一种非光滑损失坐标下降算法*

吴卫邦[1]，朱烨雷[2]，陶　卿[2]
（1. 镇江船艇学院 船艇指挥系，江苏 镇江 212003；2. 陆军军官学院 五系，合肥 230031）

摘　要：针对非光滑损失问题提出一种新的坐标下降算法，采用排序搜索的方式求解子问题解析解。分析了算法的时间复杂度，并给出了三种提高收敛速度的实用技巧。实验表明算法对正则化 Hinge 损失问题具有良好的性能，达到了预期的效果。

关键词：机器学习；优化；坐标下降；非光滑损失；Hinge
中图分类号：TP301.5　　**文献标志码**：A　　**文章编号**：1001-3695(2012)10-3688-05
doi:10.3969/j.issn.1001-3695.2012.10.022

New coordinate descent algorithm for non-smooth losses

WU Wei-bang[1], ZHU Ye-lei[2], TAO Qing[2]
(1. *Dept. of Watercraft Commanding, Zhenjiang Watercraft College of PLA, Zhenjiang Jiangsu* 212003, *China*; 2. *The* 5*th Dept., Army Officer Academy, Hefei* 230031, *China*)

Abstract: For non-smooth losses, this paper presented a new coordinate descent algorithm, to get the closed form solution of the single variable problem by using the sorting and searching method. It analyzed the time complexity of algorithm and gave three practical skills to improve the convergence rate. The experiments demonstrate the expected efficiency of the proposed algorithms in the regularized Hinge loss.

Key words: machine learning; optimization; coordinate descent; non-smooth loss; Hinge

统计机器学习算法无论是在理论还是应用上都取得了极为丰硕的成果[1]。当前对机器学习问题的研究一般在损失函数 + 正则化项的框架下进行，这是对统计机器学习理论框架的有效拓展。近几年来，机器学习所处理的问题逐渐呈现出大规模高维的特点。关于解大规模数据的线性方法不断涌现，其中较为优秀的算法当属坐标下降(coordinate descent, CD)算法[2,3]，取得了 $O(\log(1/\varepsilon))$ 的超线性收敛速度。

CD 算法通过逐个优化解向量的每一维特征(坐标)，实现一次外循环；内循环中，在优化某一坐标时，固定权向量中的其余 $n-1$ 维坐标不动，对该维坐标求解单变量子问题。CD 算法之所以能够获得如此快的收敛速度，主要有三个原因：a)单变量子问题可以精确求取解析解(closed form solution)，既不需要跟梯度算法一样作梯度逼近，也不需要搜索最优步长；b)批处理遍历完所有维数后对权向量作一次更新，相比而言，CD 算法每作一次坐标更新相应的权向量也得到更新，CD 算法更新(fresh)信息更快[4]；c)计算一个坐标的方向导数比计算函数值或全梯度简单得多，著名学者 Nesterov认为如果没有廉价(cheap)的坐标方向导数，CD 算法也就失去了其实际意义[5]。

1992 年 Luo 等人[6]对可微凸目标函数讨论了坐标下降算法的收敛性；文献[7]对光滑问题进一步研究，得到了超线性的收敛性。针对信号领域的 L1 正则化最小二乘回归(LASSO)问题，文献[8]提出了用坐标下降加以求解的办法。近几年，坐标下降算法的强大优势才得以充分地发掘[9,10]。针对 LASSO 问题，文献[8]指出如果能够合理地实现坐标下降算法，将可以得到比当前主流算法快得多的算法。2008 年，Lin C J 研究小组成功地将坐标下降算法引入机器学习领域[2,3]，取得了比 Pegasos[11]、割平面方法[12]、信赖域方法[13]更好的理论结果和实验效果。文献[2]讨论了 L2 正则化 + 可微损失函数原始问题的坐标下降算法。该方法将目标函数的正则化项与损失函数看做一个整体，对单变量子问题用牛顿法近似求解，通过线搜的方法保证了目标函数的单减性，并借助文献[6]的收敛分析获得了超线性的收敛速度。文献[3]讨论了 Hinge 损失和 L2 损失(改进的最小二乘损失)SVM 对偶问题的坐标下降算法。该方法就对偶形式的子问题关于单变量进行线性展开，在盒约束域中求解。随后文献[14]讨论了 SVM 对偶问题(有线性约束的光滑目标函数优化问题)的坐标下降算法，文献[15]讨论了 L1 正则化的 Logistic 和最小二乘损失的坐标下降的随机算法。

分析上文不难发现，以上研究工作的共同点在于研究损失函数是光滑的(Lipschitz 连续可导)，这就可以直接将函数进行一阶或二阶近似展开求解子问题。而机器学习中最常见的 Hinge 损失是非光滑且次可微，使得绝大多数基于梯度的方法难以直接应用。文献[3]求解了 L2 正则化 + Hinge 损失问题，虽然对偶目标函数迅速收敛，但在某些数据库(样本个数远大于维数)不一定保证原问题目标函数的收敛。同时，很多非光滑损失的原问题没有对偶形式，如 L1 正则化 Hinge 损失情况。因此，如何求解非光滑损失函数的坐标下降算法是一个值得关注而又亟需解决的问题。

收稿日期：2012-03-27；**修回日期**：2012-04-23　　**基金项目**：国家自然科学基金资助项目(60975040)

作者简介：吴卫邦(1981-)，男，讲师，主要研究方向为军事运筹学、人工智能；朱烨雷(1985-)，男，硕士研究生，主要研究方向为模式识别、人工智能、数据挖掘(zhu.ye.lei1985@gmail.com)；陶卿(1965-)，男，教授，博导，CCF 高级会员，主要研究方向为统计机器学习、人工智能.

