根据上述向量 a,构造 $T(\Omega)$上的连续函数 $L'(t)=a^T\cdot t$.既然 $T(\Omega)$有界,故 $L'(t)$在 $T(\Omega)$上必有下确界,即存在 c,使得 $\forall t\in T(\Omega)\Rightarrow L'(t)=a^T\cdot t\geqslant c$.根据 $T(\Omega)$的定义,上式等价于

$$\forall x\in\Omega\Rightarrow a^T\cdot T(x)\geqslant c \tag{7}$$

令 $\rho(x)=a^T\cdot T(x)-c$.不难看出,上述式(6)对应于秩函数定义中的条件(c),式(7)对应于条件(a).因此,根据式(6)和式(7)以及秩函数定义,易知 $\rho(x)=a^T\cdot T(x)-c$ 恰好是程序 Q 的秩函数,且是代数分式型的秩函数.从上述分析可以看出,若使得式(5)或者式(6)成立的向量 a 被找到,那么式(7)自然成立,从而可以构造得到程序 Q 的秩函数 $\rho(x)=a^T\cdot T(x)-c$.因此,计算程序 Q 的秩函数,我们可以通过计算可满足式(5)或式(6)的向量 a 来得到.由此可得下列命题.

命题 4. 记号同上.若存在 a^T 使得 $\forall u\in U(\Omega)\Rightarrow L(u)=a^T\cdot u\geqslant 1$, 那么程序 Q 有代数分式秩函数.要寻找使得式(5)成立的向量 a^T,我们可先对Ω的像集 $U(\Omega)$进行分析.

情形(I):$O\in U(\Omega)$.不难看出,倘若 $\mathbb{R}^k$ 空间中的原点 O 落在 $U(\Omega)$中,即 $O\in U(\Omega)$,那么,对任意固定的向量 a^T,式(5)都不可能成立.这是因为 $a^T\cdot O\equiv 0\not\geqslant 1$. 这等价于,倘若存在点 $x\in\Omega\subseteq\mathbb{R}^n$, 使得 $T(x)-T(M(x))=0$. 那么,对任意固定的向量 a^T, $a^T\cdot(T(x)-T(M(x)))\equiv 0\not\geqslant 1$. 而程序 Q 被给定时,像集 $U(\Omega)$将依赖于 $T(x)$的选择.秩函数的计算依赖于 $T(x)$的选择.因此,在选择 $T(x)$时,需要首先满足下面的限制条件:

$$\forall x\in\Omega\subseteq\mathbb{R}^n\Rightarrow T(x)-T(M(x))\neq 0 \tag{C1}$$

限制条件(C1)成立等价于 $O\notin U(\Omega)$.

命题 5. 记号同上.假设 $U(x)$在Ω上有定义.倘若 $\mathbb{R}^k$ 空间中的原点 $O\notin U(\Omega)$, 那么,计算使得式(5)成立的向量 a^T 的问题就等价于:在空间 $\mathbb{R}^k$ 中寻找可将原点 $O\in\mathbb{R}^k$ 与像集 $U(\Omega)\in\mathbb{R}^k$ 严格分离的超平面.

证明:若存在 a^T 使得式(5)成立,则 $\forall u\in U(\Omega)\Rightarrow a^T\cdot u-1\geqslant 0$. 令 $\pi(u)=a^T\cdot u-\frac{1}{2}$, 则有

$$\forall u\in U(\Omega)\Rightarrow a^T\cdot u-1=\left(a^T\cdot u-\frac{1}{2}\right)-\frac{1}{2}=\pi(u)-\frac{1}{2}\geqslant 0.$$

故 $\forall u\in U(\Omega)\Rightarrow\pi(u)\geqslant\frac{1}{2}>0$. 同时,因为选择的 $T(x)$满足限制条件(C1),故 $O\notin U(\Omega)$, $\pi(O)=0-\frac{1}{2}=-\frac{1}{2}<0$.

故超平面 $\pi(u)=a^T\cdot u-\frac{1}{2}$ 严格分离原点 O 与像集 $U(\Omega)$.反之,若存在超平面 $\pi(u)=a^T\cdot u+k$ 严格分离原点 O 与像集 $U(\Omega)$,则有两种情况:

(i) $\forall u\in U(\Omega)\Rightarrow\pi(u)=a^T\cdot u+k>0$ 且 $\pi(O)<0$;

(ii) $\forall u\in U(\Omega)\Rightarrow\pi(u)=a^T\cdot u+k<0$ 且 $\pi(O)>0$.

下面将证明无论哪种情况发生,满足式(5)的向量 a^T 都必然存在.不失一般性,这里仅证明(i),(ii)的证明是类似的.考虑情形(i),即 $\forall u\in U(\Omega)\Rightarrow\pi(u)=a^T\cdot u+k>0$ 且 $\pi(O)<0$, 则有 $\pi(O)=0+k=k<0$. 因此, $a^T\cdot u-(-k)>0\Rightarrow\frac{a^T}{-k}\cdot u-1>0$. 显然,向量 $\frac{a^T}{-k}$ 满足式(5).对情形(ii),同理可得,向量 $\frac{a^T}{-k}$ 满足式(5). □

命题 5 将程序 Q 的秩函数计算问题与两个点集(单点集$\{O\}$与像集 $U(\Omega)$的隔离问题建立了等价关系.下面的情形(II)表明,即便原点 $O\notin U(\Omega)$, 能严格分离单点集$\{O\}$与像集 $U(\Omega)$的超平面也未必存在.

