基于 SVM 的二次下降有效集算法

丁晓剑,赵银亮,李远成
(西安交通大学电子与信息工程学院,陕西西安 710049)

摘 要: 针对现有的有效集方法应用到支持向量机(support vector machine,SVM)优化问题时收敛速度较慢的问题,提出了一种基于二次下降法和推测赋值法的有效集算法.该算法在每次迭代过程中利用映射因子将迭代向量值限制在优化问题的不等式约束中,并通过调整步长使目标优化问题的函数值较传统的有效集算法进一步下降.由于函数值在每次迭代后保证了严格快速下降,所以提出的算法能够快速收敛到全局最优解.实验结果表明该方法的迭代次数和迭代时间有明显减少.

关键词: 支持向量机; 有效集; 二次下降法; 迭代
中图分类号: TP319 **文献标识码:** A **文章编号:** 0372-2112 (2011) 08-1766-05

Secondary Descent Active Set Algorithm Based on SVM

DING Xiao-jian, ZHAO Yin-liang, LI Yuan-cheng
(*School of Electronics and Information Engineering, Xi'an Jiaotong University, Xi'an, Shaanxi* 710049, *China*)

Abstract: To solve the slow convergence rate of the existing active set methods applied into optimization formulation of support vector machine, an active set algorithm based on the secondary descent method and the speculative assignment method is proposed. At each iteration of the algorithm, a projection operator is used to restrict the iterative vector onto the inequality constraints of optimization formulation, and then an adjustable step size is used to ensure the functional value of optimization formulation make further descent compared to the traditional active set method. As functional value ensure rapid and strictly descent at the end of each iteration, the global optimum solution can be obtained with rapid convergence rate. Experimental results show that iterations time and training time of the proposed method have been decreased obviously.

Key words: support vector machine; active set; secondary descent method; iteration

1 引言

目前求解支持向量机[1](support vector machine, SVM)的二次规划问题的方法可以分为分解法[2~7]和有效集法[8,9].文献[2]证明了 SVM 优化问题可以分解为一系列子问题来求解.当前应用广泛的 SVM 软件包 SVM[3]和 LIBSVM[4]都是将原始优化问题划分为一系列子优化问题来求解.Platt[5]提出的 SMO(sequential minimal optimization)方法是常用的求解 SVM 优化问题的方法之一,SMO 使用了一种极端分解策略,每次迭代只需求解含有两个工作变量的子优化问题.Fang 等人[6]针对简化 SVM 提出了改进的 SMO 算法,具有较少的支持向量数和较高的支持向量识别速度.Yang 等人[7]提出的自适应迭代方法可以有效确定最小二乘 SVM 回归问题中支持向量的数目,在回归精度相近的情况极大地提高了算法学习的速度.有效集法通过迭代求解子优化问题求得原始优化问题的最优解.与分解法不同的是,有效集法是一种增量迭代方法,迭代的过程就是不断识别正确的有效集和工作集的过程,当工作集和有效集都被正确识别出时,优化问题得到最优解.文献[8]提出的有效集方法在基准数据集的实验中表现出比 SMO 方法更低的计算代价.

虽然有效集方法[8,9]在很多基准数据集和不同性能指标测试中表现出了高效性,但是这些有效集方法的识别策略仍然存在两个问题:(1)迭代次数过多.在有效集识别过程中,由于采用单向下降策略,每次迭代的有效集识别阶段只能识别一个有效集变量.对于 N 个训练样本,一般会产生 $O(3N/2)$次迭代,迭代次数过多明显会影响样本训练的时间;(2)误迭代率较高.有效集算法一般先正确识别完所有有效集变量,才开始工作集变量的识别.如果上次迭代结束时所识别的有效集变量在本次迭代后仍然判定到边界约束上,那么此次迭代称为

收稿日期:2010-10-14;修回日期:2011-03-02
基金项目:国家 863 高技术研究发展计划(No.2008AA01Z136)

误迭代,显而易见误迭代会增加算法的计算代价.