1　坐标下降算法简介

以二分类为例,对于独立同分布的训练样本集 $S=\{(\boldsymbol{X}_1,y_1),\cdots,(\boldsymbol{X}_m,y_m)\}\in \mathrm{R}^n\times\{+1,-1\}$,求解下述凸优化问题:

$$\min_{\boldsymbol{W}} F(\boldsymbol{W})=\sum_{i=1}^{m} f_i(\boldsymbol{W})+P(\boldsymbol{W}) \tag{1}$$

其中:$\boldsymbol{W}\in \mathrm{R}^n$ 是优化问题的解向量;损失函数 $f(\boldsymbol{W})=\sum_{i=1}^{m} f_i(\boldsymbol{W})$ 表示由所有样本造成的损失,用于控制模型的训练精度。本文主要讨论非光滑损失函数为 Hinge(L1)损失:$f_i(\boldsymbol{W})=\max\{0,1-y_i(\langle \boldsymbol{W},\boldsymbol{X}_i\rangle)\}$;$P(\boldsymbol{W})=\|\boldsymbol{W}\|_p^{\sigma}$ 称为正则化项,用于避免模型的过拟合。$\|\cdot\|_p^{\sigma}$ 表示 l_p 范数的 σ 次方。本文主要研究 L1 正则化项:$P(\boldsymbol{W})=\lambda\|\boldsymbol{W}\|_1$ 和 L2 正则化项:$P(\boldsymbol{W})=\lambda\|\boldsymbol{W}\|_2^2$ 两种情况。通过调整参数 λ,可以得到兼有训练精度和泛化能力的模型。

坐标下降算法将 $\boldsymbol{W}$ 的每一维看成是一个坐标,根据选取优化坐标方式的不同,坐标下降算法有许多形式,如循环坐标下降(cyclic coordinate descent,CCD)和随机坐标下降(stochastic coordinate descent,SCD)等算法。循环坐标下降就是按照次序依次循环优化解向量的每一维;随机坐标下降算法则是随机选取一维坐标并对其进行优化。以原始坐标下降算法为例,其迭代的过程分为内循环和外循环两部分[2]。迭代过程从起始点 $\boldsymbol{W}^0$ 开始依次迭代出 $\boldsymbol{W}^1,\boldsymbol{W}^2,\cdots,\boldsymbol{W}^k$。从 $\boldsymbol{W}^k$ 到 $\boldsymbol{W}^{k+1}$ 的过程称为一次外循环。$\boldsymbol{W}^{k+1}$ 通过更新 $\boldsymbol{W}^k$ 的 n 个变量来实现一次外循环,每一次外循环包含 n 次内循环。

每次内循环生成 $\boldsymbol{W}^{k,j}\in \mathrm{R}^n(j=1,\cdots,n)$,且 $\boldsymbol{W}^{k,1}=\boldsymbol{W}^k$,$\boldsymbol{W}^{k,n+1}=\boldsymbol{W}^{k+1}$,$\boldsymbol{W}^{k,j}=[w_1^{k+1},\cdots,w_{j-1}^{k+1},w_j^k,\cdots,w_n^k]^{\mathrm{T}}(j=2,\cdots,n)$首尾相接。对于 $\boldsymbol{W}^{k,i}$ 到 $\boldsymbol{W}^{k,j+1}$ 的更新,通过求解如下单变量子问题得到:

$$\min_z \boldsymbol{W}^{k,j}=[w_1^{k+1},\cdots,w_{j-1}^{k+1},w_j^k+z,w_{j+1}^k,\cdots,w_n^k]^{\mathrm{T}}=\min_z f(\boldsymbol{W}^{k,j}+z\boldsymbol{e}_j)\quad \boldsymbol{e}_j=[0,\cdots,1,\cdots,0]^{\mathrm{T}} \tag{2}$$

具体算法流程如算法 1 所示。

算法 1　坐标下降算法

start with any initial $\boldsymbol{W}^0$

for $k=0,1,\cdots$(outer iterations)

　for $j=1,2,\cdots,n$(inter iterations)

　　fix $w_1^{k+1},\cdots,w_{j-1}^{k+1},w_{j+1}^k,\cdots,w_n^k$ and approximately solve the sub-problem(2)

以光滑损失为例,介绍一种常见的求解方法。令 $a_j(z)=f(\boldsymbol{W}^{k,j}+z\boldsymbol{e}_j)$,式(2)的单变量子问题可以写成如下形式:

$$\min_z \quad a_j(z)+P(\boldsymbol{W}^{k,j}+z\boldsymbol{e}_j) \tag{3}$$

对式(3)的损失项用二阶近似展开:

$$\min_z \quad a_j'(0)z+\frac{1}{2}a_j''(0)z^2+p(w_j^{k,j}+z)-p(w_j^{k,j}) \tag{4}$$

