NPA 算法及其修正*)

NPA Algorithm and it's Modification

王国胜

(北京邮电大学信息工程学院 北京100876)

Abstract NPA is currently one of the best training algorithm for support vector machine classifier. In this paper, by a large amount of experiments and further analysis based on [4], we find some defects in NPA and then present a modification. Computational experiments show that our algorithm has good performance.

Keywords Machine learning, Support vector machine, Iteration, Algorithm

1 引言

支撑向量机(Support Vector Machine,简称 SVM)是九十年代中期发展起来的机器学习技术[1,6],随着这项技术越来越受到人们重视,一些训练算法应运而生。。1998年,Platt[5]提出一个迭代算法称为 SMO 算法(Sequential Minimal Optimization),基本思想是运用一种顺序最小优化方法解二次规划的对偶问题,这是目前最好的算法之一。最近,Keerthi 等人[3]提出最近点 NPA 算法(Nearest Point Algorithm),它基于以下事实:支撑向量机的优化问题等价于两个凸多面体之间的最近点问题。Keerthi 等人的工作是把求最近点问题的 Gilbert 算法和 MDM 算法[2,4]巧妙地结合在一起。NPA 与 SMO 相比训练速度毫不逊色,甚至还有优势。本文在大量实验和深入分析的基础上,发现 NPA 算法还存在某些欠缺,进而加以修正。新算法与 NPA 相比,训练速度又明显提高。

2 支撑向量机与最近点问题

2.1 支撑向量机

本文研究具二次惩罚项的支撑向量机。设 x_k 表示输入向量,F 是特征空间(高维内积空间),特征向量 $z_k\in F, z_k=\Phi(x_k)$,Φ 是特征映射。核函数 $K(x_k,x_l)=\Phi(x_k)\cdot\Phi(x_l)$,"·"表示 F 中的内积。设 $\{x_k\}_{k=1}^m$ 是训练集,I,J 分别是第一、第二类模式的指标集,$I\neq\phi, J\neq\phi, I\cup J=\{1,2,\cdots,m\}$,$\{z_k\}$ 线性可分(Non-Violation)时与支撑向量机相关的优化问题是

$$\min \quad \frac{1}{2}\|w\|^2$$

$$\text{s.t.} \quad w\cdot z_i+b\geqslant 1, \forall\, i\in I; w\cdot z_j+b\leqslant -1, \forall\, j\in J \quad \text{(SVM-NV)}$$

$\{z_k\}$ 非线性可分时采用二次惩罚(Violation-Qardratic),相关的优化问题是

$$\min \quad \frac{1}{2}\|w\|^2+\frac{\tilde{c}}{2}\sum_k \xi_k^2$$

$$\text{s.t.} \quad w\cdot z_i+b\geqslant 1-\xi_i, \forall\, i\in I; w\cdot z_j+b\leqslant -1+\xi_j, \forall\, j\in J \quad \text{(SVM-VQ)}$$

用二次惩罚比线性惩罚来描述支撑向量机有如下优点:第一,约束条件中不含 $\xi_k\geqslant 0$,因为 $\xi_k\geqslant 0$ 一定不是最小值点;第二,通过以下线性变换,可转化为 SVM-NV 问题:

$$\tilde{w}=\begin{pmatrix} w \\ \sqrt{\tilde{c}}\,\xi \end{pmatrix}; \tilde{b}=b; \tilde{z}_i=\begin{bmatrix} z_i \\ \frac{1}{\sqrt{\tilde{c}}}e_i \end{bmatrix}, i\in I; \tilde{z}_j=\begin{bmatrix} z_j \\ -\frac{1}{\sqrt{\tilde{c}}}e_j \end{bmatrix}, j\in J$$

其中 e_k 是 m 维向量,它的第 k 个元素为1,其余元素为0。换句话说,把特征空间进一步扩大后,$\{\tilde{z}_i\}_{i\in I}$ 与 $\{\tilde{z}_j\}_{j\in J}$ 线性可分了,相应的核函数 $\tilde{K}(x_k,x_l)=K(x_k,x_l)+\frac{1}{\tilde{c}}\delta_{kl}$,其中 $\delta_{kl}=1$,$k=l$;$\delta_{kl}=0$,$k\neq l$;第三,Keerthi 等人的实验表明[3],基于 SVM-VQ 的支撑向量机有时能得到更高的识别率。这样只要研究如何解 SVM-NV 就够了。