不同于单向下降法(single directional descent method, SDDM)[8]每次迭代只能识别一个有效集变量,本文提出的二次下降法(secondary descent method, SDM)能够在每次迭代时识别多个有效集变量,使目标优化问题函数值快速下降,可以减少算法的迭代次数,进而缩短训练样本的时间.另外采用推测赋值方法能够减少误迭代情况的发生,从而进一步降低训练过程中的计算代价.

2 SVM 的有效集算法回顾

2.1 SVM 优化问题

本文研究 SVM 的两类分类问题,给定一个含有 N 个样本的训练样本集$\{\boldsymbol{x}_i, y_i\}_{i=1}^N$,其中输入为 d 维向量 $\boldsymbol{x}_i \in \boldsymbol{R}^d$,输出为 $y_i \in \boldsymbol{R}$.通常训练样本集在输入空间中是线性不可分的,需要引入映射函数 $\varphi(\boldsymbol{x})$将 $\boldsymbol{x}_i$ 映射到高维空间 $\varphi(\boldsymbol{x}_i)$中.根据 KKT(Karush-Kuhn-Tucker)定理[10]和拉格朗日乘子理论,训练 SVM 等价于求解如下对偶优化问题

$$\begin{aligned}&\text{Minimize: } f(\boldsymbol{\alpha}) = \frac{1}{2}\boldsymbol{\alpha}^{\mathrm{T}} Q\boldsymbol{\alpha} - \boldsymbol{e}^{\mathrm{T}}\boldsymbol{\alpha}\\ &\text{Subject to: } \boldsymbol{\alpha}\boldsymbol{y} = 0,\\ &\qquad 0 \leqslant \alpha_i \leqslant C, i = 1, \cdots, N \end{aligned} \tag{1}$$

其中 $Q = K\boldsymbol{y}\boldsymbol{y}^{\mathrm{T}} \in \boldsymbol{R}^{N\times N}$为半正定矩阵,$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i)\cdot\phi(\boldsymbol{x}_j)$为满足 Mercer 定理的核函数,$\boldsymbol{y} = [y_1, \cdots, y_N]^{\mathrm{T}} \in \boldsymbol{R}^N$,$\boldsymbol{\alpha} = [\alpha_1, \cdots, \alpha_N]^{\mathrm{T}} \in \boldsymbol{R}^N$ 为拉格朗日乘子,$\boldsymbol{e} = [1, \cdots, 1]^{\mathrm{T}} \in \boldsymbol{R}^N$.

定义集合 $L = \{i \mid \boldsymbol{\alpha}_i = 0\}$, $U = \{i \mid \boldsymbol{\alpha}_i = C\}$和 $S = \{i \mid 0 < \boldsymbol{\alpha}_i < C\}$,有效集算法在每次迭代中的主要计算代价是求解式(1)的子优化问题[8]

$$\begin{aligned}&\text{Minimize: } q(\boldsymbol{\alpha}_S) = \frac{1}{2}\boldsymbol{\alpha}_S^{\mathrm{T}} Q_{SS}\boldsymbol{\alpha}_S + \boldsymbol{\alpha}_U^{\mathrm{T}} Q_{US}\boldsymbol{\alpha}_S - \boldsymbol{e}^{\mathrm{T}}\boldsymbol{\alpha}_S\\ &\text{Subject to: } \boldsymbol{y}_S^{\mathrm{T}}\boldsymbol{\alpha}_S + \boldsymbol{y}_S^{\mathrm{T}}\boldsymbol{\alpha}_U = 0 \end{aligned} \tag{2}$$

其中 $\boldsymbol{\alpha}_S$ 和 $\boldsymbol{\alpha}_U$ 是由下标分别属于集合 S 和 U 的 $\boldsymbol{\alpha}$ 的分量构成的向量,其它向量同理可得.

有效集算法通过每次迭代求解式(2)得到式(1)的最优解,算法[8]利用 2 层循环来求解目标优化问题:外循环判断 $\boldsymbol{\alpha}$ 中边界约束元素是否都满足 KKT 条件,如果都满足,算法终止.如果不满足,则选取违反 KKT 条件的变量进入内循环求解.内循环目标是求解式(2),使得最优解 $\boldsymbol{\alpha}_S$ 满足上下界约束,并且利用 SDDM 使函数值下降.

