第46卷第9期 中国科学技术大学学报 Vol. 46, No. 9
2016年9月 JOURNAL OF UNIVERSITY OF SCIENCE AND TECHNOLOGY OF CHINA Sep. 2016

文章编号:0253-2778(2016)9-0727-09

一种增量式的代价敏感支持向量机

权 鑫[1,2],顾韵华[1,2],郑关胜[1,2],顾 彬[1,2]

(1. 江苏省网络监控工程中心,江苏南京 210044;
2. 南京信息工程大学计算机与软件学院,江苏南京 210044)

摘要:代价敏感学习是机器学习中一个重要的领域.由Masnadi等提出的代价敏感的支持向量机通过将铰链损失函数代价敏感化来处理代价敏感问题,比传统的代价敏感学习方法具有更好的泛化精度.现实中的数据往往是通过在线增量式获取的,而传统的全量式学习算法每次增加样本时都需要重新从头计算,因此浪费了很多时间.为了使得代价敏感的支持向量机能够在在线学习的场景下具有更高的效率,提出了一种增量式的代价敏感支持向量机算法.该算法可以在新增样本时直接更新已有的训练过的模型,不需要从头开始重新训练.在多个数据集上的实验结果也显示出了该方法与传统的批处理方法相比,在速度上的具有显著的优势.

关键词:在线学习;增量式学习;代价敏感学习;支持向量机

中图分类号:TP181 **文献标识码**:A doi:10.3969/j.issn.0253-2778.2016.09.003

引用格式:权鑫,顾韵华,郑关胜,等.一种增量式的代价敏感支持向量机[J].中国科学技术大学学报,2016,46(9):727-735.

QUAN Xin, GU Yuanhua, ZHENG Guansheng, et al. An incremental cost-sensitive support vector machine[J]. Journal of University of Science and Technology of China, 2016,46(9):727-735.

An incremental cost-sensitive support vector machine

QUAN Xin[1,2], GU Yuanhua[1,2], ZHENG Guansheng[1,2], GU Bin[1,2]

((1. *Jiangsu Engineering Center of Network Monitoring, Nanjing* 210044, *China*;
2. *College of Computer Science & Software, Nanjing University of Information Science & Technology, Nanjing* 210044, *China*))

Abstract: Cost-sensitive learning is an important field in machine learning, which widely exists in real-world applications, such as cancer diagnosis, credit application, etc. Cost-sensitive support vector machine proposed by Masnadi et al. handles cost-sensitive problems through making the hinge loss function cost-sensitive, which has better generalization accuracy than other traditional cost-sensitive algorithms. In practice data are obtained one batch after another. Conventional batch algorithms would waste a lot of time when appending samples, because they should re-train the model from scratch. To make the cost-sensitive support vector machine more practical in on-line learning problems, an incremental cost-sensitive support vector machine algorithm was proposed, which can directly update the trained model without re-training it from scratch when appending samples. Experiment study on several datasets show that our algorithm is

收稿日期:2016-03-01;**修回日期**:2016-09-17

基金项目:国家自然科学基金(61573191),江苏省大数据分析技术重点实验室开放课题(KXK1405),江苏省自然科学基金(BK20161534)资助.

作者简介:权鑫,男,1991年生,硕士生.研究方向:数据挖掘. E-mail: quanxin@nuist.edu.cn

通讯作者:顾彬,博士/副教授. E-mail: jsgubin@163.com

significantly more efficient than batch algorithms of the cost-sensitive support vector machine.
Key words: on-line learning; incremental learning; cost sensitive learning; support vector machine

0 引言

分类问题的学习目标是从训练数据集中训练出分类器,并用训练得到的分类器来对测试数据集中的样本进行类标的预测.在训练过程中,通常采用最大化分类精度(accuracy)或最小化误分率作为优化目标.在标准的分类学习中假设所有的分类错误将会产生相同的代价,也就是说不同的误分类将导致相同的结果.相应地,代价敏感学习[1]中将考虑不同的误分类导致的不同结果.

代价敏感学习(cost-sensitive learning)是机器学习中一个重要的领域.在现实中经常会碰到不同的误分类导致不同代价的场景.例如,在癌症检测[2]问题中,将一个癌症患者判定为健康人有可能会耽误治疗,甚至导致死亡;而将一个正常人检测为癌症患者则仅仅会导致精神上的惊吓以及金钱上的损失.在这个例子中,不同的误分类就会导致不同的误分代价.同理,还有许多类似于此的某些类别的样本数量比较少但是代价比较高的例子,如信用级别判定、恐怖分子甄别等问题.代价敏感学习中则将不同误分类导致的不同代价引入了算法,从而可以处理上述代价敏感问题.同时,代价敏感学习还能够比较好地处理训练集中不同类别样本数量不平衡的问题[3],即不平衡学习(imbalanced learning).