这里 $a_i''(0)$ 是 $a_i'(0)$ 在 0 处的(次)梯度。子问题变成一个关于单变量的二项式,式(4)最小值即为当前子问题的解析解。文献[2]针对 L2 损失(光滑),用牛顿法(Newton method)近似求解了子问题。

2　损失坐标下降算法推导

由于 Hinge 损失是非光滑且非连续可导的,故式(4)无法成立。因而在非光滑情况下求解式(1)的子问题,就不能直接使用式(4)的二阶近似展开的方式。本章从一种新的角度来求解非光滑损失的坐标下降问题,主要讨论 L2 正则化项 + Hinge 损失和 L1 正则化项 + Hinge 损失两种情况子问题的算法推导,并针对子问题的具体形式给出直接排序搜索求解子问题的方法。

2.1　L2 正则化项 + Hinge 损失

当 $P(\boldsymbol{W})=\lambda\|\boldsymbol{W}\|^2$ 时,式(1)变成

$$\min_{\boldsymbol{W}} F(\boldsymbol{W})=\lambda\|\boldsymbol{W}\|^2+\sum_{i=1}^{m}\max(0,1-y_i\langle \boldsymbol{W},\boldsymbol{X}_i\rangle) \tag{5}$$

将 max( )函数写成绝对值项加非绝对值项之和的形式,代入式(5)得

$$F(\boldsymbol{W})=\lambda\|\boldsymbol{W}\|^2+\sum_{i=1}^{m}\left[\frac{1}{2}|1-y_i\langle \boldsymbol{W},\boldsymbol{X}_i\rangle|+\frac{1}{2}(1-y_i\langle \boldsymbol{W},\boldsymbol{X}_i\rangle)\right] \tag{6}$$

固定 $n-1$ 维,处理当前的第 j 维,式(6)变换为

$$F(w_j)=\frac{1}{2}\left\{2\lambda w_j^2+\sum_{i=1}^{m}|1-y_i\langle \boldsymbol{W},\boldsymbol{X}_i\rangle|+\sum_{i=1}^{m}(1-y_i\langle \boldsymbol{W},\boldsymbol{X}_i\rangle)\right\} \tag{7}$$

其中:$1-y_i\langle \boldsymbol{W},\boldsymbol{X}_i\rangle=1-y_i(w_1x_i^1+\cdots+w_jx_i^j+\cdots+w_nx_i^n)=[1-y_i(w_1x_i^1+\cdots+w_nx_i^n)]-y_iw_jx_i^j-y_ix_i^j=A_i^j$,表示第 j 维第 i 样本的一次项系数,$1-y_i(w_1x_i^1+\cdots+w_nx_i^n)=B_i^j$ 表示常数项系数,将 A_i^j 和 B_i^j 代入式(7)得

$$F(w_j)=2\lambda w_j^2+\sum_{i=1}^{m}|A_i^jw_j+B_i^j|+\sum_{i=1}^{m}(A_i^jw_j+B_i^j)=2\lambda w_j^2+\sum_{i=1}^{m}A_i^jw_j+\sum_{i=1}^{m}B_i^j+\sum_{i=1}^{m}|A_i^jw_j+B_i^j| \tag{8}$$

再令 $S_i(w_j)=|A_iw_j+B_i|$,$\partial S_i(w_j)$ 为 $S_i(w_j)$ 在 w_j 处的次梯度,满足

$$\partial S_i(w_j)=\begin{cases}-A_i^j & \text{if } A_i^jw_j+B_i^j<0\\ [-A_i^j,A_i^j] & \text{if } A_i^jw_j+B_i^j=0\\ A_i^j & \text{if } A_i^jw_j+B_i^j>0\end{cases}$$

对式(8)w_j 求偏导,令 $\partial S_i(w_j)=s_i(w_j)$,可得

$$\partial F(w_j)=4\lambda w_j+\sum_{i=1}^{m}A_i^j+\sum_{i=1}^{m}s_i(w_j)$$

$F(w_j)$ 是关于 w_j 的二次凸函数,$\partial F(w_j)=0$ 为全局最优值,此时最优解 w_j^* 可通过下式求得:

$$w_j^*=\{w_j|4\lambda w_j+\sum_{i=1}^{m}A_i^j+\sum_{i=1}^{m}s_i(w_j)=0\} \tag{9}$$

由于 $s_i(w_j)$ 在 $-B_i^j/A_i^j=u_i^j$ 处出现阶跃,$\partial F(w_j)$ 函数趋势如图 1 所示。从图 1 可以发现,$\partial F(w_j)$ 与 w_j 轴的交点处所对应的 w_j^* 就是当前解析解的值。

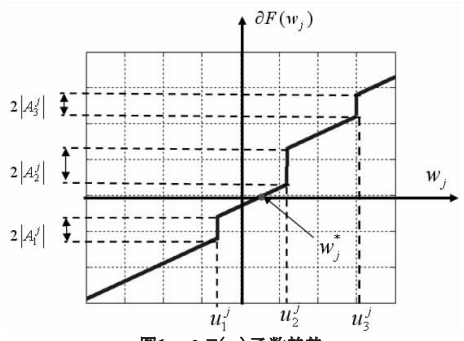

图1　$\partial F(w_j)$ 函数趋势

2.2　L1 正则化项 + Hinge 损失

当 $P(\boldsymbol{W})=\lambda\|\boldsymbol{W}\|_1$ 时,式(1)变换为

$$\min_{\boldsymbol{W}} F(\boldsymbol{W})=\lambda\|\boldsymbol{W}\|_1+\sum_{i=1}^{m}\max(0,1-y_i\langle \boldsymbol{W},\boldsymbol{X}_i\rangle) \tag{10}$$