2.2 最近点问题

设 S 是一非空集合,coS 是 S 的凸包即 $\text{coS}=\{\Sigma\beta_k s_k : s_k\in S, \beta_k\geqslant 0, \Sigma\beta_k=1\}$,记 $U=co\{z_i : i\in I\}$,$V=co\{z_j : j\in J\}$,因为 I,J 是有限集,所以 U,V 是凸多面体。考虑 U 与 V 之间的最近点问题:

$$\min \quad \|u-v\|; \text{s.t.} \quad u\in U, v\in V. \qquad \text{(NPP)}$$

有趣的是,SVM-NV 和 NPP 是等价的。

定理1[3] (w^*,b^*) 是 SVM-NV 的解,当且仅当存在 $u^*\in U, v^*\in V$,使得 (u^*,v^*) 是 NPP 的解,且 $w^*=\frac{2}{\|u^*-v^*\|^2}(u^*-v^*)$;$b^*=\frac{\|v^*\|^2-\|u^*\|^2}{\|u^*-v^*\|^2}$。

定理1的几何意义如图1所示。

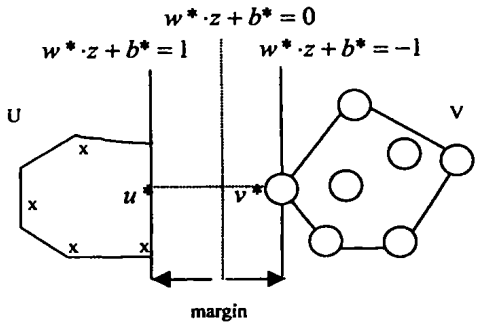


图1

若 $Z=\{\hat{z}_1,\cdots,\hat{z}_r\}\subset F$,$P=\text{coZ}$,$h_P(\eta)=\max\{\eta\cdot p : p\in P\}$,因凸多面体上线性函数的最大点必是顶点,故 $h_P(\eta)=\max\{\eta\cdot\hat{z}_k : k=1,\cdots r\}$,记 $s_Z(\eta)=\arg\max\{\eta\cdot\hat{z}_k : k=1,\cdots,r\}$,$g_P(\eta,p)=h_P(\eta)-\eta\cdot p$。定义函数 $g(u,v)=g_U(v-u,u)+g_V$

*)国家自然科学基金资助(69982001)。王国胜 博士研究生,副教授,研究方向:机器学习,神经网络。

$(u-v,v)$。

定理2[3] 设 $u\in U, v\in V, z=u-v$，则 $g(u,v)\geqslant 0$ 且 (u,v) 是 NPP 的解当且仅当 $g(u,v)=0$。

定理3[3] 设 (w^*,b^*) 是 *SVM-NV* 的解。任给 $0<\varepsilon<1$，若 $\hat{u}\in U, \hat{v}\in V, \hat{z}=\hat{u}-\hat{v}$ 且 $g(\hat{u},\hat{v})\leqslant\varepsilon\|\hat{z}\|^2$，则 $\hat{w}=\frac{2}{-h_U(-\hat{z})-h_V(\hat{z})}\hat{z}, \hat{b}=\frac{h_V(\hat{z})-h_U(-\hat{z})}{h_V(\hat{z})+h_U(-\hat{z})}$ 是 SVM-NV 的可行解且 $1-\varepsilon\leqslant\frac{\|w^*\|}{\|\hat{w}\|}\leqslant 1$。