近年来,代价敏感学习引起了越来越多研究者的注意,提出了多种代价敏感学习方法,包括改变样本分布的 MetaCost[4] 与重采样技术[5-6]、集成学习技术[7-8]以及修改原非代价敏感算法使其代价敏感化[9-15]等方法.这其中又包括多种代价敏感的支持向量机算法[11-15].本文的研究主要集中在支持向量机上.传统的非代价敏感的支持向量机[16]中使用一组松弛变量来处理线性不可分问题,并使用惩罚参数 C 来控制松弛变量.与其他的非代价敏感算法类似,标准的支持向量机假设不同的误分类会引起相同的代价.为了使得支持向量机能够处理代价敏感问题,BP-SVM[11,17-19]引入了另外两个分别对正类和负类的松弛变量的惩罚参数 C_+ 和 C_-.BP-SVM 存在一定的问题,当惩罚参数 C 非常大的时候,BP-SVM 将退化为标准 SVM.为了解决这个问题,Masnadi 等提出了 CS-SVM[14-15].这种代价敏感的支持向量机算法通过代价敏感的铰链损失函数进行推导得到,从而可以保证与代价敏感的贝叶斯风险的一致性,并且在 C 较大的情况下以及线性可分的情况下仍然可以很好地处理代价敏感问题,比传统的代价敏感学习方法具有更好的泛化精度.因而本文以 CS-SVM 作为研究对象.

代价敏感的支持向量机与传统的支持向量机一样,都是二次规划问题.这种二次规划问题通常使用分解算法等批处理算法来解决.而在实际的机器学习和数据挖掘应用场景中,数据大多是在在线(online)环境[20-21]下增量提供的,即随着时间的推移,会不断有新的样本加入训练数据集.在分解算法等批处理算法中,所有的样本一次性地被获取并全部用来训练支持向量机的模型.如果在模型建立之后再次添加样本,批处理算法需要从头重新训练所有的样本.尽管有一些类似于序列最小优化算法(SMO)[22]的快速训练算法被提出,但从头重新训练所有数据还是耗费了大量的时间,往往不具有实时性.针对这个问题,使用在线学习的方法,如增量式学习[23-27],就可以直接吸收额外的数据样本而不需要对原有数据进行重新计算,故而对于增量式支持向量机的研究将显得很有意义.

本文针对 CS-SVM 提出了一种增量式算法.该算法可以在代价敏感问题中新增样本时直接更新已有的训练过的模型而不需要从头开始重新训练.目前为止未见有研究者从事代价敏感的支持向量机的增量式算法的研究成果发表.在多个数据集上的实验结果不仅证实了代价敏感的支持向量机的有效性,同时也显示出了本文的方法与传统的批处理方法相比具有显著的效率上的优势.

1 代价敏感的支持向量机及其 KKT 条件

下面介绍代价敏感的支持向量机的形式以及其相应的 KKT 条件.

假设有二类分类的训练数据集 $\{(x_i,y_i)\mid i\in S\}$.其中,$x_i\in\chi\subseteq R^d$ 且 $y_i\in\{+1,-1\}$.代价敏感的支持向量机的主要目的是学习一个判别函数,如下式所示:

$$f(x)=\boldsymbol{w}^{\mathrm{T}}\varphi(x)+b \tag{1}$$

式中，$\varphi(x)$ 表示一个映射函数，该映射函数通常将样本的特征空间映射到一个更高维的空间，甚至是无限维的空间. 参数 w 和 b 可以通过解下面公式(2)得到，公式(2)成为代价敏感的支持向量机的原问题.

$$\left.\begin{aligned}\min_{w,b,\xi_i}\quad & \frac{1}{2}\|\boldsymbol{w}\|^2+C\Big[\beta\sum_{i\in S_+}\xi_i+\lambda\sum_{i\in S_-}\xi_i\Big]\\ \text{s.t.}\quad & f(x_i)\geqslant 1-\xi_i,\quad i\in S_+\\ & f(x_i)\leqslant -\kappa+\xi_i,\quad i\in S_-\\ & \xi_i\geqslant 0,\quad i\in S\end{aligned}\right\}\tag{2}$$

式中，

$$\beta=C_+,\quad \lambda=2C_--1,\quad \kappa=\frac{1}{2C_--1}.$$

需要注意的是，C、C_+ 与 C_- 为正则化参数，它们是代价敏感的支持向量机中的几个关键参数. 为了下面表述方便，本文将原问题简记为：

$$\left.\begin{aligned}\min_{w,b,\xi_i}\quad & \frac{1}{2}\|w\|^2+\sum_{i\in S}C_i\xi_i\\ \text{s.t.}\quad & y_if(x_i)\geqslant\Omega_i-\xi_i\\ & \xi_i\geqslant 0\end{aligned}\right\}\tag{3}$$

式中，

$$C_i=\begin{cases}CC_+, & i\in S_+\\ C(2C_--1), & i\in S_-\end{cases};$$

$$\Omega_i=\begin{cases}1, & i\in S_+\\ \dfrac{1}{2C_--1}, & i\in S_-\end{cases}.$$

为了解决原问题，本文引入拉格朗日乘子 $\alpha_i,\mu_i\geqslant 0$，即可以将原问题对应的拉格朗日函数写为：

$$L=\frac{1}{2}\|\boldsymbol{w}\|^2+\sum_{i\in S}C_i\xi_i+\sum_{i\in S}\alpha_i(\Omega_i-\xi_i-y_if(x_i))-\sum_{i\in S}\mu_i\xi_i\tag{4}$$

将拉格朗日函数分别根据 w、b 和 ξ_i 求偏导数，并令其等于 0，可以得到如下三个公式.