2.2 SDDM

设 $\boldsymbol{\alpha}_S^k$ 为第 k 次迭代的解向量,则第 $k+1$ 次迭代过程可表示为:$\boldsymbol{\alpha}_S^{k+1} = \boldsymbol{\alpha}_S^k + \mu_k d^k$.其中 d^k 为下降方向,μ_k 为搜索步长.SDDM 的核心思想是通过调整步长 μ_k 使迭代后的解向量 $\boldsymbol{\alpha}_S^{k+1}$ 满足框式约束$[0, C]$.具体的做法是将 $\boldsymbol{\alpha}_S^k$ 的每个分量都会乘以相同的步长以保证函数 $q(\boldsymbol{\alpha}_S)$值的严格下降,使得 $q(\boldsymbol{\alpha}_S^{k+1}) < q(\boldsymbol{\alpha}_S^k)$.该方法虽然保证函数值严格下降,但是每次迭代只能识别一个有效集变量.如果正确识别所有的有效集变量,将会导致较多的迭代次数.

3 提出的方法

3.1 SDM

本节对有效集算法内循环中 SDDM 进行了改进,提出了 SDM 算法,能够使函数 $q(\boldsymbol{\alpha}_S)$值 SDDM 进一步下降.

SDM 与 SDDM 的主要区别是 $\boldsymbol{\alpha}_S^k$ 每个分量的调整步长可以不同.先设置一个初始步长,利用 SDDM 得到 $\boldsymbol{\alpha}_S^{k+1}$,然后通过映射的方法使 $\boldsymbol{\alpha}_S^{k+1}$ 的所有分量都满足式(1)的不等式约束.定义映射因子 Π

$$\Pi(\boldsymbol{\alpha})_i = \begin{cases} 0, & \text{if } \alpha_i \leqslant 0\\ \alpha_i, & \text{if } 0 < \alpha_i < C\\ C, & \text{if } \alpha_i \geqslant C \end{cases} \tag{3}$$

定义 μ_{k1}为第 k 次迭代中满足 SDDM 的最优步长,μ_{k2}为第 k 次迭代中满足 SDM 的最优步长.令 $\boldsymbol{\alpha}_S^{k+1} := \Pi(\boldsymbol{\alpha}_S^k + \mu_{k2} d^k)$为第 k 次迭代求解二次下降法得到的解向量.则第 k 次迭代过程的 SDM 可描述如下:

SDM 算法

输入:变量 $\boldsymbol{\alpha}_S^k, d^k$ 和实数 $M > 1$

输出:变量 $\mu_{k2}, \boldsymbol{\alpha}_S^{k+1}$

步骤 1　计算 $\mu_{\max} \leftarrow \max\{\mu \geqslant 0 \mid \boldsymbol{\alpha}_S^k + \mu d^k \in [0, C]\}$,设 $\mu \leftarrow \min\{\mu_{\max}, 1\}$.

　　if $\mu_{\max} > 1$ then $\mu_{k2} \leftarrow 1, \boldsymbol{\alpha}_S^{k+1} = \boldsymbol{\alpha}_S^k + d^k$,终止本次迭代

　　else 转向步骤 2, end if

步骤 2　设 $\mu_{k1} \leftarrow \mu$,计算式(2)的函数值 $F \leftarrow q(\boldsymbol{\alpha}_S^k + \mu_{k1} d^k)$.

步骤 3　设 $\mu_{\text{temp}} \leftarrow M\mu$,计算 $\boldsymbol{\alpha}_S^{\text{temp}} = \Pi(\boldsymbol{\alpha}_S^k + \mu_{\text{temp}} d_k)$和式(2)的函数值 $q(\boldsymbol{\alpha}_S^{\text{temp}})$.