由于 $P(\boldsymbol{W})=\lambda\|\boldsymbol{W}\|_1$,根据 2.1 节的方法无法直接求出式(10)的解析解。在 RDA[16] 算法的启发下,式(10)增加辅助项 $\frac{\beta_\tau}{\tau}h(\boldsymbol{W})$,并随着迭代次数满足单调递减性。

$$\min_{W} F(W) = \lambda \| W \|_1 + \sum_{i=1}^{m} \max(0, 1 - y_i \langle W, X_i \rangle) + \frac{\beta_\tau}{\tau} h(W) \tag{11}$$

其中:τ 为迭代次数;$\{\beta_\tau\}$ 是非负、不减的序列;$h(W)$ 为任意强凸函数,为便于分析本文取 $h(W) = \frac{1}{2} \| W \|^2$。因此求 w_j^* 的值转换为求解式(11)子问题解析解的问题。据式(5)~(7)可得

$$F(w_j) = \lambda | w_j | + \sum_{i=1}^{m} | A_i^j w_j + B_i^j | + \sum_{i=1}^{m} (A_i^j w_j + B_i^j) + \frac{\beta_\tau}{\tau} \times \frac{1}{2} w_j^2$$

为便于计算,这里将正则化项 $|w_j|$ 看做 $|A_i w_j + B_i|$ 的一种特殊形式,令 $A_0^j = \lambda, B_0^j = 0, s_0(w_j) = \partial |\lambda w_j|$,则

$$\partial F(w_j) = \frac{\beta_\tau}{\tau} w_j + \sum_{i=1}^{m} A_i^j + \sum_{i=1}^{m} s_i(w_j) + s_0(w_j) =$$

$$\frac{\beta_\tau}{\tau} w_j + \sum_{i=1}^{m} A_i^j + \sum_{i=0}^{m} s_i(w_j)$$

此时最优解 w_j^* 可通过下式求得:

$$w_j^* = \{ w_j | \frac{\beta_\tau}{\tau} w_j + \sum_{i=1}^{m} A_i^j + \sum_{i=0}^{m} s_i(w_j) = 0 \} \tag{12}$$

2.3 直接排序求精确解

前两节均给出了 w_j^* 的解析解的形式,但由于次梯度 $s_i(w_j)$ 的存在,无法直接求得 w_j^* 的值。下面用排序的方法递推搜索 $\partial F(w_j) = 0$ 处所对应的 w_j^* 的值。以 L2 正则化项 + Hinge 损失为例,具体操作步骤如下:

a)初始化。

(a)给定权向量 $W = 0$;

(b)计算第 j 维所有样本的常数系数 $A_i^j (i = 1, \cdots, m_j)$,根据式(9)令 $C_j = \sum_{i=1}^{m} A_i^j$,计算次梯度项的最小系数 $D_j = \min \sum_{i=1}^{m} s_i(w_j) = \sum_{i=1}^{m} - |A_i^j|$。

b)计算第 j 维中所有样本的一次项系数 $B_i^j (i = 1, \cdots, m_j)$,并得出阶跃点 $u_i^j = -B_i^j / A_i^j$。

c)对 u_i^j 从小到大排序[value, index] = sort(u_i^j)。value = $\{u_{\text{index}(1)}^j, u_{\text{index}(2)}^j, \cdots, u_{\text{index}(m_j)}^j\}$,满足 $u_{\text{index}(1)}^j \leqslant u_{\text{index}(2)}^j \leqslant \cdots \leqslant u_{\text{index}(m_j)}^j$,index 是 value 在原序列 u_i^j 映射的序号。

d)按照阶跃点从小至大的顺序搜索 $\partial F(w_j) = 0$ 处所对应的 w_j^* 的值。

e)根据算法 1,依次计算各维,$j = 1, \cdots, n$,执行步骤 b)~d),直到满足某种收敛条件停止。

第 j 维解析解精确求解流程伪代码如下:

```
for i = 1 to m_j
min f = 4λ * value(i) + C_j + D_j
max f = min f + 2 | A^j_index(i) |
if  min f  > 0
w_j = -(C_j + D_j)/4λ  终止
else
if  max f ≥ 0
w_j = value(i)  终止
else
D_j = D_j + 2 | A^j_index(i) |
end
end
next
```

从上面的步骤可以发现,步骤 a)初始化求的值可以存储至内存中,后面的算法可以直接调用。整个算法的时间复杂度主要集中在步骤 c)的排序上,为了减少计算时间,提高算法的执行效率,本文引入红黑树(red-black tree)进行排序。红黑树是一种自平衡二叉查找树,是在计算机科学中用到的一种数据结构,其典型用途是实现关联数组。它是在 1972 年由 Bayer 发明的,被称为对称二叉 B 树,现在的名字是 Guibas 等人于 1978 年写的一篇论文中获得的。它是复杂的,但其操作有着良好的最坏情况运行时间,并且在实践中是高效的,可以在 $O(\log m)$ 时间内进行查找、插入和删除,这里的 m 是树中元素的数目。