$$\left.\begin{aligned}\frac{\partial L}{\partial w}=0\quad\Leftrightarrow\quad & w=\sum_{i\in S}\alpha_iy_i\varphi(x_i)\\ \frac{\partial L}{\partial b}=0\quad\Leftrightarrow\quad & \sum_{i\in S}\alpha_iy_i=0\\ \frac{\partial L}{\partial \xi_i}=0\quad\Leftrightarrow\quad & C_i-\mu_i-\alpha_i=0\end{aligned}\right\}\tag{5}$$

将公式(5)带入公式(4)可以得到代价敏感的支持向量机的对偶问题. 如下式所示：

$$\left.\begin{aligned}\max_{\alpha}\quad & \frac{1}{2}\boldsymbol{\alpha}^{\mathrm{T}}Q\boldsymbol{\alpha}+\boldsymbol{\alpha}^{\mathrm{T}}\Omega\\ \text{s.t.}\quad & \boldsymbol{\alpha}^{\mathrm{T}}y=0\\ & 0\leqslant\alpha_i\leqslant C_i\end{aligned}\right\}\tag{6}$$

式中，$Q\in R^{n\times n}$，每个元素 $Q_{ij}=y_iy_jK(x_i,x_j)$，并有 $K(x_i,x_j)=\varphi(x_i)\varphi(x_j)$. $K(x_i,x_j)$ 为核函数，它可以在不需要知道映射函数 $\varphi(\cdot)$ 的情况下计算映射之后的 $\varphi(x_i)$ 和 $\varphi(x_j)$ 的乘积. 对应上述对偶问题的 KKT 条件如下式所示：

$$\left.\begin{aligned}&\alpha_i(\Omega_i-\xi_i-y_if(x_i))=0\\ &\xi_i(C_i-\alpha_i)=0\end{aligned}\right\}\tag{7}$$

此时，通过引入拉格朗日乘子 α_i 与核函数 $K(x_i,x_j)$，判别函数(1)可以表示为：

$$f(x)=\sum_{i\in S}\alpha_iy_iK(x,x_i)+b\tag{8}$$

式中，偏置项 b 可以通过 KKT 条件(7)得到. 在取到最优解的情况下，$y_if(x_i)$ 和其相应的拉格朗日乘子 α_i 之间的关系必须满足：

$$\left.\begin{aligned}&y_if(x_i)>\Omega_i\quad\Rightarrow\quad\alpha_i=0\\ &y_if(x_i)=\Omega_i\quad\Rightarrow\quad 0\leqslant\alpha_i\leqslant C_i\\ &y_if(x_i)<\Omega_i\quad\Rightarrow\quad\alpha_i=C_i\end{aligned}\right\}\tag{9}$$

根据上述公式，本文定义了 3 个对应的下标集合.

$$\left.\begin{aligned}&O=\{i:y_if(x_i)>\Omega_i,\alpha_i=0\}\\ &M=\{i:y_if(x_i)=\Omega_i,0\leqslant\alpha_i\leqslant C_i\}\\ &I=\{i:y_if(x_i)<\Omega_i,\alpha_i=C_i\}\end{aligned}\right\}\tag{10}$$

下面将定义一些在下文的表述中使用的符号. 文中使用 $\boldsymbol{v}_M$ 表示向量 $\boldsymbol{v}$ 的子向量，在这个子向量中所有的元素以集合 $\boldsymbol{M}$ 中的值为下标. 类似的，$\boldsymbol{M}_{M,I}$ 表示 $\boldsymbol{M}$ 的子矩阵，它以集合 $\boldsymbol{M}$ 中的值索引行、以集合 $\boldsymbol{I}$ 中的值索引列. 另外，如果一个子矩阵的行列索引集合相同，例如 $\boldsymbol{M}_{M,M}$，文中将简记为 $\boldsymbol{M}_M$.

2　增量式的代价敏感支持向量机

本节提出了增量式的代价敏感支持向量机(ICSSVM)的推导过程，并给出了详细的增量式算法描述.

假设向现有训练样本集中添加一个新增样本 (x_c,y_c)，其对应的对偶问题中的拉格朗日乘子为 α_c，即 $\alpha=(\alpha_1,\cdots,\alpha_n,\alpha_c)$，另有 $Q\in R^{(n+1)\times(n+1)}$.

接下来将讨论 α_c 的值. 文中首先初始化 $\alpha_c=0$. 此时，如果 $y_cf(x_c)>\Omega_c$，则将该样本直接添加进集合 O，因为它已经满足了最优化条件(10). 同理，如果 $y_cf(x_c)=\Omega_c$，则将该样本添加进集合 M. 若上述条件都不满足，则逐步增加 α_c，并在增加的过程中保证最优化条件(10)中的集合不会发生变动，即没有样本跨越集合边界，从一个集合进入另一

个集合.因此需要确定 α_c 的增量,文中使用 $\Delta\alpha_c$ 来表示,并令其满足:

$$\Delta\alpha_c = \eta(C_c - \alpha_c) \tag{11}$$

式中,η 为 α_c 增量的步长,且满足 $0 \leqslant \eta \leqslant 1$.