情形(II):虽然原点 $O\notin U(\Omega)$, 但原点 O 却落在 $U(\Omega)$的边界上,即 O 与 $U(\Omega)$之间没有间隙.譬如: $U(\Omega)=\{(u_1,u_2),u_2^2-u_1>0\}$. 显然,原点 $O=(0,0)\notin U(\Omega)$,但在边界上.由命题 5 可知,既然寻找使得式(5)成立的向量 a^T 等价于寻找可将原点 O 与像集 $U(\Omega)$严格分离的超平面,那么,当情形(II)发生时,那样的严格分离超平面显然不可能存在.但是,倘若 $U(\Omega)$是个闭的,那么,当 $O\notin U(\Omega)$ 时,原点 O 与 $U(\Omega)$具有间隙,因而有可能存在超平面将 O 与 $U(\Omega)$严格分开.不幸的是,根据程序 Q 定义及Ω的定义不等式系统可知,既然不等式系统 $\boldsymbol{C}(\boldsymbol{x})\triangleright 0$ 中至少含有一个严格不等式,故Ω不是闭的,因而其在 U 映射下的像 $U(\Omega)$不是闭的.尽管 $U(\Omega)$不是闭的,但 O 与像集 $U(\Omega)$之间的严格分离超平面的探测问题可以放松为去探测 $\mathbb{R}^k$ 空间中可严格分离原点 O 与某个包含 $U(\Omega)$的闭集的严格

分离超平面 $\pi(u)$的问题.为此,首先给出一些记号.

令 $\overline{\Omega}=\{x\in\mathbb{R}^n:\boldsymbol{C}(\boldsymbol{x})\geqslant 0\}$,$\Omega^*=\overline{\Omega}\setminus\{O\}$, 这里,$\{O\}$为包含原点 O 的单点集.显然,$\overline{\Omega}$ 是包含Ω的闭集.假设 $U(x)$在Ω*上有定义,即 $U(x)$中各分量的分母在Ω*上均不为 0,既然 $\Omega\subseteq\Omega^*$, 那么,$U(x)$在Ω上也有定义.不难看出,$U(\Omega)\subseteq U(\Omega^*)$. 下面我们将证明 $U(\Omega^*)$是有界闭集.

命题 6. 记号同上.$U(\Omega^*)$是 $\mathbb{R}^k$ 中有界闭集.

证明:既然Ω*在几何上是一个由从原点出发的射线构成的锥,根据此前分析可知,对任意的λ>0,有 $U(x)=U(\lambda x)$.这表明,同一条射线上的任意两个非零点在映射 $U(x)$下的像是相同的.因此,既然同一条射线上的任意两个非零点在映射 $U(x)$下的像是相同的,故可在Ω*中的每条射线上各自寻找代表元.既然 $O\notin\Omega^*$, 则那样的代表元可以设置为 $\frac{x}{|x|},x\in\Omega^*$. 该代表元刚好是Ω*中的每条从原点出发的射线与单位球Θ相交的交点,即 $\frac{x}{|x|}\in\Omega^*\cap\Theta$. 因此, $U(x)=U\left(\frac{x}{|x|}\right)$, 故而有 $U(\Omega^*)=U(\Omega^*\cap\Theta)$. 此外,因为 $\overline{\Omega}\cap\Theta=(\Omega^*\cup\{O\})\cap\Theta=\Omega^*\cap\Theta$. 显然,因为 $\overline{\Omega}$ 是闭的,Θ是有界闭的,故 $\overline{\Omega}\cap\Theta$ 也是有界闭的.由上式可知 $\Omega^*\cap\Theta$ 也是有界闭的.既然前面已经假设 $U(x)$在Ω*上有定义,因此 $U(x)$在Ω*上连续.由连续函数性质可知,$\Omega^*\cap\Theta$ 在 $U(x)$的像 $U(\Omega^*\cap\Theta)$ 是有界闭的.

既然 $U(\Omega^*)=U(\Omega^*\cap\Theta)$, 故有 $U(\Omega^*)$是有界闭的. □

如果 $O\notin U(\Omega^*)$, 根据命题 6,既然 $U(\Omega^*)$是有界闭集,那么在 O 与像集 $U(\Omega^*)$之间存在间隙,因而有可能在二者之间存在严格分离超平面.既然 $U(\Omega)\subseteq U(\Omega^*)$, 当 $O\notin U(\Omega^*)$ 时,如果存在超平面 $\pi(u)$能够严格分离原点 O 与像集 $U(\Omega^*)$,那么 $\pi(u)$也能严格分离原点 O 与 $U(\Omega)$.类似于命题 5 的证明,我们有下列结论.

定理 7. 记号同上.假设 $U(x)$在Ω*上有定义且原点 $O\notin U(\Omega^*)\subseteq\mathbb{R}^k$. 如果存在超平面 $\pi(u)$可将原点 O 与像集 $U(\Omega^*)$严格分离,那么一定存在向量 a^T 使得式(5)成立.