步骤 4　if $(\mu_{\text{temp}} \geqslant 1$ or $q(\boldsymbol{\alpha}_S^{\text{temp}}) \geqslant F)$ then

　　if $\mu_{\text{temp}} = 1$ then $\mu_{k2} \leftarrow \mu, \boldsymbol{\alpha}_S^{k+1} = \boldsymbol{\alpha}_S^k + \mu d^k$,终止本次迭代

　　else $\mu_{k2} \leftarrow \mu_{\text{temp}}/M, \boldsymbol{\alpha}_S^{k+1} = \Pi(\boldsymbol{\alpha}_S^k + \mu_{k2} d^k)$,终止本次迭代

　　end if

　　else $F \leftarrow q(\boldsymbol{\alpha}_S^{\text{temp}}), \mu_{\text{temp}}/M\mu, \mu \leftarrow \mu_{\text{temp}}$,转向步骤 3, end if

SDM 算法主要是寻找最优搜索步长 μ_{k2}的过程.步骤 1 判断由 d^k 计算得到的向量是否满足不等式约束:如果满足不等式约束,通过计算 $\boldsymbol{\alpha}_S^{k+1} = \boldsymbol{\alpha}_S^k + d^k$ 得到解向量;如果不满足,则转向步骤 2 继续寻找最优步长.根据步骤 1 阶段 μ 的计算公式和判断条件可知,步骤 2 初始阶段得到的 μ 即 SSDM 的最优步长 μ_{k1}.步骤 3 设置迭代步长 $\mu_{\text{temp}} \in (\mu, 1]$,步骤 4 通过临时步长 μ_{temp}判断函数值 $q(\boldsymbol{\alpha}_S^{\text{temp}})$能否较 $q(\boldsymbol{\alpha}_S^k + \mu_{k1} d^k)$进一步下降,如果

可以继续下降,重复选取 μ_{temp}直至函数值不再下降为止.

当找到满足条件的最优步长 μ_{k2}后,计算 $\boldsymbol{\alpha}_S^{k+1}=\Pi(\boldsymbol{\alpha}_S^k+\mu_{k2}d^k)$时可能会产生一个问题.经过映射得到的 $\boldsymbol{\alpha}_S^{k+1}$中所有分量的值都可能等于0或者等于 C,这样会导致工作集为空,从而影响下一次迭代.对于这种情况,先从指标集 I 中约减掉有效集指标,然后从 I 集合中随机选取一个指标加入工作集中,并对相应的变量赋予0到 C 间的任意值,以进行下一次迭代.

3.2 推测赋值法

在步骤1中,如果不等式约束满足,可以得到解向量 $\boldsymbol{\alpha}_S^{k+1}=\boldsymbol{\alpha}_S^k+d^k$,然后判断KKT条件是否满足:如果都满足条件,就得到式(1)的最优解;如果不满足,选取违反KKT条件最大的指标,将之加入到工作集中继续迭代.

当选定一个指标后,一般的有效集方法[8]是将该指标直接加入到工作集中.这种做法就导致一个问题:该指标从有效集中取出再放入到工作集中,但是它对应的变量在边界上(等于0或者等于 C),而工作集对应的变量的值应该在0和 C 之间.这种做法会产生矛盾,容易使下次迭代成为误迭代.由于工作集变量的值可能是在0到 C 之间的任意值,直接的方法是将该值赋为 $C/2$,能够减少工作集变量迭代到最优值花费的计算代价.

3.3 算法收敛性分析

定理1 设第 k 次迭代的解向量 $\boldsymbol{\alpha}_S^k$,下降方向 d^k,SDDM的最优搜索步长 μ_{k1},SDM的最优搜索步长 μ_{k2},则 $q(\Pi(\boldsymbol{\alpha}_S^k+\mu_{k2}d^k))\leqslant q(\boldsymbol{\alpha}_S^k+\mu_{k1}d^k)$成立.