需要注意的是,当增加 α_c 时,其它参数需要保证始终满足 KKT 最优化条件.由 $y_i f(x_i) = \Omega_i$, $i \in \boldsymbol{M}$,可以得到

$$Q_{ic}\Delta\alpha_c + \sum_{j\in M} Q_{ij}\Delta\alpha_j + y_i\Delta b = 0,\ i \in M \tag{12}$$

同理,由对偶问题(6)的等式约束可以得到

$$y_c\Delta\alpha_c + \sum_{j\in M} y_j\Delta\alpha_j = 0 \tag{13}$$

将上述两个公式写成矩阵形式可以得到

$$\boldsymbol{M}\begin{bmatrix}\Delta b\\ \Delta\alpha_M\end{bmatrix} + \begin{bmatrix}y_c\\ \boldsymbol{Q}_{M,c}\end{bmatrix}\Delta\alpha_c = 0 \tag{14}$$

式中,

$$\boldsymbol{M} = \begin{bmatrix}0 & \boldsymbol{y}_M^T\\ \boldsymbol{y}_M & \boldsymbol{Q}_M\end{bmatrix}.$$

根据3个下标集合 O、M 与 I 的定义(10),如下3个条件必须同时满足.

$$\left.\begin{aligned} y_i(f(x_i)+\Delta f(x_i)) > \Omega_i, &\quad i \in O\\ 0 \leqslant \alpha_i + \Delta\alpha_i \leqslant C_i, &\quad i \in M\\ y_i(f(x_i)+\Delta f(x_i)) < \Omega_i, &\quad i \in I \end{aligned}\right\} \tag{15}$$

根据上述条件,由于 $f(x_c) \geqslant \Omega_c$ 的新增样本在没有进入该步骤的时候已经结束了,故此时新增样本满足:

$$y_c(f(x_c)+\Delta f(x_c)) < \Omega_c \tag{16}$$

在 α_i 从0增加到 C_i 的过程中,当不等式(16)变为等式的时候,即可将该新增样本添加到 $\boldsymbol{M}$ 集合中.同理,在 α_i 增加到 C_i 的过程中不等式(16)始终满足,则可以将其添加到 I 集合中.

将式(11)带入线性系统(14)中可以得到关于 η 的增量表示如下:

$$\begin{bmatrix}\Delta b\\ \Delta\alpha_M\end{bmatrix} = \eta\varphi \tag{17}$$

式中,

$$\varphi = -\boldsymbol{M}^{-1}\begin{bmatrix}y_c\\ \boldsymbol{Q}_{M,c}\end{bmatrix}(C_c - \alpha_c) \tag{18}$$

文中假设核矩阵 $\boldsymbol{Q}$ 是正定的,于是矩阵 $\boldsymbol{M}$ 可逆.若 $\boldsymbol{Q}$ 半正定,则矩阵 $\boldsymbol{M}$ 可能是不可逆的.此时,我们可以通过在核矩阵的对角线上添加一个较小的正常量来避免.为了决定步长 η,我们需要检查不等式(15)与(16),之后寻找到能够恰好使得一个样本碰到集合 O、M 与 I 的边界.通过引入按元素乘的Hadamard乘 $*$ 运算[28],我们可以得到:

$$y * \Delta f = \eta\psi \tag{19}$$

式中,

$$\psi = \begin{bmatrix}y & \boldsymbol{Q}_{S,M}\end{bmatrix}\varphi + Q_{S,c}(C_c - \alpha_c).$$

由于式(17)和(19)是关于 η 的线性系统,我们可以计算出步长 η.

得到步长 η 之后,即可以通过式(17)更新 α_M 和 b,通过式(11)更新 α_c.若有样本碰到集合 O、M 与 I 的边界,还需要更新这3个集合.之后,重新计算 φ 和 ψ,并确定下一轮迭代步长.

本文算法的细节描述如下.

算法 2.1 增量式的代价敏感支持向量机

输入:优化的 $\alpha=[\alpha_1,\cdots,\alpha_n]^T$,b

下标集合 O、M、I

新增样本(x_c,y_c)

1:令 $\alpha_c=0$,$\eta=0$

2:while$\eta=1$ do

3:通过线性系统(18)计算 φ

4:通过(20)计算 ψ

5:通过(15)和(16)计算 η

6:更新$[b\quad \alpha_M^T]\leftarrow[b\quad \alpha_M^T]+\eta\varphi$

7:更新 $\alpha_c\leftarrow\alpha_c+\eta(C_c-\alpha_c)$

8:更新下标集合 O、M、I

9:end while

输出:α,b

接下来将简要说明增量式的代价敏感支持向量机的有限收敛性.受限于篇幅,具体证明步骤可以参照Gu等对于在线 ν 支持向量机的证明过程[29].

首先可以证明在增量式算法的迭代过程中,目标问题是严格单调递减的.在增量式算法的每一步迭代过程中,目标函数的改变都对应着 α 的改变,同时对应着一个唯一的 $\boldsymbol{M}$ 集合.由于 $\boldsymbol{M}$ 集合的组合数是有限的,所以目标函数的改变序列一定是有限的.从而 α 的改变序列是有限的,并且可以证明其序列是严格递增序列.因此增量式算法具有有限性.