证明:如果存在超平面 $\pi(u)=a^T\cdot u+k$ 可将原点 O 与像集 $U(\Omega^*)$严格分离,那么,既然 $U(\Omega)\subseteq U(\Omega^*)$, 则超平面 $\pi(u)=a^T\cdot u+k$ 也将严格地分离原点 O 与像集 $U(\Omega^*)$.根据命题 5 及其证明可知,一定存在向量 $a^T:=\frac{a^T}{-k}$ 满足式(5),即 $\forall u\in U(\Omega)\Rightarrow\frac{a^T}{-k}\cdot u\geqslant 1$. □

定理 7 表明,当定理的前提假设被满足时,程序 Q 的秩函数计算问题可以最终归结为原点 O 与 $U(\Omega^*)$的超平面分离问题.倘若那样的超平面被找到,那么 $\rho(x)=\frac{a^T}{-k}\cdot T(x)$ 就恰好是程序 Q 的代数分式秩函数.而后者显然是一个二分类问题.通常,二分类问题可以利用支持向量机(SVM)算法来解决.这里,令 $A=\{O\}$为一单点集,$B=U(\Omega^*)$.我们接下来的问题就是如何利用 SVM 算法求得一个可将集合 A 与 B 严格分离的超平面.一般地,在通常的二分类问题中,涉及到的两个数据集均是有限点集.而这里的二分类问题中,集合 B 显然是一个无穷点集.因此,为了便于利用 SVM 算法计算 A 和 B 的超平面,我们将采取下面的步骤.

S1:从 B 集中取出有限个点记为 $B_0\subseteq B$. 转 S2;

S2:调用 SVM 算法计算单点集 A 与有限点集 B_0 的分离超平面,记为 $\pi(u)=a^T\cdot u+k$. 转 S3;

S3:验证式(5): $\forall u\in U(\Omega)\Rightarrow\pi_0(u)=\frac{a^T}{-k}\cdot u\geqslant 1$ 是否成立.若成立,则表明程序 Q 有代数分式型秩函数,即 $\rho(x)=\frac{a^T}{-k}\cdot T(x)$. 反之,从 B 集中取出新的有限点集 $B_1(B_0\subset B_1)$. 令 $B_0:=B_1$, 转 S2.

上述步骤 S1 中,采用打格子的方式进行 B_0 的构造.这一点将在下一小节中加以具体介绍.在 S2 中,当调用 SVM 算法计算单点集 A 与有限点集 B_0 的分离超平面 $\pi_0(u)$时,理论上会有失败的可能,即那样的超平面可能不存在,但在具体的大量实验中我们发现,那样的超平面总存在.在 S3 中,式(5)的验证可归结为验证: $\forall x\in\Omega\Rightarrow\frac{a^T}{-k}\cdot(T(x)-T(M(x)))\geqslant 1$. 同时,既然 $T(x)$含有根式表达式,故我们利用强有力不等式验证器 BOTTEMA 来验证

上式是否成立.在 S3 中,若划分 A 与 B_0 的超平面经过验证不可能对应一个分式秩函数时,我们将通过更加细分格子的思路来得到含有更多点的有限点集 B_1.下面是基于上述 3 个步骤建立的基于 SVM 的秩函数探测算法.

2.2 有限点集B_0的构造

在上述的 S1 中,需要从 $B=U(\Omega^*)$中取出有限个点构成 B_0.由命题 8 的证明可知,$U(\Omega^*)=U(\Omega^*\cap\Theta)$.这里,$\Omega^*\cap\Theta$ 是一个有界闭集,Θ为单位球.因此,要构造 B_0,我们可以先从有界闭集 $\Omega^*\cap\Theta$ 中取出有限点集 C,然后通过 $U(x)$映射得到点集 $U(C)$,从而得到 $B_0=U(C)$.显然,直接在球面Θ上通过打格点的方式取点并不方便.但单位球Θ总是可以被包含在某个超立方体Δ中.故,为了构造 $\Omega^*\cap\Theta$ 上的点集 C,我们采取如下思路:先在区域 $\Omega^*\cap\Delta$ 上构造点集 S.这里,$\Delta=[-m,m]^n$ 为一超立方体.然后对点集 S 中的点进行归一化,得到归一化后的点集 S^0.显然,$S^0\subseteq\Omega^*\cap\Theta$.最后令 $C:=S^o$. 而在超立方体Δ上打格子是相对容易的.因此,在区域 $\Omega^*\cap\Delta$ 上构造点集 S 时我们可以通过对超立方体Δ打格子,然后将落入Ω^*的格点收集起来构成点集 S.图 2 所示为一个算法化的取点过程.