3 NPA 算法

解最近点问题的著名算法有 Gilbert 算法、MDM 算法和 NPA 算法，MDM 算法是 Gilbert 算法的改进，NPA 算法又是 MDM 算法的改进。

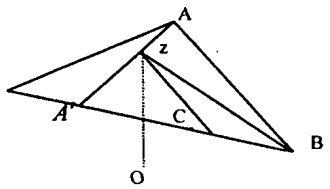


图2

设 $Z=U-V, W=\{z_i-z_j, Z: i\in I, j\in J\}$ 易证 $Z=coW$。我们通过图2来说明 NPA 算法的设计思想，$z=\Sigma\gamma_i w_i$ 是 Z 中的点，其中 $\Sigma\gamma_i=1, \gamma_i\geqslant 0, w_i\in W$，$A=w_{i\min}, i_{\min}$ 满足 $-z\cdot w_{i_{\min}}=\min_{i,\gamma_i>0}(-z\cdot w_i)$，$B=s_Z(-z)$，$C=z+\gamma_{i\min}(s_Z(-z)-w_{i\min})$，$A'=(z-\gamma_{i\min}w_{i\min})/(1-\gamma_{i\min})$，易见 A' 的凸组合中不含 $w_{i\min}$。实际上，线段 zB 是 Gilbert 算法[2]的搜索范围，线段 zC 是 MDM 算法[4]的搜索范围。因为 $C=z+\gamma_{i\min}(s_Z(-z)-w_{i\min})=(1-\gamma_{i\min})A'+\gamma_{i\min}B$，故 C 在线段 $A'B$ 上。考虑 $\Delta zA'B$，由于线段 zC 位于其中，因此它涵盖了 Gilbert 和 MDM 算法的搜索范围。NPA 正是选取 $\Delta zA'B$ 作为搜索范围。

为进一步提高收敛速度并节约内存，在实现上 NPA 算法还采取了一些其它措施。设 $u\in U, v\in V, z=u-v$，称下标 $k\in I\cup J$ 在 (u,v) 满足条件 Λ，如果下面两种情形之一成立：

$$k\in I\Leftrightarrow -z\cdot z_k+z\cdot u\geqslant\frac{\varepsilon}{2}\|z\|^2;$$

$$k\in J\Leftrightarrow z\cdot z_k-z\cdot v\geqslant\frac{\varepsilon}{2}\|z\|^2$$

显然若一切 k 都使 Λ 不成立，必有 $g_U(-z,u)\leqslant\frac{\varepsilon}{2}\|z\|^2$，$g_v(z,v)\leqslant\frac{\varepsilon}{2}\|z\|^2$，从而 $g(u,v)\leqslant\frac{\varepsilon}{2}\|z\|^2$，根据定理3，$u,v$ 即为满足给定精确度的近似解。

若 $u=\sum_{i\in\hat{I}}\beta_i z_i, v=\sum_{j\in\hat{J}}\beta_j z_j, \hat{I}\subset I, \hat{J}\subset J$ 且 $\beta_k>0, \forall k\in\hat{I}\cup\hat{J}, \sum_{i\in\hat{I}}\beta_i=1, \sum_{j\in\hat{J}}\beta_j=1$，则称 $z_k, k\in\hat{I}\cup\hat{J}$ 为支撑向量。

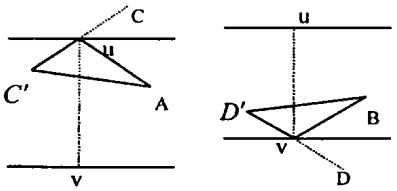
NPA 算法每轮循环由两个检验环节和一个迭代环节构成，首先进行第一类检验，在非支撑向量中寻找满足 Λ 的 k，然后第二类检验在支撑向量中寻找满足 Λ 的 k，如果在某个检验过程中找到了这样的 k，则进入迭代环节；之后重新检验，如此继续直至某轮循环中找不到满足 Λ 的 k 为止。为减少计算代价，每轮循环中只进行一次第一类检验，而第二类检验连续进行多次直到支撑向量中不存在满足 Λ 的 k 或检验次数达到某个预先设定的阈值为止，即在支撑向量上再形成一个子循环。这一点对提高算法的整体性能非常重要。