关于算法的收敛性可以采用反证法来证明.若 α 序列不收敛于KKT条件,由于算法的有限性,α 序列停止时将不满足KKT条件,这与算法的停止条件相矛盾,因此 α 在算法的迭代过程中将收敛于KKT条件.进一步可以证明目标函数将在有限的步骤内收敛;因此本文提出的增量式算法是有限收敛的.

3 实验及分析

本节使用Matlab实现了本文提出的增量式代

价敏感的支持向量机算法(ICSSVM)以及传统批处理代价敏感支持向量机(BCSSVM),并在多个基准数据集上进行了比较.此外,本节还给出了 ICSSVM 算法收敛的迭代次数以及边缘向量的数量的实验分析.实验中,ICSSVM 和 BCSSVM 算法均使用 Matlab 实现,其中 BCSSVM 算法根据 LIBSVM 论文[30]中采用的 SMO 算法[22]进行实现.LIBSVM 是被研究者广泛使用的支持向量机的实现版本,由于其中暂未实现本文中研究的 CS-SVM 算法,本文对其进行了实现,并在实验中与增量式算法进行了对比.

3.1　实验方案

为了消除不同属性值之间取值范围差异对算法的影响,本文按照如下公式对数据做了归一化.该过程使得所有的属性值均为 0 到 1 之间的小数.

$$\text{value}=\frac{\text{value}-\min}{\max-\min} \tag{20}$$

式中,max 和 min 分别表示某属性中的最大值和最小值,value 表示当前待归一化的属性值.在归一化过程中,有一种情况值得被注意.当某属性的所有值完全相同时,$\max-\min=0$,通过公式(20)计算将会产生除 0 错误.在本文的预处理过程中,此情况下,若该属性的值为 0,则归一化之后仍然为 0;否则为 1.

实验在 8 个基准数据集上对比了 ICSSVM 和 BCSSVM 算法,这些基准数据集分别来自 UCI 机器学习知识库、LIBSVM 网站以及 KEEL.使用到的基准数据集列在表 1 中给出.其中"比例"表示两个不同类别之间的比例.

表 1　基准样本集说明

Tab.1　Benchmark datasets

数据集	样本数	属性数	比例
abalone19	4 174	8	130
ijcnn1	49 990	22	9.3
page-blocks0	5 472	10	8.8
segment0	2 308	19	6
shuttle-c0-vs-c4	1 829	9	13.9
svmguide3	1 243	22	3.2
winequality-red-4	1 599	11	29
yeast4	1 484	8	28

所有的实验均在 Mac OS X 10.10.2 系统上的 Matlab2015a 平台上运行,硬件配置为 1.4GHz Intel Core i5 CPU,8G RAM.实验中核函数包括三种,分别为:线性核函数为 $K(x_i,x_j)=x_i\cdot x_j$;多项式核函数为 $K(x_i,x_j)=(2x_i\cdot x_j+1)^d$,其中指数 $d=2$;高斯核函数为 $K(x_i,x_j)=\exp(-\|x_i-x_j\|^2/2\sigma^2)$,其中参数 $\sigma=1.414$.另外,实验中固定参数 $C=10$,$C_+=1$,$C_-=2$.所有实验重复 10 次,取其均值作为最终结果.

3.2　运行效率分析

首先,实验对比了 ICSSVM 和 BCSSVM 的运行效率.实验包括线性核、多项式核、高斯核等 3 种核函数.为了对比增量式和全量式算法的运行效率,实验分别在包含不同样本的数据集上进行实验.实验假设在每个尺寸的数据集上均有训练过的模型,并在此基础上增加一个样本,比较增量式算法和全量式算法的运行效率.实验结果如图 1 所示.

每个数据集有 6 条曲线,分别表示 ICSSVM 和 BCSSVM 两种算法在 3 个核上的运行时间.其中实线表示增量式算法,虚线表示全量算法.圆形表示高斯核,菱形表示多项式核,三角形表示线性核.

从 8 个基准数据集上的训练时间效率图中可以看出,增量式算法显著地比全量算法快.并且随着样本数量 增多,增量式算法的训练时间基本不变,而全量算法的训练时间有明显的稳定增长.原因是显而易见的.本文提出的增量式算法每次仅更新一个样本,即可以直接更新现有的模型而不需要从头开始训练,因而对于增量式算法来说每次添加样本后的问题规模是基本不变的,因而问题规模小,训练速度快;全量算法则每次都需要从头开始训练,每次添加样本后问题规模会随之变大,随着问题规模的增大,大量的重复运算必然会使得算法运算时间随之增长.

此外,由图可以发现不同的核会有不同的效率.对于全量算法,线性核和多项式核在大多数据集中比高斯核耗费的时间更多.因为与线性核和多项式核比起来,高斯核有更快的收敛速度,尽管高斯核的核函数的运算复杂度更高,当迭代次数显著少于其他核的时候,依然可以得到较高的运行效率.而对于增量式算法,该规律不明显.由于对于增量式算法来说,每次添加一个样本,相对于全量算法来说,训练的迭代次数显著减少,因而迭代次数对最终时间的影响不甚明显.