输入:程序 Q 对应的Ω^*;超立方体$\Delta=[-m,m]^n$;程序变量 $x=(x_1,\ldots,x_n)$;取点间距 h;
输出:B_0.
过程:
1:for $x_1=-m,-m+h,-m+2h,\ldots,m$ do
2: for $x_2=-m,-m+h,-m+2h,\ldots,m$ do
3: ⋮
4: for $x_n=-m,-m+h,-m+2h,\ldots,m$ do
5: if $x=(x_1,\ldots,x_n)\in\Omega^*$ then
6: 将 x 放入 S 中;
7: 将点 x 归一化得到 $x^0=\frac{x}{|x|}$, 并将 x^0 保存到训练样本集 S^o 中;
8: end if
9: end for
10: ⋮
11: end for
12: end for
13: 令 $C:=S^o$
14:输出 $B_0=U(C)$

Fig.2 The algorithm of getting the sample points

图 2 取样本点算法

在上述算法中,需要设置 m 的值.在我们的实验中,m 被取为 100.此外,取点的间距 h 需要根据计算机的计算能力进行适当的调整,在计算机的实际计算能力范围内,间距越小,取得的样本点越多,最终计算出的代数分式函数经过验证成为真正秩函数的可信度就越高.

2.3 例 子

本节将通过一个例子来阐述上文提出的方法.

例 1:考虑下列循环:

$$Q_1 \quad \text{while} \quad 4x_1^2+x_2^2+16\leqslant 16x_1+6x_2\wedge x_1+4x_2\geqslant x_2^2+5 \quad \text{do}$$
$$\{x_1'=x_1^2-x_2^2;x_2'=x_1+x_2+1;\}$$

该循环若使用 RegularChains[26]或者 redlog[27]来求解线性或者非线性秩函数,均会因复杂度太高,而无法在有效的时间内得出结论.接下来将演示如何通过上文提出的基于 SVM 的方法来探测循环程序 P_1 的代数分式秩函数.

(a) 首先将循环程序 Q_1 进行齐次化处理,变成终止性等价的循环程序 Q_1' ,如下所示:

$$Q_1' \quad \text{while} \quad 4x_1^2+x_2^2+16x_3^2\leqslant 16x_1x_3+6x_2x_3\wedge x_1x_3+4x_2x_3\geqslant x_2^2+5x_3^2\wedge x_3>0 \quad \text{do}$$
$$\{x_1'=x_1^2-x_2^2;x_2'=x_1x_3+x_2x_3+x_3^2;x_3'=x_3^2;\}$$

其中, $\Omega=\{x:4x_1^2+x_2^2+16x_3^2\leqslant 16x_1x_3+6x_2x_3\wedge x_1x_3+4x_2x_3\geqslant x_2^2+5x_3^2\wedge x_3>0\}$ 是齐次循环程序 Q_1' 的循环条件所围成的空间,它是一个锥,也即,针对任意的 $x\in\Omega$, 都有 $\lambda x\in\Omega,\lambda>0$. 令

$\bar{\Omega}=\{x:4x_1^2+x_2^2+16x_3^2\leqslant 16x_1x_3+6x_2x_3\wedge x_1x_3+4x_2x_3\geqslant x_2^2+5x_3^2\wedge x_3>0\}$. 令 $\Omega^*=\bar{\Omega}\setminus\{O\}$. 这里,$O$ 为原点(0,0,0). 对该例,我们可令 $T(x)=\left(\frac{x_1}{|x|},\frac{x_2}{|x|},\frac{x_3}{|x|}\right)$. $T(x)$的每一个分量的奇点均为原点 O,而Ω^*中并不包含原点,故 $T(x)$在Ω^*上连续.这里, $x=(x_1,x_2,x_3,),|x|=\sqrt{x_1^2+x_2^2+x_3^2}$. 令 $T(x')=\left(\frac{x_1'}{|x'|},\frac{x_2'}{|x'|},\frac{x_3'}{|x'|}\right)$, 其中, $x'=(x_1',x_2',x_3')$. 既然 $x'=M(x)=(x_1^2-x_2^2,x_1x_3+x_2x_3+x_3^2,x_3^2)$, 因此,

$$U(x)=T(x)-T(x')=T(x)-T(M(x))=\left(\frac{x_1}{|x|}-\frac{x_1'}{|x'|},\frac{x_2}{|x|}-\frac{x_2'}{|x'|},\frac{x_3}{|x|}-\frac{x_3'}{|x'|}\right).$$

这里,$|x'|=\sqrt{(x_1')^2+(x_2')^2+(x_3')^2}$. 因此,$U(x)$将原三维空间(原像空间)中的点映射到了另外一个三维空间(像空间).根据定理 7,我们可通过支持向量机 SVM 的方法在像空间中去探测可将原点 O 与像点 $U(\Omega^*)$严格分隔开的超平面 $\pi(u)=a^T\cdot u+k$.若那样的超平面被找到,则 $\rho(x)=\frac{a^T}{-k}\cdot T(x)$ 就恰好是程序 Q_1' 的代数分式秩函数.但是,我们需要事先验证该例是否满足定理 7 的前提条件,即判断:$U(x)$在Ω^*上是否有定义且是否原点 $O\notin U(\Omega^*)$.

(i) 判断 $U(x)$在Ω^*上是否有定义.这等价于判断是否存在使得 $|x|=0\vee|x'|=0$ 成立的点会落入到Ω^*中.利用 Maple 中的命令 solve 不难验证,使得 $|x|=0\vee|x'|=0$ 成立的点仅有原点 O=(0,0,0).而Ω^*并不包含原点.故 $U(x)$在Ω^*上有定义.

