.1007/978-3-642-23154-4_26]

[7] Schapire RE, Freund Y. Boosting: Foundations and Algorithms. Cambridge: The MIT Press, 2012.

[8] Fan W, Stolfo SJ, Zhang J, Chan PK. AdaCost: Misclassification cost-sensitive boosting. In: Proc. of the 16th Int'l Conf. on Machine Learning. San Francisco: Morgan Kaufmann Publishers, 1999. 97−105.

[9] Sun Y, Wong AK, Wang Y. Parameter inference of cost-sensitive boosting algorithms. In: Proc. of the 4th Int'l Conf. on Machine Learning and Data Mining in Pattern Recognition. Berlin: Springer-Verlag, 2005. 21−30. [doi: 10.1007/11510888_3]

[10] Sun Y, Kamel MS, Wong AK, Wang Y. Cost-Sensitive boosting for classification of imbalanced data. Pattern Recognition, 2007, 40(12):3358−3378. [doi: 10.1016/j.patcog.2007.04.009]

[11] Ting KM. A comparative study of cost-sensitive boosting algorithms. In: Proc. of the 17th Int'l Conf. on Machine Learning (ICML 2000). San Francisco: Morgan Kaufmann Publishers, 2000. 983−990.

[12] Masnadi-Shirazi H, Vasconcelos N. Asymmetric Boosting. In: Proc. of the 24th Int'l Conf. on Machine Learning (ICML 2007). San Francisco: Margan Kaufmann Publishers, 2007. 609−619. [doi: 10.1145/1273496.1273573]

[13] Masnadi-Shirazi H, Vasconcelos N. Cost-Sensitive Boosting. IEEE Trans. on Pattern Analysis and Machine Intelligence, 2011, 33(2):294−309. [doi: 10.1109/TPAMI.2010.71]

[14] Landesa-Vázquez I, Alba-Castro JL. Shedding light on the asymmetric learning capability of AdaBoost. Pattern Recognition Letters, 2011,33(3):247−255. [doi: 10.1016/j.patrec.2011.10.022]

[15] Lin Y. A note on margin-based loss functions in classification. Statistics & Probability Letters, 2004,68(1):73−82. [doi: 10.1016/ j.spl.2004.03.002]

[16] Zou H, Zhu J, Hastie T. New multicategory boosting algorithms based on multicategory fisher-consistent losses. The Annals of Applied Statistics, 2008,2(4):1290−1306. [doi: 10.1214/08-AOAS198]

[17] Friedman J, Hastie T, Tibshirani R. Additive logistic regression: A statistical view of boosting (with discussion and a rejoinder by the authors). The Annals of Statistics, 2000,28(2):337−407. [doi: 10.1214/aos/1016218223]

曹莹(1987－),女,陕西西安人,博士生,主要研究领域为机器学习,网络安全.
E-mail: yingcao@stu.xidian.edu.cn

刘家辰(1988－),男,博士生,主要研究领域为机器学习,计算机安全.
E-mail: jcliu@stu.xidian.edu.cn

苗启广(1972－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为智能图像处理,机器学习,计算机安全.
E-mail: qgmiao@mail.xidian.edu.cn

高琳(1964－),女,博士,教授,博士生导师,主要研究领域为数据挖掘,计算生物信息学,图论与组合优化算法及其应用.
E-mail: lgao@mail.xidian.edu.cn