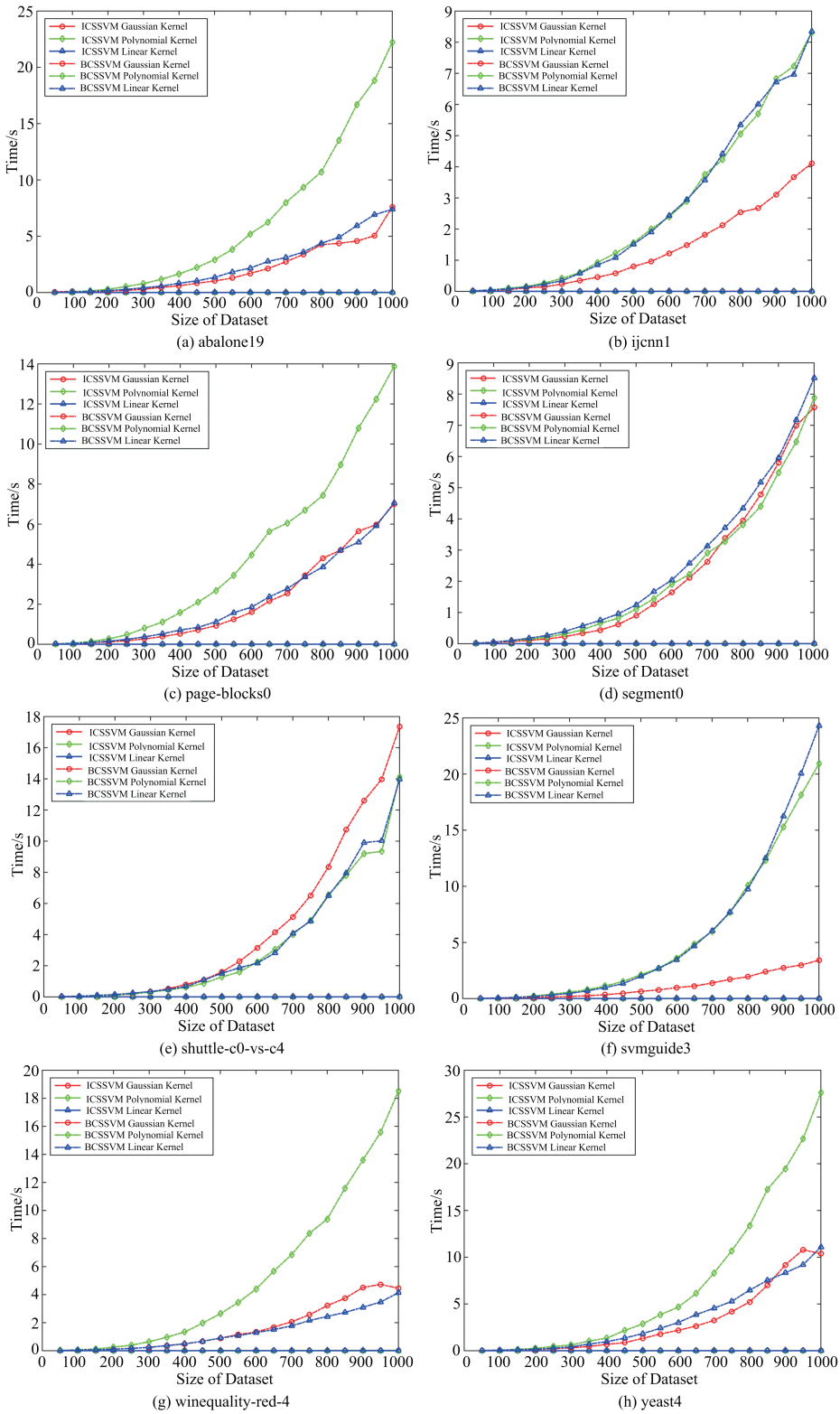

图 1 ICSSVM 和 BCSSVM 的运行时间

Fig. 1 Runtime of ICSSVM and BCSSVM

3.3 增量式算法迭代次数分析

实验还统计了文中提出的增量式算法的迭代次数. 如图 2 所示. 其中圆形表示高斯核,菱形表示多项式核,三角形表示线性核.