(ii) 判断是否原点 $O\notin U(\Omega^*)$. 这等价于判断下列系统是否有解:

$$\left.\begin{aligned}&\frac{x_1}{|x|}-\frac{x_1'}{|x'|}=0\wedge\frac{x_2}{|x|}-\frac{x_2'}{|x'|}=0\wedge\frac{x_3}{|x|}-\frac{x_3'}{|x'|}=0\wedge x_1'=x_1^2-x_2^2\wedge x_2'=x_1x_3+x_2x_3+x_3^2\wedge x_3'\\&=x_3^2\wedge 4x_1^2+x_2^2+16x_3^2\\&\leqslant 16x_1x_3+6x_2x_3\wedge x_1x_3+4x_2x_3\\&\geqslant x_2^2+5x_3^2\wedge x_3\\&\geqslant 0\wedge(x_1\neq 0\vee x_2\neq 0\vee x_3\neq 0)\wedge|x'|\\&=\sqrt{(x_1')^2+(x_2')^2+(x_3')^2}\end{aligned}\right\}\tag{8}$$

若式(8)无解,则表明 $O\notin U(\Omega^*)$. 由于代数分式以及根式的出现使得我们直接计算上述系统是否有解较为困难,因此,我们采取了一些化简措施.比如,将上述系统中的代数分式化为多项式,利用平方手段消除根号等.由此我们得到下列多项式系统:

$$\left.\begin{aligned}x_1^2((x_1')^2+(x_2')^2+(x_3')^2)&=(x_1')^2(x_1^2+x_2^2+x_3^2)\wedge x_2^2((x_1')^2+(x_2')^2+(x_3')^2)\\&=(x_2')^2(x_1^2+x_2^2+x_3^2)\wedge x_3^2((x_1')^2+(x_2')^2+(x_3')^2)\\&=(x_3')^2(x_1^2+x_2^2+x_3^2)\wedge x_1'=x_1^2-x_2^2\wedge x_2'=x_1x_3+x_2x_3+x_3^2\wedge x_3'\\&=x_3^2\wedge 4x_1^2+x_2^2+16x_3^2\leqslant 16x_1x_3+6x_2x_3\wedge x_1x_3+4x_2x_3\geqslant x_2^2+5x_3^2\wedge x_3\geqslant 0\end{aligned}\right\}\tag{9}$$

显然,式(8)的任何一个解也必是式(9)的解.调用 Maple 中的命令 solve 计算可得式(9)仅有零解(0,0,0).显然, $(0,0,0)\notin\Omega^*$. 因为零解不可能是式(8)的解,故式(8)无解.这表明原点 $O\notin U(\Omega^*)$.

综上所述,该循环程序满足定理 7 的前提条件.因此,根据定理 7,秩函数计算问题可归结为 O 与 $U(\Omega^*)$之间严格分离超平面的计算问题.

(b) 取样本点.

采用第 2.2 节介绍的取点算法,并将该算法中的参数 m=100.在本例中我们令取点间距 h=1.由于在 $\bar{\Omega}$ 约束条件中有 $x_3\geqslant 0$ 的条件,故 x_3 不必从–100 开始取点.通过取点算法,我们得到 $U(\Omega^*)$中的有限点集 $B_0=U(C)$.将 B_0 中的点的标签置为 1,而令单点集 $A=(O)$的标签为–1.接下来调用 SVM 算法计算 A 与 B_0 的严格分离超平面

$\pi_0(u)=a^T\cdot u+k$.

(c) 使用 LIBSVM 软件包计算出像空间中的划分超平面.

本文采用 MATLAB 调用 LIBSVM 软件包的方式来计算一个“硬分隔”的超平面.令 $\overline{D}=A\cup B_0$. 在求得划分超平面之后,需要对 $\overline{D}$ 中的点进行一次完整的测试,测试结果必须是 100%完全精确才能进行下一步操作.对该例通过计算得到划分超平面中 a^T 和 k 的值,

$$a=(2.798739925884132,-6.058645127373750,6.167554957795089),k=-1.000000000061875.$$

由上述计算得到的 a^T 和 k 的值可确定一个严格分离超平面 $\pi_0(u)=a^T\cdot u+k$,它将 $A=\{O\}$ 与 $B_0(\subseteq U(\Omega^*))$ 严格分离.即:

(i) $\forall u\in B_0\Rightarrow\pi_0(u)=a^T\cdot u+k>0$ 且 $\pi_0(O)<0$;

(ii) $\forall u\in B_0\Rightarrow\pi_0(u)=a^T\cdot u+k<0$ 且 $\pi_0(O)>0$.

由类似于命题 5 的证明可知,无论哪种情形发生,都将有

$$\forall u\in B_0\Rightarrow\frac{a^T}{-k}\cdot u-1>0.$$

接下来将验证超平面 $\pi_0(u)$是否能将 $A=\{O\}$与 $B=U(\Omega^*)$严格分离.为此,就需要验证下列两种情形之一是否成立.

(i) $\forall u\in U(\Omega^*)\Rightarrow\pi_0(u)=a^T\cdot u+k>0$ 且 $\pi_0(O)<0$;

(ii) $\forall u\in U(\Omega^*)\Rightarrow\pi_0(u)=a^T\cdot u+k<0$ 且 $\pi_0(O)>0$.

