文章编号：1001-0920(2009)12-1895-04

在原始空间用 Rosenbrock 算法训练线性支持向量机

刘叶青[1,2]，刘三阳[1]，谷明涛[3]

(1. 西安电子科技大学 数学科学系，西安 710071；2. 河南科技大学 理学院，
河南 洛阳 471003；3. 解放军 96251 部队，河南 洛阳 471003)

摘 要：为了加快并行下降方法(CD)用于线性支持向量机(SVM)时的最终收敛速度，将 Rosenbrock 算法(R)用于线性 SVM. 在内循环，R 通过解一个单变量子问题来更新 w 的一个分量，并同时固定其他分量不变；在外循环，采用 Gram-schmidt 过程构建新的搜索方向. 实验结果表明，与 CD 相比，R 加快了最终的收敛，在分类中能更快地获得更高的测试精度.

关键词：支持向量机；模式识别；分类；Rosenbrock 算法；并行下降

中图分类号：TP18 **文献标识码**：A

Training linear support vector machine by Rosenbrock algorithm in the primal space

LIU Ye-qing[1,2], *LIU San-yang*[1], *GU Ming-tao*[3]

(1. Department of Mathematical Sciences, Xidian University, Xi'an 710071, China; 2. School of Science, He'nan University of Science and Technology, Luoyang 471003, China; 3. PLA Unit96251, Luoyang 471003, China. Correspondent: LIU Ye-qing, E-mail: xiangshui15@163.com)

Abstract: To improve the speed of final convergence of coordinate descent method (CD) when it is applied to linear support vector machine (SVM), Rosenbrock algorithm (R) is applied to linear SVM. In inter iterations, R updates one component of w by solving a one-variable sub-problem while fixing other components. In outer iterations, the new search direction is constructed by the Gram-schmidt procedure. Experimental results show that, compared with CD, R accerlates final convergence and achieves higher testing accuracy more quickly in classification.

Key words: Support vector machine; Pattern recognition; Classification; Rosenbrock algorithm; Coordinate descent

1 引 言

支持向量机(SVM)[1]是很受欢迎的数据分类工具. 由于 SVM 是带有不等式约束的二次优化问题，大多文献都使用 Lagrange 定理来求解它的对偶问题. 然而，最近几年人们提出用原始优化问题来训练 SVM[2-5]，这些新出现的算法是循环算法，一次更新 w 的所有分量，由此产生了 w 的一个序列 $\{w_k\}_{k=0}^{\infty}$.

文献[6]将并行下降方法用于线性 SVM，每次通过解一个单变量的子问题来更新 w 的一个分量. 该方法虽然开始时收敛速度快，但它的最终收敛却很慢. 本文根据该方法的特点，在每次进入下一个循环前求解一个新的搜索方向. 新方向的构建采用 Gram-schmidt 过程，而修改后的这种方法正是连续的 Rosenbrock 方法，Rosenbrok 方法是用一个全局牛顿算法来求解单变量子问题. 在 UCI 数据集上的实验结果显示，Rosenbrock 方法不仅加速了最终的收敛，而且在分类中能够更快地获得更高的测试精度.

2 并行下降方法

给定一个训练数据集 $\{(x_1,y_1),(x_2,y_2),\cdots,(x_m,y_m)\}$，$x_i\in R^n$，$y_i\in\{-1,1\}$. 用 SVM 求解下面的无约束优化问题：

$$\min_w f(w)=\frac{1}{2}w^{\mathrm{T}}w+C\sum_{i=1}^{m}\max(0,1-y_iw^{\mathrm{T}}x_i)^2. \quad (1)$$

收稿日期：2008-12-25；**修回日期**：2009-03-21.

基金项目：国家自然科学基金项目(60574075,60705004).

作者简介：刘叶青(1979—)，女，山东聊城人，博士生，从事模式识别、机器学习等研究；刘三阳(1959—)，男，西安人，教授，博士生导师，从事最优化理论及其应用等研究.

并行下降方法[7] 使用坐标轴作为搜索方向，即在每个循环中沿着并行方向 $d_1,\cdots,d_n$ 搜索. 这里 d_j 为第 j 个位置是 1，其余位置都是 0 的向量. 这样沿着搜索方向 d_j，w_k 的第 j 个分量将被改变，而其他分量都保持不变.