证明 SDM算法先找出SDDM的最优步长 $\mu_{k1}\in[0,1]$,并得到函数值 $q(\boldsymbol{\alpha}_S^k+\mu_{k1}d^k)$.步骤4寻找步长 μ_{temp}使函数值较 $q(\boldsymbol{\alpha}_S^k+\mu_{k1}d^k)$进一步下降,可以分为两种情况:

(1)如果步长 μ_{temp}越界($\mu_{\text{temp}}>1$)或者 $q(\boldsymbol{\alpha}_S^{\text{temp}})\geqslant q(\boldsymbol{\alpha}_S^k+\mu_{k1}d^k)$,进行回溯操作,令 $\mu_{k2}=\mu$,由于先前已经令 $\mu_{k1}=\mu$,此时 $\mu_{k1}=\mu_{k2}$,由SDDM算法[8]可知,此时向量 $\boldsymbol{\alpha}_S^k+\mu_{k1}d^k$ 仅有一个分量在边界上,其余分量都在界内.根据映射函数 Π 定义可知,$\Pi(\boldsymbol{\alpha}_S^k+\mu_{k1}d^k)=\boldsymbol{\alpha}_S^k+\mu_{k1}d^k$,即 $\Pi(\boldsymbol{\alpha}_S^k+\mu_{k2}d^k)=\boldsymbol{\alpha}_S^k+\mu_{k1}d^k$,所以有 $q(\Pi(\boldsymbol{\alpha}_S^k+\mu_{k2}d^k))=q(\boldsymbol{\alpha}_S^k+\mu_{k1}d^k)$.

(2)如果步长 μ_{temp}不越界($\mu_{\text{temp}}\in[0,1]$)并且 $q(\boldsymbol{\alpha}_S^{\text{temp}})<q(\boldsymbol{\alpha}_S^k+\mu_{k1}d^k)$,循环执行步骤3和4直至 $q(\boldsymbol{\alpha}_S^{\text{temp}})$的值不再降低为止,此时有 $\mu_{k2}=\mu_{\text{temp}}$,又 $\boldsymbol{\alpha}_S^{\text{temp}}=\Pi(\boldsymbol{\alpha}_S^k+\mu_{\text{temp}}d_k)$,所以有 $q(\Pi(\boldsymbol{\alpha}_S^k+\mu_{k2}d^k))<q(\boldsymbol{\alpha}_S^k+\mu_{k1}d^k)$.

根据上面两种情况,必然有 $q(\Pi(\boldsymbol{\alpha}_S^k+\mu_{k2}d^k))\leqslant q(\boldsymbol{\alpha}_S^k+\mu_{k1}d^k)$成立,定理得证.

从SDM的算法流程可以看出,SDM是基于SDDM来判断函数值是否能进一步下降.令 $q(\boldsymbol{\alpha}_S^k+\mu_{k1}d^k)$为SDDM得到的函数值,$q(\Pi(\boldsymbol{\alpha}_S^k+\mu_{k2}d^k))$为SDM得到的函数值,根据定理1有 $q(\Pi(\boldsymbol{\alpha}_S^k+\mu_{k2}d^k))\leqslant q(\boldsymbol{\alpha}_S^k+\mu_{k1}d^k)$.文献[8]证明了SDDM的有效集方法是收敛的,证明的核心思想是利用了 $q(\boldsymbol{\alpha}_S^k+\mu_{k1}d^k)<q(\boldsymbol{\alpha}_S^k)$的性质,即保证每次迭代函数值的严格下降.由于 $q(\Pi(\boldsymbol{\alpha}_S^k+\mu_{k2}d^k))\leqslant q(\boldsymbol{\alpha}_S^k+\mu_{k1}d^k)$,所以 $q(\Pi(\boldsymbol{\alpha}_S^k+\mu_{k2}d^k))<q(\boldsymbol{\alpha}_S^k)$必然成立,利用SDDM的证明方法,SDM也必然是收敛的.由于SDM和SDDM得到的最优函数值相同,而SDM在每次迭代能得到更小的函数值(定理1),所以它的收敛速度比SDDM要快.

4 实验与性能分析

为了全面地分析SDM算法的性能,将SDM算法与经典的SSDM算法[8]和改进的有效集算法(ESSDM)[9]进行对比.ESSDM算法利用Cholesky快速分解的方法和Rank-one更新 Q_{SS}矩阵法求解式(2),有效地提高了算法的效率.