既然像点 $u\in U(\Omega^*)$且由 $U(\Omega^*)$的定义可知 $U(\Omega^*)=\{u\in\mathbb{R}^k:u=U(x),x\in\Omega^*\}$, 那么验证上述(i),(ii)两式就等价于验证:

(i*) $\forall x\in\Omega^*\Rightarrow\beta(x)=a^T[T(x)-T(M(x))]+k>0$ 且 $k<0$;

(ii*) $\forall x\in\Omega^*\Rightarrow\beta(x)=a^T[T(x)-T(M(x))]+k<0$ 且 $k>0$.

这是因为,显然地,若(i)(或(ii))成立,那么(i*)(或(ii*))也必成立.反之亦然.

(e) 使用 BOTTEMA 软件包进行验证.

既然 $k=-1.000000000061875<0$,那么我们需要验证:

$$\forall x\in\Omega^*\Rightarrow\beta(x)=a^T[T(x)-T(M(x))]+k>0$$

是否成立.由于最终需要验证式(5)或式(6)是否成立,所以,在本例中,我们直接验证:

$$\forall x\in\Omega\Rightarrow\beta(x)=a^T[T(x)-T(M(x))]+k>0$$

是否成立.BOTTEMA 是一款强有力的不等式证明软件.它可以验证带复杂根式的代数表达式的不等式是否成立.我们可以在 Maple 上调用 BOTTEMA 中的各类函数.通过验证发现,

$$\forall x\in\Omega\Rightarrow\beta(x)=\frac{a^T}{-k}[T(x)-T(M(x))]-1>0$$

的确成立.故,

$$\rho(x)=\frac{a^T}{-k}\cdot T(x)=\frac{(2.798\,739\,925\,884\,132,-6.058\,645\,127\,373\,750,6.167\,554\,957\,795\,089)^T}{-1.000\,000\,000\,061\,875}\left(\frac{x_1}{|x|},\frac{x_2}{|x|},\frac{x_3}{|x|}\right)$$

是该程序的代数分式型秩函数.因此循环程序是可终止的.

3 实 验

下面给出基于 SVM 的多项式循环程序秩函数探测的具体算法.

实验是在一台装有 7.86GB 的可用内存、处理器频率为 2.59GHz、操作系统 64 位的 Windows 操作系统的计算机上进行的.下面我们选取了 4 个循环程序用于对比.每个循环程序具体的计算方法可以参照前文中的例 1.

针对表 1 中的 6 个循环程序,我们先后用量词消去工具 Redlog 和 Maple 中的 RegularChains 软件包来计算

各自的线性秩函数,从表 2 可以看出,除了循环程序 P_4 和 P_5 外,其他 5 个循环程序都不能通过量词消去方法来证明其是终止的.原因在于,量词消去算法的复杂度太高,当循环程序的赋值语句或循环条件中含有非线性项时,无法在有效的时间内给出计算结果.在表 2 中,unknown 表示计算时间超过 10 个小时仍未得出结果.Terminating 表示相应的线性秩函数或代数分式秩函数被成功找到,故表明循环是终止的.特别地,在 Terminating 后面的括号中给出了计算秩函数所花费的时间.显然,当循环中的表达式是线性时,基于量词消去的工具即能有效地计算它们的线性秩函数.但,当表达式中含有非线性项时,实验结果表明,量词消去工具较难在可接受的时间内计算线性秩函数.此外,本文提出的基于 SVM 的方法所涉及的时间开销包括:样本点集的构造所需时间、SVM 计算时间以及用 BOTTEMA 验证代数分式是否为秩函数所需时间.在实验中我们发现,本文方法最主要的时间开销在于样本点集的构造.譬如,循环 P_3,样本点集的构造花费了 76 分钟.当然,我们也可以尝试使用量词消去的方法来合成非线性秩函数,但是,非线性项的引入同样会带来高复杂度的问题.从表 2 中可以看出,本文提出的基于 SVM 的秩函数探测方法能够探测出上述 6 个循环程序的确具有代数分式型秩函数,从而证明了这 6 个程序均是可终止的.上述 6 个循环程序在使用 SVM 的方法时,各自的取点间距 h 以及所产生的划分超平面 $a^T\cdot u+k$ 中的参数 a^T 和参数 k 的计算值见表 3.

输入:单分支循环程序 $\tilde{P}$;

输出:如果验证通过输出相应的非线性秩函数和 Terminating;否则,输出 Unknown.