由于大规模数据一般都是稀疏的,这里定义 $\bar{m}$ 为各维样本的平均数目。通过分析可以知道在最坏的情况下执行一次内循环的时间复杂度为 $O(\bar{m} \log \bar{m})$。

3 算法的加速技巧

上一章中给出一次内循环的时间复杂度的上界为 $O(\bar{m} \times \log \bar{m})$,这个时间开销还是比较大的。因此,根据算法的自身性质给出三种加速的技巧,使算法能够快速地收敛。

3.1 分段技巧

根据 $\partial F(w_j)$ 的单调性和图 1 可知,$\partial F(w_j)$ 与 w_j 轴的交点必在某一个区域中,w_j^* 的值可在某一区域中求取,而其他的区域跟 w_j^* 的关系不大。因此,确定 w_j^* 所在的区域以及区域两边端点的值,并在较小的区域内进行排序求解,就是分段所要做的工作。令第 j 维的样本数为 m_j,具体操作流程如下:

a)$1 \sim m_j$ 中产生任意随机正整数,mid_index = mod(rand(M), m_j) + 1 为分段的节点序号,其中 rand(M)是一个较大的随机数,满足 rand(M) > m_j。

b)判断所有节点 u_i^j 中比 $u_{\text{mid\_index}}^j$ 小的值和对应的序号,将得到的结果存储到 left_value 和 left_index 的数组中,记录左分段节点的数目 left_count,并计算左分段区域右端点值 D_mid = $D_j + 2 \sum_{i=1}^{\text{left\_count}} | A_{\text{index}(i)}^j |$,这里下标 index 表示 u_i^j 中比 $u_{\text{mid\_index}}^j$ 小的序号,满足 $\{\text{index}(i) | u_{\text{index}(i)}^j < u_{\text{mid\_index}}^j\}$;再把 u_i^j 中剩下的值和对应的序号作为右分段,存储到 right_value 和 right_index 的数组中,这样就把 $\partial F(w_j)$ 分成了两段。

c)确定 w_j^* 大概位置。计算左分段的右端点值 $LR = 4\lambda u_{\text{mid_index}}^j + C_j + D$_mid,分两种情况进行讨论:

(a)如果 $LR \geqslant 0$ 表示 w_j^* 的位置在左分段区域,则把 left_value、left_index 及 D_mid 值存储到临时数组 temp_value、temp_index 及 temp_mid 中,以供下一次的分段或排序使用。

(b)如果 $LR < 0$,又分两种情况讨论:①如果 $LR + 2 | A_{\text{mid\_index}}^j | \geqslant 0$,则表示 w_j^* 在 mid_index 节点的阶跃处的直线上,此时 $w_j^* = u_{\text{mid\_index}}^j$,所有内循环终止;②如果 $LR + 2 | A_{\text{mid\_index}}^j | < 0$,说明 w_j^* 的位置在右分段区域,把 right_value、right_index 及 D_mid 的值存储到需要排序的临时数组 temp_value、temp_index 及 temp_mid 中。

d)反复执行步骤 a)~c),最后按照某一标准停止,如分段执行的次数或排序的规模等。最后按照 2.3 节中排序搜索策略求解 w_j^* 的值。

分段的好处在于,充分利用各个区域之间相互独立并对 w_j^* 不产生影响的性质,无须对某一维的所有节点进行完全排序。简单计算一下这种分段技巧的时间复杂度。为了便于分析,假设每次都在节点的中间位置分段,即 mid_index = $\bar{m}/2$,分段过程执行 l 次。第一次执行分段需要的时间复杂度为 $\bar{m}$,第二次为 $\bar{m}/2$,依次类推第 l 次为 $\bar{m}/2^l$,形成一个等比数列。因

此分段需要总共的时间复杂度为 $O(2[1-(1/2)^l]\bar{m})$,且上界是 $O(2\bar{m})$。此时排序的时间复杂度为 $O\left(\frac{\bar{m}}{l}\log\frac{\bar{m}}{l}+2\bar{m}\right)$。

按照上面的理论计算,可以发现 l 值越大,时间复杂度就越低,但在程序实际执行过程中并不能保证算法收敛的时间越快。这是因为分段技巧中存在一系列的赋值、判断等运算,占用了一定的时间开销。因此,针对不同的数据库可以找到一个临界值 l,确保算法以最快的速度收敛。不管怎样,这种分段技巧在降低时间复杂度是非常可观的,能够大大提高算法的执行效率。

3.2　线搜方式求精确解

直接排序搜索求解析解的方法其时间开销是比较大的,本节提出一种线搜的策略缩小排序的节点数,减少时间复杂度,同时保证解析解是精确解。

$\partial F(w_j)$ 是单调增函数,要寻找 $\partial F(w_j)=0$ 的点,只要找到两个点 a、b,使得 $\partial F(a)\leqslant 0$,$\partial F(b)\geqslant 0$,那么 $w_j^*\in[a,b]$。取初始点为当前值 w_j,步长 d,$k\geqslant 1$ 的正整数,计算 $\partial F(w_j)$。分三种情况讨论:

a)当 $\partial F(w_j)=0$,则 $w_j^*=w_j$,终止本次内循环;

b)当 $\partial F(w_j)>0$,则计算 $\partial F(w_j-d)$,直到 $\partial F(w_j-kd)\,\partial F(w_j-(k-1)d)<0$;

c)当 $\partial F(w_j)<0$,则计算 $\partial F(w_j+d)$,直到 $\partial F(w_j+kd)\,\partial F(w_j+(k-1)d)<0$。