设 $i\min=\arg\min\{-z\cdot z_i: i\in\hat{I}\}$，

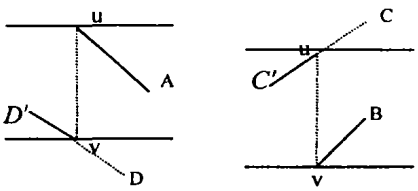
$j\min=\arg\min\{z\cdot z_j: j\in\hat{J}\}$

$$k\min=\begin{cases} i\min & if -z\cdot z_{i\min}+z\cdot u<z\cdot z_{j\min}-z\cdot v \\ j\min & \text{其它}, \end{cases}\quad \mu=\beta_{k\min}/(1-\beta_{k\min})$$

迭代环节分四种情形：



情形 1　　情形 2



情形 3　　情形 4

1. $k\in I, k\min\in I$. 求 v 到 $\Delta uC'A$ 的最近点，其中 $A=z_k, C'=(u-\beta_{k\min}z_{k\min})/(1-\beta_{k\min})=u+\mu(u-z_{k\min})$。

2. $k\in J, k\min\in J$. 求 u 到 $\Delta vD'B$ 的最近点，其中 $B=z_k, D'=(v-\beta_{k\min}z_{k\min})/(1-\beta_{k\min})=v+\mu(v-z_{k\min})$。

3. $k\in I, k\min\in J$. 求线段 uA 和线段 vD′ 之间的最近点，A，D′ 同上。

4. $k\in J, k\min\in I$. 求线段 uC' 和线段 vB 之间的最近点，C'，B 同上。

第二类检验比第一类检验细致，要计算 $i\max=\arg\max\{-z\cdot z_i: i\in\hat{I}\}$ 和 $j\max=\arg\max\{z\cdot z_j: j\in\hat{J}\}$，

令 $k=\begin{cases} i\max & if -z\cdot z_{i\max}+z\cdot u>z\cdot z_{j\max}-z\cdot v \\ j\max & \text{其它} \end{cases}$

NPA 算法的详细描述见文[3]。Keerthi 等人把 NPA 与 SMO 算法做了对比实验[3]，在学习速度上，当惩罚系数小时二者难分上下；惩罚系数大时，NPA 算法居绝对优势，因此它是目前最好的迭代算法之一。

4 修正的 NPA 算法

我们用 NPA 算法做了大量实验，结果显示，随着惩罚系数逐渐增大，第二类检验及其伴随的迭代的累积次数会几十倍几百倍地增加，而第一类检验及其伴随的迭代的累积次数却相对稳定。因此，若把第二类检验下的迭代过程进一步优化，NPA 的性能必将得到改善。

在第二类检验中，虽然进行了更细致的计算，获得了更多信息，但在其后的迭代过程中，并没有把这些信息全部利用起来（$i\max$ 或 $j\max$ 被丢弃了）。

迭代环节的第3，4两种情形，分别求线段 uA 和 vD'，uC' 和 vB 之间的最近点，这种选择缺乏有力的理论依据。

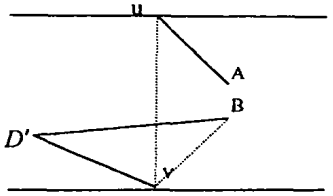


图3

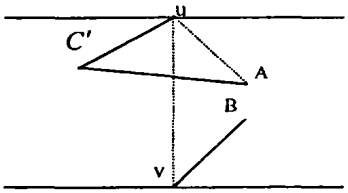


图4

针对上述问题，我们提出修正的 NPA 算法。在情形3，之所以选择求线段 uA 和 vD' 之间的最近点，一方面是为了保证 $\|u-v\|$ 呈下降趋势，另一方面是适时地去掉 V 中不必要的支撑向量(如 D)，然而这样做并不能及时有效地去掉 V 中不必要的支撑向量。我们注意到，D' 的凸组合中不含 D，而 B 在某种意义上常常靠近 U，B 在第二类检验中既已求出，若把 D' 和 B 两点连接起来，则线段 $D'B$ 上的点既与 D 无关又与 U 靠近，所以用 $D'B$ 代替 vD' 一举两得(如图3所示)。但是仅仅这样却不能保证 $\|u-v\|$ 连续下降，无法保证算法的收敛性，为此我们把原来的3修改成：计算 v 到线段 vA 的最近点 u_1 和最近距离 d_1，线段 uA 和线段 $D'B$ 的最近点 $(u_2,\tilde{v})$ 和最近距离 d_2，如果 $d_1<d_2$，令 $u=u_1$(v 不变)；否则 $u=u_2 v=\tilde{v}$。同理在情形4，如图4所示，计算 u 到线段 vB 的最近点 v_1 和最近距离 d_1，线段 $C'A$ 与线段 vB 的最近点 $(\tilde{u},v_2)$ 和最近距离 d_2，如果 $d_1<d_2$，令 $v=v_1$(u 不变)；否则 $u=\tilde{u}$，$v=v_2$。