需要指出的是，并行下降方法中究竟应将哪个坐标轴方向放在前面、哪个放在后面进行一维搜索，将会对算法的收敛速度产生较大的影响. 虽然在实际问题中可以按各因素（变量）对实验结果的影响大小依次自前往后排列坐标轴方向，但当变量很多时，这种方法的效率不高.

3 解线性 SVM 的 Rosenbrock 方法

3.1 Rosenbrock 算法

Rosenbrock 方法是对并行下降方法的改进方法. 并行下降方法每次循环使用的搜索方向都是并行方向，而 Rosenbrock 方法在每次循环前需要求解一个新的搜索方向. 下面首先给出 Rosenbrock 方法的具体描述，即算法 1；然后给出求解新搜索方向的 Gram-schmidt 过程.

算法 1 线性 SVM 的 Rosenbrock 方法.

Step1：给定初始点 w_1，初始搜索方向 $d_1,\cdots,d_n$. 令

$$z_1 = w_1,\ k = j = 1,\ \varepsilon_1 > 0.$$

Step2：while $j \leqslant n$，解子问题

$$\text{minimize}\, f(z_j + \lambda d_j),$$

得最优解 λ_j，令 $z_{j+1} = z_j + \lambda_j d_j$，$j = j + 1$.

end while.

Step3：令 $w_{k+1} = z_{n+1}$，如果 $\| w_{k+1} - w_k \| < \varepsilon_1$，则停止；否则，$z_1 = w_{k+1}$，$k = k + 1$，转 Step4.

Step4：根据 Gram-schmidt 过程求解搜索方向，并用 $d_1,\cdots,d_n$ 标注. 令 $j = 1$，转 Step2.

在 Step1 中，初始搜索方向一般选为并行方向，这样 $d_1,\cdots,d_n$ 每个向量的范数都等于 1，而且是互相垂直的. 从当前点 w_k 出发，目标函数 f 沿着每个方向被最小化，得到点 w_{k+1}，从而

$$w_{k+1} - w_k = \sum_{j=1}^{n} \lambda_j d_j,$$

这里 λ_j 为沿方向 d_j 移动的距离. 根据 Gram-schmidt 过程，产生新搜索方向 $\bar{d}_1,\cdots,\bar{d}_n$ 如下：

$$a_j = \begin{cases} d_j,\ \lambda_j = 0; \\ \sum\limits_{i=j}^{n} \lambda_i d_i,\ \lambda_j \neq 0. \end{cases}$$

$$b_j = \begin{cases} a_j,\ j = 1; \\ a_j - \sum\limits_{i=1}^{j-1} (a_j^{\mathrm{T}} \bar{d}_i)\bar{d}_i,\ j \geqslant 2; \\ \bar{d}_j = b_j / \| b_j \|. \end{cases}$$

文献[7] 已经证明，用上述方法产生的搜索方向是线性无关和垂直的.

在算法 1 中，需要求解下面的单变量子问题：

$$\min_{\lambda} f(z_j + \lambda d_j).$$

可将其重新写为

$$D_j(\lambda) = f(z_j + \lambda d_j) = \frac{1}{2}(z_j + \lambda d_j)^{\mathrm{T}}(z_j + \lambda d_j) + C \sum_{i \in I(z_j + \lambda d_j)} (b_i(z_j + \lambda d_j))^2. \tag{2}$$

这里：$b_i(w) = 1 - y_i w^{\mathrm{T}} x_i$，$I(w) = \{ i \mid b_i(w) > 0 \}$. 由于 $D_j(\lambda)(\lambda \in R)$ 是分段平方函数，这里用牛顿方法最小化. $D_j(\lambda)$ 的一阶导数为

$$D_j'(\lambda) = z_{jj} + \lambda - 2C \sum_{i \in I(z_j + \lambda d_j)} y_i x_{ij} (b_i(z_j + \lambda d_j)). \tag{3}$$

这里：z_{jj} 为 z_j 的第 j 个分量，x_{ij} 为 x_i 的第 j 个特征.