这样只要针对 $u_i^j\in[w_j-kd,w_j-(k-1)d]$ 或者 $u_i^j\in[w_j+(k-1)d,w_j+kd]$ 进行排序就可以了。

分析线搜策略的时间复杂度:计算导数 $\partial F(w_j)$ 的时间复杂度是 $O(\bar{m})$,假定用 k 次计算就满足了线搜停止标准,线搜所产生的时间复杂度为 $O(k\bar{m})$。经过线搜后需要排序的节点被限定在一个更小的区域内,其需要排序的个数 $m_0<\bar{m}$。因此时间复杂度为 $O(k\bar{m}+m_0\log m_0)$。如果 k 和 m_0 远小于 $\bar{m}$,那么这种线搜策略就比较高效。

3.3　非精确线搜方式求近似解

前面子问题的各维 w_j^* 都是精确解,并且都使用到了排序,时间复杂度较高。在文献[2]的启发下用 $\tilde{w}_j$ 近似解来代替 w_j^* 的精确解,以达到降低时间复杂度的目的。

如果当前的值为 $F(w_j)$,只要找一个比 $F(w_j)$ 小的值即可。算法具体实现如下:同样设初始点为当前值 w_j,优化后的值为 $\tilde{w}_j$,并预先设置步长 d,$k\geqslant 1$ 的正整数,计算 $\partial F(w_j)$。

a)当 $\partial F(w_j)=0$,则 $\tilde{w}_j=w_j$,终止本次内循环。

b)当 $\partial F(w_j)>0$,则计算 $\partial F(w_j-d)$。

如果 $\partial F(w_j-d)<\partial F(w_j)$,则 $\tilde{w}_j=w_j-d$;

如果 $\partial F(w_j-d)\geqslant\partial F(w_j)$,则计算 $\partial F(w_j-\delta d)$,$\delta\in(0,1)$,直至找到 $\partial F\left(w_j-\frac{\delta}{k}d\right)<\partial F(w_j)$ 的点,此时 $\tilde{w}_j=w_j-\frac{\delta}{k}d$,停止计算。

c)当 $\partial F(w_j)<0$,则计算 $\partial F(w_j+d)$。

如果 $\partial F(w_j+d)<\partial F(w_j)$,则 $\tilde{w}_j=w_j+d$;

如果 $\partial F(w_j+d)\geqslant\partial F(w_j)$,则计算 $\partial F(w_j+\delta d)$,$\delta\in(0,1)$,直至找到 $\partial F\left(w_j+\frac{\delta}{k}d\right)<\partial F(w_j)$ 的点,此时 $\tilde{w}_j=w_j+\frac{\delta}{k}d$,停止计算。

这样就能确保 $F(\tilde{w}_j)<F(w_j)$,其时间复杂度仅为 $O(k\bar{m})$。

本文处理的是大规模稀疏数据。大规模高维机器学习问题的数据来源具有一定的结构特点:a)样本是独立同分布,在绝大多数情形下,少部分样本已足以反映样本集合的统计规律;b)样本是稀疏的,绝大部分属性是0,各维中对应的样本的个数也是不确定的,少则一两个,多则几千乃至几万个。因此,在计算的时候应该区别处理这些问题,对于样本(节点)个数少的维数直接用红黑树排序求解,对于样本个数多的维数可以同时灵活使用上面三种技巧。

4　数值实验

本文通过实验验证理论分析的正确性并证明算法的性能。本实验在 Sun Ultra45 工作站(1.6 GHz UltraSPARC Ⅲi 处理器,4 GB 内存,Sorlaris10 操作系统)上实现。大规模数据的坐标优化方法在 LIBLINEAR 平台上实现。LIBLINEAR 是由 Lin C J 教授带领的工作组编写完成的开源代码(http://www.csie.ntu.edu.tw/~cjlin/),它既是可以独立运行的软件系统,又是可以借以继续开发新算法的平台。该平台设计了科学合理的数据结构,用多种语言进行实现,可以高效处理数据和安排内存开销,受到了众多国家教育科研机构的青睐。本实验采用标准 C/C++语言实现,可以方便地跨平台运行。

本文用到四个大规模数据库,即 astro-physic、CCAT、a9a 和 covtype,它们都属于文本分类库,如表 1 所示。四个数据库可在 http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/进行下载。

表 1　大规模库描述

数据集	训练样本	测试样本	维数
astro-physic	29 882	32 487	99 757
CCAT	10 000	9 996	1 355 191
a9a	24 703	7 858	123
covtype	522 911	58 101	54