本文仅进行二类分类问题的比较,实验数据来自UCI标准数据集[11]和Statlog数据集[12],数据的输入归一化到[0,1]范围内,数据的具体描述见表1.所有算法都是基于Pentium 4,2.53GHZ,MATLAB 2007环境下比较.

表1 标准数据描述

数据集	属性数	训练样本数	测试样本数
Breast-cancer	10	300	383
liver-disorders	6	200	145
Diabetes	8	576	192
ionosphere	34	300	251
Pimadata	8	400	368
Pwlinear	10	100	100
Sonar	60	100	158
Monk's Problem 1	6	124	432
Monk's Problem 2	6	169	432
Splice	60	1000	2175
heart	13	70	200
Australian	14	300	390

4.1 参数设定

SVM优化问题的核函数选择高斯核函数:$K(\boldsymbol{u},\boldsymbol{v})=\exp(-\|\boldsymbol{u}-\boldsymbol{v}\|^2/2\gamma^2)$.其中核参数 γ 和代价参数 C 的选择方法参照文献[13,14].对于每个数据集选取15个 γ 值和15个 C 值,然后在225种参数对中选择性能最好的组合(C,γ).15个 C 值分别取:0.001,0.01,0.05,0.1,0.2,0.5,1,2,5,10,20,50,100,1000和

10000.15 个 γ 值分别取:0.001,0.01,0.1,0.2,0.4,0.8,1,2,5,10,20,50,100,1000 和 10000.SDM 算法中的参数 M 取值为 2.

4.2 SDM 与 SDDM 和 ESSDM 的比较

由于有效集算法需要设置初始迭代向量,在此 3 种算法的初始迭代向量都作如下设置:前一半分量的指标作为初始工作集,值设为 $C/2$,后一半分量的指标作为初始有效集,值设为 C.由于有效集算法都是求解式(2),得到的最优解是一致的,所以它们的测试精度是一样的.在此利用迭代次数,训练时间作为算法性能的评价指标.表 2 给出了 3 种算法的比较结果,表 3 给出了 3 种算法的各项参数,包括边界支持向量(bounded support vector, BSV),非边界支持向量(non-bounded support vector, NBSV),初始函数值和最优解函数值.

表 2 SDM 与 SDDM 和 ESSDM 的比较

数据集	SDDM		ESSDM		SDM	
	迭代次数	训练时间	迭代次数	训练时间	迭代次数	训练时间
Breast-cancer	321	0.2375	289	0.2021	191	0.0822
liver-disorders	190	0.0905	175	0.0898	132	0.0514
Diabetes	715	1.8233	674	1.2398	523	0.5857
ionosphere	181	0.0590	167	0.0520	153	0.0419
Pimadata	518	0.6710	481	0.5287	341	0.3936
Pwlinear	116	0.0332	103	0.0303	78	0.0239
Sonar	218	0.0648	211	0.0597	153	0.0497
Monk's Problem 1	279	0.0852	285	0.0921	216	0.0671
Monk's Problem 2	408	0.2087	394	0.1998	334	0.1884
Splice	979	10.94	878	5.282	583	1.123
heart	164	0.0516	167	0.0534	162	0.0503
Australian	476	0.3888	421	0.312	328	0.2455

表 3 SDM 与 SDDM 和 ESSDM 算法的参数

数据集	(C,γ)	#NBSV	#BSV	初始函数值	最优解函数值
Breast-cancer	(5,50)	4	219	9.5386e+002	−7.5523e+002
liver-disorders	(10,2)	9	155	2.0524e+004	−1.4667e+003
Diabetes	(1,1)	10	330	8.8695e+003	−3.1305e+002
ionosphere	(5,5)	40	2	2.4865e+004	−4.9571e+001
Pimadata	(10^3,50)	9	222	2.5246e+009	−2.2394e+005
Pwlinear	(10^4,10^3)	5	56	2.0142e+008	−5.0409e+005
Sonar	(20,1)	73	0	2.6694e+004	−8.5923e+001
Monk's Problem 1	(10,1)	70	0	1.1908e+004	−7.6653e+001
Monk's Problem 2	(10^4,5)	53	24	3.2482e+010	−3.3522e+005
Splice	(2,20)	42	662	7.3972e+002	−1.1364e+003
heart	(10^4,5)	15	44	1.1472e+009	−3.3186e+005
Australian	(2,2)	25	173	7.8305e+003	−3.7012e+002