因为 $D_j(\lambda)$ 不是二次可微的，所以用与文献[6] 相同的方法来定义广义二阶导数

$$D_j''(\lambda) = 1 + 2C \sum_{i \in I(z_j + \lambda d_j)} x_{ij}^2. \tag{4}$$

本文不使用文献[6] 中解子问题的方法，而是用一个全局牛顿算法来求解. 该算法的全局收敛性已在文献[8] 中得到证明. 本文根据 $D_j''(\lambda) > 0$ 进行了相应的修改，其详细步骤如下：

算法 2 用全局牛顿算法求解子问题.

Step1：给定初始值 λ_1，令 $k = 1$，选择 $\varepsilon_2 > 0$.

Step2：计算 $D_j'(\lambda_k)$，$D_j''(\lambda_k)$. 如果 $| D_j'(\lambda_k) | < \varepsilon_2$，则停止；否则，令 $\alpha_k = 1$.

Step3：如果

$$D_j(\lambda_k - \alpha_k D_j'(\lambda_k)/D_j''(\lambda_k)) \leqslant D_j(\lambda_k) - \frac{\alpha_k}{4}[D_j'(\lambda_k)]^2 / D_j''(\lambda_k),$$

则转 Step4；否则，令 $\alpha_k = \alpha_k / 2$，转 Step3.

Step4：令 $\lambda_{k+1} = \lambda_k - \alpha_k D_j'(\lambda_k)/D_j''(\lambda_k)$，$k = k + 1$，转 Step2.

算法 1 的收敛对于分类精度而言是很重要的，下面给出如下定理来证明算法 1 的收敛.

定理 1 由算法 1 产生的点列 $\{w_k\}$ 收敛.

证明 根据文献[7] 的定理 7.3.5，只需证明：

1) 算法产生的点列 $\{w_k\}$ 包含在 R^n 的一个紧子集内；

2) f 沿任意方向的最小值是唯一的.

考虑水平集

$$\Lambda = \{ w : f(w) \leqslant f(w_1) \},$$

其中 w_1 是初始点. 显然算法产生的点列 $\{w_k\}$ 包含在水平集 Λ 内，从而用反证法易证 Λ 是有界闭集，即

紧集；然后根据 Weierstrass 定理及 $f(w)$ 严格凸，可知 f 沿任意方向的最小值是唯一的. □

3.2　算法性能分析

算法 1 每次外部循环的计算复杂度主要取决于 Step2 求解子问题(算法 2) 和 Step4 求新搜索方向.

对于算法 2(用全局牛顿算法求解子问题)，为计算 $D_j'(\lambda_k)$ 和 $D_j''(\lambda_k)$，对于所有的 i，需计算 $b_i(z_j+\lambda d_j)$. 计算 $b_i(z_j+\lambda d_j)$，$i=1,\cdots,m$，需 $O(mn)$ 次运算. 因此，计算 $D'_j(\lambda_k)$ 和 $D''_j(\lambda_k)$ 的花费为 $O(mn)$. 在 Step3 中需计算几个线性搜索步长 α_k，对于每个 α_k，主要的运算在于计算

$$D_j\left(\lambda_k-\alpha_k\frac{D_j'(\lambda_k)}{D_j''(\lambda_k)}\right)-D_j(\lambda_k).$$

将式(2) 带入上式，可知包含 $O(mn)$ 次运算. 实验中发现，α_k 一般取前几个数值 1，1/2 和 1/4，则算法 2 的花费为 $O(mn)+O(mn)\times$ 求线性搜索步长的次数，即算法 2 的花费大约为 $O(mn)$.

对于算法 1 的 Step4，求所有 a_j 最多包含 $O(n^2)$ 次运算，求每个 b_j 有 $2(j-1)n$ 次运算，因此求所有 b_j，$j=1,\cdots,n$，共需 $O(n^3)$ 次运算. 求所有 $\bar{d}_j$ 需 $O(n^2)$ 次运算，则求新搜索方向需 $O(n^3)$ 次运算.

综上所述，一般而言 Rosenbrock 算法每个外部循环的计算复杂度为 $O(mn^2)+O(n^3)$.