实验中所有的大规模数据库的调谐参数均取 $\lambda=1/m$。一般情况下,随机坐标下降算法的性能一般要优于其对应的循环坐标下降算法性能,但为获得稳定可靠的实验结论,本文采用循环的坐标方法进行实验比较。讨论的坐标下降算法可以分为三种:

a)直接排序求取原问题的精确解 w_j^*(primal coordinate descent,PCD),只使用红黑树排序与分段技巧,直接对子问题的解析解进行精确求解;

b)线搜方式求原问题的精确解 w_j^*(primal coordinate descent with exact line-search,PCD-EL),使用红黑树排序与分段技巧,并采取 3.2 节中的线搜方法求精确解;

c)非精确线搜方式求原问题的近似解 $\tilde{w}_j$(primal coordinate descent with non-exact line-search,PCD-NEL),使用红黑树排序与分段技巧,并采取 3.3 节中的非精确线搜方法近似求解析解。

本文实验的主要目的在于比较上述三种不同求解方式对收敛速度的影响并考察算法的单调性。

1)L1 损失 L2 正则化的实验比较

考虑 L1 损失和 L2 正则化项(L1-R-L2)。由于 L2 正则化项具有强凸、一阶可微等性质,求解此类问题的算法比较多。最有效的原始算法有批处理 comid(batch comid)[17] 和批处理 RDA(batch RDA)[16]。文献[3]中对偶坐标下降(DCD)能够有效求解 L1-R-L2 的问题,其实验表明 DCD 速度超过许多优化方法,如 Pegasos、TRON 和 SVM^{perf}。这里选择 PCD、PCD-EL、PCD-NEL 与循环对偶坐标下降算法(CDCD)、batch RDA 和

batch Comid 进行比较。本实验采用习惯性的方法计算 $F(W^t)$ 并记录对应的 CPU 时间,实验结果如图 2 ~ 5 所示。

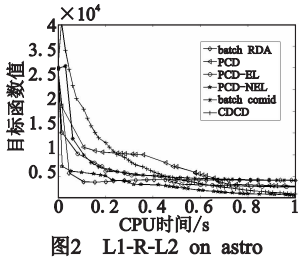

图2 L1-R-L2 on astro

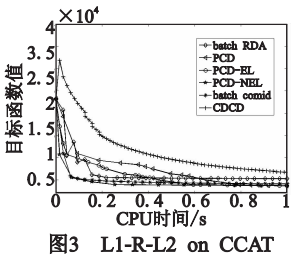

图3 L1-R-L2 on CCAT

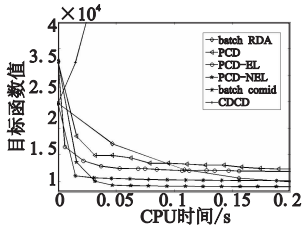

图4 L1-R-L2 on a9a

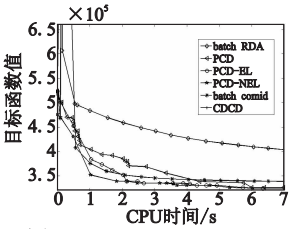

图5 L1-R-L2 on covtype

2)L1 损失 L1 正则化的实验比较

考虑 L1 损失和 L1 正则化(L1-R-L1)的情况。由于 L1-R-L1 问题的损失函数和正则化项都是非光滑次可微,且正则化项又是一般凸次可微,很少文献或算法讨论过 L1-R-L1 问题,在文献[18]中也没有涉及到此类情况。目前,没有公开报道非光滑损失加的坐标下降算法,为了验证算法的有效性,实验选择与 batch comid 以及 batch RDA 相比较,实验结果如图 6 ~ 9 所示。

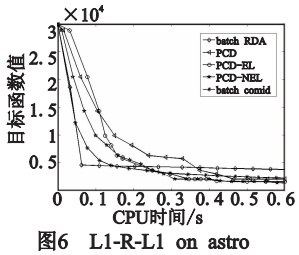

图6 L1-R-L1 on astro

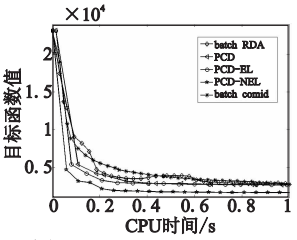

图7 L1-R-L1 on CCAT

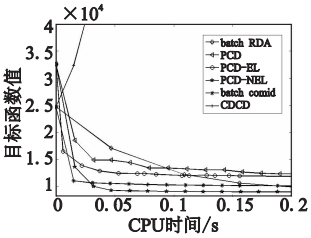

图8 L-R-L1 on a9a

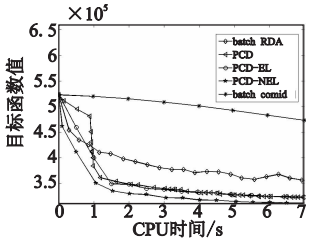

图9 L1-R-L1 on covtype

3)实验结论及分析

从实验中可以发现以下三点现象:

a)PCD、PCD-EL 和 PCD-NEL 与 batch comid、batch RDA 及 CDCD 的收敛速度相当,但是 PCD、PCD-EL 和 PCD-NEL 在原问题中目标函数都是单调递减的,而其他算法不具有严格的单调性。

b)PCD、PCD-EL 与 PCD-NEL 之间收敛速度满足 PCD≤PCD-EL≤PCD-NEL,目标函数值之间存在 PCD-NEL ≤PCD-EL≤ PCD。

c)当 $m < n$ 时,在 astro-physics 和 CCAT 库上,PCD 和 DCD 相比收敛速度相当;当 $m \gg n$,在 a9a 和 covtype 库上,PCD 比 DCD 表现出更加优越的性能。这说明当样本个数远大于维数时,DCD 就显得力不从心,甚至在某些数据库中原始问题的目标函数都不能够收敛。