过程:

1: 将循环程序齐次化为 P.记其循环条件围成的区域为Ω,赋值语句为 $M(x)$;

2: 根据循环程序 P 中 x 的维数设置相应的单项式向量 $T(x)$,令 $U(x)=T(x)-T(M(x))$;

3: 将 P 的循环条件Ω中的严格不等式全部替换成非严格的不等式得到 $\bar{\Omega}$.令 $\Omega^*=\bar{\Omega}\setminus\{O\}$. 验证 $U(x)$在Ω^*上是否有定义且原点是否 $O\notin U(\Omega^*)$;

4: 选择合适的取点间距 h,并利用上述取样本点算法从 $U(\Omega^*)$选取训练样本集 B_0;

5: 将原点 O 添加进去构成 $\overline{D}=\{O\}\cup B_0$, 并将原点 O 的标签设置为−1,设置 B_0 中点的标签为 1;

6: 调用 LIBSVM,在 $\overline{D}$ 中计算一个超平面 $\pi_0(u)=a^T\cdot u+k$,将 B_0 中的点与原点 O 分隔开;

7: 利用工具 BOEETA,验证向量 $\frac{a^T}{-k}$ 是否满足公式(6);

11: if 验证通过 then

12: $\rho(x)=\frac{a^T}{-k}\cdot T(x)$ 是该程序的代数分式型秩函数.输出 $\rho(x)$和 Terminating;

13: else

14: 输出 Unkown;

15: end if

16: end if

Fig.3 Ranking function detection of polynomial loop programs based on SVM

图 3 基于 SVM 的多项式循环程序秩函数探测

Table 1 More examples

表 1 更多实例

P_2: while $(y^2+10\leqslant x+6y\wedge x^2+6\leqslant 4x+y)$ $\begin{cases}x_1=x+y;\\ y_1=y-1;\end{cases}$	P_3: while $(x^2+y^2\leqslant 1)$ $\begin{cases}x_1=x-y^2+1;\\ y_1=y+x^2-1;\end{cases}$
P_4: while $(x-y\geqslant 1\wedge x+y\geqslant 1\wedge x\leqslant 10)$ $\begin{cases}x_1=y;\\ y_1=y-1;\end{cases}$	P_5: while $(x\geqslant 0\wedge y\leqslant -1)$ $\begin{cases}x_1=x+y;\\ y_1=y-1;\end{cases}$
P_6: while $(4\leqslant x\leqslant 5\wedge 1\leqslant y\leqslant 2)$ $\begin{cases}x_1=x;\\ y_1=-xy+y^2+1;\end{cases}$	P_7: while $(x\geqslant 0\wedge y-2x\geqslant 1)$ $\begin{cases}x_1=-x^2-4y^2+1;\\ y_1=-xy-1;\end{cases}$

从表 3 可以看出,循环程序 P_3 和 P_4 所采用的取点间距为 0.5,而循环程序 P_2 采用的取点间距为 0.2.从理论上来说,取点间距越小,所选取的样本点越多,最终计算出的代数分式函数被成功验证为秩函数的可能性就越

大.但是考虑到计算机的实际计算能力,针对不同的循环程序,取点间距要作适当的调整.

Table 2 Comparison of several tools

表 2 几种工具的比较

循环程序	Redlog	RegularChains	SVM
P_2	unknown	unknown	Terminating(53.9min)
P_3	unknown	unknown	Terminating(79.5min)
P_4	Terminating(0.24s)	Terminating(125min)	Terminating(74.3min)
P_5	Terminating(0.2s)	Terminating(6.156s)	Terminating(41.7min)
P_6	unknown	unknown	Terminating(2.7min)
P_7	unknown	unknown	Terminating(1.5min)

Table 3 The parameters generated using the SVM method

表 3 使用 SVM 的方法所产生的参数

循环程序	取点间距 h	a^T	k
P_2	0.2	[–2.323273254461750,3.655297165341511,0.743880981998555]	–0.999 999 997 404 251
P_3	0.5	[–5.303998785021788,5.303967565999045,4.733548642017029]	–1.000 213 463 236 426
P_4	0.5	[114.542235487223684,136.417264854622173,–199.707636874516821]	–0.999 859 927 713 084
P_5	0.5	[82.9547,194.7744,–1.8830]	–0.999 8
P_6	0.01	[–0.4922,5.1377,2.5196]	–1
P_7	0.5	[0.9720,1.0254,0.2133]	–0.999 9

4 总 结

本文提出了一种基于支持向量机(SVM)理论的求解单重无条件分支多项式循环程序秩函数的方法.该方法将秩函数计算问题归结为二分类问题,然后利用SVM工具计算出一个候选的代数分式型函数,最后借助不等式自动验证工具BOTTEMA对其进行验证,以确定该代数分式型函数是否为循环程序的秩函数.由于SVM方法内部采用数值计算,因此相较于现有的基于量词消去的秩函数计算方法,本文方法能在可接受时间内探测形式较为复杂的秩函数.实验结果表明,针对某些循环程序,传统的量词消去方法不能在合理时间内判断出它们是否有线性秩函数,但通过本文提出的方法却可以找到其代数分式型秩函数.

References:

[1] Turing A. On computable numbers, with all application to the entscheidungsproblem. London Mathematical Socity, 1936,42(2): 230–265.

[2] Yang L, Zhou CC, Zhan NJ, Xia BC. Recent advances in program verification through computer algebra. Frontiers of Computer Science, 2010,4(1):1–16.

[3] Tiwari A. Termination of linear programs. Computer Aided Verification, 2004,3114:70–82.

[4] Braverman M. Termination of integer linear programs. In: Ball T, Jones RB, eds. Proc. of the Computer Aided Verification (CAV 2006). Springer-Verlag, 2006. 372–385.