迭代次数与训练样本的特点以及新增样本相关. 通过图 2 可以看出,在每次增加一个样本的实验

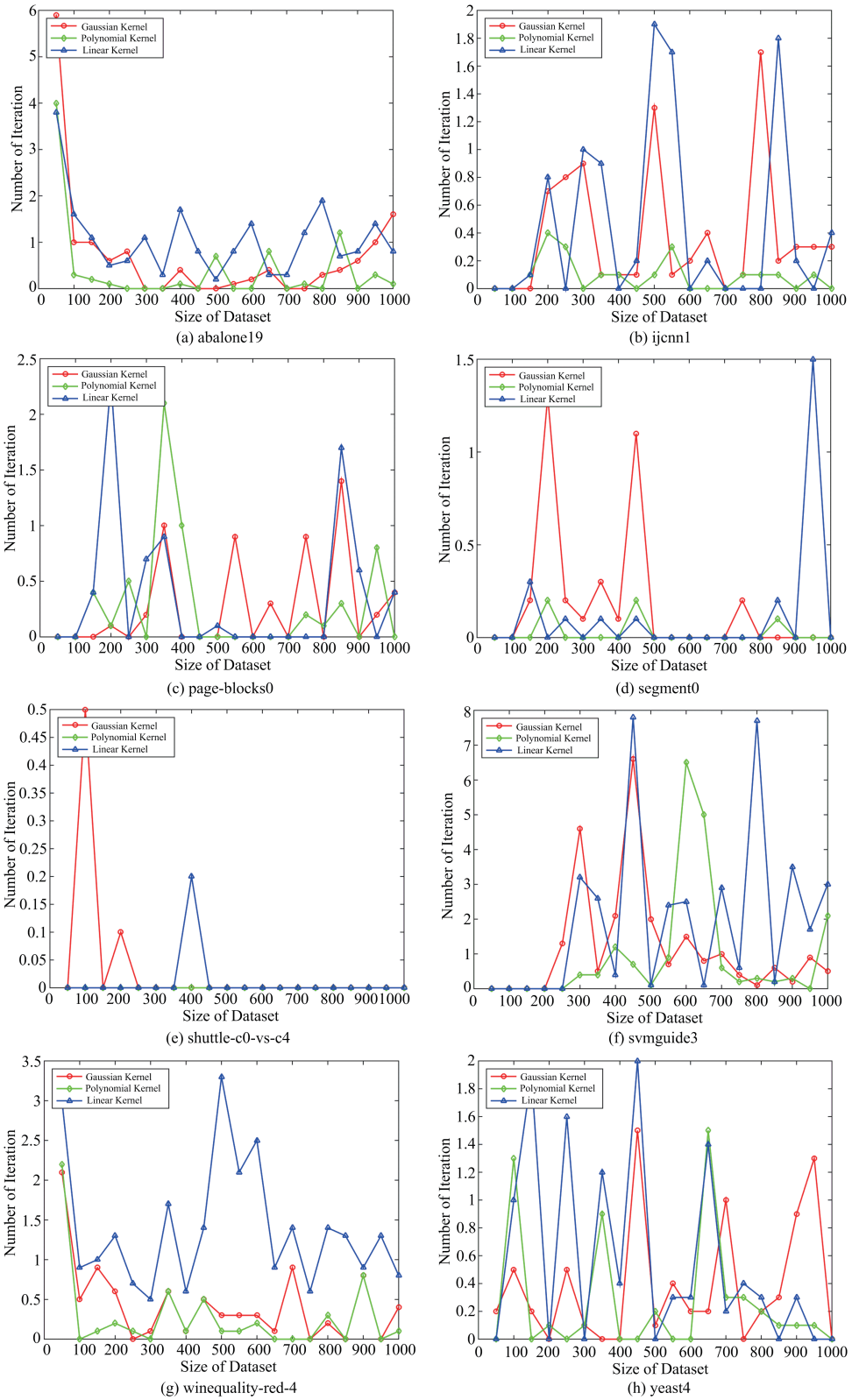

图 2　ICSSVM 在不同核函数上的迭代次数

Fig. 2　Iteration of ICSSVM with different kernel

方案下，在大多数数据集上多数结果落在 0 到 2 次之间. 即对于每个样本点，平均迭代 2 次以内即可以收敛. 在少数样本集上有少数样本点的迭代次数会超出 2 次，不过最高不超过 8 次. 同时，可以注意到，

在很多数据集上加入某些样本点时，迭代次数小于1.这是因为本文在实验中取10次实验均值作为最终结果，而在实验过程中出现了不需要迭代的情况.迭代次数为0表示样本在初始化过程中即满足KKT条件，未进入迭代过程.由于只需要较少的迭代次数就可以收敛，迭代次数曲线均在小范围内波动，未见有明显的规律.

由于增量式算法有很少的迭代次数和很快的收敛速率，具有非常高的效率.随着数据集增大，算法运算时间增长十分缓慢，不同核函数之间的差异表现不明显，即算法效率非常稳定.

4 结论

本文提出了一种代价敏感支持向量机(CS-SVM)的增量式算法.该算法可以在新增样本时直接更新现有已训练模型，而不需要重新从头训练所有已训练样本.文中首先给出了原始CS-SVM的修改形式，之后对增量式算法进行推导并给出了详细算法.

实验研究表明，本文提出的增量算法的运行效率显著高于全量算法.文中进一步在线性核、多项式核、高斯核等三种核函数上比较了增量算法的迭代次数与边界向量的数量.通过对比发现，随着数据集的不断增大，增量式算法迭代次数并不会随之增加，而是在较小的迭代次数附近小范围波动，进而导致总体运行时间也相对稳定，不会显著增加.

理论上来说，文中用于增量式算法的技术同样可以用于减量式算法，进而可以在不需要重复计算所有历史数据的前提下从样本集中移除样本.

参考文献(References)

[1] SHENG V S, LING C X. Thresholding for making classifiers cost-sensitive[C]// Proceedings of the 21st National Conference on Artificial Intelligence. Boston: AAAI Press, 2006: 476.

[2] PARK Y J, CHUN S H, KIM B C. Cost-sensitive case-based reasoning using a genetic algorithm: Application to medical diagnosis [J]. Artificial Intelligence in Medicine, 2011, 51(2): 133-145.

[3] ELKAN C. The foundations of cost-sensitive learning [C]// Proceedings of the 17th International Joint Conference on Artificial Intelligence. San Francisco: Morgan Kaufmann Publishers, 2001: 973-978.