从表 2 可以看出,在 12 个标准数据集测试中,SDM 相比 SDDM,迭代次数平均减少 1/3 左右,训练时间也有大幅的降低.从表 3 可以看出,对于 BSV 个数较多的数据集,SDM 算法训练时间减少较多,对于 NBSV 个数较多的数据集,SDM 算法训练时间减少较少.这是由于当 BSV 个数较多时,根据引入的映射函数,SDM 算法在每次迭代中有更高的几率将工作集变量判定到边界上,同时会使得函数值二次下降,进而减少算法的训练时间.

4.3 迭代向量初始值敏感性分析

迭代向量初始值选取的不同会对算法性能造成较大影响,本节利用 Splice 数据集比较单向下降法和二次下降法的不同初始迭代向量的性能.由于式(1)的不等式约束的边界值为 0 和 C,在此利用两个策略来分析不同的迭代向量初始值对算法的影响.策略 1 每次测试初始迭代向量的前 $(N/10)*i$ 个分量作为初始工作集,值设为 $C/2$,余下分量的指标作为初始有效集,值设为 C.策略 2 每次测试初始迭代向量的前 $(N/10)*i$ 个分量作为初始工作集,值设为 $C/2$,余下分量的指标作为初始有效集,值设为 0,其中 $i=1,\cdots,10$.

对于每个数据集利用策略 1 和策略 2 时,都有 10 次测试,对于每次测试利用迭代次数和训练时间作为性能测试指标.图 1 ~ 2 是两个算法在 Splice 数据集上基于策略 1 的比较结果.图 3 ~ 4 是两个算法在 Splice 数据集上基于策略 2 的比较结果.

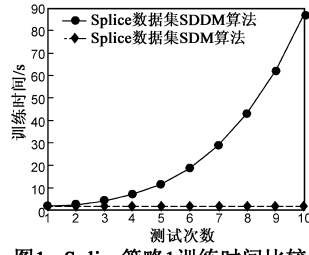


图1 Splice策略1训练时间比较

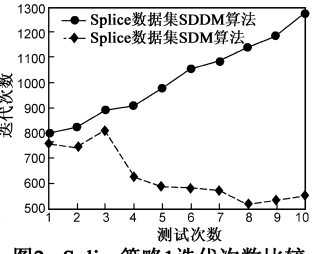


图2 Splice策略1迭代次数比较

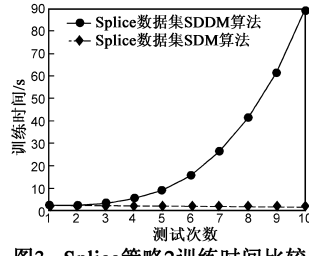


图3 Splice策略2训练时间比较

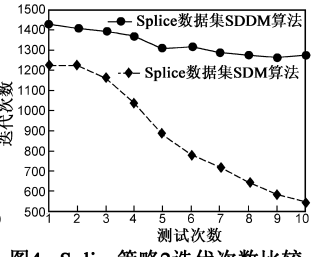


图4 Splice策略2迭代次数比较

从图 1 ~ 2 可以看出,SDDM 训练时间受迭代向量初始值变化相当明显,这是由于初始工作集变量很多时单向下降法每一次迭代的计算复杂度太高.从图 4 可以看出,SDM 在初始工作集变量变多时迭代次数下降明显.从两个策略的比较可以看出,SDDM 对迭代向量初始值变化敏感,在初始迭代时工作集变量较少时可以表现出较好的性能.SDM 在初始工作集变量较多时性能较好.