5 模拟实验

算法用 C 语言实现，在 PentiumII_MMX300MHZ 机器上运行，使用与文[3]相同的三组标准样本数据：Checkers[3]，Adult_1和 Adult_4[5]，并采用高斯核函数 $K(x_i,x_j)=\exp(-0.5\|x_i-x_j\|^2/\sigma^2)$，每组实验相应的 σ^2，训练样本的维数 n 和样本容量 m，见表1。

表1

	σ^2	n	m
Checkers	0.5	2	465
Adult_1	10.0	123	1605
Adult_4	10.0	123	4781

针对不同的惩罚系数 $\tilde{c}$，实验进行了若干次，每次均设 $\varepsilon=0.001$，训练初值取每类在训练集中第一个出现的样本。实验除考察训练时间 T(CPU 时间：秒)之外，还有支撑向量机的 margin 和支撑向量个数 #SV。对相同的 $\tilde{c}$，修正的 NPA 与 NPA 得到几乎相同的 margin 和 #SV，故这两项结果未在此列出，训练时间 T 有显著差别，具体结果见表2～4。

表2 Checkers

$\tilde{c}$	10	50	100	500	10^3	10^4	10^5	10^6	10^7	10^8
NPA	2	3	5	10	13	28	57	83	66	85
修正的 NPA	2	3	4	8	12	27	53	66	75	66

表3 Adult_1

$\tilde{c}$	0.03	0.1	0.2	0.3	0.6	1.0	2.0	3.0	5.0	10	50	100	500	10^3
NPA	72	68	72	71	79	87	93	102	116	143	238	320	509	634
修正的 NPA	63	61	65	64	70	79	84	91	103	124	200	267	415	503

表4 Adult_4

$\tilde{c}$	0.03	0.1	0.2	0.3	0.6	1.0	2.0	3.0	5.0	10	50	100	500	10^3
NPA	586	573	636	688	808	861	949	994	1181	1519	3158	3957	8652	12449
修正的 NPA	527	526	583	629	745	791	873	899	1055	1323	2669	3303	7106	10293

一般来说，当 $\tilde{c}$ 增大(margin 减小)时，训练时间逐渐增大，支撑向量个数逐渐减小。在 $\tilde{c}$ 相同的条件下，与 NPA 相比我们的算法获得几乎相同的 margin 与支撑向量个数，训练时间却减少了，平均减幅在10%以上，$\tilde{c}$ 增大时减幅更大。每组实验都呈现出同样的特点，说明新算法的性能是稳定的。

参考文献

1 Cortes, Vapnik V. support vector networks. Machine Learning, 1995,20:273～297

2 Gilbert E G. Minimizing the quadratic form on a convex set. SIAM J. Contr. ,1966,4:61～79

3 Keerthi S S,et al. A fast iterative nearest point algorithm for support vector machine classifier design. IEEE tran. neur. networks, 2000,11(1)

4 Mitchell B F,Demyanov V F, Malozemov V N. Finding the point of a polyhedron closest to the origin. SIAM J. Contr. 1974,12:19～26

5 Platt J C. Fast training of support vector machines using sequential minimal optimization. in Advances in Kernel Methods: Support Vector Machines, B. Scholkopf, C. Burges, and A. Smola, Eds. Cambridge, MA:MIT Press, Dec. 1998

6 Vapnik V. The nature of statistical learning theory . New York: Sprinter-Verlag, 1995

7 王国胜，钟义信. 支撑向量机的若干新进展. 电子学报，2001,10