[4] DOMINGOS P. Metacost: A general method for making classifiers cost-sensitive[C]// Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Lisbon: ACM Press, 1999: 155-164.

[5] CHAWLA N V, BOWYER K W, HALL L O, et al. SMOTE: Synthetic minority over-sampling technique [J]. Journal of Artificial Intelligence Research, 2011, 16(1): 321-357.

[6] DRUMMOND C, HOLTE R C. C4. 5, class imbalance, and cost sensitivity: Why under-sampling beats over-sampling[C]// Proceedings of the LCML Workshop on Learning from Imbalanced Datasets II. 2003: 1-8.

[7] MARGINEANTUD D, DIETTERICH T G. Bootstrap methods for the cost-sensitive evaluation of classifiers [R]. Corvallis, OR: Oregon State University, 2000.

[8] TING K M. A comparative study of cost-sensitive boosting algorithms [C]// Proceedings of the 17th International Conference on Machine Learning. San Francisco: Morgan Kaufmann, 2000: 983-990.

[9] TING K M. An instance-weighting method to induce cost-sensitive trees [J]. IEEE Transactions on Knowledge and Data Engineering, 2002, 14(3): 659-665.

[10] ZHOU Z H, LIU X Y. Training cost-sensitive neural networks with methods addressing the class imbalance problem[J]. IEEE Transactions on Knowledge and Data Engineering, 2006, 18(1): 63-77.

[11] LIN Y, LEE Y, WAHBA G. Support vector machines for classification in nonstandard situations [J]. Machine Learning, 2002, 46(1): 191-202.

[12] GEIBEL P, BREFELD U, WYSOTZKI F. Perceptron and SVM learning with generalized cost models[J]. Intelligent Data Analysis, 2004, 8(5): 439-455.

[13] SCHÖLKOPF B, SMOLA A J. Learning with Kernels: support Vector Machines, Regularization, Optimization, and Beyond [M]. Cambridge: MIT Press, 2001.

[14] MASNADI-SHIRAZI H, VASCONCELOS N. Risk minimization, probability elicitation, and cost-sensitive SVMs[C]// Proceedings of the 27th International Conference on Machine Learning. Haifa, Israel: IEEE Press, 2010: 759-766.

[15] MASNADI-SHIRAZI H, VASCONCELOS N, IRANMEHR A. Cost-Sensitive Support Vector Machines [J]. arXiv preprint, 2012: arXiv: 1212.0975.

[16] CORTES C, VAPNIK V. Support-vector networks [J]. Machine learning, 1995, 20(3): 273-297.

[17] BACH F R, HECKERMAN D, HORVITZ E. Considering cost asymmetry in learning classifiers[J].

Journal of Machine Learning Research, 2006, 7(4): 1713-1741.

[18] DAVENPORT M, BARANIUK R G, SCOTT C D. Controlling false alarms with support vector machines [C]// Proceedings of the International Conference on Acoustics, Speech and Signal Processing. IEEE Press, 2006: 589-592.

[19] CHANG C C, LIN C J. LIBSVM: A library for support vector machines [J]. ACM Transactions on Intelligent Systems and Technology, 2011, 2(3): 389-396.

[20] WANG J, ZHAO P, HOI S C. Cost-sensitive online classification [J]. IEEE Transactions on Knowledge and Data Engineering, 2014, 26(10): 2425-2438.

[21] ZHENG J, SHEN F, FAN H, et al. An online incremental learning support vector machine for large-scale data [J]. Neural Computing and Applications, 2013, 22(5): 1023-1035.

[22] PLATT J C. Fast training of support vector machines using sequential minimal optimization [C]// Advances in Kernel Methods. Cambridge, USA: MIT Press, 1999: 185-208.

[23] POGGIO G C T. Incremental and decremental support vector machine learning [C]// Proceedings of the 2000 Conference on Advances in Neural Information Processing Systems, MIT Press, 2001: 409.

[24] GU B, WANG J D, CHEN H. On-line off-line ranking support vector machine and analysis [C]// IEEE International Joint Conference on Neural Networks. 2008: 1364-1369.

[25] GU B, WANG J D, YU Y C, et al. Accurate on-line ν-support vector learning [J]. Neural Networks, 2012, 27: 51-59.

[26] GU B, SHENG V S, WANG Z, et al. Incremental learning for ν-support vector regression [J]. Neural Networks, 2015, 67(C): 140-150.

[27] GU B, SHENG V S, TAY K Y, et al. Incremental support vector learning for ordinal regression [J]. Neural Networks and Learning Systems, IEEE Transactions on, 2015, 26(7): 1403-1416.

[28] SCHOTT J R. Matrix analysis for statistics [J]. Exvi Print, 2005, 30(5): xvi,456.

[29] GU B, SHENG V S. Feasibility and finite convergence analysis for accurate on-line-support vector machine [J]. IEEE Transactions on Neural Networks and Learning Systems, 2013, 24(8): 1304-1315.

[30] CHANG C C, LIN C J. LIBSVM: A library for support vector machines [J]. ACM Transactions on Intelligent Systems and Technology (TIST), 2011, 2(3): 389-396.