对于并行下降算法，其子问题不管用全局牛顿算法还是文献[6] 中的算法，它的每个外部循环的计算复杂度均为 $O(mn^2)$.

比较两个算法的计算复杂度. 因为 UCI 中的分类数据基本上都是 $m>n$，因此在这种情况下两个算法的计算复杂度相同，都是 $O(mn^2)$.

4　实　　验

为了验证 R 是否加速了最终的收敛，下面在并行下降方法(CD) 中使用全局牛顿算法来求解单变量子问题.

所有实验均在 AMDsempron(tm)2400 +，512 MB 内存的机器上进行，采用 MATLAB7.0.1 语言编程，实验中使用的数据来自 UCI.

在上面的讨论中，为了方便而省略了一个偏项 b，在接下来的实验中，通过将每个样本和 w 作如下变换而保留了偏项 b：

$$x_i^{\mathrm{T}}\rightarrow[x_i^{\mathrm{T}},1],\ w^{\mathrm{T}}\rightarrow[w^{\mathrm{T}},b].$$

实验中选择 $w_1=\mathrm{ones}(n+1,1)$，$\lambda_1=1$. 数据的 4/5 用于训练，1/5 用于测试. 对每个数据使用 5-折交叉有效验证来选择最好的参数 C.

4.1　R 和 CD 收敛速度的比较

实验 1　根据 4 个数据集上 $\|w_{k+1}-w_k\|$ 的收敛性来比较 R 和 CD. 图 1 为 $\|w_{k+1}-w_k\|$ 的值随着训练时间直到最小值的情况. 图 1 中省略了训练时间和 $\|w_{k+1}-w_k\|$ 的前几个值，因为对于后面的值而言这些值太大了，不易观察最终的收敛情况. 在数据集 monks 上，两种方法都一次收敛到零.

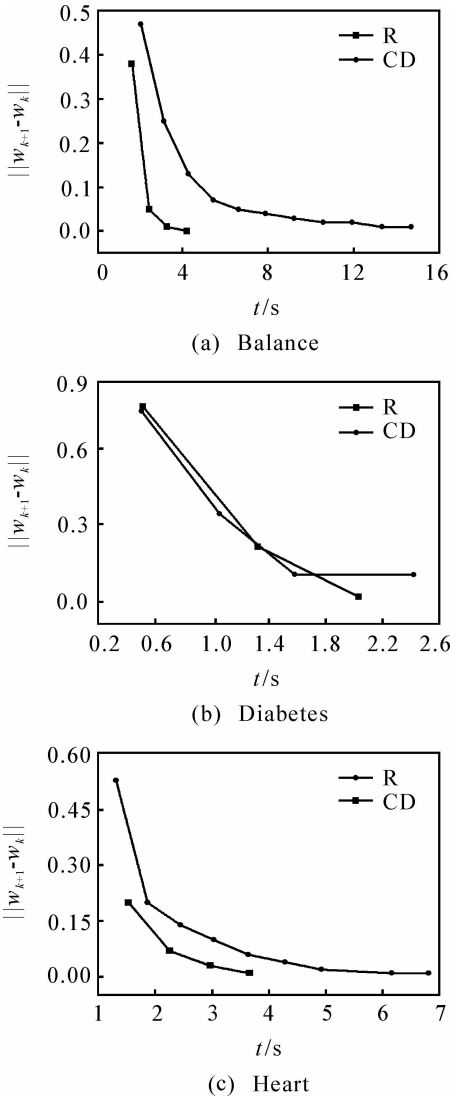

图 1　$w_{k+1}-w_k$ 的 2- 范数对训练时间

从图 1 可以看出，在没有前几个值(收敛的最快) 的情况下，CD 在开始阶段收敛的较快，但对于最终的收敛而言却很慢，而 R 比 CD 收敛快. 该结果意味着 Rosenbrock 算法确实改进了并行下降方法最终收敛慢的问题.

4.2　R，CD 及文献[4] 中 SSVM 分类结果比较

实验 2　检查训练时间和测试精度. 表 1 给出了参数取值情况，表 2 为比较结果，黑体表示最好的结果. 从实验结果可以看出，不论是训练时间还是测试精度，R 都明显优于 CD 和 SSVM 方法.