[5] Xia BC, Yang L, Zhan NJ, Zhang ZH. Symbolic decision procedure for termination of linear programs. Formal Aspects of Computing, 2009,23(2):171–190.

[6] Xia BC, Zhang ZH. Termination of linear programs with nonlinear constraints. Journal of Symbolic Computation, 2010,45(11): 1234–1249.

[7] Li Y. Witness to non-termination of linear programs. Theoretical Computer Science, 2017,681:75–100.

[8] Liu J, Xu M, Zhan NJ, Zhao HJ. Discovering non-terminating inputs for multi-path polynomial programs. Journal of Systems Science and Complexity, 2014,27:1284–1304.

[9] Chen YF, Heizmann M, Lengal O, Li Y, Tsai MH, Turrini A, Zhang LJ. Advanced automate-based algorithms for program termination checking. In: Proc. of the ACM SIGPLAN Conf. on Programming Language Design and Implementation (PLDI 2018). ACM, 2018. 135–150.

[10] Zhang J. A path-based approach to the detection of infinite looping. In: Proc. of the 2nd Asia-Pacific Conf. on Quality Software (APAQS 2001). Hong Kong: IEEE Computer Society, 2001. 88–96.

[11] Leike J, Heizmann M. Geometric nontermination arguments. In: Proc. of the 24th Int'l Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2018). Berlin, Heidelberg: Springer-Verlag, 2018,10806:266–283.

[12] Colón M, Sipma H. Synthesis of linear ranking functions. In: Margaria T, Yi W, eds. In: Proc. of the 7th Int'l Conf. on Tools and Algorithms for Construction and Analysis of Systems (TACAS 2001). Springer-Verlag, 2001. 67–81.

[13] Podelski A, Rybalchenko A. A complete method for the synthesis of linear ranking functions. In: Proc. of the Verification, Model Checking, and Abstract Interpretation (VMCAI). Berlin: Springer-Verlag, 2004. 239–251.

[14] Bagnara R, Mesnard F, Pescetti A, Zaffanella E. A new look at the automatic synthesis of linear ranking functions. Information and Computation, 2012,215:47–67.

[15] Bagnara R, Mesnard F. Eventual linear ranking functions. In: Proc. of the 15th Symp. on Principles and Practice of Decla-rative Programming (PPDP). Madrid: ACM Press, 2013. 229–238.

[16] Li Y, Zhu G, Feng Y. The *L*-depth eventual linear ranking functions for single-path linear constraint loops. In: Proc. of the 10th Int'l Symp. on Theoretical Aspects of Software Engineering (TASE 2016). IEEE, 2016. 30–37.

[17] Leike J, Heizmann M. Ranking templates for linear loops. In: Proc. of the 20th Int'l Conf. on Tools and Algorithms for Construction and Analysis of Systems (TACAS 2014). Berlin, Heidelberg: Springer-Verlag, 2014,8413:172–186.

[18] Ben-Amram AM, Genaim S. On multiphase-linear ranking functions. In: Computer Aided Verification. Cham: Springer-Verlag, 2017,10427:601–620.

[19] Chen YH, Xia BC, Yang Lu, Zhan NJ, Zhou CC. Discovering non-linear ranking functions by solving semi-algebraic systems. In: Proc. of the Int'l Colloquium on Theoretical Aspects of Computing (ICTAC 2007). Berlin: Springer-Verlag, 2007. 34–49.

[20] Cousot P. Proving program invariance and termination by parametric abstraction, langrangian relaxation and semidefinite programming. In: Proc. of the 6th Int'l Conf. on Verification, Model Checking, and Abstract Interpretation (VMCAI 2005). Springer-Verlag, 2005. 1–24.

[21] Shen LY, Wu M, Yang ZF, Zeng ZB. Generating exact nonlinear ranking functions by symbolic-numeric hybrid method. Journal of Systems Science and Complexity, 2013,26(2):291–301.

[22] Li Y. Termination of semi-algebraic loop programs. In: Proc. of the Int'l Symp. on Dependable Software Engineering: Theories, Tools and Applications (SETTA 2017). Springer-Verlag, 2017. 131–146.

[23] Platt J. Sequential minimal optimization: A fast algorithm for training support vector machines. Technical Report, MSR-TR-98-14, Microsoft Research, 1998.

[24] Hsu CW, Chang CC, Lin CJ. A practical guide to support vector classification. Technical Report, Department of Computer Science, Taiwan University, 2003.

[25] Zhou ZH. Machine Lerning. Beijing: Tsinghua University Press, 2016 (in Chinese).

[26] The RegularChains Library. http://regularchains.org/index.html

[27] Redlog—Computing with Logic. http://redlog.eu/

附中文参考文献:

[25] 周志华.机器学习.北京:清华大学出版社,2016.

李轶(1980－),男,重庆人,博士,副研究员,CCF 专业会员,主要研究领域为程序验证,符号计算.

冯勇(1965－),男,博士,研究员,博士生导师,主要研究领域为数值混合计算.

蔡天训(1993－),男,工程师,主要研究领域为程序验证,嵌入式系统.

吴文渊(1976－),男,博士,研究员,主要研究领域为同伦计算.

樊建峰(1993－),男,硕士生,CCF 学生会员,主要研究领域为程序验证,区块链.