5 结论和将来的工作

针对传统有效集方法求解 SVM 优化问题出现的迭代速度较慢和容易出现误迭代的问题,本文提出了基

于 SDM 和推测赋值法的改进算法,能够使 SVM 优化问题的目标函数较传统的 SDDM 进一步下降,从而有效地减少迭代次数和降低计算复杂度,并且能有效减少误迭代的情况.实验表明该方法的训练时间对迭代向量初始值的变化不敏感,当初始迭代时工作集变量增加时,提出的算法的迭代次数可以有效地减少.

本文的后续工作将研究在初始迭代阶段能识别正确的工作集变量的方法,这就保证了每次迭代选取多个违反 KKT 条件的变量的可行性,从而进一步减少迭代次数和降低计算复杂度.

参考文献

[1] Cortes C, et al. Support vector networks[J]. Machine Learning, 1995, 20(3): 273 – 297.

[2] Osuna E, et al. An improved training algorithm for support vector machines[A]. Proceedings of the 1997 IEEE Workshop Neural Networks for Signal Processing VII[C]. J Principe, L Giles, N Morgan, E Wilson, editors, Amelia Island, Florida, USA, 1997. 276 – 285.

[3] Joachims T. Making Large-Scale SVM Iearning Practical[M]. Advances in Kernel Methods-Support Vector Learning, B Schölkopf, C J C Burges, A J Smola, Eds Cambridge, MA: MIT Press, 1998.

[4] Chang C C, et al. LIBSVM: A library for support vector machines[CP/OL]. http://www.csie.ntu.edu.tw/~cjlin/libsvm, 2001-02-12.

[5] Platt J. Fast Training of Support Vector Machines Using Sequential Minimal Optimization[M]. Advances in Kernel Methods – Support Vector Learning, chapter 12, Cambridge, MA: MIT Press, 1999. 185 – 208.

[6] 方景龙,等.复杂分类问题支持向量机的简化[J].电子学报,2007,35(5):858 – 861.
Fang J L, et al. A simplification to support vector machine for complicated recognition problem[J]. Acta Electronica Sinica, 2007, 35(5): 858 – 861. (in Chinese)

[7] 杨滨,等.自适应迭代最小二乘支持向量机回归算法[J].电子学报,2010,38(7):1621 – 1625.
Yang B, et al. Adaptive and iterative training algorithm of least square support vector machine regression[J]. Acta Electronica Sinica, 2010, 38(7): 1621 – 1625. (in Chinese)

[8] Vishwanathan S V N, et al. SimpleSVM[A]. 20th International Conference on Machine Learning[C]. Washington DC: AAAI Press, 2003.

[9] Scheinberg K. An efficient implementation of an active set method for SVMs[J]. Journal of Machine Learning Research, 2006, 7(10): 2237 – 2257.

[10] R Fletcher. Practical Methods of Optimization, Constrained Optimization, Vol 2[M]. USA: John Wiley & Sons, 1981.

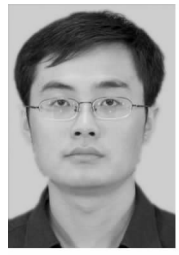
[11] Blake C L, et al. UCI repository of machine learning databases [EB/OL]. http://www.ics.uci.edu/~mlearn/MLRepository.html, 1998-04-02.

[12] Michie D, et al. Machine Learning, Neural and Statistical Classification[EB/OL]. ftp: ftp.ncc.up.pt/pub/stalog/, 2001-02-19.

[13] Huang GB, et al. Optimization method based extreme learning machine for classification[J]. Neurocomputing, 2010, 74(1 – 3): 155 – 163.

[14] 丁晓剑,赵银亮.优化极限学习机的序列最小优化方法[J].西安交通大学学报,2011,45(6):7 – 13.
Ding X J, Zhao Y L. Sequential minimal optimization method based on optimization ELM[J]. Journal of Xi'an Jiao tong University, 2011, 45(6): 7 – 13.

作者简介



丁晓剑 男,西安交通大学电子与信息工程学院博士研究生.主要研究方向为神经网络和机器学习.

E-mail: xjding@stu.xjtu.edu.cn

赵银亮 男,西安交通大学电子与信息工程学院教授.主要研究方向为并行计算,数据挖掘和新一代编程模型.

E-mail: zhaoy@mail.xjtu.edu.cn