R 比 CD 训练时间短的原因是新增加的搜索方向加快了收敛速度. 虽然 SSVM 具有二次收敛性，但频繁地求解式(2) 中的一阶导数和二阶导数，使得训练时间比前面两种方法长.

表 1 数据集大小,R 和 CD 的最优参数值

Dataset size	R CD SSVM		
	C	ε_1	ε_2
Diabetes 766×9	0.1	2	1
	0.1	0.1	0.1
	0.1	1	0.1
Balance 625×5	0.1	1	1
	0.01	0.1	0.1
	0.01	0.01	0.1
Heart 270×14	0.0001	0.1	0.001
	0.1	0.1	0.1
	0.01	0.1	0.01
Monks 429×7	0.001	5	0.01
	0.0001	0.1	0.1
	0.001	0.1	0.01

表 2 R 和 CD 的测试精度和训练时间

Dataset size	testing accuracy/% training time/s		
	R	CD	SSVM
Diabetes	**75.817**	**75.817**	73.203
	2.625	4.047	6.923
Balance	**96**	92	91.726
	1.469	1.985	5.107
Heart	**85.185**	83.333	81.481
	0.125	0.719	1.807
Monks	**72.093**	70.93	65.116
	0.042	0.125	3.520

5 结 论

本文将直接搜索方法中的 Rosenbrock 算法用于线性 SVM,证明了算法的收敛性,并与并行下降算法进行了比较.实验结果表明,Rosenbrock 算法改进了并行下降算法最终收敛慢的问题,并以更快的速度取得了更高的测试精度.

Rosenbrock 算法和并行下降算法都属于非线性规划中的直接搜索方法,这类算法只需求解一维子问题而没有用到导数的信息.因此,算法简单、易执行,尤其在实际问题中,当目标函数的导数很难求解或者根本不能求解时,这类算法与用导数信息的算法相比更具优势.直接搜索方法虽然在非线性规划中得到了广泛研究,但用于 SVM 的并不多,而将其用于 SVM 时如何提高其收敛速度是很重要的一个研究方向.

参考文献(References)

[1] VapnikV. Satistical learning theory[M]. New York: Springer Verlag, 1995.

[2] Mangasarian O L, Wild E W. Multisurface proximal support vector machine classification via generalized eigenvalues[J]. IEEE Trans on Pattern Analysis and Machine Intelligence, 2006, 28(1): 69-74.

[3] Chapelle O, Sindhwani V, Keerthi S S. Optimization techniques for semi-supervised support vector machines [J]. J of Machine Learning Research, 2008, 9(1): 203-233.

[4] Lee Y J, Mangasarian O L. SSVM: A smooth support vector machine for classification [J]. Computational Optimization and Applications, 2001, 22(1): 5-21.

[5] Chapelle O. Training a support vector machine in the primal[J]. Neural Computation, 2007, 19(5): 1155-1178.

[6] Chang K W, Hsieh C J, Lin C J. Coordinate descent method for large-scale L_2-loss linear support vector machines[J]. J of Machine Learning Research, 2008, 9(4): 1369-1398.

[7] Bazaraa M S, Shetty C. Nonlinear programming: Theory and algorithms[M]. New York: Wiley, 1979.

[8] 袁亚湘. 非线性规划数值方法[M]. 上海:上海科学技术出版社, 1993.
(Yuan Y X. Numerical methods for nonlinear programming[M]. Shanghai: Shanghai Scientific and Technical Publishers, 1993.)

(上接第 1894 页)

[8] 王玉龙, 杨光红. 具有时变采样周期的网络控制系统的 H_∞ 控制[J]. 信息与控制, 2007, 36(3): 278-285.
(Wang Y L, Yang G H. H_∞ control of networked control system with time-varying sampling period[J]. Information and Control, 2007, 36(3): 278-285.)

[9] 俞立. 鲁棒控制——线性矩阵不等式处理方法[M]. 北京: 清华大学出版社, 2002.
(Yu L. Robust control — Dealing with linear matrix inequalities[M]. Beijing: Tsinghua University Press, 2002.)