从目标函数值来看,实验中可以得到 PCD-NEL ≤PCD-EL≤PCD的规律。从第 3 章算法分析可知 PCD-EL≤ PCD 都是精确求子问题的解析解,只是 PCD-EL 相比 PCD 用了线搜策略,计算复杂度有所降低,但实际每一步解析解的值跟 PCD 一样,因此两个算法收敛后目标值是相等。

相比 PCD-EL 和 PCD 每一步的精确求解,PCD-NEL 每一次迭代均是近似解,但是它所得到的目标值一般情况下均小于前两种算法。原因在于 CD 算法是一种近似算法,通过固定解向量中的 $n-1$ 个坐标不动,对某一个坐标求解单变量子问题。虽然每一步都得到子问题的精确解对于单变量而言是最优的,而对于全局变量并不一定是最优的,在很多情况下把每一步优化得最好反而会影响全局解的最终结果。

基于上述实验比较,说明了 PCD 解大规模非光滑原始问题的正确性与高效性,也进一步论证了相关的算法分析。

5 结束语

本文以求解大规模机器学习问题为背景,针对机器学习中非光滑损失问题,提出了一种新的求解 Hinge 损失的坐标下降算法。分别对 L2/L1 正则化项 + Hinge 损失子问题的求解进行了详细的推导,采用排序搜索方式求解析解。在此基础上又提出三种加速策略提升算法的计算效率,最后通过实验验证了算法的正确性和高效性。在实验中发现线搜步长 d 的选择对算法的收敛速度影响较大,因此下一步将探索线搜的改进策略。

参考文献:

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

[2] CHANG Kai-wei, HSIEH C J, LIN C J. Coordinate descent method for large-scale L2-loss linear support vector machines[J]. Journal of Machine Learning Research,2008,9(7):1369-1398.

[3] HSIEH C J, CHANG Kai-wei, LIN C J, *et al*. A dual coordinate descent method for large-scale linear SVM[C]//Proc of the 25th International Conference on Machine Learning. New York: ACM Press, 2008:408-415.

[4] SAHA A, TEWARI A. On the finite time convergence of cyclic coordinate descent methods[EB/OL]. (2010-05-12). http://arxiv.org/abs/10052146.

[5] NESTEROV Y. Efficiency of coordinate descent methods on huge-scale optimization problems[R]. Belgium: University Catholique de Louvain, Center for Operations Research and Econometrics,2010:2-20.

[6] LUO Zhi-quan, TSENG P. On the convergence of the coordinate descent method for convex differentiable minimization[J]. Journal of Optimization Theory and Applications,1992,72(1):7-35.

[7] LUO Zhi-quan, TSENG P. On the linear convergence of descent methods for convex essentially smooth minimization[J]. SIAM Journal Control Optimization,1992,30(2): 408-425.

[8] FU Wen-jiang. Penalized regressions: the bridge versus the lasso[J]. Journal of Computational and Graphical Statistics,1998,7(3): 397-416.

[9] FRIEDMAN J, HASTIE T, HÖFLING H, *et al*. Pathwise coordinate optimization[J]. Annals of Applied Statistics,2007,1(2):302-332.

[10] WU Tong-tong, LANGE K. Coordinate descent algorithms for lasso penalized regression[J]. Annals of Applied Statistics,2008,2(1): 224-244.

(下转第 3700 页)

(上接第 3692 页)

[11] SHALEV-SHWARTZ S, SINGER Y, SREBRO N. Pegasos: primal estimated sub-gradient solver for SVM[C]//Proc of the 24th International Conference on Machine Learning. New York: ACM Press, 2007: 807-814.

[12] JOACHIMS T. Training linear SVMs in linear time[C]//Proc of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. New York: ACM Press, 2006: 82-95.

[13] SMOLA A J, VISHWANATHAN S V N, LE Q. Bundle methods for machine learning[C]//Advances in Neural Information Processing Systems. 2008: 1377-1384.

[14] TSENG P, YUN S. A coordinate gradient descent method for nonsmooth separable minimization[J]. Mathematical Programming, 2009, 117(1-2): 387-423.

[15] SHALEV-SHWARTZ S, TEWARI A. Stochastic methods for L1 regularized loss minimization[C]//Proc of the 26th Annual International Conference on Machine Learning. New York: ACM Press, 2009: 929-936.

[16] XIAO Lin. Dual averaging methods for regularized stochastic learning and online optimization[J]. Journal of Machine Learning Research, 2010, 11(3): 2543-2596.

[17] DUCHI J, SHALEV-SHWARTZ S, SINGER Y, *et al*. Composite objective mirror descent[C]//Proc of the 23rd Annual Conference on Learning Theory. 2010: 14-26.

[18] YUAN Guo-xun, CHANG Kai-wei, HSIEH C J, *et al*. A comparison of optimization methods and software for large-scale L1-regularized linear classification[J]. Journal of Machine Learning Research, 2010, 11(3): 3183-3234.

[19] FRANC V, SONNENBURG S. Optimized cutting plane algorithm for support vector machines[C]//Proc of the 25th International Conference on Machine Learning. New York: ACM Press, 2008: 907-914.